Universidad de la República
Facultad de Ingeniería

RoCo: Robots Comunicados

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Gabriel Bibbó, Mariana Gelós, Martín Randall

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutores

Pablo Belzarena	Universidad de la República
Federico La Rocca	Universidad de la República
Leonardo Steinfeld	Universidad de la República

Tribunal

Gabriel Gómez	Universidad de la República
Sebastián Fernández	Universidad de la República
Federico Lecumberry	Universidad de la República

Montevideo
Miércoles 20 de Diciembre, 2017

RoCo: Robots Comunicados, Gabriel Bibbó, Mariana Gelós, Martín Randall.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 224 páginas.
Compilada el lunes 19 febrero, 2018.
http://iie.fing.edu.uy/

No fracasé, sólo descubrí 999 maneras de cómo no hacer una bombilla.

Thomas Alva Edison

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

A pesar del riesgo probable de aburrir al lector, queremos destinar algunas líneas para agradecer a quienes en algún sentido nos brindaron ayuda durante la realización de este proyecto. Pedimos disculpas si se nos escapa algún nombre, sépase que no es de forma intencional, sino producto del olvido y de la gran cantidad de colaboradores con que tuvimos la suerte de contar.

Queremos empezar por agradecer a nuestros tutores, Pablo Belzarena, Federico La Rocca y Leonardo Steinfeld, por los consejos brindados y la guía durante este "casi año y medio".

También a varios docentes y compañeros del IIE, que supieron estar siempre dispuestos a responder dudas, recomendar material y pensar con nosotros problemas y soluciones: a Claudina Rattaro y Benigno Rodríguez sobre las antenas, a Rafael Canetti y Mauricio González sobre el GPS, a Francisco Veirano sobre la impresora 3D, a Pablo Pérez sobre las soldaduras de los B200 mini, a Germán Capdehourat y Gabriel Gómez sobre los encabezados de 802.11, al grupo de TallerInE Robótico sobre las pilas, a Paola Romero sobre GWN-RR, a Gregory Randall y Gonzalo Belcredi sobre la documentación, a Santiago Paternain sobre el problema de optimización, a Martha Delgado por su ayuda para las gestiones de compras, y especialmente a Víctor González, que colaboró en el desarrollo de la CSMA/CA y con varios consejos, algunos de los cuales lograron pasar los filtros de la terquedad y la juventud. En particular, también a los compañeros de la oficina 1135 "El Establo" por soportarnos en convivencia durante este año: Pablo, Martín, Gonzalo y Mauricio.

Queremos agradecer al Club de Bochas del Parque Rodó, tanto al local como a los deportistas, que nos permitieron con mucha gentileza usar el club para pruebas. También a Daniel Randall con su contribución al logo del proyecto, a Rosario Balparda y Andrés Gelós por sus comentarios sobre la documentación, a Luis Bibbó por sus aportes en la corrección de estilo, y a Ana Frau por su ayuda en la construcción de los robots.

Queremos reflejar también nuestra gratitud al vasto mundo casi anónimo conformado por los foros, debates, y códigos abiertos del mundo de la electrónica, de las telecomunicaciones y demás que nos fueron de gran utilidad para encontrar pistas, soluciones y respuestas.

Finalmente, agradecemos a nuestros amigos, familias y compañeros, por su apoyo constante durante la realización de este proyecto.

Esta página ha sido intencionalmente dejada en blanco.

Martín desea dedicar este trabajo:
A Pablo V. Carlevaro.
A su familia: Papá y Mamá, Lía y Guzmán, Daniel y Mariana, Guillermo y Emma, y demás; y a sus amigos: ComeRanas, Murito y otros.
A sus compañeros, que mantienen este presente de lucha y ese futuro de esperanza.

Mariana desea dedicar este trabajo:
A su familia: Rosario, Andrés, Gonzalo y María Pía y a sus amigos.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

Este proyecto se enmarca en el trabajo que ha llevado adelante el grupo ARTES[1] en los últimos años. Una de las vertientes que ha explorado el grupo es el desarrollo de una extensión de GNU Radio[2] que permita trabajar con SDR[3] en el modelado e implementación de redes de datos (GWN)[4]. Por otro lado la robótica cooperativa ha tenido un avance importante en los últimos tiempos, y ha planteado desafíos interesantes a resolver a nivel de comunicación entre robots.

El trabajo realizado en este proyecto se enfoca en ensayar la combinación de SDR y robótica móvil. Para organizar el intercambio de datos entre los nodos, se utiliza un protocolo de acceso al medio (CSMA/CA)[5] desarrollado sobre GWN. Los robots consisten en un chasis con cuatro ruedas y sus respectivos motores, fuentes de energía, una computadora, un microcontrolador, el SDR y sensores para posicionarse y esquivar obstáculos.

El problema planteado consiste en lograr el mayor acercamiento posible de un robot (Explorador) a una ubicación dada, manteniendo siempre comunicación con una estación base (con cierta QoS[6]). En el escenario en que existe pérdida de calidad de servicio, por ejemplo porque aumenta la distancia entre los nodos, entra en juego un segundo robot (Repetidor) que al actuar como repetidor mejora el intercambio y permite al Explorador acercarse más a la posición objetivo. Este ejercicio se puede pensar como una forma muy sencilla de robótica cooperativa comunicada por SDR.

[1] Grupo ARTES (Análisis de Redes, Tráfico y Estadísticas de Servicio): grupo académico de la Universidad de la República vinculado al estudio de las redes de datos `https://iie.fing.edu.uy/investigacion/grupos/artes/es/inicio/`.

[2] GNU Radio: herramienta de desarrollo libre y abierta para la implementación de sistemas de radio definida por software.

[3] SDR (Software Defined Radio): radios definidas por software, tecnología de comunicación inalámbrica que busca trasladar la mayor cantidad de funciones posibles a su ejecución en software, dejando en hardware las tareas mínimas indispensables.

[4] GWN (GNU Radio Wireless Networks): redes inalámbricas en GNU Radio.

[5] CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance): acceso múltiple por detección de portadora y prevención de colisiones.

[6] QoS (Quality of Service): calidad de servicio, se refiere a determinadas garantías que ofrece una comunicación.

El proyecto se centró en dar solución a tres aspectos fundamentales: la comunicación, la lógica general del sistema y la construcción de los robots autónomos. Las tres vertientes implicaron una etapa de búsqueda, diseño, implementación y pruebas. En varios casos se trabajó sobre modelos y se realizaron simulaciones para testear y corregir la solución propuesta previo a su implementación final.

Lograr una implementación final que resuelva el problema planteado constituye el *alma mater* del proyecto, por lo que el documento refleja un enfoque descriptivo de los problemas encontrados y las soluciones propuestas.

Prefacio

Este documento busca representar el trabajo colectivo del grupo de proyecto RoCo durante el desarrollo del proyecto de fin de carrera. A la vez que redactado para disposición del tribunal que evaluará los resultados del proyecto, se piensa como un documento que pueda ser entendido por lectores con escasos conocimientos en la temática, por lo que se solicita cierta tolerancia frente a introducciones aparentemente obvias, o su contracara, debates que abordan problemas técnicos específicos.

Los autores

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación

"El hombre (y la mujer) *es por naturaleza sociable, con lo cual quiero decir que los hombres, aparte de la necesidad de auxilio mutuo, desean invenciblemente la vida social".* [1] Estemos de acuerdo o no con esta afirmación de Aristóteles, es claro que las relaciones sociales se han desarrollado como un componente fundamental de la vida misma. El ser humano se estructura y se organiza en sociedad. No es posible concebir la existencia humana sin vínculos con sus semejantes. En este contexto, la comunicación ha cobrado una importancia tan grande, que incluso hemos definido períodos enteros de existencia de la humanidad en torno a la misma: el inicio de la escritura se asocia al inicio de la historia. Desde la aparición del lenguaje articulado (considerado uno de los elementos distintivos de la condición humana) hasta la expresión de ideas en menos de 280 caracteres en las redes, las comunicaciones describen y escriben a la sociedad.

El desarrollo del lenguaje articulado y de la escritura para el intercambio de información vivió desde el siglo XIX transformaciones gigantescas, con la posibilidad de comunicarse en tiempo real a larga distancia. Y no es que este impulso se haya marchitado con el paso del tiempo, al revés, la importancia de las comunicaciones para la producción económica, las guerras o situaciones de emergencia, los asuntos particulares, en fin, para el progreso (o retroceso) de la civilización y la vida de sus individuos no ha hecho más que aumentar.

Con una fuerte apuesta desde los estados nacionales, acompañados y retroalimentados por la economía de mercado, las comunicaciones han pasado a ser un espacio de disputa y renovación permanente, donde encuentran aplicación y desarrollo múltiples avances tecnológicos. En el siglo XX, la revolución que provocó la aparición del mundo digital multiplicó las posibilidades de intercambios, abriendo nuevos horizontes para las comunicaciones.

Es que hoy ya no sólo debemos considerar que cada individuo puede dispo-

ner de varios sistemas de comunicaciones en su bolsillo (por ejemplo, un teléfono portátil incluye numerosos protocolos), sino que además se avanza hacia que los aparatos, las cosas en definitiva, tengan sus propios sistemas de comunicación.

Las comunicaciones digitales inalámbricas encontraron utilidad ya no solamente entre seres humanos, sino también para distribuir tareas y comunicar agentes y procesos, como pueden ser robots. La multiplicación de dispositivos inalámbricos ha planteado no pocos problemas: desde la distribución del espectro (el canal a través del cual enviamos nuestros mensajes) hasta la organización eficiente de los intercambios (protocolos). También ha introducido numerosos debates en torno a la dependencia existente de las multinacionales de las telecomunicaciones (pensar que "wifi", ahora incluido en el diccionario, es el nombre de una marca), o frente a la pertinencia y el uso de las tecnologías de comunicación: para guerras, violaciones de derechos humanos o control social.

Así surgen modos alternativos de desarrollo tecnológico, como ser el software y hardware libre, que se enfrentan a la concentración de conocimiento. La disminución de la tasa de ganancia obtenida con las apuestas a hardware específico en un mercado restrictivo, y el desarrollo de comunidades cooperativas de base tecnológica ha planteado cambios en las formas de producción de tecnologías vinculadas a las telecomunicaciones. Aparecen entonces con fuerza apuestas que buscan que con el mismo hardware se logre establecer diferentes vías de comunicación -desarrolladas con sus especificidades en software- lo que permite flexibilidad, adaptabilidad, disminución de costos y aprovechamiento de infraestructura. Este concepto aplicado a comunicaciones inalámbricas tiene por nombre SDR: Software Defined Radio (radios definidas por software), y si bien ha tenido un impulso reciente con mejoras tecnológicas en las capacidades de procesamiento, se remonta en sus orígenes a los finales de la década del 80 [48].

También van cobrando importancia conceptos como robótica cooperativa, donde un conjunto de robots desarrolla tareas de manera organizada, de forma tal que el resultado global sea superior a la sumatoria de los resultados particulares. Esto permite sistemas con mayor inmunidad ante fallas, de menor costo, cumpliendo tareas de mayor complejidad, adaptables y presentando soluciones inherentemente distribuidas [81] [21].

Aunque las ventajas que presentan estos sistemas son importantes, también vienen asociadas algunas dificultades. Por un lado, presenta las complejidades propias del manejo de una red de datos, donde los recursos deben ser administrados para optimizar el resultado final. A esto se agregan los problemas propios de los robots móviles, que surgen cuando un agente autónomo sale a un ambiente dinámico, donde las condiciones pueden cambiar sin información previa y el robot debe ser capaz de adaptarse. La unión de ambos conlleva la aparición de dificultades adicionales, por ejemplo las referidas a problemas geométricos y la planificación de trayectorias para evitar colisiones.

Es evidente la importancia de la comunicación entre los agentes para el desarrollo de las funciones que se les encomienda. Muchas veces estos sistemas deben trabajar en escenarios donde no es posible establecer comunicación a través de los medios tradicionales, por no estar en zonas de cobertura o por dificultades intrínsecas del problema planteado. Por ejemplo en el escenario de robótica móvil para minería, o en casos de rescates en ambientes poco amigables (en el océano o un incendio), o de guerra tras líneas enemigas.

A su vez, la idea de que un único protocolo es suficiente para ordenar de manera homogénea y eficiente el uso del canal, ya ha perimido. En redes de datos, se encarga a un área específica el control de acceso al medio (subcapa MAC, por sus siglas en inglés). Esta subcapa cuenta con numerosos protocolos de organización de las comunicaciones, siendo uno de los más populares el acceso múltiple por detección de portadora y prevención de colisiones (CSMA/CA).

En los últimos tiempos, la necesidad de tener varios tipos de MAC se enfrenta a que éstas se implementan en tarjetas de red, en general con hardware y software propietario, lo que las hace difíciles de adaptar y modificar [57]. En particular varios trabajos han avanzado hacia la búsqueda de soluciones combinando software específico implementado sobre hardware de propósito general (como SDR) [88].

Por otro lado, SDR y GNU Radio [69] (el software con el que trabajamos en SDR) han concentrado esfuerzos en torno al tratamiento de flujos continuos de muestras, representando -por ejemplo- números complejos. Esto se aleja de las redes de datos, donde se intercambian segmentos, paquetes, tramas, en fin: un conjunto de muestras que sigue una estructura dada con inicio y final. El grupo ARTES de la Universidad de la República ha presentado recientemente un marco de trabajo para redes de datos en GNU Radio llamado GWN [34]. GWN (GNU Radio Wireless Networks) presenta un conjunto de herramientas y funcionalidades que facilitan la implementación de redes de datos sobre SDR, permitiendo investigación, exploración, testing y desarrollo, utilizando un entorno sencillo, completo y amigable.

Las SDR no se han generalizado como aplicaciones de usuario final. Según varios autores, esto se debe en particular a características como la velocidad de la conversión analógica-digital (que es considerado uno de los cuellos de botella para la ejecución de funciones con alta velocidad), o problemas debidos al alto consumo energético [48]. Es en este sentido que este proyecto representa un trabajo exploratorio de implementación, ya que buscamos combinar SDR con robots móviles autónomos. A nivel de las funciones de acceso al medio, si bien no entramos en el detalle fino de los tiempos de ejecución, nuestro sistema permite iniciar trabajo y estudio en esa dirección.

Las ventajas que podría representar la utilización de tecnologías flexibles y

versátiles como las SDR para las comunicaciones son muchas. Por un lado un mejor aprovechamiento de los recursos disponibles (hardware, por ejemplo). Por otro, posibilita el desarrollo de protocolos y sistemas de comunicación a medida de las necesidades. No en vano uno de los principales impulsores de la radio definida por software es el ejército de los Estados Unidos y en consecuencia importantes complejos industriales y centros académicos [48].

Entre las aplicaciones posibles a SDR se encuentra su utilización para el manejo de robots autónomos, en aplicaciones de robótica cooperativa o en el estudio de redes de nodos móviles. En estos escenarios se debe considerar la participación de múltiples agentes comunicados entre sí, y por ende compartiendo el canal. Esto implica necesariamente la organización de la comunicación y del uso del medio por parte de los actores involucrados, es decir, un protocolo de control de acceso al medio (por ejemplo CSMA/CA).

La combinación de SDR, robots móviles y control de acceso al medio representa el corazón de este proyecto, en el que se busca implementar con GWN funciones de acceso al medio (CSMA/CA) en radios definidas por software montadas sobre una estación base y robots móviles autónomos, que cooperan para el desarrollo de una red inalámbrica con garantías mínimas de calidad de servicio en un área geográfica determinada.

El trabajo realizado en este proyecto podría ser de utilidad a futuro en aplicaciones de robots móviles comunicados por SDR para el cumplimiento de funciones de mayor complejidad. Presenta igualmente la posibilidad de avanzar hacia el estudio de redes de nodos móviles, por ejemplo utilizando un esquema como el desarrollado en el proyecto para pruebas o simulaciones. El estudio del sistema como un problema de optimización permite mejorar la calidad de servicio y ofrecer resultados interesantes en relación a las comunicaciones en redes móviles. Finalmente, también se puede pensar en profundizar el trabajo realizado sobre el control de acceso al medio y SDR, buscando soluciones competitivas frente a las propuestas por el hardware especializado, o pensando aplicaciones vinculadas a redes cognitivas.

Pasaremos ahora a plantear los antecedentes, el problema a resolver, los objetivos generales y específicos, el alcance, e introduciremos los capítulos siguientes.

1.2. Antecedentes

Como antecedentes contamos con la experiencia del grupo ARTES del departamento de Telecomunicaciones en el proyecto CSIC - Grupos. En particular una de las líneas que se ha trabajado es el desarrollo de librerías e implementaciones de comunicaciones inalámbricas para SDR a través de GNU Radio.

En este sentido ya se han desarrollado proyectos de fin de carrera en el marco

de SDR, aunque no necesariamente referidos a la temática de comunicaciones entre robots. Como ejemplo se puede mencionar el proyecto "Implementación en un FPGA de la etapa de sincronismo de un receptor OFDM para recepción de señales de DTV del estándar ISDB-T", realizado en el período 2015-2016 por Florencia Ferrer y Daniel Contrera.

Por otro lado se pueden mencionar las pruebas realizadas por uno de nuestros tutores (Pablo Belzarena), acerca del funcionamiento de GNU Radio sobre Raspberry Pi corriendo en Arch Linux; así como las pruebas realizadas de envío de órdenes a robots a través de SDR, entre otras.

1.3. Descripción del proyecto

Se le encomendará la siguiente tarea a dos robots y una estación base que se encuentran comunicados a través de SDR. La tarea consistirá en que uno de los robots deba acercarse lo más posible a una coordenada (latitud, longitud) manteniendo un nivel mínimo en la calidad de comunicación. .

La estación base (o radiobase) es un PC con GNU Radio cuyo SDR es una placa Universal Software Radio Peripheral (de aquí en más USRP [73]) B200 mini.

Cada robot cuenta con:

- Un chasis y sus 4 ruedas, 4 motores, driver de dos canales, pilas y conversor DC/DC.
- Sensores para su ubicación, orientación y movilidad:
  - 2 encoders
  - GPS
  - Magnetómetro
  - Sensores de luz infrarroja para la detección de obstáculos
- Un microcontrolador Arduino
- Un SDR USRP B200 mini
- Un Raspberry Pi[1] comunicado con el Arduino y el B200 mini a través de USB

Se utilizan dos microcontroladores con el fin de independizar las tareas que le corresponden a cada uno de ellos. Por un lado, el microcontrolador Arduino es

[1]Raspberry Pi: computadora de placa única, de tamaño y costo reducidos.

el encargado de dirigir al robot a las coordenadas que reciba utilizando los sensores.

Paralelamente el Raspberry Pi se encarga de comunicarse con el otro robot y la estación base a través del B200 mini. Recibe de la radiobase hacia dónde debe moverse y se lo informa al Arduino para que lo haga. En el Pi corre el sistema operativo Arch Linux, una distribución de Linux. Sobre el Arch Linux corre GNU Radio.

Para la comunicación en capa física y MAC se parte de una capa física básica funcionando y de protocolos ARQ implementados en máquina de estados. Se debe implementar una MAC con una versión simple de CSMA/CA.

1.4. Objetivo General

Como objetivo general del proyecto se plantea:

- Explorar las complejidades que surgen de conjugar el mundo SDR con la robótica móvil cooperativa en vistas a identificar las dificultades existentes en las distintas etapas de su implementación.
- Validar y expandir, a partir de casos reales de utilización de protocolos de acceso al medio, el trabajo realizado por el grupo ARTES sobre SDR y redes inalámbricas.
- Construir los robots que ofrezcan las prestaciones necesarias para la realización del proyecto.

Para cumplir con esto se propuso el problema de establecer una comunicación entre dos robots y una estación base a través de GNU Radio. Los robots deberán llegar a cierta posición, la cual será indicada por la estación base, y la comunicación deberá mantener una QoS mínima en todo momento.

1.5. Objetivos Específicos

Los objetivos específicos planteados para este proyecto son:

- Comprender el funcionamiento de GNU Radio.
- Integrar en cada robot el control por Arduino, dirigido por un Raspberry Pi sobre el que corra GNU Radio.
- Estimar correctamente la posición de cada robot en todo momento.
- Controlar a distancia la posición de los robots desde la estación base.
- Estimar la QoS entre los robots y la estación base.

- Establecer un algoritmo sencillo que permita mantener una QoS mínima.
- Implementar un protocolo de comunicación utilizando uno de los robots como repetidor.
- Documentar el proyecto.

1.6. Alcance

Está comprendido dentro del alcance del proyecto:

- Lograr que al ubicar la radio base y determinar una posición razonablemente cercana, los robots se posicionen de forma tal de garantizar cierta QoS.
- Medición y estimación de la QoS según SNR o throughput.
- Desarrollar un algoritmo que indique la posición del robot que actúa como repetidor, de forma tal de mejorar la QoS.
- Establecer mecanismos de comunicación que permitan controlar los movimientos del robot y obtener datos del mismo (posición, calidad de servicio).

1.7. Criterios de éxito

Se considera que el proyecto es exitoso si:

- Se logra cierta calidad de servicio (QoS) que permita la transmisión de mensajes cortos, lo más cerca posible de la posición de destino. La radiobase debe poder recibir los mensajes, aún cuando ambos robots participen del enlace.
- Se logra un algoritmo competente para garantizar la QoS mínima.
- Se prueba la implementación de capa física proporcionada por el grupo ARTES y logra la implementación de la capa MAC.
- Se logra que los robots se comuniquen entre ellos.
- Se logra que el Explorador vaya a la posición indicada.
- Se logra calcular la posición a la que debe ir el segundo robot para actuar como repetidor, de forma de mejorar la calidad de servicio.

1.8. Estructura del documento

En el segundo capítulo (2) explicaremos el sistema completo que diseñamos para cumplir con las tareas planteadas en la descripción del proyecto. Se presentará la lógica general del sistema, así como en qué consiste cada agente a nivel de hardware y software, explicando brevemente las funciones que realiza cada una de las partes.

Continuaremos deteniéndonos en la comunicación. El capítulo (3) empieza por una breve explicación de las herramientas utilizadas: SDR, GNU Radio y GWN, y el concepto de máquinas de estados finitos (FSM). Esto dará paso a la descripción del protocolo CSMA/CA y a nuestra implementación del mismo, detallando otros protocolos en juego en la capa de enlace. Sigue la presentación de la capa física, encargada de la modulación y demodulación de las tramas. Se continúa con una somera descripción del hardware SDR utilizado, dando lugar a las pruebas realizadas, los problemas encontrados y las soluciones propuestas. Finalmente, se estudiará cómo se desarrolló la medida de la calidad de servicio y se plantean conclusiones del capítulo.

El capítulo (4) se centra en el movimiento de los robots. Se detallan las características y limitaciones de la estructura mecánica y el sistema de alimentación elegidos para los robots. Se describe la lógica y las funciones utilizadas en el microcontrolador Arduino. Abarca la elección, descripción y funcionamiento de los sensores empleados, en particular del GPS que se usa para el posicionamiento, incluyendo ciertas incompatibilidades encontradas con su incorporación. Se finaliza con conclusiones parciales respecto del hardware utilizado y posibles mejoras.

El capítulo (5) se concentra en la inteligencia central del sistema, deteniéndose en la implementación de la misma en cada nodo. Incluye la descripción de la incorporación de SDR y GWN al Raspberry Pi usando el sistema operativo Arch Linux. Se explica el algoritmo general de funcionamiento. En particular las funciones desarrolladas en cada nodo (incluida la estación base), así como las interacciones entre la inteligencia central y sus periféricos: con Arduino a través de la comunicación serial, con SDR a través de sockets. Finalmente, se presentan conclusiones y trabajo a futuro, como plantear al sistema como un problema de optimización para la elección de la posición del Repetidor.

El documento continúa en el capítulo (6) con la exposición de las pruebas realizadas respecto de nuestra solución propuesta, así como un análisis de los resultados obtenidos.

Para finalizar se establecen conclusiones generales (7) sobre las soluciones encontradas. Se evalúa el desarrollo del proyecto, incluida la gestión del mismo y el cumplimiento de los objetivos propuestos. Se termina con la detección de líneas de trabajo futuras, así como mejoras posibles para el desarrollo de sistema similares.

En los apéndices pueden encontrarse informaciones útiles respecto de pruebas realizadas, pasos para la instalación del software utilizado, mejoras posibles del sistema de navegación a considerar a la hora de realizar proyectos de robótica móvil y demás información específica que nos pareció adecuado compartir. El proyecto cuenta con un git `https://github.com/MartinRandallC/RoCo/wiki` donde se pueden encontrar los códigos utilizados, y parte del anexo que nos pareció bueno compartir públicamente.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Robots Comunicados: el sistema completo

Nuestro sistema consta de tres actores: la estación base, el robot Explorador y el robot Repetidor. El requerimiento común para los tres es la utilización de SDR para la comunicación. Explicaremos ahora qué funciones debe cumplir cada uno de los nodos y cómo está compuesto (hardware). La descripción del funcionamiento de cada nodo será profundizado en el capítulo 5, y la navegación de los robots será detallada en el capítulo 4. Sin embargo es necesario tener una visión general del sistema para entender mejor las partes. El esquema de la figura 2.1 pretende introducir a grandes rasgos cómo están formados los tres nodos del sistema: comunicación mediante SDR, Raspberry Pi para la inteligencia de los robots, y Arduino para controlar el movimiento e integrar la información de los sensores.

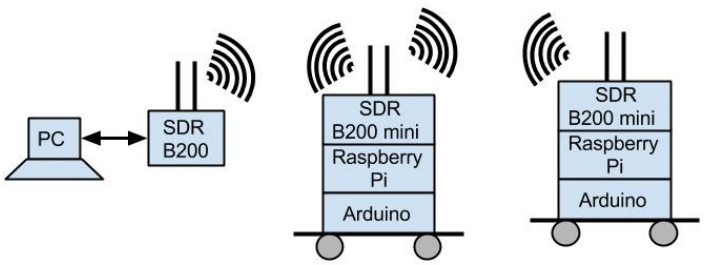


Figura 2.1: Esquema de los tres actores del sistema

El sistema global funciona como sigue:

- Se elige una posición de destino. Los nodos están prendidos y dispuestos a hablar y escucharse. La estación base mide la QoS y si es buena (inicialmente lo es porque están cerca), inicia el funcionamiento del conjunto indicando al robot Explorador la posición a la cual dirigirse. Continúa la medida de la calidad de servicio mientras este último avanza.

- El robot Explorador avanza por tramos. Para esto se orienta hacia dónde debe dirigirse y avanza una distancia fija en linea recta mientras no se encuentre con ningun obstáculo en el camino. En caso de que esto suceda, el robot se detiene y esquiva al obstáculo. En cada uno de los tramos que avanza, se está midiendo la calidad de servicio y si ésta es menor a un cierto umbral, el robot se detiene. Esto lo decide la estación base, que detiene o confirma el movimiento del robot mediante el envío de mensajes de parada o de avance a la posición de destino. Se considera la posición de destino es alcanzable por los robots, aunque para alcanzarla deban esquivar algunos obstáculos.

- Si el Explorador llega al destino se lo comunica a la estación base, en cuyo caso el problema ha sido resuelto exitosamente.

- Si entre el Explorador y la estación base se pierde la comunicación, es decir que la QoS pasa a estar por debajo del umbral planteado, la estación base le indica al Repetidor una posición de destino. Ésta se encuentra sobre el segmento de recta que va desde el origen (la posición de la estación base) y el destino final, de forma tal que el Repetidor se ubique entre la estación base y el Explorador.

- Para que el Repetidor llegue a su posición se realizan intercambios similares a los descriptos para el movimiento del Explorador. Al llegar éste, se vuelve a medir la calidad de servicio punta a punta (estación base a Explorador).

- De superar el umbral, el Explorador puede continuar avanzando nuevamente hasta llegar al destino, o hasta quedar tan cerca como se pueda sin pasar por debajo del umbral de QoS mínima.

2.1. Diseño implementado

Explicaremos brevemente cómo está compuesto en hardware cada nodo, introduciendo qué función debe cumplir cada una de las partes.

En la estación base se encuentra el gobierno del conjunto: se realiza gran parte del control del sistema. Ella es quien indica la posición de destino final al robot Explorador, controla su avance, realiza la medida de calidad de servicio, indica las posiciones a las que se debe dirigir el Repetidor.

Se compone de una computadora portátil y de un USRP B200. En la computadora están corriendo dos programas: la aplicación Python ("estacion_base.py") y un programa en Python ("Full_CSMA_EB.py") generado a partir de GNU Radio.

En la aplicación Python se dan las órdenes a los robots, que son pasadas a GNU Radio y enviadas a través del SDR. La comunicación con GNU Radio se realiza a través de sockets, y se usa tanto para mensajes internos como para enviar

las órdenes y recibir las posiciones. También la medida de calidad de servicio se realiza en la aplicación, a la que GNU Radio le pasa los resultados del envío de paquetes y los reintentos. Para la toma de decisiones sobre el posicionamiento de cada robot, la estación base le solicita a los robots información sobre su posición actual. Decidido el accionar que debe seguir cada robot (si quedarse quieto, ir a una posición, etc) la estación base envía la orden correspondiente.

El robot Explorador no tiene ese nombre en vano. Está construido sobre el chasis más robusto con que contamos, que soporta:

- un Raspberry Pi 3 modelo B
- un Arduino Uno
- la alimentación del sistema (pilas y batería portátil)
- los sensores
- un USRP B200 mini y sus antenas respectivas

El Raspberry Pi cumple el rol de supervisor del sistema: incluye el inicio automático, los loops de control, y corre el diagrama de flujo de la comunicación. Realiza el procesamiento del intercambio con los otros nodos a través de GNU Radio. Mediante la comunicación serial le envía a Arduino la posición a la que debe dirigirse y recibe la posición actual registrada por el GPS del robot la que posteriormente es enviada a la estación base.

Arduino controla los sensores y obtiene información de estos, además de controlar los motores. Es decir que la elección y el recorrido del camino hasta el destino recibido son tareas que cumple Arduino. Usa para orientarse un compás, para medir el avance los encoders, para avanzar los motores, para ubicarse un GPS, y para evitar obstáculos los sensores infrarrojos.

La alimentación del sistema está compuesta por pilas (para los motores) y una batería portátil de la que dependen el Raspberry PI, el USRP B200 mini, Arduino y los sensores.

El USRP es controlado por el programa de GNU Radio que corre en el Raspberry Pi, el cual cuenta con la CSMA/CA, la capa física y la comunicación por sockets.

El Repetidor cuenta con hardware similar al del Explorador. En el diagrama de flujo de GNU Radio está habilitado un modo para que los mensajes que no están dirigidos hacia él sean retransmitidos, que llamamos función de repetidor. Al mismo tiempo, el chasis y los motores son más débiles, y cambia la alimentación, aunque bajo la misma premisa de combinar pilas y batería portátil.

La tabla 2.1 presenta un listado de las tareas y sistemas operativos de cada actor.

Tabla 2.1: Funciones a realizar para cada agente.

Características	**Estación Base**	**Robot Repetidor**	**Robot Explorador**
Sistema Operativo	Ubuntu 16	Arch Linux	Arch Linux
Tareas a desarrollar	Solicitar ubicacion de los robots.	Enviar ubicación.	Enviar ubicación.
	Enviar ubicación destino al Explorador.	Orientarse y avanzar hacia la ubicación destino.	Orientarse y avanzar hacia la ubicación destino.
	Estimar la QoS.	Función de repetidor.	Avisar en caso de llegada a destino.
	Calcular y enviar ubicación destino al Repetidor.		

La imagen 2.2 detalla los procesos que se ejecutan en la estación base y en los robots (en éste último caso diferenciando entre Raspberry Pi y Arduino). Además se ilustran las interfaces, tanto entre las distintas placas mediante conectores USB como a la interna de cada sistema anfitrión mediante sockets.

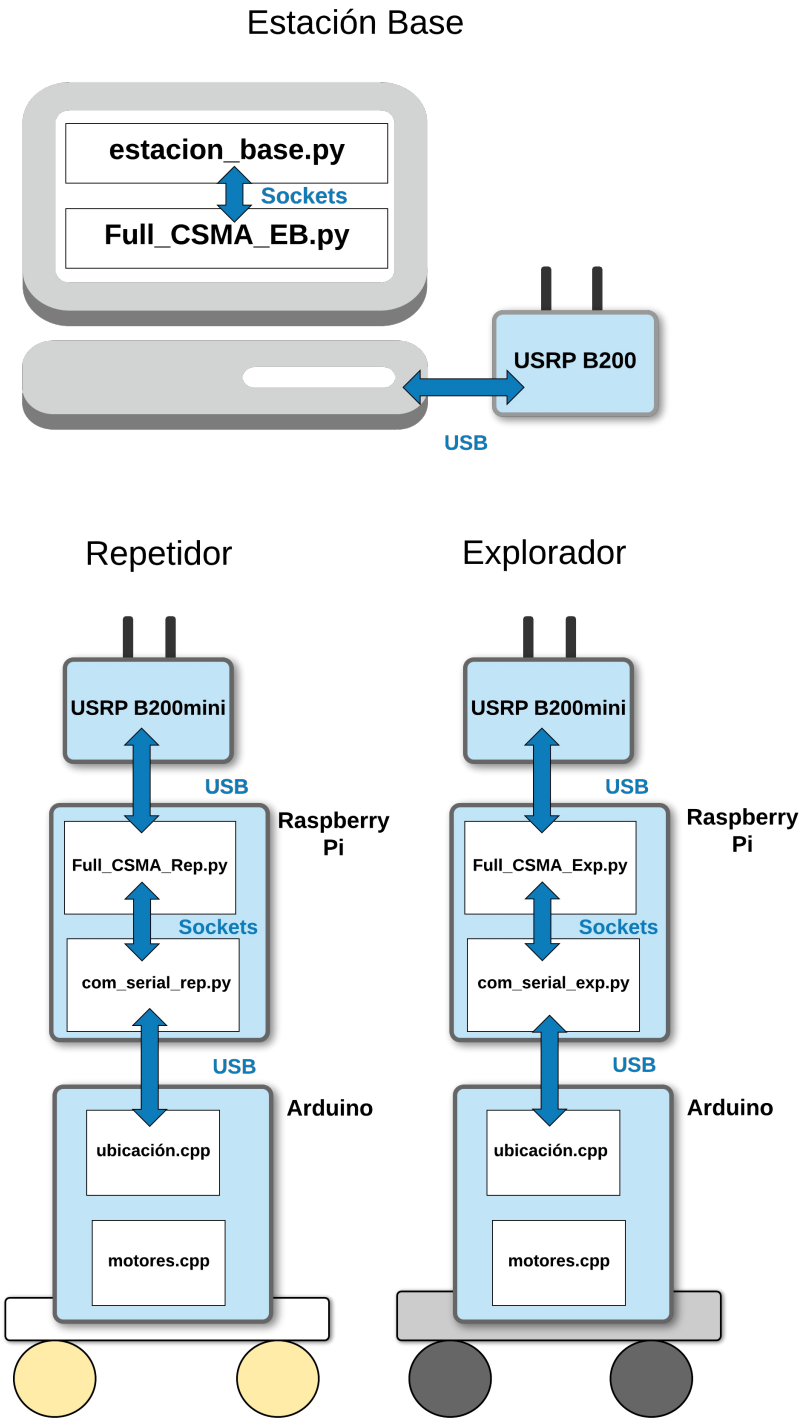


Figura 2.2: Esquema de los tres actores del sistema, detallando los procesos que corren en cada nodo y las interfaces

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Sistema de comunicación

3.1. SDR

El importante desarrollo que han tenido las tecnologías de la comunicación vivió en carne propia los impactos del impulso de la era digital. En particular, el tratamiento digital de las señales propone una amplia variedad de posibilidades para su procesamiento [15], limitadas por la capacidad de cómputo del hardware en que corre el software diseñado y la creatividad de los programadores. A partir de esta característica es que se ha planteado el interés en desarrollar hardware de propósito general, cuyas tareas específicas sean establecidas por software. Esta idea se enfrenta al paradigma de tener un hardware específico para cada tarea específica, que aún sigue siendo la opción más difundida (radio, router, celular).

Es en este marco que recientemente se han explorado mecanismos para la definición de redes por software (SDN) [87], la virtualización de funciones de red (NFV) [87], los centros de datos definidos por software (SDDC) [56], y para comunicaciones inalámbricas la radio definida por software (SDR). El objetivo general de estas propuestas es justamente pasar a software parte de las funciones típicamente realizadas en hardware [15]. Esto permite mayor flexibilidad, ya que con el mismo equipo uno puede desarrollar un transmisor de FM [17], un receptor de televisión digital [42], o implementar algunos protocolos sencillos de comunicaciones inalámbricas similares a los empleados en el protocolo 802.11 [57] (como es nuestro caso). Incluso pensando en el futuro de las comunicaciones inalámbricas, se pone en valor la utilidad de tecnologías como SDR para el desarrollo de -por ejemplo- redes inalámbricas cognitivas [52] [28].

De esta forma se logra además disminuir los problemas de interfaces entre hardware especializado de diferentes orígenes, lo que lleva a reducciones importantes en costos por disminución de recursos dedicados a lograr compatibilidades, o por evitarse la compra de equipos para fines muy específicos. Se escapa entonces en cierto modo de la situación de rehén de compañías que pueden discontinuar la producción de componentes o el soporte de ciertos productos, principalmente si se

trata de software y hardware libre. Se evitan además los problemas asociados a la falta de estándares cuando cada compañía maneja lenguajes propios [87].

Al mismo tiempo, el poder reutilizar un equipo simplemente cambiando el software puede implicar grandes ahorros. Este es un aspecto no menor, ya que es cada vez más reducida la posibilidad de realizar grandes inversiones para equipos muy especializados sin retornos rápidos, lo que queda reservado para pocas multinacionales.

Del lado de las empresas proveedoras de tecnología, pueden concentrar esfuerzos en producir equipos de propósito general con mayores capacidades, aprovechando las crecientes velocidades de cómputo y el bajo costo de componentes generales de alta precisión [87], dejando de lado el costoso desarrollo de equipos especializados con poca (aunque existente) demanda.

Del lado del usuario, problemas de compatibilidad y un costo excesivo para equipos con un uso reducido plantean las alternativas mencionadas como posibilidades interesantes. Es que la programación sobre equipos de propósito general abre muchas oportunidades para las que no es necesario contar con tanta infraestructura ni inversión. De hecho, muchas de las grandes Telcos han optado por mantener un pie en cada lado de la calle, si bien continúan desarrollando sus equipos específicos, comienzan a transitar por el cambio de paradigma [30] [31].

Es también el avance de la capacidad de cómputo para la constitución del hardware de propósito general el que ha permitido acercar muchísimo la velocidad de las funciones realizadas digitalmente a los tiempos (igual siempre menores) del mundo analógico. Asimismo, la digitalización también introduce ventajas a varios niveles: además de la flexibilidad mencionada, la distinción binaria entre niveles de tensión hace más robusto frente al ruido al mundo digital que al mundo analógico [15]. Por otro lado, el impulso que ha tenido la computación en la nube y el desarrollo del software colaborativo han sido grandes propulsores de estos cambios de modelo de las tecnologías, y a su vez han forzado a las grandes empresas de las telecomunicaciones a migrar hacia este sentido [31] [29].

En el caso de SDR, se llama de igual manera al paradigma mencionado que al hardware que se encarga del muestreo y transmisión (o recepción y muestreo). Como se puede apreciar en la figura 3.1, este consta de (al menos) una FPGA, la conversión AD y DA, el sistema de pasaje a banda base de las muestras recibidas, la etapa de RF (con amplificadores y antenas), y en el otro extremo, la conexión a la PC donde se generan (o reciben) las muestras.

De manera simplificada, en recepción las antenas capturan a la frecuencia elegida muestras que le pasan a la PC, donde corre el software que las procesa para obtener el resultado deseado. En transmisión se da el camino inverso: la PC entrega muestras procesadas a la FPGA, que luego son emitidas por las antenas a la

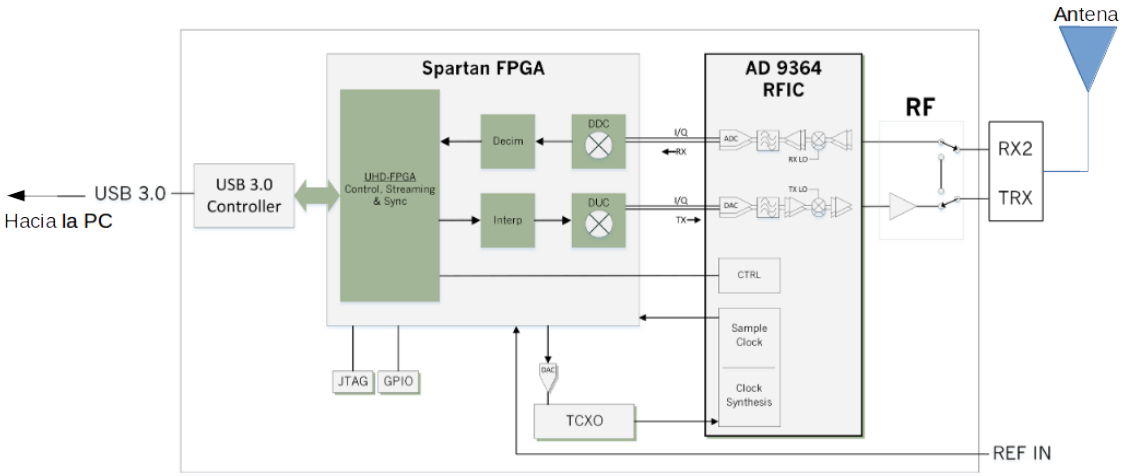


Figura 3.1: Esquema de un USRP, con su conexión al computador que funciona como anfitrión (izquierda) y a las antenas (derecha). Figura tomada de la hoja de datos del USRP B200 mini.

frecuencia y tasa de muestreo deseadas.

El objetivo de las SDR es empujar al mundo analógico lo más posible hacia las antenas, llevando al plano digital la mayor parte de las funciones posibles. Como se ve en la figura 3.2, la mayor parte de las tareas se realizan ya sea en la FPGA (programable, parte del hardware SDR), o en la computadora de propósito general a la que se conecta el SDR.

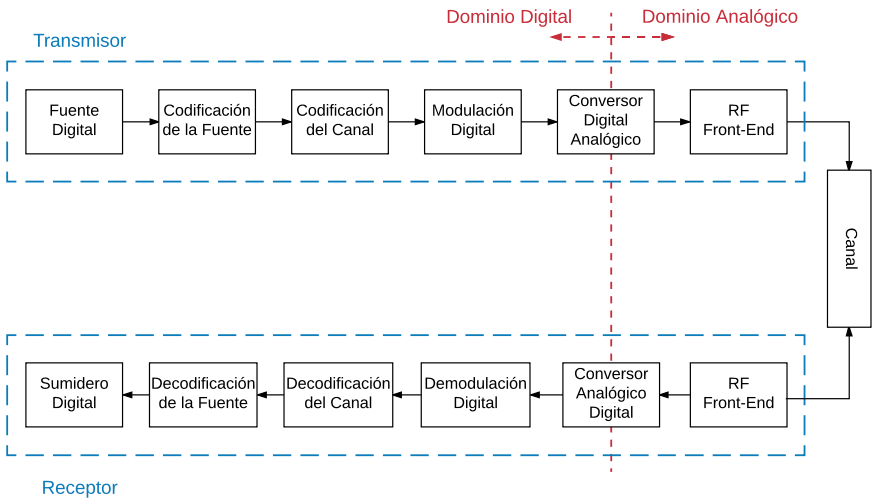


Figura 3.2: Camino desde un transmisor a un receptor, con la división del procesamiento analógico y digital

En Uruguay hace ya varios años que el grupo ARTES del IIE está trabajando

con SDR con fines educativos, académicos, exploratorios. Es una línea de trabajo presentada en el Proyecto Grupos – CSIC [8], donde se plantea como objetivo específico (entre otros): *"Desarrollar un framework y bibliotecas públicas y de uso libre que se integren con equipos SDR y software de capa física como GNU Radio y que permitan implementar una red inalámbrica de futura generación. Este objetivo busca contar con hardware y software libre que permita fundamentalmente a países que actualmente no son productores de tecnología inalámbrica poder hacer enseñanza, investigación y prototipos industriales con esta tecnología."*. Dentro del Instituto de Ingeniería Eléctrica se destacan particularmente los trabajos vinculados a la realización de un receptor de televisión digital en SDR que fue Tesis de Maestría de Pablo Flores Guridi [35], así como el desarrollo de cursos vinculados al uso de SDR y GNU Radio (Comunicaciones Inalámbricas, Taller de Introducción a las Telecomunicaciones) o de proyectos de fin de carrera [20].

En particular, el marco de trabajo para software utilizado para el desarrollo sobre SDR en el IIE es GNU Radio, que es a su vez el más usado en el ambiente.

3.2. GNU Radio

GNU Radio es una herramienta para trabajar con SDR y para realizar simulaciones. Es muy popular en el mundo de SDR por su gran variedad de librerías. Está bajo licencia GPL (GNU General Public License (GPL), versión 3), y los derechos del código son de la Free Software Foundation [69].

La utilización de software libre para el desarrollo de las librerías y entornos de trabajo no es ya un capricho ideológico. Aunque bien pudiera haber una inclinación al desarrollo colaborativo por motivos filosóficos, el software colaborativo es hoy una realidad, y en particular para los países dependientes, que no cuentan con los medios para generar sus propias infraestructuras de trabajo y no desean ser presa de un monopolio rígido y probablemente insuficiente en su oferta. La colaboración abierta permite no solo el acceso gratuito a materiales (y a tecnología de punta), sino justamente la complementación en los esfuerzos. Se suman a esto ciertas ventajas intrínsecas asociadas a su carácter abierto, en particular la robustez ante bugs (errores) y las dificultades para que el software tenga puertas traseras u otros elementos asociados a seguridad.

En este sentido, podemos ver que no es sólo un impulso alternativo o tercermundista la apuesta al software libre, sino que las grandes multinacionales empiezan de a poco a abrazarse a este paradigma [31]. Para un equipo de investigadores o profesionales de un país como el Uruguay, presenta ventajas y oportunidades como ser: la participación en proyectos más amplios, para los cuales no se contaría con infraestructura o posibilidades de otra manera; la investigación y el desarrollo sobre tecnologías de punta; el intercambio con colegas y gente del entorno, para pruebas, motivos académicos, o la colaboración activa en la resolución de pro-

blemas mundiales de primer nivel. Para las grandes empresas claro que también hay ventajas: en lugar de tener un equipo gigante de programadores, uno puede fácilmente tercerizar funciones a grupos más pequeños y distribuidos en el mundo, contando ahora con mayor oferta y permitiendo vínculos rápidamente cambiantes. Esto traslada parte del riesgo de investigacion y desarrollo (un riesgo grande) de la empresa matriz a las subsidiarias, y permite el aprovechamiento de recursos entre empresas, algo que antes no era muy común pero hoy es necesario (entre otras cosas, para no afrontar caminos en solitario, que pueden desembocar en inversiones sin retorno).

Tal como lo planteaba el grupo ARTES en su propuesta a Grupos – CSIC [1], para los países dependientes SDR y GNU Radio presentan la oportunidad de poder probar, simular, explorar e implementar con menores costos. Las ventajas del software colaborativo y abierto pueden hasta cierto punto combatir las falencias de los países pequeños, como el nuestro, donde la ausencia de industria del hardware y la falta de infraestructura puede representar un obstáculo para el desarrollo de conocimiento, aplicaciones, pruebas, etc.

Establecidas algunas ventajas del software libre, y sin entrar en aspectos profundamente filosóficos como ser el cuestionamiento a la apropiación del conocimiento humano por particulares, intentaremos ahora presentar el funcionamiento básico de GNU Radio.

Uno puede imaginarse el procesamiento que realiza GNU Radio como un camino que transitan los datos. Este camino se recorre en el dispositivo anfitrión ("host"), una computadora en nuestro caso, cuyas muestras resultantes son enviadas al SDR para su transmisión por aire. GNU Radio funciona con bloques interconectados, que podrían pensarse como escalas en el camino de origen-destino. Cada bloque cuenta con una cierta cantidad de entradas y salidas, y tiene (al menos) una función concreta. Un scheduler interno de GNU Radio va recorriendo

[1] *"Es importante resaltar el impacto (particularmente para países como el nuestro) que tiene para la enseñanza, la investigación y el diseño industrial, la existencia de hardware y software libre que permita desarrollar equipamiento inalámbrico. En cuanto a la investigación, hasta ahora quienes trabajan en el área de redes, particularmente inalámbricas, desarrollando un nuevo algoritmo o protocolo, se ven obligados a probarlo en un simulador. Salvo contadas excepciones, no es posible probarlo en un equipo real ya que el hardware y software de los equipos inalámbricos es propietario y no modificable por el usuario. En países productores de estos equipos habitualmente las universidades tienen acuerdos con fabricantes y en algunos casos pueden hacerlo, pero ciertamente no en los países de nuestra región. Las posibilidades se limitan a probarlo en un simulador y publicar los resultados en algún artículo pero nunca se pasa a la fase de desarrollo. Esto es un límite evidente para el desarrollo industrial en torno a estos temas en la región. El mismo problema existe para enseñar sobre redes inalámbricas. Lo que es posible hacer hoy es explicar la teoría y mostrar como caja negra lo que hacen los equipos propietarios de los fabricantes o usar un simulador. La existencia de hardware y software libre abre las puertas a implementar equipos operativos y ver cómo funcionan, modificando las partes que se deseen."* [8]

este "flow graph" (diagrama de flujo), siguiendo el recorrido que realizan las muestras, indicando a cada bloque su turno de actuar y modificándose las muestras en consecuencia, a través de la utilización de memoria compartida. El flow graph (los bloques a usar y el camino a recorrer) se define en un archivo Python (por defecto llamado **top_block**). Si bien en general se trata de un flujo continuo de muestras, veremos que nuestro caso se escapa de esta norma.

En particular GNU Radio permite la creación de bloques propios. Estos se programan en C++ o Python, y si bien una gran parte de las funcionalidades esperadas ya se encuentra disponible en GNU Radio y sus extensiones, siempre se puede ajustar o crear un bloque propio, a gusto del consumidor. Acceder al código fuente es útil al momento de entender mejor lo que hace un bloque.

Existe un entorno gráfico, llamado GNU Radio Companion, que no es más que una interfaz amigable para poder generar los diagramas de flujo. En el mismo, alcanza con arrastrar los bloques e interconectarlos, seteando los parámetros deseados. Luego el GNU Radio Companion genera el **top_block** que corre el programa entero. Correr el flow graph es ejecutar el archivo Python, lo que puede hacerse desde terminal (como se ejecuta cualquier script de Python): **$ python top_block.py**. Si bien estos flow graphs se pueden codificar desde cero, definiendo los bloques, luego sus conexiones, y finalmente la ejecución de las tareas, es mucho más sencillo armarlo en la interfaz de GNU Radio Companion, visualizando todos los parámetros y bloques utilizados, ahorrando también tiempo de programación. También se puede ejecutar el programa desde la propia interfaz, que incluye bloques de depuración y gráficos para visualizar en tiempo real el espectro de la señal o la constelación, entre otras posibilidades.

De querer instalar GNU Radio y GNU Radio Companion (así como GWN), recomendamos seguir las instrucciones detalladas en el Anexo "C".

La posibilidad de generar bloques propios fue utilizada por el grupo ARTES para el desarrollo de un conjunto de herramientas para el trabajo con comunicaciones inalámbricas orientadas a mensajes, permitiendo un entorno más adecuado para el trabajo sobre redes de datos. Estas librerías (GWN) están muy conectadas con nuestro trabajo en el proyecto, que podría pensarse como un caso de uso de las mismas, ya sea utilizando bloques allí implementados o creando los nuestros bajo ese concepto, contribuyendo al desarrollo de GWN, y por extensión, de GNU Radio.

3.3. GNU Radio Wireless Network (GWN)

La versatilidad de SDR y en particular de GNU Radio se había encontrado con un obstáculo en su aplicación a redes de datos. Este es que, a diferencia de otras comunicaciones inalámbricas, las redes de datos requieren trabajar con eventos

finitos, y no con un flujo continuo de datos. En el esquema tradicional de GNU Radio, el flujo continuo de datos (bytes, complejos, enteros) era procesado con funciones particulares en cada bloque (filtros, operadores matemáticos, corrección de errores, instrumentos gráficos). En el caso de las redes de datos, es necesario estar orientado a mensajes o eventos, finitos.

Es por esto que desde hace algunos años, el grupo ARTES del IIE ha venido trabajando en el desarrollo de un marco de trabajo que posibilite la experimentación con mensajes, paquetes, tramas, es decir, datos delimitados. Al mismo tiempo, para trabajar con redes de datos, son necesarias algunas funcionalidades en los bloques como ser el manejo de tiempos (timeouts), la orientación a mensajes, el manejo de eventos. Es por esto que surge GNU Radio Wireless Networks, o por sus siglas GWN, que propone un conjunto de herramientas para el trabajo de experimentación y diseño de redes de datos sobre SDR [34].

Se propone en GWN una clase de la cual heredan los bloques a crear, que permite no sólo lo ya mencionado, sino que facilita la implementación en máquina de estados (FSM) de las funciones deseadas. Esto quiere decir que cualquier bloque nuevo, creado bajo la clase GWN, además de ser compatible con los bloques generales de GNU Radio, tendrá características particulares orientadas hacia las redes de datos.

Para la comunicación, GWN utiliza mensajes, que ya se habían implementado en GNU Radio con dos propósitos: la comunicación aguas arriba entre bloques, el intercambio con aplicaciones externas (como ser a través de sockets), y la discretización deseada del flujo. Los mensajes que entran y salen de un bloque de GWN son del tipo PMT (Polymorphic Data Type) [68], un tipo genérico de datos que se utiliza en GNU Radio para el intercambio de mensajes y flujos etiquetados. Un mensaje PMT es un contenedor opaco de datos que puede incluir una gran variedad de tipos de datos a ser utilizados en la topología de GNU Radio. Son muy usados en intercambios de mensajes entre interfaces y en flujos etiquetados, y pueden representar (entre otros): booleanos (verdadero/falso); strings (cadenas de caracteres); números enteros, flotantes o complejos; pares, tuplas o diccionarios [70]. En GWN se utiliza la flexibilidad que brinda PMT para acercarse a una realidad más similar a las redes de datos.

El modelo de un bloque de la clase GWN es el que se aprecia en la figura 3.3. Las entradas son mensajes PMT que se traducen para su tratamiento interno al tipo event de GWN. El procesamiento se realiza en la función process_data, con el manejo de eventos internos (como los temporizadores) y de la máquina de estados (de existir). A la salida se convierten los eventos nuevamente a mensajes PMT. El manejo de eventos permite la realimentación de los bloques, yendo aguas arriba en el diagrama de flujo (que con el tradicional flujo de complejos de GNU Radio no se podía), y al mismo tiempo abre la puerta a explotar las facilidades de la programación en máquinas de estados finitas.

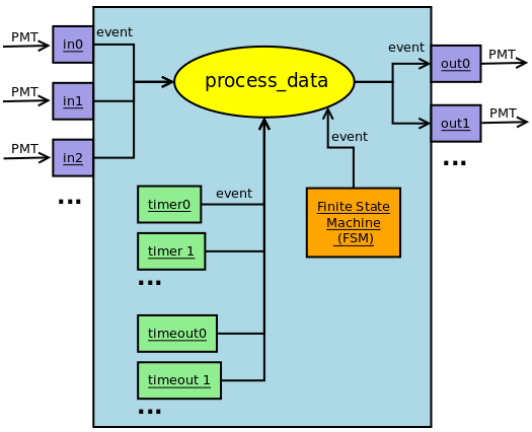


Figura 3.3: Representación de un bloque perteneciente a la clase GWN. Nótese las conversiones de PMT a eventos en las entradas y salidas. Cortesía del Grupo ARTES.

Para una mejor aproximación al modelo de capas se crean mensajes encapsulados que se definen como eventos (de la clase GWN Event), que será nuestra unidad de comunicación. La conversión de PMT a Eventos es invisible para los programadores (nosotros en este caso), por lo que podemos considerar que ingresan y egresan eventos a nuestros bloques. Los eventos incluyen un nombre (nickname) y un tipo (por ejemplo si son referidos a tiempos, a contenido, a mensajes de control). Los eventos pueden usarse tanto para transmisión de datos, mensajes de control o administrativos, y pueden ser internos al bloque. Por ejemplo los temporizadores, al expirar, generan un evento del tipo Timer, de la clase EventTimer, y con nombres EventTimer, TimerACKTout, etc. Además, al tener un diccionario asociado permiten agregar particularidades, pudiendo acercarse más al concepto deseado para trabajar en cada capa o protocolo a implementar.

GWN incluye también el manejo asíncrono de tiempos. Esto es necesario para los protocolos de redes de datos inalámbricos: ya sea para retransmitir un paquete perdido como para implementar mecanismos de acceso a medios compartidos. En este sentido existen dos modalidades: la utilización de una cantidad determinada de intervalos temporales, donde expirado cada intervalo se genera un evento para manejo interno; la generación de un temporizador que, al expirar, genera un único evento interno. El manejo de tiempos a través de eventos permite además su integración sencilla con las máquinas de estados.

La utilización de máquinas de estados para la implementación de protocolos de comunicaciones es muy común. Alcanza con ver los pseudocódigos utilizados en el libro Redes de computadoras de Tanenbaum (libro de referencia de los cursos de redes de la FING) para ver lo intuitivo que resulta pensar protocolos en torno a máquinas de estados. En nuestro caso, GWN permite implementar una máquina de estados finita extendida (extended FSM). Explicaremos más adelante su funcionamiento, y daremos un ejemplo de implementación. De manera resumida, la

idea es que según el evento que llega a mi bloque, y dependiendo del estado en que me encuentro, de verificarse una condición dada ejecutaré una cierta acción y transición a un nuevo estado.

Si bien GWN representa un marco de trabajo, también contiene varios bloques ya implementados, ya sea para ejemplificar o utilizar concretamente en aplicaciones. Nosotros usamos varios de estos bloques, como comentaremos posteriormente, a la vez que modificamos y creamos bajo el modelo de GWN otros que permiten la implementación concreta de una CSMA/CA sobre un protocolo sencillo de ARQ de tipo Stop And Wait.

La creación de un bloque bajo GWN incluye además dos códigos que se generan automáticamente. Uno, en lenguaje XML, será el encargado de construir el bloque para el GNU Radio Companion, es decir la interfaz de trabajo de GNU Radio. En éste se deben detallar los parámetros que el bloque contiene, así como su tipo de dato y valor por defecto. El otro código que se genera es lo que se llama un test de QA (aseguramiento de la calidad). El objetivo es que en este código se establezca un diagrama de flujo para realizar las pruebas que permitan asegurar el funcionamiento del bloque, detectando bugs, errores o descuidos en la lógica del bloque. Al mismo tiempo se actualizan los códigos necesarios para, ejecutando CMAKE se carguen los bloques en GNU Radio y GNU Radio Companion. Más información sobre la instalación de GWN y sobre estos bloques se encuentra en el Apéndice “C.1”, así como en `https://github.com/vagonbar/gr-gwn/wiki`.

Las posibilidades que abre GWN son muchas, tanto para el ámbito académico como para aplicaciones finales. Para quienes estén interesados en instalar, utilizar, y si lo desean, contribuir a las librerías, se recomienda visitar el repositorio `https://github.com/vagonbar/gr-gwn`. En el mismo se encuentra una guía para la instalación de GWN, y tutoriales sobre la construcción de un bloque de GWN, el manejo de tiempos y el trabajo sobre máquinas de estados finitas.

Uno de los objetivos específicos de este proyecto fue justamente la implementación de la mencionada CSMA/CA, que permite tanto su utilización como producto final (en un chat, o en un proyecto como el nuestro, con varios nodos comunicándose sobre un medio compartido) así como una forma de probar la completitud de GWN como marco de trabajo, y su fácil utilización por parte de estudiantes de grado.

3.4. Explicación de los protocolos en juego: CSMA y ARQ

En el modelo OSI de capas, cada capa busca ser independiente del resto (una abstracción), y tiene funciones concretas a cumplir hacia las capas superiores e

inferiores. En el caso de la capa de enlace (capa 2), en un servicio confiable, su función principal consiste en: entregar en orden el contenido de cada trama a las capas superiores (capa de red), confirmando la llegada de las tramas al emisor; y hacia la capa inferior (capa física) utilizar mecanismos que permitan mejorar la utilización del medio físico, buscando garantizar el uso eficiente del canal. La primera subcapa es llamada de control de enlace lógico, mientras la segunda se denomina de control de acceso al medio.

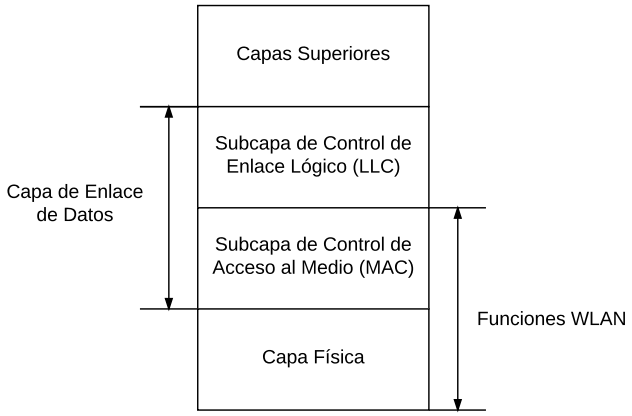


Figura 3.4: La capa de Enlace: subcapa de Control de Enlace Lógico (Logical Link Layer - LLC) y subcapa de Cotrol de Acceso al Medio (Medium Access Control - MAC)

Llamaremos protocolo a un conjunto ordenado de reglas y acciones a seguir para diferentes casos, con una lógica general orientada al cumplimiento de por lo menos un objetivo. Por ejemplo, la espera de 15 minutos por parte de la hinchada de un cuadro de fútbol (en general del cuadro menos violento) para la salida del Estadio, y así evitar el cruce entre hinchadas, podría considerarse un protocolo: de haber posibilidad de violencia, se busca evitar los encuentros que puedan dar lugar a la misma. Existen, y más en ingeniería de redes, protocolos con mayor sentido, y sobre todo mejor eficacia para el cumplimiento de los objetivos propuestos.

La gran cantidad de artefactos que cuentan con conectividad inalámbrica (un número aún en crecimiento) genera la necesidad de desarrollar protocolos sencillos, confiables, flexibles y aplicables a diferentes comunicaciones y escenarios, principalmente en la subcapa de control de acceso al medio (MAC) [88] [8].

Las soluciones para esto se realizan en base a 5 supuestos (como lo plantea Tanenbaum en el libro Redes de Computadoras [85]):

- Modelo de estación. El modelo considera N estaciones independientes que generan y reciben tramas.
- Supuesto de canal único. Todas las estaciones comparten un único canal.

- Supuesto de colisión. Cuando dos tramas se transmiten en tiempos muy cercanos, colisionan y se pierde la información que llevan. Por lo tanto se deben retransmitir.
- Tiempo continuo / ranurado. En nuestro caso se consideró el tiempo como continuo, ya que en cualquier momento se puede transmitir una trama (aunque los temporizadores se generan a partir de una unidad temporal). En el caso de tiempo ranurado, se establecen períodos (ranuras) de tiempo, y únicamente al inicio de cada ranura se transmiten tramas.
- Detección de portadora / sin detección de portadora. Las estaciones pueden o no detectar la ocupación del canal. En nuestro caso utilizamos justamente la detección de portadora, es lo que llamamos "sensado del medio físico".

Los medios inalámbricos son medios compartidos que presentan problemas particulares, tales como el de la estación oculta y el de la estación expuesta. El problema de estación oculta se da cuando un nodo C queriendo hablar con un nodo B realiza un sensado del medio y no detecta un tercer nodo A comunicándose con B (caso a de la figura 3.5). Por otra parte, el nodo C podría detectar una señal como ocupando el canal, y elegir entonces no transmitirle a D, aunque tal vez no interferiría con la comunicación entre A y B (caso b de la figura 3.5), esto es el problema de la estación expuesta. Para resolver estos problemas y establecer pautas de acceso a medios compartidos se crearon diversos protocolos, entre ellos el CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).

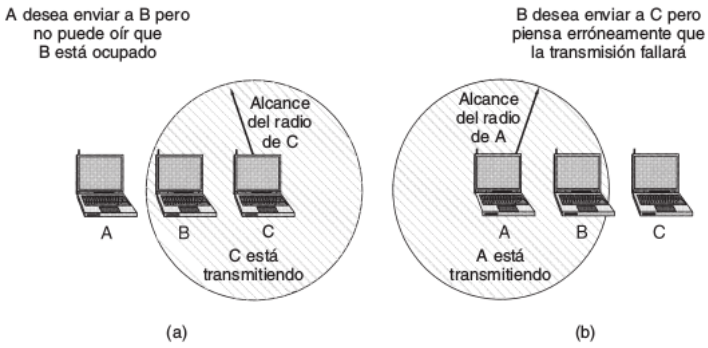


Figura 3.5: a) El problema de la estación expuesta b) El problema de la estación oculta. [85]

El protocolo CSMA/CA pertenece a la familia de protocolos CSMA (Carrier Sense Multiple Access), y fue definido por Kleinrock y Tobagi [41]. Pertenecen a la mencionada capa de enlace de datos, en particular a la subcapa MAC. Como fue mencionado, en esta subcapa se busca organizar la utilización del medio, es decir del canal físico a través del que nos comunicaremos.

Estos protocolos se usan por ejemplo en las redes WLAN (redes inalámbricas de área local), que son las que usamos en general bajo el nombre popular de "wifi".

La versión de CSMA/CA que se utiliza en este caso está definida en el estándar IEEE 802.11 [83]. El problema de estación oculta y expuesta se soluciona en las versiones de CSMA/CA que cuentan con el mecanismo de RTS/CTS (Request To Send/Clear To Send). Nuestra implementación no incluye este mecanismo, por lo que nuestra versión de la CSMA/CA se plantea más como un mecanismo de distribución del medio físico y prevención de colisiones que resolviendo los problemas de estación oculta y expuesta.

El objetivo de la CSMA es disminuir los choques entre paquetes (colisiones), lo que empeora la comunicación. Para esto se realiza un sensado del medio, es decir se chequea si el canal (inalámbrico en nuestro caso) está libre u ocupado. Si alguien está hablando se detecta que el medio está ocupado. Esto permite evitar intentos inútiles (y más bien disruptivos) de establecer comunicación. Es una manera de distribuir un recurso compartido, como es el medio físico en el caso de las comunicaciones inalámbricas, intentando optimizar su uso.

De estar ocupado el medio compartido, se sortea un tiempo aleatorio dentro de un rango que se va incrementando con cada reintento de establecer comunicación. Esto es porque no se quiere que todos los que están sensando el medio a la vez choquen en cada reintento. Esta función se explicará con mayor detalle luego, pero se ejemplificará a continuación.

Un caso de utilidad de una CSMA/CA podría ser en una conferencia de prensa. A uno de los periodistas se le dará la palabra para preguntar, tiempo durante el cual el resto quedará esperando. Al terminar la respuesta de, por ejemplo, el Maestro Tabárez, todos los periodistas querrán preguntar a la vez, hablando uno sobre el otro e impidiendo la realización de una pregunta clara. De contar con una CSMA/CA, y sortearse quien levanta la mano primero, es altamente probable que sólo uno de los periodistas lo haga, logrando la atención del Maestro. Para el resto de los reporteros, estará entonces ocupado Tabárez, y tendrán que volver a esperar.

El sorteo aleatorio para la asignación del uso de un bien compartido encierra un concepto maravilloso, o por lo menos asombroso: lo aleatorio, y por ende incierto, funciona bien al momento de impartir justicia, y es solución a problemas concretos de la vida real. Por más asombroso que parezca, esto no es un concepto nuevo, bien lo sabían ya los antiguos griegos, que habían desarrollado un sistema de asignación aleatoria a sus ciudadanos para servir en los cargos públicos, e incluso habían diseñado una máquina específica para ello: el kleroterion [97].

Existen diversas versiones del protocolo CSMA/CA, con detección del canal físico o del canal virtual (esta última utiliza mensajes de Request to Send y Clear to Send). En este proyecto se utilizó la primera versión, descartando la utilización de paquetes para solicitar permiso y otorgar permiso (RTS/CTS). Estos mensajes de control, para nuestra implementación sencilla de una CSMA/CA en GWN, no fueron considerados. En particular, porque el encabezado que usaremos de los

paquetes es muy grande y enviar cada trama ocupa durante bastante tiempo el canal, por lo que estos mensajes generarían más una obstrucción del canal que una garantía.

El protocolo CSMA/CA se puede describir de la siguiente forma:

Carrier Sense: Previo a la transmisión, la estación sensa el canal para determinar si otro nodo está transmitiendo.

Collision Avoidance:

- Si el canal está ocupado, se sortea un slot de tiempo dentro de un conjunto de slots posibles, antes de volver a sensar el medio. En el siguiente punto se detalla cómo se determina el conjunto de slots posibles. Cada slot tiene una longitud igual al tiempo de propagación de ida y vuelta de una trama [85], que para 802.11b es de $20\mu s$ [42].
- Si el canal está libre, se transmite la trama y se espera el acuse de recibo (ACK). Si el ACK no llega en tiempo y forma, se asume que la trama colisionó y se aplica el retroceso exponencial binario, esto es, se duplica la cantidad de slots posibles antes de volver a sensar el medio. Inicialmente se espera un número aleatorio de entre 0 y $W_{min} - 1$ slots, luego de una colisión se sortea entre 0 y $2W_{min} - 1$ slots, luego de una segunda colisión se sortea entre 0 y $4W_{min} - 1$ slots, y así sucesivamente hasta un valor $W_{max} = 2^m W_{min}$, siendo reseteada la cantidad de slots al llegar un ACK correctamente. Los valores de W_{min} y W_{max} para 802.11b son de 31 y 1023 respectivamente, permitiendo un máximo de 5 reintentos [42].

El funcionamiento del protocolo se puede ver en la figura 3.6.

La CSMA/CA trabaja sobre un protocolo Stop and Wait sencillo, que explicaremos a continuación.

Además de la CSMA/CA que busca optimizar el acceso al medio, son necesarios otros protocolos, también en la capa de enlace de datos, que trabajan sobre la lógica de envío de datos, procurando en particular que los datos lleguen y en orden al remitente. Para esto se desarrollaron los protocolos llamados ARQ (Automatic Repeat reQuest), que pertenecen a la subcapa de control de enlace lógico, mencionada al inicio del capítulo. Si bien estos protocolos de control de errores pueden ser utilizados en otras capas, como capa de transporte, nos centraremos en su utilización a bajo nivel.

La forma de garantizar la entrega de un dato es a través de realimentación, es decir, de información sobre el estado de la trama enviada. Lo que necesitamos entonces es una confirmación de la recepción del mensaje por parte del destinatario. El mensaje que se envía con la confirmación de la correcta recepción del mensaje se denomina acuse de recibo (ACK). Un mensaje que no obtiene su correspondiente

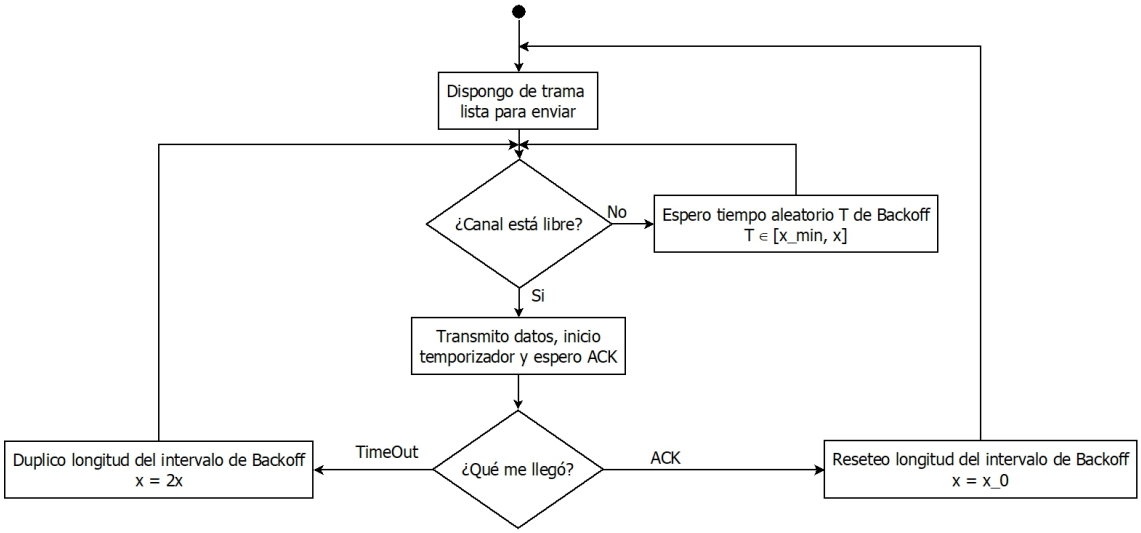


Figura 3.6: Diagrama simplificado del protocolo CSMA/CA

ACK se considera perdido luego de cierto tiempo, así como uno que recibe su ACK se considera transmitido con éxito. Esto se profundizará en el estudio del bloque ACK_RX, que es el encargado de esta tarea.

Algunos protocolos utilizan el concepto de ventana deslizante para poder mandar varios datos a la vez. Para esto se requiere de mayor inteligencia y memoria para un buffer en el receptor (para el ordenamiento de las tramas previo a su entrega a las capas superiores, y/o para el envío de Not ACKnowledged cuando el receptor detecta de que le falta una trama). Estos protocolos mejoran la tasa de transmisión de datos, ya que puedo retransmitir únicamente los datos cuyos ACK no llegaron, y no freno el envío de todos los datos esperando por cada ACK. En nuestro caso, se optó por la implementación de la CSMA/CA sobre el protocolo Stop and Wait, aunque también se generaron en GWN usando FSM los bloques para protocolos con ventanas deslizantes (Go Back N, Selective Repeat).

En el caso de un Stop and Wait tradicional, se considera que la capacidad de la capa de red receptora es menor a la del emisor, lo que es el motivo de la pérdida de tramas [85]. El caso del Stop and Wait es el más sencillo de los protocolos de la familia ARQ, ya que no contamos con ventanas deslizantes, y la realimentación es binaria: o llega un acuse de recibo (la trama llegó bien) o pasado un cierto tiempo asumo que ésta no llegó. Es decir que no envío nunca más de una trama, y detengo los envíos hasta que no llegue la confirmación de la trama enviada. Su funcionamiento se describe en la figura 3.7.

Tanto el diagrama presentado como el desarrollo del protocolo Stop and Wait en máquina de estados bajo GWN fue realizado por Víctor González Barbone. Nosotros tomamos como base para el desarrollo de la CSMA/CA su trabajo en este bloque.

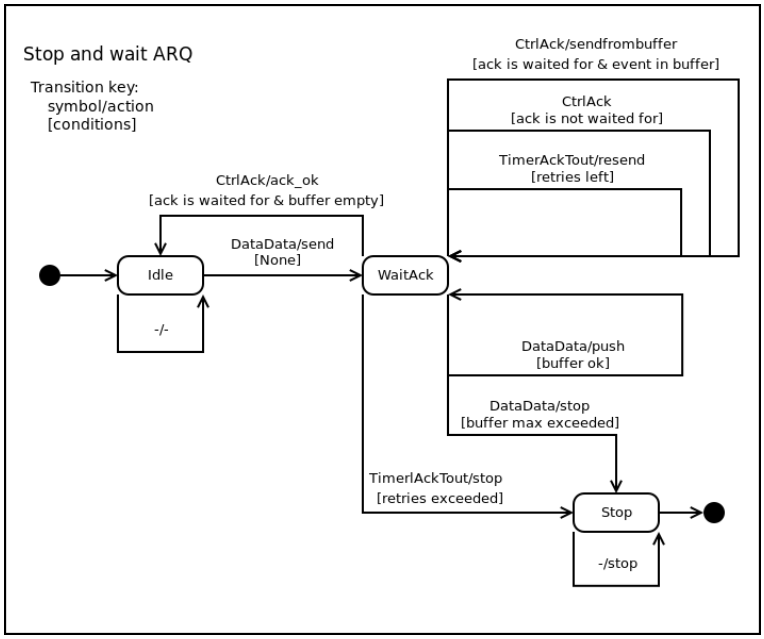


Figura 3.7: Máquina de estados del protocolo Stop and Wait. Cortesía de Víctor González Barbone.

3.5. Implementación en GWN de la CSMA (y pruebas) como FSM

Existen diferentes formas de programar protocolos. Entre ellas, la máquina de estados finitos (o FSM, por sus siglas en inglés) es una abstracción que ha sido muy utilizada para la representación de protocolos de acceso al medio. Se elige esta forma de programación en particular para los protocolos de acceso al medio porque las FSM son muy efectivas modelando operaciones de control secuencial. Incluso, varios protocolos de la subcapa MAC son definidos como máquinas de estados [88].

Para el trabajo de implementación de la CSMA/CA en máquina de estados, se estudió el protocolo Stop and Wait, ya implementado como máquina de estados. Esto se había realizado previamente para el desarrollo de otros protocolos ARQ (Go Back N y Selective Repeat) por parte de un integrante del grupo de proyecto. Entendida la lógica de xFSM, y teniendo ya experiencia en programación y verificación de protocolos similares, se emprendió el camino de la construcción de la CSMA/CA.

Lo primero fue trabajar sobre papel. Es decir, se dejó la computadora a un lado y nos concentramos en pensar qué estados, transiciones, acciones y condiciones necesitaríamos. Esto implicó entender en profundidad el protocolo, y la toma de decisiones respecto a qué aspectos del mismo era necesario ajustar.

Los protocolos, más allá de las definiciones existentes en distintas normas, no fueron considerados como una lógica cerrada y necesariamente a replicar con exactitud. Al revés, se comprendió su rol y función, y se trabajó para desarrollar un protocolo ajustado a nuestras necesidades concretas, tanto sea para no salir del mundo GWN, como para priorizar un bloque funcional y práctico por sobre uno que sea réplica de lo definido pero poco aplicable. Vale la pena destacar además que existen, como hemos dicho, muchas formas de acceso al medio con sensado, debido a la multiplicación de las comunicaciones inalámbricas y de sus aplicaciones. No es lo mismo pensar la MAC en 802.11 tradicional que orientado a redes mesh, o en enlaces de larga distancia [57].

En nuestro caso, se quiso replicar una CSMA/CA sencilla, como la que describimos en el capítulo anterior. Esto resultó en la máquina de estados presentada en la figura 3.8.

Como se observa, se identificaron 4 estados:

- **Idle:** Estado inactivo. La máquina se encuentra en este estado mientras espera que llegue una trama para transmitir.
- **WaitACK:** Estado de espera de ACK. La máquina se mantiene en este estado cuando envió una trama, hasta que llegue el ACK esperado en tiempo y forma.
- **BackOff:** Estado de retroceso. En este estado se espera un tiempo aleatorio antes de volver a sensar el canal. Se llega cuando un sensado del medio da ocupado.
- **Stop:** Estado de parada. Se llega a este estado cuando se supera la capacidad del buffer o la cantidad máxima de intentos de retransmisión. Cuando se llega a este estado, se envía un mensaje a las capas superiores ("Stop!"), para notificar que se paró el funcionamiento de la capa de enlace de datos. En el código de la aplicación (**chat_client.py**, o **estacion_base.py**) se puede incorporar por ejemplo que si llega este mensaje, se vuelva a ejecutar el código del diagrama de flujo de GNU Radio.

Para aclarar, **chat_client.py** o **estacion_base.py** son las aplicaciones que corren en la máquina anfitriona a la que está conectada el SDR. Se comunican con GNU Radio para enviar o recibir datos (a través de sockets).

Cada transición de la máquina de estados se compone de tres elementos, además de los estados inicial y final:

- El **evento** que llega, el cual puede ser un dato para enviar, un TimeOut o un ACK. En este punto es importante aclarar que la máquina de estados maneja dos TimeOuts distintos: uno representa el tiempo máximo de espera

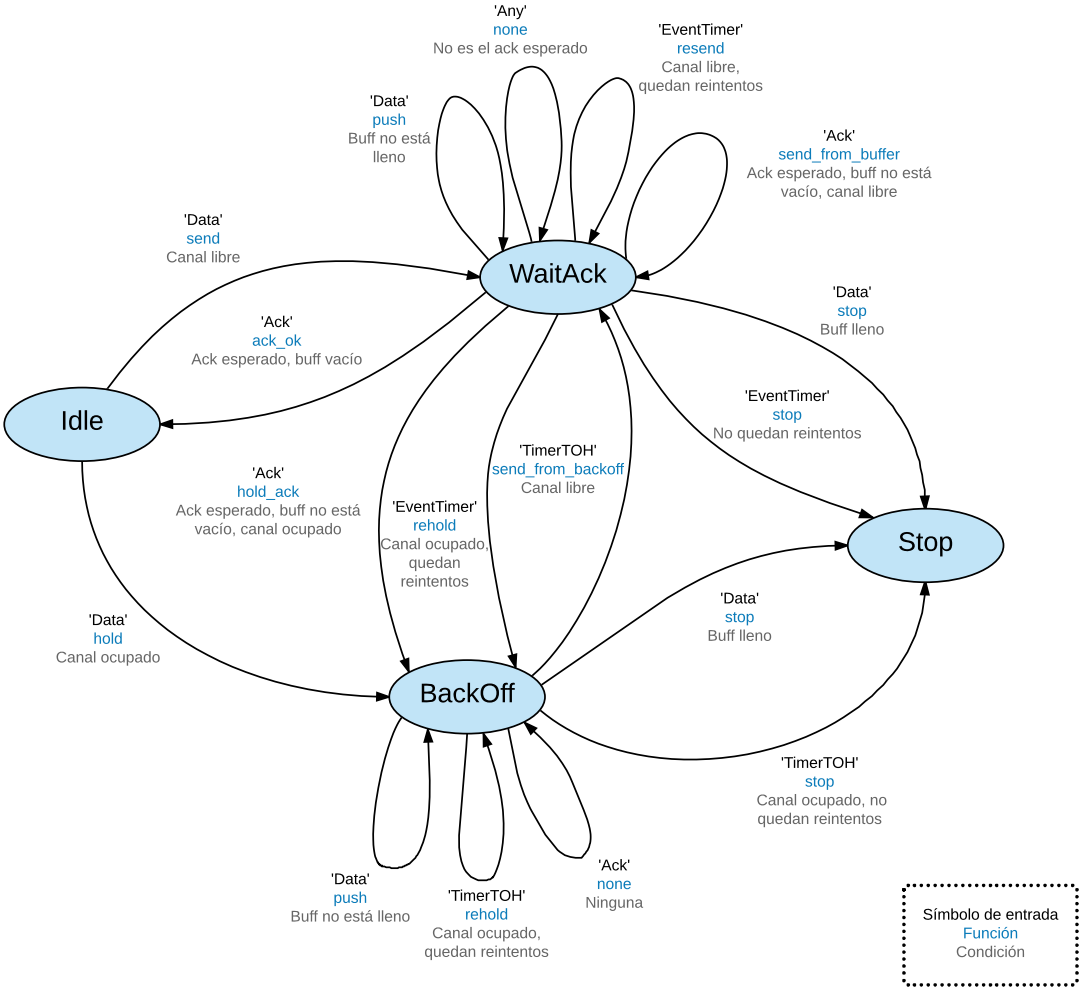


Figura 3.8: Máquina de estados implementada del protocolo CSMA/CA

de un ACK (en el diagrama se encuentra bajo el nombre de TimeACK) y otro representa el tiempo de BackOff sorteado (se encuentra bajo el nombre de TimeOut).

- La **función** que se ejecuta antes que la máquina pase al siguiente estado (por ejemplo enviar o guardar en el buffer un dato).
- Las **condiciones** que se deben verificar. Se pueden mencionar algunas a modo de ejemplo: que el canal esté libre, que el buffer no esté lleno o que el ACK sea el esperado.

La programación se hizo en Python, trabajando de cerca con los ejemplos provistos por el grupo ARTES. Los pasos a seguir para la creación de un bloque en GWN se encuentra detallados en el wiki de GWN: `https://github.com/`

`vagonbar/gr-gwn/wiki/Extending-GWN`.

La confirmación de la llegada de una trama en el orden correcto al destino se da con la llegada al receptor de un ACK. Este ACK, si bien en un Stop and Wait entre dos interlocutores puede no requerir números de secuencia (usando paridad por ejemplo), en nuestro caso, donde pueden intervenir más nodos, sí lo requiere. La confirmación se realiza entonces por Origen-Destino y número de secuencia, con memoria para evitar duplicados.

Una parte importante del desafío que representó la construcción de la CSMA/CA está dado en el sensado del medio. Éste se tiene que desarrollar con rapidez (no tiene sentido sensar el canal y demorar para tomar las decisiones) y se debe llevar adelante en la interfaz con el medio (las antenas en nuestro caso). Para esto, utilizamos un bloque llamado Probe Avg Magˆ2, que devuelve el cuadrado de la potencia recibida por la antena. Este valor es adquirido en las transiciones correspondientes, y comparado con un umbral (ajustable) para detectar si el canal está libre u ocupado. Esto se logró usando el lenguaje Cheetah embebido en XML (lenguaje de construcción de los bloques de GNU Radio) para obtener como variable en el bloque CSMA/CA la medida del sensado.

De hecho, uno de los obstáculos que tuvimos se dio en torno a esta solución. El bloque de la CSMA/CA va definiendo transición por transición la decisión a tomar, y medidas diferentes del sensado del medio durante la evaluación de las transiciones podían dejar al programa funcionando incorrectamente. Los problemas se darían en el caso en que el estado del canal cambiara mientras que se están recorriendo las transiciones, y el resultado podría ser desde descartar paquetes hasta quedar en un estado incorrecto. Por ejemplo, quiero mandar un dato y detecto al canal ocupado, entonces paso a chequear la transición siguiente, que antes de guardar el dato e iniciar el temporizador de BackOff, chequea la condición “canal ocupado”, y que en este segundo sensado el canal aparezca como libre. Se decidió entonces proceder según la RFC, y realizar primero el sensado del medio y luego decidir la lógica, sin sensar para cada transición. Esto puede no ser lo óptimo, y es todavía una línea a trabajar el estudio de los tiempos de sensado e inteligencia, para ver de aumentar la eficiencia de la CSMA/CA propuesta.

Otras arbitrariedades vinculadas al desarrollo del bloque son los ajustes realizados para la aplicación concreta del proyecto, que mencionaremos después, como por ejemplo para la medida de la calidad de servicio. Otros son configurables como parámetros del bloque (por ejemplo los tiempos de retransmisión y la unidad de tiempo básica para el estado de BackOff), cuya elección concreta para la utilización de la CSMA/CA en el proyecto será debidamente explicada cuando nos enfrentemos a la implementación en la vida real de la CSMA/CA.

Pero por ahora no tenemos todavía canal físico, y lo único que hemos logrado es construir el bloque de la CSMA/CA. El bloque resultante en GNU Radio Com-

panion se puede apreciar en la figura 3.9.

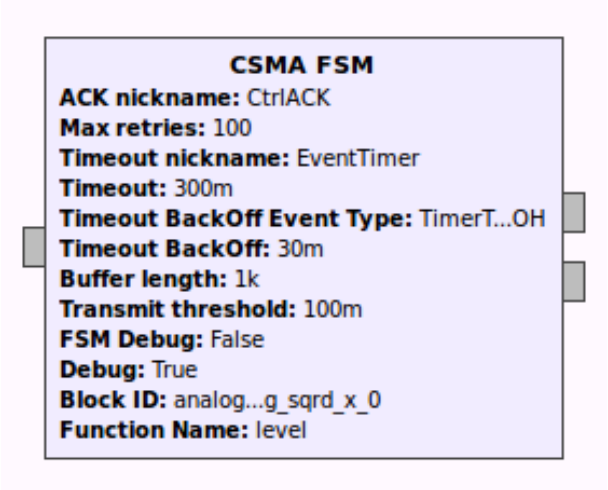


Figura 3.9: Figura del bloque que implementa la CSMA/CA

Sus parámetros son:

- **ACK nickname:** este es el nombre (nickname) del evento de tipo que esperamos que nos llegue como reconocimiento (ACK). En la máquina de estados se controla que coincida su número de secuencia con el esperado.
- **Max retries:** este parámetro fija la cantidad máxima de reintentos para cada paquete. Luego de superada esta cantidad se entra en el estado Stop, es decir cortamos la comunicación, dándola por finalizada.
- **Timeout nickname:** nombre del evento de tipo Timer que indica que expiró el tiempo que inicia cuando enviamos un dato válido. Al expirar este temporizador, consideramos que la trama se perdió, y debemos reenviarla (lo que llamamos un reintento).
- **Timeout:** valor del temporizador de reintento.
- **Timeout BackOff Event Type:** nombre del evento de tipo Timer que indica que expiró el temporizador asociado a la espera en el estado de Back Off.
- **Timeout BackOff:** valor de base del temporizador asociado al estado de Back Off. Inicialmente, de estar ocupado el canal, se esperará este valor, luego se sorteará entre [1; 2]*Timeout BackOff, como lo indica el algoritmo de retroceso exponencial binario. De volver a estar ocupado el canal, se sorteará un tiempo entre [1; 2; 3; 4]*Timeout BackOff. En los protocolos este valor se ha fijado en el tiempo de ida y vuelta de una trama. Nosotros lo fijamos arbitrariamente en un valor cercano a este tiempo. El tiempo de

procesamiento en GNU Radio se estima en 15ms, algo un poco mayor que los resultados de pruebas realizadas. Como observación, el tiempo de viaje de una trama entre dos nodos en nuestro caso es despreciable frente a estos tiempos de procesamiento (considerando la velocidad de la luz, alrededor de 0.06 microsegundos).
Se eligió entonces un valor de 30ms, considerando los tiempos de procesamiento de GNU Radio en ambos nodos como el grueso del tiempo de ida y vuelta de las tramas. Luego de 3 medidas del canal ocupado consecutivas, se fija el umbral del retroceso exponencial binario en el valor máximo de 11, para que no se generen tiempos de espera demasiado grandes, que enlentezcan más la comunicación de lo deseado. Considerando el valor inicial, con 11 reintentos se llega a una espera total de un segundo en el peor caso posible, que para nuestro sistema es razonable: los intercambios planteados (en modalidad de chat) no son tan extensos como para tener que esperar mucho, por lo que no es conveniente enlentecer la comunicación sin obtener como beneficio un mejor throughput (ancho de banda). Hay varios trabajos en torno a cómo se debe fijar este valor, por lo pronto resaltamos el trabajo de Bianchi sobre MAC en 802.11 [12]. En este, Bianchi trabaja con un valor de estados de BackOff máximo de hasta 8 estados, y considera que desde un valor máximo de 5 en adelante hay poca variación en el ancho de banda (throughput) resultante.

- **Buffer length:** tamaño de la ventana de transmisión. Superada esta capacidad, se descartan los datos que ingresan a la CSMA.

- **Transmit threshold:** este es un parámetro importante para que tenga sentido la CSMA. Es el valor de referencia para el sensado del medio, es decir que si el resultado del sensado es mayor que este valor, consideramos al canal ocupado, y si es menor, libre. Lo fijamos en un valor intermedio entre la potencia (al cuadrado) mínima para la recepción correcta de una trama y la potencia (al cuadrado) del ruido en recepción.

- **FSM Debug:** admite valores booleanos (Verdadero, Falso). Es una opción que de estar encendida muestra las transiciones y demás información de depuración de la CSMA.

- **Debug:** es una opción común en los bloques de GWN. Permite ver información de depuración, por ejemplo qué datos ingresan y egresan del bloque, con otra información relevante para tener noción de lo que está sucediendo (tamaño del buffer, número de secuencia esperado).

- **Block ID:** nombre del bloque del cual extraemos el valor del sensado del medio. En nuestro caso será el nombre del bloque Probe Avg Magˆ2.

- **Function Name:** el nombre de la función que leeremos para obtener la medida del sensado del medio. La función pertenece al bloque anteriormente definido, en nuestro caso es la función “level”, que retorna el valor deseado.

Para probar el funcionamiento de la CSMA/CA se establecieron un conjunto de pruebas que forzara el paso por todas las transiciones y estados posibles del sistema. El esquema de la prueba es el que se plantea en la figura 3.10.

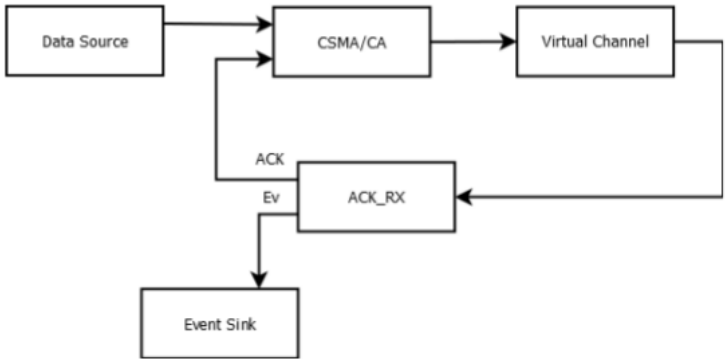


Figura 3.10: Esquema de prueba utilizado para la CSMA/CA

Esto se traduce al flow graph de la figura 3.11 donde se fijaron parámetros externos de manera de poder realizar estas pruebas, por ejemplo el estado del canal, o la pérdida de paquetes en la comunicación. El flow graph se desarrolló sobre el código que por defecto genera la creación de un bloque para la prueba del mismo.

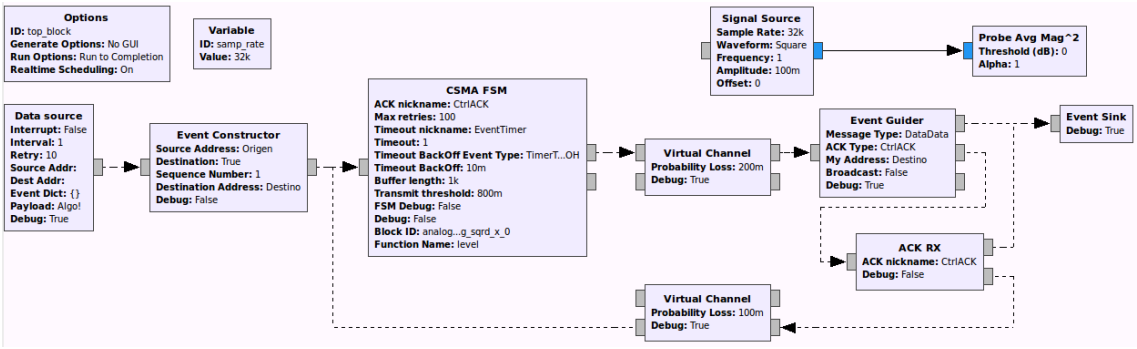


Figura 3.11: Diagrama de flujo de la prueba de la CSMA/CA

Nótese que este flow graph puede servir como prueba general de protocolos ARQ. El bloque Data Source genera datos, que pasan a la CSMA, luego del bloque Event Constructor, que explicaremos más adelante, pero asigna direcciones de origen y destino. La CSMA envía los datos cuando corresponde, simulando el sensado del medio acorde a lo fijado para la prueba, en este caso a través del bloque Signal Source generando un coseno de amplitud y frecuencia variables, a ser leído por el Probe Avg Mag^2. Los eventos pasan por un canal virtual (que simula el medio físico, con cierta probabilidad de pérdida), hasta llegar al receptor (bloque ACK_RX). Este pasa los datos al bloque Event Sink, es decir las capas superiores

del receptor, si llegan en orden, y envía a la CSMA el ACK correspondiente.

El bloque ACK_RX fue desarrollado originalmente por Víctor González Barbone para el protocolo Stop and Wait, pero tuvo que ser modificado para el proyecto, a fin de considerar algunas necesidades de la comunicación entre varios nodos. Las funciones del bloque serán vistas con mayor profundidad en el capítulo siguiente, al mencionar los bloques creados y ajustados para la implementación final. Por ahora quedémonos con la noción de que en caso de recibir una trama esperada (o nueva, pero en todo caso válida), se generará el acuse de recibo correspondiente (ACK) y se lo enviará al emisor, para que pueda dar como recibida la trama.

Las pruebas realizadas pueden consultarse con mayor detalle en el Apéndice "C". Las pruebas contribuyeron a corregir errores y falencias en el código, y permiten afirmar que la CSMA cumple correctamente su función desde el punto de vista de su inteligencia.

Para descartar otros errores, que pudieran aparecer en el pasaje del mundo simulado al mundo real, y para una prueba más completa, que incorporara la comunicación por Sockets con capas superiores, se utilizaron dos SDR para implementar un chat entre dos computadoras. La capa física usada fue provista por Pablo Belzarena, siendo la genérica que se ha utilizado en las pruebas de GWN, y será estudiada con mayor detenimiento. A la par del programa de GNU Radio debe correrse el archivo Python que figura en el Git del proyecto como **chat_client.py** (`https://github.com/MartinRandallC/RoCo/`), que abre una comunicación por socket en el puerto vinculado al bloque respectivo en GNU Radio, permitiendo enviar por socket lo que se escribe por terminal e imprimir en pantalla lo que llega al socket. Cada nodo debe estar entonces corriendo el programa de GNU Radio, con su SDR conectado, y ejecutando el script de python, y de esta manera se puede chatear por terminal.

Los resultados, utilizando USRP B100, fueron alentadores, ya que nuevamente la CSMA se comportó como debía, a menos de un cambio que tuvimos que hacer en bloque ACK_RX para permitir la comunicación entre más de 2 nodos. A continuación veremos el flow graph utilizado en cada nodo, en el que ya vemos aparecer todos los bloques utilizados para la implementación de la CSMA.

3.6. Diagrama de flujo de GNU Radio utilizado

Tomaremos el flow graph mencionado para introducir los bloques en juego en nuestra implementación final. Consideremos entonces la figura 3.12, con una diferenciación por capas y funciones.

El área marcada arriba a la izquierda en color azul tiene por cometido establecer la comunicación por socket con la aplicación, que corre en la terminal. La

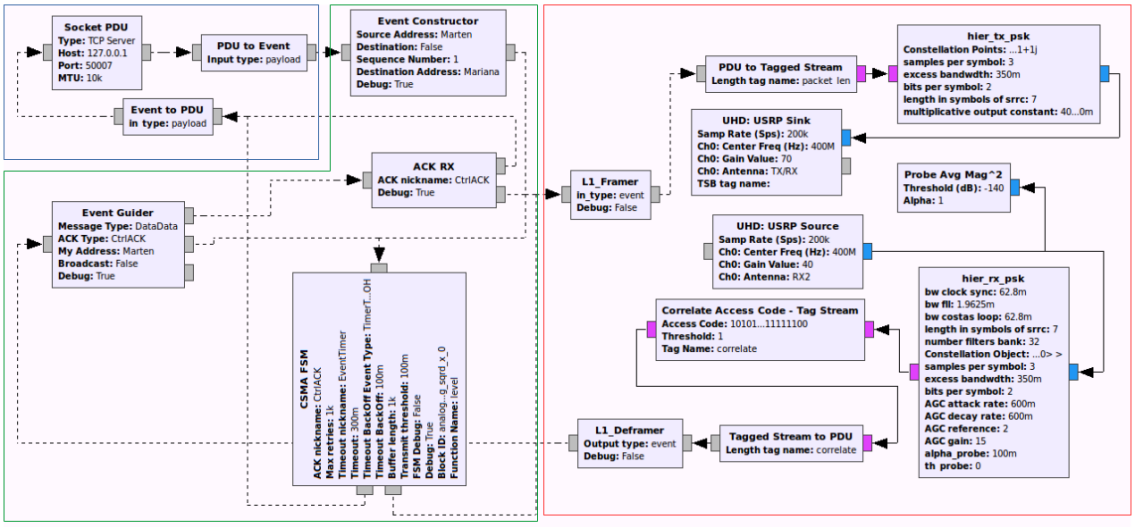


Figura 3.12: Diagrama de flujo utilizado. El área azul son las capas superiores (3 a 5). El área marcada en verde representa nuestra capa de enlace. El área rojo representa la capa física.

aplicación en nuestro ejemplo era el **chat_client.py**, ajustado para escuchar y hablar en el puerto deseado (socket TCP con dirección 127.0.0.1 y puerto 50007), que debe ser coherente con lo planteado en el flow graph. Asumamos que este es el caso, para poder concentrarnos en los bloques de GNU Radio. Cuando ingresan datos (es decir que alguien escribió en la terminal que corre **chat_client.py**), esta zona se encarga de construir el evento como nuestro sistema de comunicación lo requiere (de mensaje PDU a eventos GWN). PDU (Protocol Data Unit) se le dice a un conjunto de datos ordenados acorde a un protocolo, en el caso de nuestra capa de enlace los PDUs son los eventos GWN, y por ejemplo en capa de red son paquetes, en capa de transporte pueden ser segmentos. Luego de convertido, pasa este evento a la capa siguiente, es decir al área marcada en verde, que representa la capa de enlace de datos.

Esta capa incluye a la CSMA para transmisión y al ACK_RX para recepción. También incluye un bloque previo al ACK_RX que fue creado para filtrar rápidamente eventos que no están destinados al nodo, y un bloque constructor de eventos que asigna al evento direcciones de origen y destino. Esta capa se comunica con las capas superiores a través del área azul ya mencionada, y con las capas inferiores a través del área roja.

El área roja representa los bloques de modulación, traducción y el control de errores con códigos de redundancia cíclica. La traducción se hace entre los tipos: de eventos que utiliza GWN a números complejos que se envían al SDR. La modulación se realiza con bloques jerárquicos de capa física que ya se utilizan en GWN y fueron provistos por el Grupo ARTES. Esta merece un capítulo aparte, ya que entenderla y ajustarle parámetros fue parte importante del trabajo que se realizó para la implementación en los USRP B200/B200mini. Se trata de una modulación

QPSK con corrección de sincronización temporal, frecuencia y fase para el receptor.

Presentadas las partes, pasamos a explicar la función de cada bloque. Para esto veremos los caminos que siguen los diferentes datos (dato en transmisión, ACK, dato en recepción). El flowgraph descrito se mantiene, pero se ocultaron los bloques que no aportan en el camino detallado, para mejor visualizar las funciones de cada uno. Intentaremos no repetirnos, ya que hay bloques que participan en más de un caso.

Recorramos entonces los bloques como lo haría un dato recién ingresado a la terminal luego de pulsar "Enter". El recorrido será el de la figura 3.13

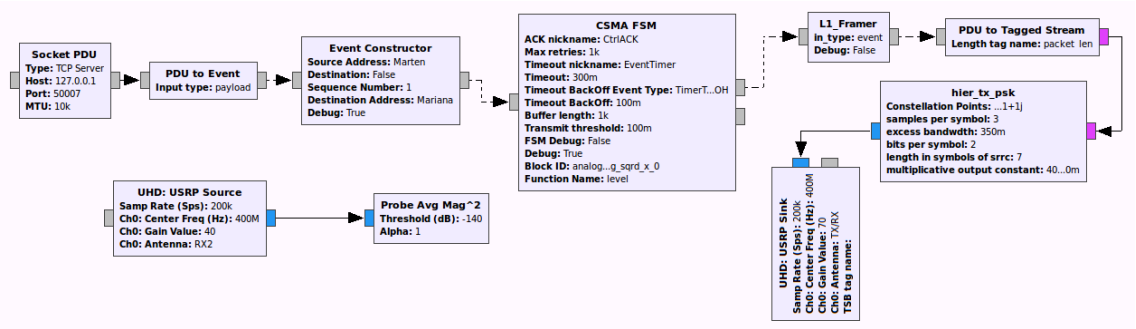


Figura 3.13: Recorrido desde las capas superiores de la pc anfitriona a través de la capa de enlace y hasta la salida del USRP.

- **Bloque Socket PDU:** Este bloque está escuchando en el puerto 50007 y dirección 127.0.0.1. En ese puerto debe estar corriendo la aplicación **chat_client.py**, como dijimos. Lo que sea ingresado a la terminal corriendo la aplicación será leído por el bloque y pasado como PDU al bloque siguiente. Este es un bloque nativo de GNU Radio, y permite usar TCP y UDP. En el caso del chat usamos TCP, pero para la aplicación final del proyecto usamos UDP. El funcionamiento de los Sockets se verá con mayor profundidad en el capítulo 4.

- **Bloque PDU to Event:** Este bloque toma el PDU y lo convierte a evento de GWN, con las características necesarias (direcciones, encabezados, etc). Lo que viene del Socket PDU es pasado como payload (carga útil, contenido) del evento. Fue creado por el Grupo ARTES.

- **Bloque Event Constructor:** Si bien en el bloque anterior se asignan direcciones al evento, agregando a los eventos un diccionario con éstas, en este bloque es que se asignan los valores de dirección. Este bloque lo creamos nosotros, y aunque podríamos haber modificado el anterior y unificarlos, no parecía prudente tocar un bloque tan fundamental y de uso genérico como

el PDU to Event para la actividad específica del proyecto. Tiene dos modalidades: asignando una dirección de origen y destino fijas, o extrayéndose éstas de lo escrito en el chat. Una vez fijado un criterio, se debe mantener durante la comunicación. Se le asigna a los eventos además un número de secuencia de manera aleatoria, reproduciendo lo que sucede en la realidad, y evitando que dos nodos queriendo hablar con un tercero envíen datos con igual número de secuencia.

El primer criterio es útil para pruebas, cuando no nos importa hablar entre varios sino mantener una comunicación fija. En ese caso se fija Destination en True, y se escriben las direcciones de origen y destino en el parámetro respectivo. Se construye entonces el diccionario del evento con estas direcciones, y se le asigna número de secuencia a cada evento a partir del 1.

La segunda opción recrea mejor lo que sucede en la capa. Se identifica del mensaje (proveniente de capas superiores) la dirección de destino (y se asigna la de origen, es decir la propia). El destino es la primera palabra de la carga útil, y debe mantenerse durante la conversación con ese destino: si quiero hablar con Vladimir, escribo en terminal entonces: <**You**> **Vladimir Hola!** . En cualquier momento puedo cambiar la dirección de destino, lo que permite que el sistema sea escalable para hablar con varios destinos.

Las direcciones deben ser únicas e invariantes durante la comunicación. Son nuestras direcciones de capa 2 (equivalente a la MAC por ejemplo).

- **Bloque CSMA FSM:** Este bloque ya fue explicado, pero tal vez el caso concreto aclare un poco su rol. Cuando le llega un dato a transmitir, debe sensar el medio y tomar la decisión en la máquina de estados. En caso de estar el medio disponible, lo envía a L1 Framer, y queda esperando el ACK respectivo. En caso de estar el medio ocupado, entra al estado de Back Off, esperando un tiempo aleatorio para volver a sensar el medio. Si estaba cumpliendo alguna de estas funciones, el dato entrará al buffer de transmisión, esperando su turno.
- **Bloque UHD: USRP Source:** Este es un bloque nativo de GNU Radio, que lee lo que hay en las antenas en la frecuencia definida y con la tasa de muestreo elegida, lo baja a banda base y lo pasa como muestras (complejas) para su tratamiento. En este bloque se fijan parámetros de bajo nivel: frecuencia de portadora, tasa de muestreo, ganancia y selección de la antena.
- **Bloque Probe Avg Magˆ2:** La clave para la lectura del medio, es decir el sensado del canal. Este bloque realiza el cálculo del módulo cuadrado del promedio de paquetes que le son pasados. Al conectarlo directamente a la salida del bloque cuya salida representa la lectura del medio, tenemos la medida del módulo cuadrado de la potencia de la señal en la frecuencia

deseada. Esta medida es leída por la CSMA a través de los parámetros Block ID y Function Level, como ya explicamos.

- **Bloque L1 Framer:** Otro aporte del grupo ARTES para traducción a capa física, el L1 Framer recibe un evento de GWN (de capa 2), le agrega código de redundancia cíclica para control de errores, y envía un PDU (Protocol Data Unit) a partir de esto, que será el dato a transmitir en capa 1.
- **Bloque PDU to Tagged Stream:** Este continúa el trabajo realizado por el L1 Framer, convirtiendo el PDU a un flujo de bytes, para su modulación. Es nativo de GNU Radio.
- **Bloque hier_tx_psk:** Bloque jerárquico otorgado por el grupo ARTES, fue brevemente descripto. No entraremos en más detalles, ya que le dedicaremos un capítulo a explicar la capa 1, y fundamentalmente este bloque y su recíproco (hier_rx_psk).
- **Bloque UHD: USRP Sink:** El bloque recíproco del UHD: USRP Source, nativo de GNU Radio. Envía por la antena (seleccionada como parámetro, así como la ganancia del amplificador del SDR) las muestras que le pasa el bloque anterior, a la frecuencia y tasa de muestreo elegidas (también pasadas como parámetros del bloque).

Veamos ahora el camino inverso, el que se recorrería en recepción. Aclaremos primero que todos los datos a enviar y recibir comparten la capa física, por lo que no repetiremos explicaciones (o eso intentaremos). Es decir que a partir de L1 Framer y hasta el bloque UHD: USRP Sink, es el mismo camino para todas las tramas que ya fue descrito. De igual manera, desde UHD: Source y hasta el L1 Deframer, es decir el camino de recepción en capa física, lo explicaremos ahora y es común a todos los datos a recibir.

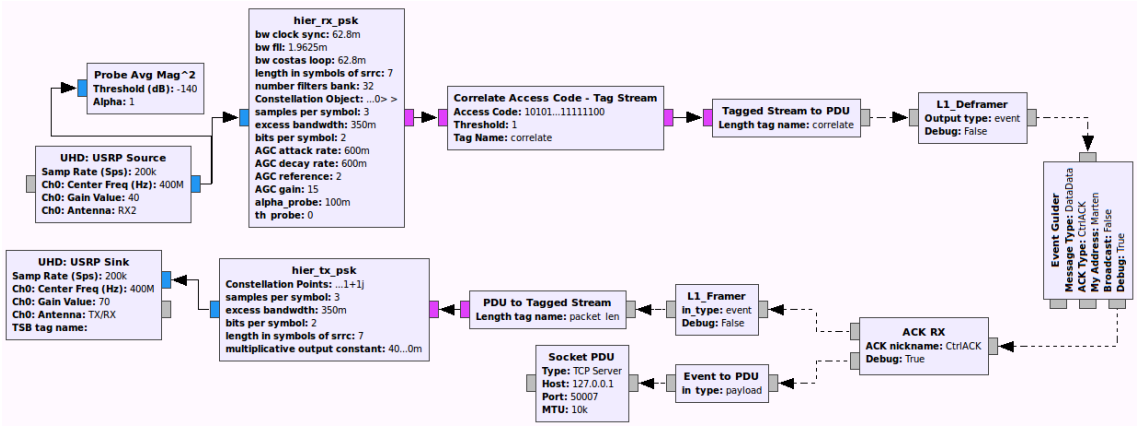


Figura 3.14: Las tramas llegan al USRP, pasan por la capa física y luego por la capa de enlace: generan el envío del ACK y la trama es pasada a las capas superiores a través del socket. Esta separación se realiza en el bloque ACK_RX.

- **Bloque UHD: USRP Source:** como dijimos, este bloque entrega las muestras (como un flujo de complejos) que recibe del aire en la frecuencia deseada, con tasa de muestreo y ganancia fijadas como parámetros.

- **Bloque hier_rx_psk:** este bloque es el recíproco del bloque hier_tx_psk, y cuenta con los mismos autores (grupo ARTES). Demodula las muestras recibidas y entrega un stream de bytes al bloque siguiente.

- **Bloque Correlate Access Code - Tag Stream** y **Bloque Tagged Stream to PDU:** en estos bloques, nativos de GNU Radio, se realiza la traducción de bytes a PDU, ubicando el valor del código de redundancia cíclica, el tamaño de la trama, etc.

- **Bloque L1_Deframer:** traduce el PDU que llega, construyendo el evento de GWN correspondiente. Realiza también un primer control de errores, verificando con el código de redundancia cíclica que la trama esté completa. Este bloque es el recíproco del bloque L1_Framer, y también fue proporcionado por el grupo ARTES.

- **Bloque Event Guider:** ahora ya entramos en capa de enlace de datos. Este bloque, creado por nosotros, representa un primer filtro, grosero, en esta capa. Comprueba si el evento que llega tiene mi dirección como destino, y en caso contrario lo descarta. En caso de que el dato que llega al Event Guider sea un ACK, lo pasará a la CSMA para su procesamiento. En caso que sea un dato dirigido a nosotros, seguirá camino hacia el ACK RX, donde se generará el ACK correspondiente a enviar, y será pasado a las capas superiores (a través del socket). En caso de querer cumplir la función de repetidor, debemos encender en True el parámetro Broadcast. Esto hace que todos los datos que llegan son reenviados. Veremos con mayor detalle esta función más adelante.

- **Bloque ACK RX:** este bloque ya fue introducido, pero es momento de verlo con detenimiento.

  Este bloque fue construido por el grupo ARTES, en principio para utilizar como receptor del protocolo Stop and Wait. Al ser muy sencillo este protocolo, el receptor también lo era. En particular, cada evento enviado por la CSMA incluía en su diccionario una palabra como identificador del ACK: 'ACK0' o 'ACK1'. En el receptor, se chequeaba si coincidía el diccionario del evento recibido con lo esperado, en cuyo caso se tomaba el dato como válido, y en caso contrario se descartaba como un duplicado.

  Este funcionamiento simplificado, válido para un Stop and Wait entre dos nodos, no era suficiente para nuestra implementación entre más de dos nodos. En particular por dos motivos:

1. El dato que generaba cada CSMA en cada nodo podría tener el mismo valor 'ACK0', y el receptor consideraría el segundo dato como un duplicado, descartándolo.
2. No se enviaban ACK duplicados. Es decir, si un dato llegaba al receptor, y éste mandaba el ACK correspondiente, asumía que el ACK llegaba bien. En caso de que el ACK se perdiera, y se reintentara enviar el dato, se descartaría en el receptor como duplicado.

Lo que se planteó fue la posibilidad de acercarnos al funcionamiento de otros protocolos de ARQ, que trabajan con número de secuencia. Este es un identificador único de cada trama, que se genera aleatoriamente al inicio de la comunicación y se incrementa por trama (se genera, al igual que las direcciones, en el bloque Event Constructor). El chequeo que se realiza en el bloque previo (Event Guider) garantiza la llegada de eventos destinados al receptor. Separamos entonces en recepción la llegada de tramas por origen y destino, generando el acuse de recibo con estas direcciones invertidas. De esta manera, en recepción podemos tener varias comunicaciones a la vez, y esperamos un evento con número de secuencia siguiente al último que llegó para esa comunicación específica.

Por otro lado, se habilitó la generación de ACK sobre eventos reenviados, pero solamente en el caso del reenvío del último evento llegado, que en el caso de Stop and Wait alcanza (no se manda nunca más de un evento a la vez). También, y por las dudas de que una comunicación se reseteara y el número de secuencia cambiara radicalmente, se habilitó la posibilidad de recibir un número de secuencia muy diferente a los esperados, considerándolo como un inicio (mejor dicho, reinicio) de la comunicación.

Estas modificaciones fueron de la mano con la construcción de los bloques Event Constructor y Event Guider. El resultado del conjunto de las modificaciones fue muy positivo: por un lado podemos resetear sólo uno de los nodos sin afectar al resto, que se adaptan a los nuevos números de secuencia y direcciones generados, esto es particularmente útil cuando tenemos nodos funcionando en el campo y no deseamos resetearlos por cambios en la estación base, por ejemplo. Por otro el sistema pasa a ser escalable. Alcanza con conocer nuestra dirección de origen (o más bien, tener una dirección propia), ya que las direcciones de destino se obtienen de lo escrito en la aplicación del chat, para poder establecer comunicación.

- **Bloque Event to PDU:** recupera del evento la carga útil, para pasarla al Socket correspondiente.
- **Bloque Socket PDU:** es el bloque nativo de GNU Radio para comunicarse por Sockets. Ya fue presentado, excepto que en este caso le entrará un dato PDU a escribir en la terminal en que estamos corriendo el **chat_client.py**.

El esquema cuyos bloques explicamos, recordemos, se trata del diagrama de flujo para una aplicación de chat entre varios nodos. Las únicas diferencias con nuestro esquema final para cada nodo se da en la utilización de Sockets UDP (entonces uno como cliente y otro como servidor), y en el rol de repetidor que puede jugar uno de los nodos.

Introduciremos entonces a continuación el concepto y la implementación de un repetidor, describiendo luego la modulación utilizada. Sigue la presentación de nuestro sistema de comunicación completo. Finalmente avanzaremos con la implementación en hardware del diagrama de flujos completo y las pruebas realizadas, detallando cómo se realizó en estas la medida de calidad de servicio.

3.7. El Repetidor

El objetivo de un repetidor es poder alargar el alcance de una comunicación, es decir aumentar la distancia máxima a la cual puedo mantener el intercambio. Con el incremento de la distancia comienza eventualmente una disminución de la calidad de la comunicación, ya sea por la introducción de errores, la atenuación del canal, la aparición de interferencias.

Es por esto que se han diseñado una gran variedad de dispositivos cuyo propósito es mantener las comunicaciones en buenas condiciones a pesar del incremento en distancia. Los repetidores son los más sencillos de éstos, ya que su única función es recibir una señal y amplificarla para enviarla. Vale la pena destacar que en esta concepción, el repetidor es invisible al origen y al destino, y que funciona en ambos sentidos. Por su sencillez, los repetidores tradicionales envían hacia el medio físico (en todos los sentidos) lo que les llega, en el caso de las comunicaciones inalámbricas es simplemente transmitir la señal llegada por aire.

Los repetidores digitales se conocen también como regeneradores, ya que realizan correcciones, buscando reconstruir los datos recibidos previo a su envío. En nuestro caso aprovechamos la lectura en capa de enlace del dato recibido para poder distinguir las tramas que eran específicas para el nodo (órdenes por ejemplo) de aquellas que buscaban ser amplificadas y enviadas (mensajes al Explorador por ejemplo). La reconstrucción de los datos también sirve para mejorar la calidad de la comunicación a nivel de capa de enlace. Sucede que en nuestro caso se busca que sea uno de los nodos quien actúe como repetidor, es decir que el repetidor incluye la comunicación con capas superiores cuando corresponde.

La forma de lograr la función de repetidor en nuestro sistema es habilitando en el bloque Event Guider una función llamada “Broadcast”, que admite el pasaje de parámetros booleanos. En caso de estar este parámetro en True, las señales que lleguen al Event Guider y no estén dirigidas a ese nodo serán reenviadas como llegaron, sin pasar por los protocolos de capa de enlace (sin pasar por el bloque

CSMA). Esto incluye tanto datos como reconocimientos, sin distinción. En caso de estar dirigido hacia ese nodo, no será reenviado, sino que se procesa como cualquier trama en el resto de los nodos. Esto es útil porque puede existir la voluntad de hablar específicamente con el repetidor, por ejemplo, ajustando su posición, o relevando datos de calidad de servicio.

Se realizaron pruebas con dos nodos comunicados y con un nodo funcionando en esta modalidad, de las cuales obtuvimos buenos resultados, confirmando el funcionamiento esperado del repetidor. La comunicación punta a punta mejoró sustancialmente al encender el repetidor, aunque luego veremos que el repetidor encendido puede afectar la medida de calidad de servicio que consideramos nosotros.

3.8. Explicación de la modulación digital utilizada

Habiendo analizado las tareas que cumple cada bloque en la capa de enlace de datos, es hora de estudiar cómo se comporta nuestro diagrama de flujo en la capa física. Es decir, entender lo que sucede en los bloques jerárquicos hier_tx_psk y hier_rx_psk, que son los encargados de la modulación y demodulación. Un bloque jerárquico es un bloque que incluye la combinación de varios bloques en su interior.

Primero veamos el proceso en su conjunto, y luego veamos qué función cumple cada uno de los bloques que componen los bloques jerárquicos.

La modulación utilizada es QPSK, que es un caso particular de modulación por cambio de fase. Esto significa que es en la fase (o la diferencia entre las fases) de nuestros complejos a transmitir y recibir que encontramos la información deseada.

Si queremos transmitir un conjunto de bits, debemos vincularlos a un conjunto finito de estados posibles (diccionario de símbolos), lo que se conoce como codificación. Por ejemplo los bits 00 serán el estado 0, los bits 01 el estado 1, etc. Luego, cada símbolo debe tener su correlato a nivel de una serie de números complejos, que serán las muestras de dos señales analógicas, una en fase y la otra en cuadratura (es decir desfasadas $\frac{\pi}{2}$.

En QPSK se eligen como símbolos 4 valores complejos separados por una fase de $\frac{\pi}{2}$, en nuestro caso: [1+j, 1-j, -1-j, -1+j]. En la figura 3.15 se ve una constelación resultante de utilizar QPSK con su par de bits asociado a cada símbolo. Cada par de bits a la vez es representado por un número complejo. Es decir que los bits 11 son representados por 1+j, los bits 01 por 1-j, los bits 00 por -1-j y los bits 10 por -1+j.

Veamos entonces qué sucede en nuestro diagrama de flujo.

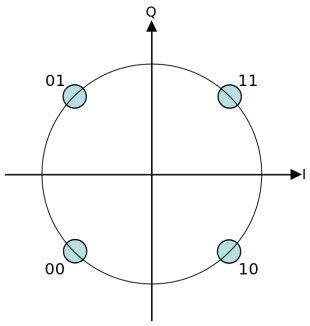


Figura 3.15: Constelación posible usando modulación QPSK [99].

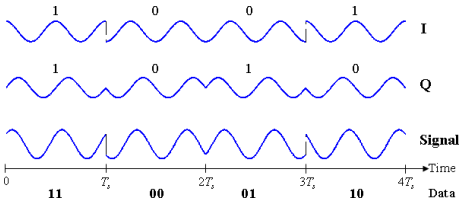


Figura 3.16: Representación de una palabra de bits a complejos [99].

Empecemos por la transmisión. Abriendo el bloque jerárquico, nos encontramos con que está compuesto por los bloques presentes en la figura 3.17.

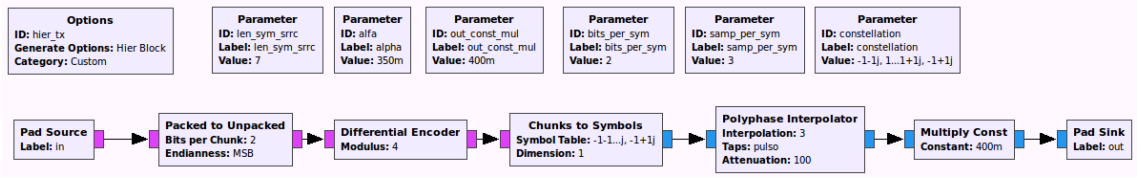


Figura 3.17: Conformación del bloque jerárquico de transmisión hier_tx (iría conectada la entrada al bloque PDU to Tagged Stream y la salida bloque UHD: USRP Sink).

Habíamos quedado en el bloque PDU to Tagged Stream, que convierte de mensajes PDU a un flujo etiquetado de bytes. Siguiendo el recorrido:

- **Bloque Packed to Unpacked:** a este bloque ingresan bytes y genera nuevos bytes, pero con sólo 2 bits útiles. Por ejemplo si entra 11011001 al bloque, generará 00000011, 00000001, 00000010 y 00000001. Esto es para obtener un tren de pares de bits que tomarán los 4 valores que asociaremos a los 4 estados posibles. Este bloque es análogo a un bloque serie a paralelo.
- **Bloque Differential Encoder:** la modulación por fase puede sufrir de problemas de cambios de fase generados por perturbaciones de la comunicación (interferencias, rebotes, fading, efecto Doppler). Para ser resistente a la pérdida de una fase, es que se utiliza lo que se llama modulación diferencial

QPSK (o DQPSK). Esto significa que no vinculamos valores de bits a una fase determinada, sino que estudiamos los cambios de fase entre valores consecutivos de las parejas de bits. Por ejemplo (esto depende de cómo se hace la constelación), si no hay cambios entre el par de bits actual y el anterior, los codificamos con el valor 00. Si hay cambios de $+\frac{\pi}{2}$, la salida será 01. Si hay cambio de π, la salida será 10. Si hay cambio de $-\frac{\pi}{2}$, la salida será 11.

- **Bloque Chunks to Symbols:** este es el bloque que encarna la conversión de los pares de bits a símbolos. Se le pasa por parámetro la tabla de la constelación deseada, en nuestro caso [-1-1j, 1-1j, 1+1j, -1+1j], que se corresponde respectivamente con [00, 01, 10, 11] como mostraba la constelación en la figura (figura de la constelación).

- **Bloque Polyphase Interpolator:** este bloque convoluciona la entrada interpolada con un filtro. El filtro se construye a partir de taps que se especifican, y debe verificar ciertas características, en particular es deseable que pertenezcan a la categoría de los pulsos de Nyquist. Usamos uno en particular, que se llama Squared Root Raised Cosine (conocido como coseno elevado).

- **Bloque Multiply Constant:** tenemos entonces un tren de pulsos complejos, este bloque no hace más que multiplicar la amplitud de los mismos para, por ejemplo que las muestras que entren al bloque de transmisión (UHD: USRP Sink) no superen la amplitud máxima admitida (1 digital).

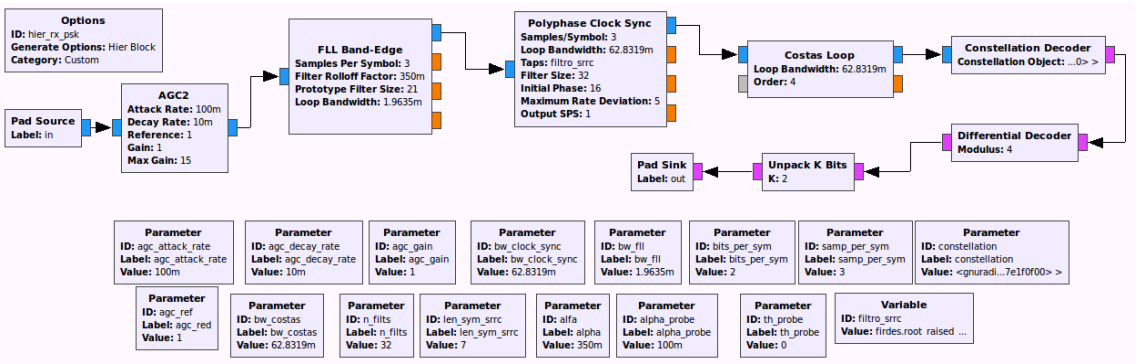


Figura 3.18: Conformación del bloque jerárquico de recepción hier_rx (el origen vendría del bloque UHD: USRP Source y la salida al bloque Correlate Access Code - Tag Stream).

Desde la recepción de las muestras hasta el tren de bits (que luego empaquetamos como eventos GWN para el trabajo en capa de enlace), tenemos los siguientes bloques (figura3.18):

- **Bloque AGC2:** a este bloque llegan las muestras que vienen directamente del medio físico, es decir lo que recibe la antena a la tasa de muestreo y frecuencias elegidas. Para poder leer estas muestras con cierta estabilidad, se utilizan bloques de control de ganancia (Automatic Gain Control), que

buscan fijar frente a un valor de referencia la amplitud de las mismas. Esto hace que aunque llegue una señal con potencia débil, si el AGC logra detectarla la buscará acercar al valor de referencia, y lo opuesto: si llegan señales con mucha potencia, se disminuirá su amplitud.

- Existen muchos errores que se introducen en las comunicaciones inalámbricas. Entre ellos, está la posibilidad (alta) de no coincidir la frecuencia la portadora entre emisor y receptor, lo que provoca un error en frecuencia. Esto puede suceder tanto por diferencias en el hardware, como por efectos del canal físico (rebotes, interferencias, movimiento). El error en frecuencia resulta en, por ejemplo, una constelación que gira, y puede desembocar en la lectura incorrecta de los símbolos. Esto es corregido por el bloque **PLL Band-Edge**.

- Otro factor que puede causar daños importantes en la recepción es que los relojes de emisor y receptor no estén sincronizados, lo que puede provocar una lectura no óptima (e incluso incorrecta) de las muestras recibidas. El objetivo siempre es buscar muestrear en los puntos que aportan mayor información de la señal, esto es en el máximo del pulso a la salida del filtro apareado [11]. Esto maximiza la relación señal a ruido, y permite tener un receptor con mayor inmunidad frente a ruido, interferencia, etc. Para sincronizar los relojes se utiliza el bloque **Polyphase Clock Sync**, que incluye al filtro apareado al pulso conformador.

- Finalmente, un error común refiere a las diferencias entre las fases de receptor y emisor. Esto puede deberse a diferencias de hardware, como también a efectos del canal. Se observa en la constelación como un desfasaje fijo. Esto es importante de corregir para evitar que se confunda la interpretación de las muestras. Por ejemplo si una muestra que debía estar en el punto 1+j cae en 1,42 (desfasaje de pi/4), no sabremos si vincularla a 1+j o a 1-j. La corrección en fase se realiza usando un bloque **Costas Loop**.

- **Bloque Constellation Decoder:** este bloque es el recíproco del Constellation Encoder presentado para transmisión. Reconstruye los bits a partir de los complejos que le llegan, es decir que si llega por ejemplo un 1+j, sale 10.

- **Bloque Differential Decoder:** como la modulación era diferencial, de igual forma debe ser la demodulación. De eso se encarga este bloque, cuyo trabajo es reconstruir el tren de bits a partir de las diferencias entre fases, realizando el proceso inverso que hizo el Differential Encoder.

- **Bloque Unpack K Bits:** finalmente, debemos reconstruir el tren de bits que conformará el mensaje etiquetado, que será a su vez traducido a PDU para terminar como eventos. En este sentido es que hay que tomar únicamente los bits útiles (con información) del flujo que tenemos en recepción. Es decir, en transmisión, en el bloque Packed to Unpacked habíamos agregado

bits (6 bits por byte), y en este bloque debemos recuperar 2 bits por byte, de manera análoga.

3.9. El sistema de comunicación completo

Veamos ahora nuestro sistema de comunicación completo. En la radiobase y el nodo Explorador tendremos el diagrama de flujo de la figura 3.19. En el Repetidor, tendremos el mismo diagrama de flujo pero con la opción Broadcast del bloque Event Guider en True. En el Apéndice E se encuentran los diagramas de flujo finales usados en cada nodo.

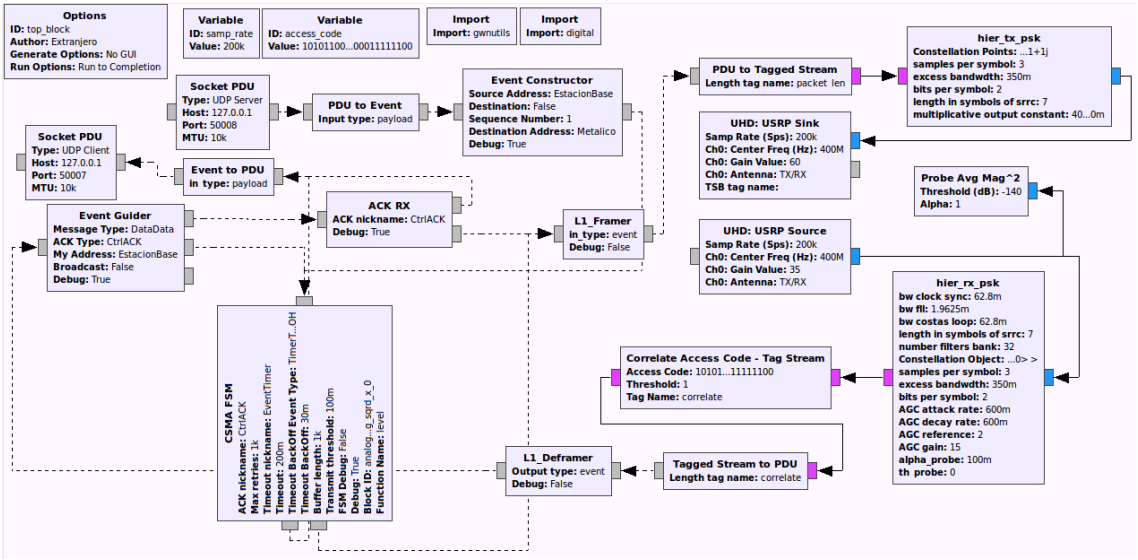


Figura 3.19: Figura general del diagrama de flujo utilizado en la Estación Base.

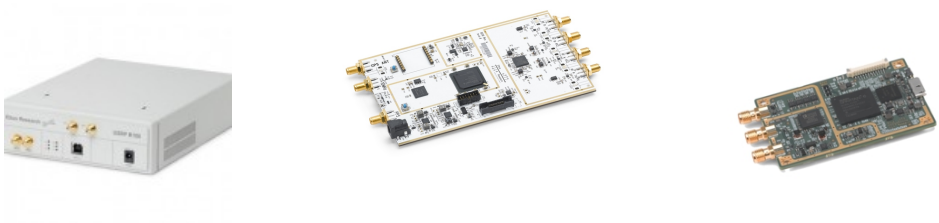
Pasemos ahora a explicar cómo se implementó en la realidad este diagrama de flujo para cada nodo, deteniéndonos en las pruebas y correcciones que hubo que realizar.

3.10. Pasaje a HW: pruebas realizadas y resultados obtenidos

Una parte importante del proyecto fue la confirmación del funcionamiento correcto de la CSMA/CA, así como del sistema de comunicación en su conjunto. Esto exigió un conjunto de pruebas que permitieran verificar nuestro trabajo. Ya hemos mencionado las pruebas realizadas a nivel de simulación y con los USRP B100 relativas exclusivamente a la CSMA/CA. Sin embargo, el pasaje a hardware del sistema de comunicación introdujo varias dificultades, algunas inesperadas, en

particular por pasar a utilizar los USRP B200.

Los USRP B200 y B200mini son SDR más modernos que los USRP B100. Son del mismo fabricante, Ettus Research, ahora incorporado a National Instruments. Para el proyecto fue necesario utilizar los B200mini, que se caracterizan por tener mucho menor tamaño y peso que el resto de los SDR de capacidades similares. Sumado a esto, los B100 se alimentan externamente desde la red eléctrica, mientras que la línea B200 se alimenta por USB. Tanto por las características de alimentación como de tamaño, los B200 mini son la opción de que se dispuso para la utilización de SDR sobre robots móviles [26].



(a) USRP B100 [58] (b) USRP B200 [100] (c) USRP B200mini [51]

Figura 3.20: Modelos de USRP utilizados

Para depurar nuestro sistema de comunicación se buscó siempre aislar el error, partiendo nuestro esquema completo en partes más sencillas a probar. El concepto detrás de esto es la verificación de cada parte por separada del diagrama de flujos.

Por ejemplo por un lado la CSMA/CA se prueba en simulaciones para descartar errores de protocolo, por otro lado se prueba la capa física. De la CSMA/CA se prueba primero el funcionamiento del protocolo Stop and Wait, considerando un canal sin pérdidas y con sólo dos nodos. Luego se confirma la función de sensado del medio, simulando interferencia. Una vez realizada la prueba que recorre todos las situaciones posibles a la que se puede enfrentar el bloque, se da como válido a nivel de simulación. Recién entonces se prueba con hardware.

Bajo esta premisa se realizaron diagramas de flujo para la verificación de cada uno de los bloques creados: Event Constructor, Event Guider, CSMA FSM y ACK RX. Los diagramas de flujo son generados automáticamente al momento de crear los bloques, bajo la forma de test de Quality Assurance, aunque luego hay que modificarlos para ajustarlos a la prueba deseada. Las pruebas de Quality Assurance buscan prevenir y eliminar los errores posibles previo a la ejecución de los procesos, a diferencia de otras pruebas como de control de calidad que se enfocan en los resultados.

Inicialmente no se realizaron pruebas sobre la capa física provista por el grupo ARTES, pero el pasaje a hardware forzó el desarrollo de las mismas. Al encontrarnos con errores en el sistema de comunicación completo que no sucedían cuando se utilizaban los USRP B100, nos concentramos en el estudio de la capa física y la búsqueda de herramientas de depuración. Los problemas encontrados se dieron cuando se utilizaron USRP B200 y USRP B200mini para la comunicación que había sido probada en USRP B100, y se observó que el sistema de comunicación no funcionaba.

Por funcionamiento correcto de la CSMA/CA, de la capa física, o más en general del sistema de comunicación, nos referimos a que los mensajes enviados son recibidos exitosamente en su mayoría. Esto implica la demodulación y el pasaje a capa superior (capa de enlace) de las tramas para su procesamiento. En el escenario de una distancia cercana (igual o menor a dos metros) deberían ser bien recibidos y leídos casi todos los mensajes, a menos de alguna colisión. Ante mayores distancias, al menos algunos de los paquetes deben llegar y ser leídos, es decir que se detectan, demodulan y pasan a la capa de enlace. Cuando se trabaja sólo con la capa física sin incluir la lógica de GWN, la confirmación del funcionamiento pasa por si se recibe la constelación correctamente, y si se reconstruye el mensaje (se utilizaron canciones en transmisión).

Lo primero fue probar con el nuevo hardware (USRP B200 y B200mini) la capa física sin incluir eventos GWN. Es decir, solamente probar la modulación digital utilizada. Las pruebas se realizaron generando un flujo de datos (una canción en formato wav) a transmitir y recibir, confirmando en recepción a través de la constelación que los datos llegaban correctamente, así como escuchando la canción transmitida.

Esta prueba, exitosa, nos permitió descartar que hubiera un problema de hardware más profundo, como la incompatibilidad de los USRP B200 y B200mini con alguno de los parámetros de nuestro diagrama de flujos, o que su mal funcionamiento se debiera a hardware defectuoso.

Surgió entonces la hipótesis de que el problema de nuestro sistema de comunicación se encontraba entonces en la vinculación entre la capa física y la utilización de eventos GWN.

Pero al momento de trabajar sobre esta tesis nos encontramos con que los bloques típicos de GNU Radio para debbugueo no eran suficientes, ya que están pensados para flujos continuos de datos. Esto es justamente una de las innovaciones que introduce GWN, y en consecuencia, tuvimos que establecer un sistema de pruebas propio.

Este consistió en guardar la historia de lo sucedido durante la ejecución de la prueba. Es decir, guardar los datos en algún momento de su procesamiento, a

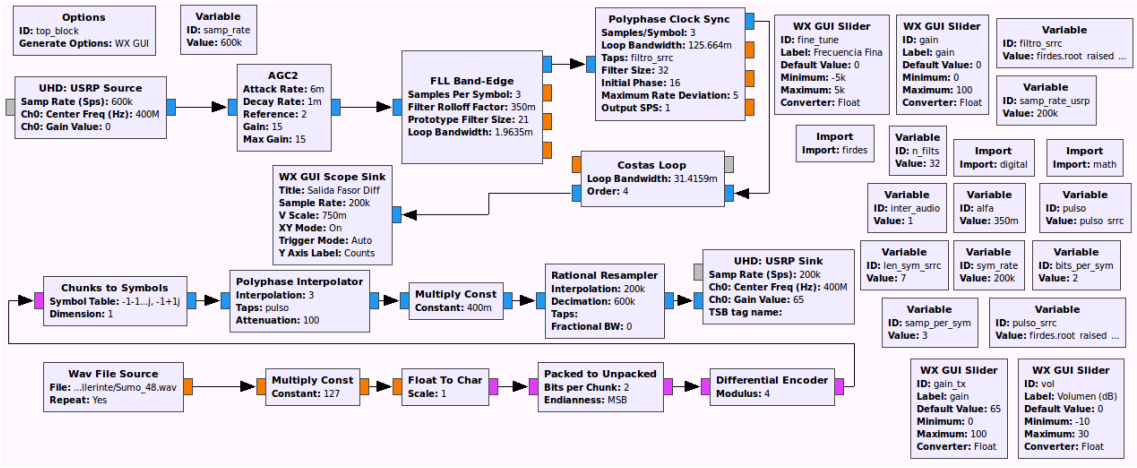


Figura 3.21: Diagrama de flujo utilizado para la prueba de la capa física. Nótese que se utiliza una canción para la generación de las muestras a transmitir (abajo a la izquierda). El camino de transmisión se encuentra abajo en la imagen, mientras que el camino de recepción se encuentra arriba.

la entrada o salida de alguno de los bloques donde suponíamos que podía estar el error buscado. Para lograr esto se utilizó el bloque File Sink, que guarda las muestras complejas que recibe en un archivo en formato binario. Estas muestras fueron luego procesadas en Octave, con un script sencillo que devuelve el flujo de datos como un vector de complejos, que es lo que queremos analizar.

Graficando el valor absoluto de este vector se visualiza la amplitud de las muestras y por ende, del ruido y de las tramas. Graficando el vector, se obtiene la constelación, es decir la distribución en el plano complejo de las muestras. El código usado se encuentra en el Git del proyecto (`https://github.com/MartinRandallC/RoCo/wiki/Pruebas`).

Para el desarrollo de las pruebas de la capa física, se construyeron diagramas de flujo sencillos, donde dos nodos se comunican sin capa de enlace. Consiste en el envío de datos desde un Data Source y su modulación en transmisión, y la demodulación y recepción a través de un Event Sink en recepción.

En el diagrama de prueba presentado, se recaban los datos a la salida del bloque que entrega las muestras desde el medio (UHD: USRP Source), y luego a la salida del bloque AGC2.

Se probó también la lectura de los datos más adelante en el diagrama de flujos, realizando pruebas alterando algunos parámetros como ser la tasa de muestras por símbolo, sin éxito. El resto de los parámetros correspondía con los valores que ya se habían utilizado en otras ocasiones, como fue durante la realización del curso de Comunicaciones Inalámbricas, donde se estudia modulación digital en SDR. Entre estas pruebas y el estudio de los valores a la salida de los primeros dos bloques, se

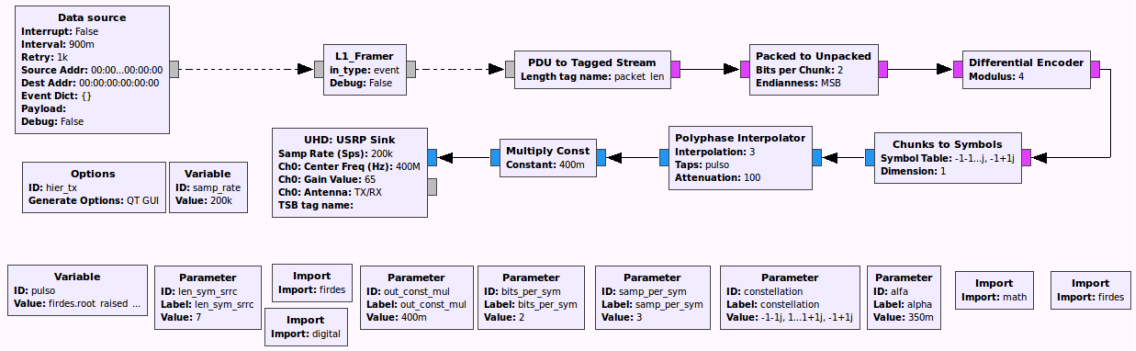


Figura 3.22: Diagrama de flujo utilizado en transmisión para la prueba de la capa física con eventos GWN.

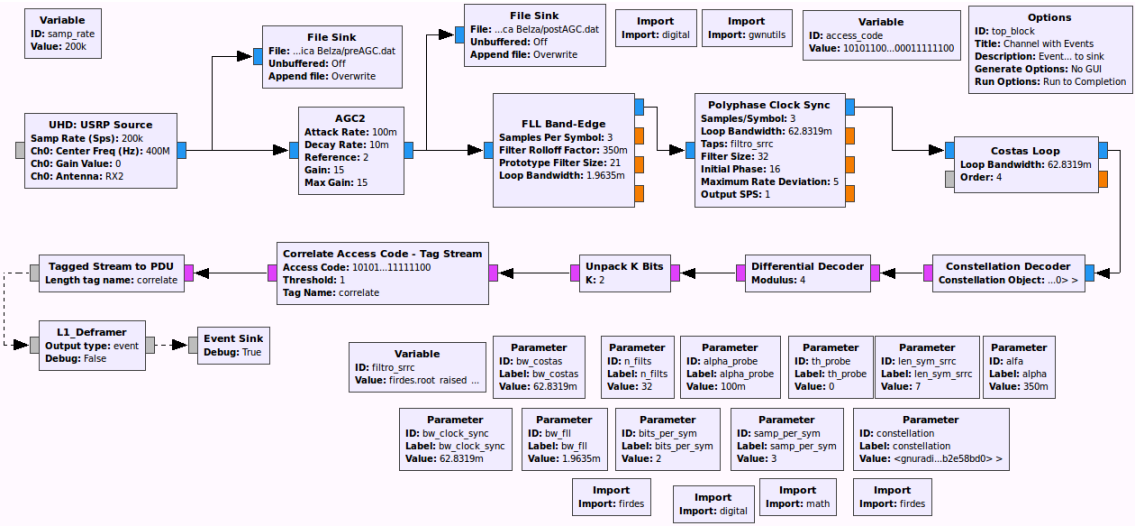


Figura 3.23: Diagrama de flujo utilizado en recepción para la prueba de la capa física con eventos GWN.

confirmó que si el AGC2 entregaba un flujo de complejos donde se distinguieran correctamente los pulsos recibidos, alejados lo suficiente del ruido, el resto de la demodulación no sería problema.

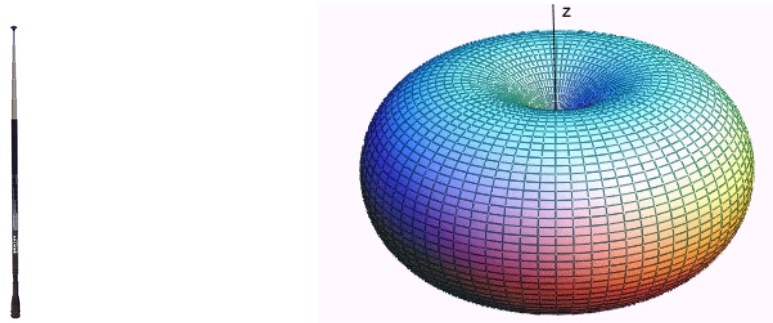
Volviendo al trabajo sobre modulación digital sin la utilización de eventos GWN, se estimó el valor de la ganancia de los amplificadores de las antenas de los USRP B200, para que se comunicaran correctamente con un USRP B100. Se observó que la "línea" B200 (que incluye al B200 mini) necesita de mayor ganancia en la etapa de RF para obtener similares resultados que el B100. Esto se comprobó analizando las hojas de datos, a pesar de que se discontinuó el B100 y hay pocos datos de referencia [74]. Se encontró que los B100 tienen una ganancia alrededor de mil veces mayor que los B200, lo que se condice con lo hallado experimentalmente (alrededor de 30 dB de diferencia) [26] [16].

Existe otra hipótesis que fue verificada en el esquema de la modulación digital sin MAC. Esta es la alta dependencia existente entre la alimentación de los USRP y la potencia de transmisión y recepción. La potencia de transmisión es tres veces mayor cuando estos se encuentran alimentados externamente (desde la red) frente a cuando los alimenta una computadora portátil desenchufada. En nuestro proyecto, al estarse tratando de nodos móviles, la alimentación de los B200 mini sería a través del Raspberry PI. En el capítulo referido a la alimentación de los nodos se encuentra mayor información. En todo caso, y para nuestro sistema de comunicación, esto es una restricción importante a la hora de estimar las ganancias. Toda prueba realizada debió entonces ser luego verificada en escenarios con la alimentación final a ser usada. El alto consumo energético que tienen los SDR se debe a que se les está trasladando parte del procesamiento de las señales, y es una de las explicaciones de porque se ha utilizado más en radiobases (con acceso a la red eléctrica como fuente de energía) que en aplicaciones de usuario final [48].

Un aspecto que tuvo que considerase fue la temperatura de los USRP. Debido a la fragilidad del hardware utilizado, además de su alto costo, para los B200 mini se realizaron cajas protectoras en una impresora 3D. Durante las pruebas de ganancia, se observó un aumento sensible de la temperatura en los conectores de las antenas, además de un aumento (menor) de la temperatura general de la caja protectora. Esto coincidió con la necesidad de aumentar las ganancias en las antenas, por lo que se asumió que el recalentamiento de los USRP provoca cambios en las potencias, además de que por precaución se optó por dejarlos enfriar cada cierto tiempo. Este aumento de temperatura se dió únicamente con aquellos USRP alimentados desde la red eléctrica o desde una computadora conectada a la red eléctrica, es decir con lo que se consideró "buena alimentación".

Otro factor que afecta la potencia de las señales, tanto en transmisión como en recepción, son las antenas a utilizar. Por un lado se debe elegir adecuadamente el modelo de antena en juego: direccional, omnidireccional, un arreglo de antenas. Por la situación de movilidad de los nodos, se eligió la utilización de antenas omnidireccionales, que permiten que aún cuando los robots estén girando, se mantengan valores similares de potencia. Por otro lado, una vez elegida la antena, debe estimarse su largo, para optimizar la relación señal a ruido en recepción. La geometría de las antenas permite aumentar la potencia de la señal recibida según la frecuencia de trabajo. Se eligió el tamaño de la antena acorde a la frecuencia de trabajo (400 MHz), lo que multiplicó por 3 la potencia de la señal en recepción. El largo de la antena se estimó igual a $\frac{5\lambda}{8}$, siendo λ la longitud de onda de nuestra señal, es decir en 47cm con el valor de frecuencia usado. Esto genera un patrón de radiación razonablemente orientado hacia la dirección de interés, y con una ganancia cercana a la máxima [9].

Las medidas de potencia (para estimar ganancias, tamaño de antenas, etc) se realizaron utilizando el script de octave, estudiando los datos crudos inmediatamente recibidos, a la salida del bloque UHD: USRP Sink.



(a) Antena omnidireccional [7] (b) Patrón de radiación [24]

Figura 3.24: Antena omnidireccional y su patrón de radiación

Una vez elegidos estos parámetros, se retomó el trabajo relativo al intercambio de datos usando eventos de GWN. Nuestro sistema de comunicación funcionaba entre dos USRP B100, y el problema se dió al incluir al USRP B200. Se realizaron pruebas para descartar que hubiera incompatibilidad entre la tasa de muestreo y el hardware, ya que el samp_rate (tasa de muestreo) debe guardar cierta relación con el reloj del sistema [61].

Existen, y esto es una virtud del conocimiento libre, varios foros con debates interesantes sobre los más diversos aspectos de GNU Radio y SDR. Algunos debates que nos parecieron interesantes se encuentran en el Git de RoCo, en la página de pruebas (`https://github.com/MartinRandallC/RoCo/wiki/Pruebas`).

También funcionó cuando un USRP B100 transmitía y un USRP B200 recibía. Sin embargo, cuando el B200 transmitía y recibía se encontró que luego del primer mensaje enviado se trancaba la comunicación. La figura 3.25 grafica lo recibido luego de pasar por el bloque AGC2.

Obsérvese que vemos dos pulsos: los propios y los ajenos. Es que la señal generada para la emisión de datos es detectada también por la antena de recepción. La cercanía entre las antenas de un mismo USRP hace que la potencia de estos pulsos “propios” sea mucho mayor a la de los pulsos ajenos. Por otro lado, los pulsos ajenos siguieron llegando con razonable potencia, es decir, similar a la del camino half-duplex, que no dió problemas para la demodulación. En la figura 3.26 se observa el ”phase trajectory plot”de un pulso ajeno, recibido previo a la llegada del pulso propio.

Nótese que tiene la forma que corresponde a QPSK con pulso SRRC con 50 % de exceso de ancho de banda (figura 3.27).

En la figura 3.28 se muestra el ”phase trajectory plot”luego de la recepción de un pulso propio.

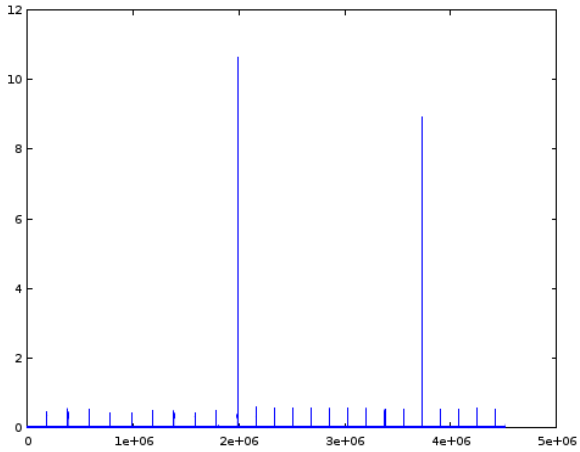


Figura 3.25: Valor absoluto de las muestras recibidas en función del tiempo. Los pulsos altos son los generados por el propio USRP.

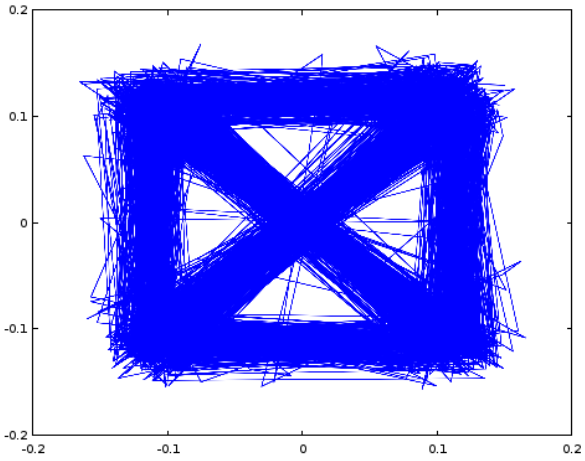


Figura 3.26: Gráfica de las muestras recibidas de un pulso previo a la llegada de un pulso "propio". Nótese el parecido con la constelación esperada de QPSK.

Esta gráfica, lejos de lo que uno desea, indica que no se identifica más que ruido con una amplitud considerable. Para el estudio de las constelaciones se seleccionaron las muestras correspondientes al pulso que se desea graficar.

Esto nos planteó una serie de pautas por donde avanzar: una opción era relativa al tamaño del encabezado de los pulsos, dónde hay bits que son deformados: mientras el AGC busca llegar al valor de referencia, puede que se pierda demasiada información como para reconstruir el mensaje. La otra refería a la amplitud de los pulsos, en concreto, a regular correctamente la ganancia de las antenas.

La ganancia de las antenas ya se había estimado en las pruebas de modulación

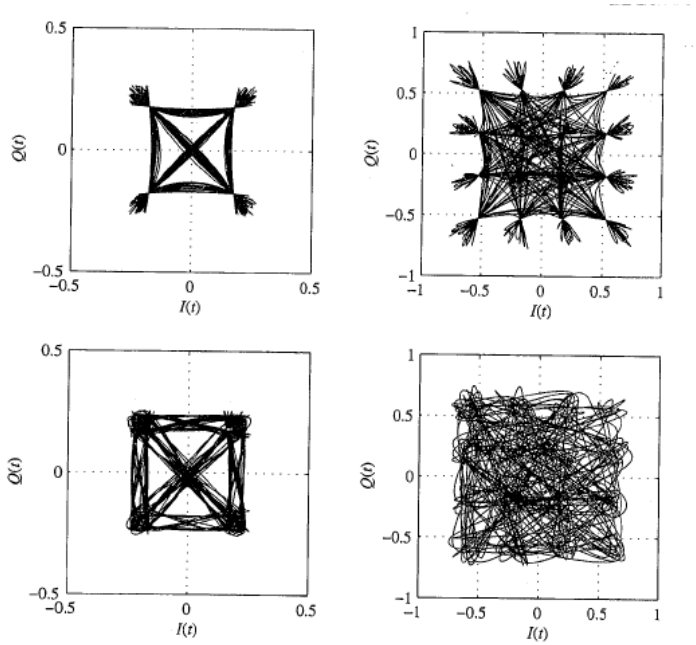


Figure 5.3.14 Phase trajectory plots for QPSK and 16-QAM using the SRRC pulse: (upper left) QPSK with 100% excess bandwidth; (upper right) 16-QAM with 100% excess bandwidth; (lower left) QPSK with 50% excess bandwidth; (lower right) 16-QAM with 50% excess bandwidth.

Figura 3.27: Constelaciones resultantes de usar un pulso SRRC para distintas modulaciones [76].

digital sin capa de enlace. Pero en estas pruebas, cada SDR cumplía únicamente una función: ora como receptor, ora como transmisor. Esto difería del funcionamiento real de nuestro sistema: cada nodo es tanto emisor como receptor. Incluso en una aplicación sencilla de receptor, este debe enviar ACKs. Como se explicó anteriormente, cada nodo recibe entonces tanto los paquetes ajenos como los propios.

Se observó que la ganancia de las tramas propias era muy grande, no sólo al llegar al AGC2, sino también a la salida del mismo. Es decir, entraban muestras de amplitud considerable, bordeando el "1 digital" que es el valor máximo que acepta GNU Radio para las muestras que vienen del sistema operativo (host, en nuestro caso la computadora portátil o el Raspberry PI) [61]. GNU Radio trabaja con muestras digitales que debe luego convertir a valores de tensión para su transmisión a través de las antenas. No se encontró mucha información al respecto de qué conversiones se realizan en este proceso, pero se manejó el criterio de que algo superando el 1 digital propio podría traer problemas [61]. A su vez, el AGC2 aumentaba esta amplitud a los valores máximos de ganancia: alrededor de 15, como se ve en la figura 3.25.

Al enviar un complejo por la antena de transmisión y recibir el mismo complejo por la antena de recepción, prácticamente sin atenuación, el módulo de las muestras que llegan es de casi $\sqrt{1^2 + (1j)^2} = 1{,}42$. El bloque AGC, está intentando

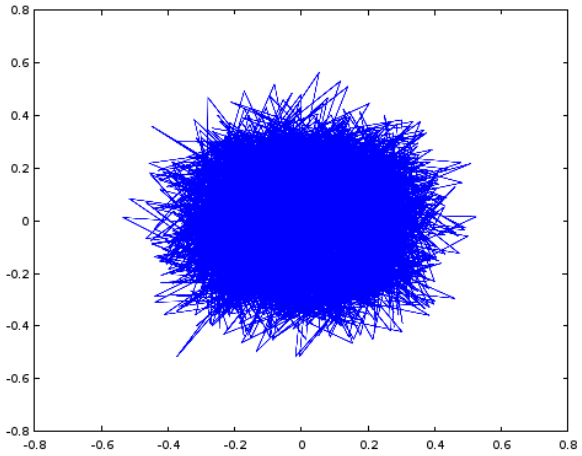


Figura 3.28: Gráfica de las muestras recibidas de un pulso posteriormente a la llegada de un pulso "propio".

llevar a la señal de recepción a los valores de referencia, es decir que maneja la ganancia máxima (de 15) cuando le llega ruido, que llega con baja potencia. Esto es lo que genera que al recibir un "pulso propio" se reciba este con la amplitud de casi 1.4 al entrar al AGC2 y salga con amplitud de alrededor de 15 al salir del mismo. Luego, de nuevo buscando seguir el valor de referencia (fijado en 2), el AGC2 busca bajar este valor, disminuyendo su ganancia. Con este valor de ganancia (disminuida) es que buscará recibir los pulsos ajenos (de potencia de 0.01, por ejemplo). Es decir que recibirá pulsos ajenos y demorará en amplificarlos hasta el valor de referencia, lo que genera dificultad en la recepción. Los tiempos de ataque y decaimiento, como veremos, también influyen en la velocidad con que el AGC2 se ajusta para lidiar con las muestras recibidas. Este bloque se verá con mayor detalle más adelante.

Se consideró la opción de que las ganancias de las antenas disminuyeran, evitando así la saturación por autoescucha. Esto resolvió el problema, pero nos planteó otro desafío: el alcance de la comunicación no superaba unas decenas de centímetros. Para resolver esto se exploró la posibilidad de usar la misma antena en transmisión y recepción. Todos los USRP utilizados (B100, B200 y B200mini) cuentan con dos antenas: una de transmisión y recepción (TXRX), y una de recepción (RX2). La capacidad de multiplexación de los USRP es alta comparada con los valores de procesamiento y los tamaños de pulsos), y esto haría que al enviar un nodo un pulso, no lo escuchara él mismo. Esto sirvió para solucionar parcialmente el problema, pero la ganancia en recepción de la antena TX/RX es baja, por lo que es una solución aplicable a algunos nodos, pero con restricciones en caso de tener una potencia de recepción y transmisión baja en el sistema. En particular, se aplicó esta solución a la estación base, mientras que no se utilizó en los robots móviles, que ya de por sí tienen peor alimentación y entonces baja potencia en recepción y transmisión.

Otra prueba realizada fue aumentando la tasa de envío de tramas. Esto se realizó para ver si podía solucionar en recepción problemas de, por ejemplo, el AGC respecto de los tiempos de convergencia a los valores de referencia. El resultado de la prueba fue que llegaban correctamente todos los datos, lo que nos orientó a seguir concentrándose en el AGC.

El bloque utilizado se llama AGC2. Recordemos que el AGC tiene por finalidad llevar la señal a dos valores de referencia: alto y bajo. El objetivo es aumentar la relación señal a ruido: aunque lleguen la señal y el ruido con valores cercanos, se lleva la señal a un valor alto de referencia y el ruido a un valor bajo de referencia. Esto además genera mayor estabilidad para las siguientes instancias en la demodulación, así como previene transitorios y glitches que pueden ocurrir. El funcionamiento de un controlador de ganancia automático se ilustra en la figura 3.29.

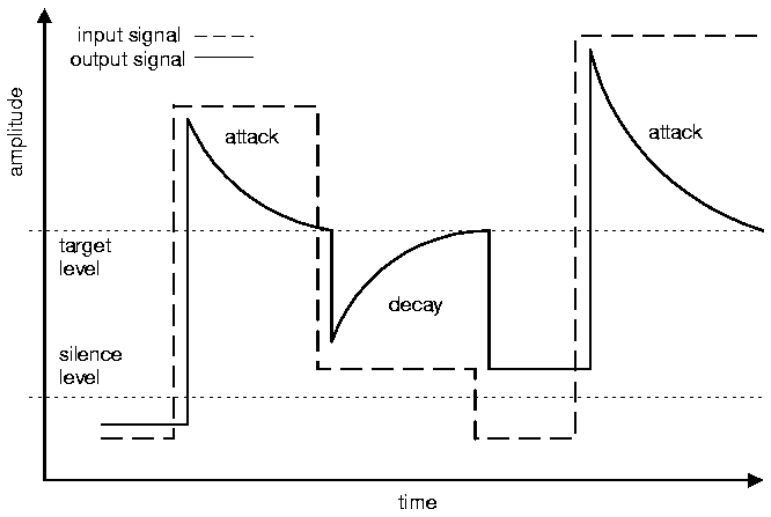


Figura 3.29: Gráfico explicativo del rol que cumplen los parámetros de un controlador automático de ganancia. [80]

Notemos que los parámetros que aparecen en la gráfica son prácticamente los mismos que los del bloque (figura 3.30). Los parámetros del bloque son:

- **Attack Rate:** esta es la tasa de amplificación del valor de las muestras. Cuanto mayor sea este valor, más rápido convergerá al valor de referencia. Es importante que pueda estabilizarse en el valor alto durante la lectura de las muestras correspondientes al encabezado, para tener un valor estable de las muestras correspondientes a los códigos de redundancia cíclica y al resto de la trama.
- **Decay Rate:** esta es la tasa de decaimiento de la amplitud de las muestras. De igual manera que en el Attack Rate, cuanto mayor este valor, más rápido

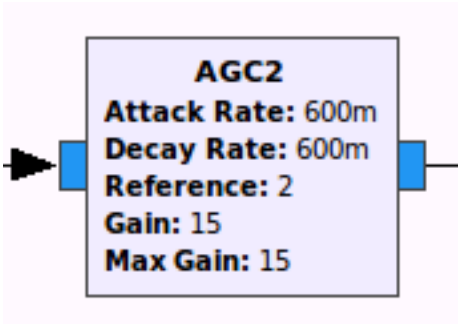


Figura 3.30: Figura del bloque AGC2.

convergerá al valor de referencia. Si no llega rápidamente al valor alto, pueden descartarse tramas por considerarlas de código de redundancia cíclica inválido (mensaje de debbuguing de Invalid CRC que arroja el bloque L1 Deframer). Si se mantiene la potencia en valores muy bajos, puede considerarse ruido y no intentar amplificarse las muestras (por debajo del nivel de silencio marcado en la figura).

- **Reference:** este es el valor de referencia al que queremos llevar la amplitud de las muestras, es decir la amplitud de salida deseada.
- **Initial Gain:** la ganancia inicial con que se trabajará.
- **Maximum Gain:** la ganancia máxima con que se trabajará. No puede ser muy alta, porque sino buscará llevar los niveles de ruido al valor de referencia, y en consecuencia cuando llegue un pulso con mayor potencia, tendremos un sobretiro que impedirá su correcta recepción.

Originalmente los valores de ataque y decaimiento se encontraban en 0.006 y 0.001 respectivamente. Si la diferencia en potencia entre un pulso propio y un pulso ajeno era muy grande, no daba el tiempo al AGC a llegar a los valores de referencia y poder recibir correctamente los pulsos. Para evitar esto, se aumentaron los valores (ambos) a 0.6. Esto logró mejorar el problema de la lectura de los datos aún luego de llegar pulsos propios al sistema. Otra medida que se tomó fue aumentar el encabezado de los paquetes. Este es un encabezado que se agrega en la construcción de los eventos GWN (archivo **gwnutils.py**), y que no contiene información útil. Su objetivo es justamente dar tiempo a la convergencia del AGC y el resto del sistema de sincronismo. Este valor, inicialmente en 160 bytes se cambió, luego de reiteradas pruebas, a un valor de 300 bytes. Ambas modificaciones fueron necesarias, porque el aumento de las tasas mejora la lectura de datos, pero no impide que la primera parte de la trama sea corrompida. A su vez, un encabezado muy grande de bytes inútiles empeora el sistema, ocupando ancho de banda y enlenteciendo la comunicación.

El largo total de nuestros pulsos es de 465 bytes que se agregan con la construcción del paquete. Además se agrega el código de redundancia cíclica, y un

postámbulo, que ocupan unos 35 bytes. Se agrega entonces el encabezado que se mencionaba anteriormente, de 300 bytes. Nótese que este encabezado representa 37.5 % del tamaño de la trama. También puede observarse que nuestra carga útil en general es de unos pocos bytes (**'Hola!'** son 5 bytes), y que el tamaño total de las tramas es de alrededor de 800 bytes. Esto se debe a la orientación de GWN de trabajar hacia eventos con funcionalidades amplias y con versatilidad, en vez de buscar optimizar la velocidad de transmisión de datos, bajo riesgo de generar estructuras insuficientes y con pocas aplicaciones. Se profundizará en este aspecto de las conclusiones.

Para el estudio del funcionamiento del AGC2, se recurrió a una de las ventajas enunciadas originalmente del trabajo con software libre: el código abierto. Se estudió el código del mismo, para poder entenderlo en profundidad. El mismo se encuentra disponible en el repositorio de GNU Radio en GitHub [79].

Estas soluciones permiten el funcionamiento del sistema de comunicación en modo full-duplex (todos los nodos enviando y recibiendo), usando eventos GWN.

Se probó modificar los restantes valores, para ver si tal vez una ganancia máxima mayor podría mejorar la lectura de los datos recibidos, pero no obtuvimos resultados alentadores. En particular, sí es importante notar que esta ganancia máxima fija la salida del AGC, es decir que si llegan datos con potencia muy pequeña no serán correctamente leídos. Esta limitante existe en todo sistema de comunicación, es necesaria una potencia mínima en recepción para la detección de señales. En nuestro caso el valor es de alrededor de 0.008 (valor absoluto del complejo recibido), relativo al 1 digital que vale 1.4142 (un complejo de parte real e imaginaria en 1).

Se podría avanzar en el estudio de los restantes bloques para entender correctamente el porqué de este valor. Sin embargo, estando el proyecto orientado a la implementación del sistema de comunicación, y habiendo otras prioridades a resolver para la culminación del proyecto, queda como pendiente para futuros trabajos, donde la optimización de la capa física y el estudio de las capacidades de los B200 y B200 mini debe profundizarse.

Luego de resueltos los problemas vinculados al intercambio de eventos GWN en modo full-duplex, hubo que trabajar sobre los nodos móviles. Ya algunas de las dificultades se anticipaban por posibles problemas de alimentación. No únicamente por la dificultad de que el Raspberry PI alimentara correctamente al B200 mini para el funcionamiento de GNU Radio, sino también por la disminución en potencia debido a esta alimentación.

Hubo entonces que regular las ganancias del amplificador de RF en los diagramas de flujo de los nodos móviles, llegando prácticamente a los límites de los SDR, más allá de los cuales corre riesgo la integridad del hardware. Según los fabricantes

del SDR B200: *"The receive frontends have 73 dB of available gain; and the transmit frontends have 89.5 dB of available gain. Gain settings are application specific, but it is recommended that users consider using at least half of the available gain to get reasonable dynamic range."* [26].

Depurar el funcionamiento de GNU Radio en los Raspberry PI no fue trivial. Por lo pronto no deseábamos tener elementos que pudieran afectar las medidas por consumo de alimentación. Se descartó entonces la posibilidad de conectar el Raspberry PI a un monitor a través de HDMI, puesto que se observó que el monitor lo alimentaba. También se descartó la utilización de SSH, por consumir corriente del Raspberry PI. Ambos consumos, si bien en general pueden considerarse despreciables, en este caso no lo eran. En particular, el Raspberry PI no reconoce siquiera al periférico (B200) si no tiene buena alimentación.

Lo que se planteó fue entonces trabajar con el relevo de muestras en el PI y su estudio posterior. Por otra parte, al ser las comunicaciones a probar en modo full-duplex, y al encenderse los leds de las antenas al enviar y recibir datos, se pudo identificar cúal era la causa de la falla en la comunicación.

Un problema al que nos enfrentamos fue la capacidad de procesamiento del Raspberry PI. GNU Radio no sólo es exigente en consumo de energía, también en consumo de procesador. Utilizando la opción **top -H** mientras corría GNU Radio con el diagrama de flujos a implementar, se observó que se estaba rozando el límite del procesador. Esto nos era indicado en primera instancia por el LED correspondiente a la antena de recepción, que en vez de estar prendido en continuo, parpadeaba. Para corregir esto, se disminuyó la tasa de muestreo, llevándola de 1 MHz a 200 KHz. También se pasó de trabajar con 5 muestras por símbolo a 3 muestras por símbolo. Esto implicó un cambio de 200K símbolos por segundo a 67k símbolos por segundo. Estas modificaciones podrían introducir problemas para la demodulación de la señal. Tanto la tasa de muestreo como las muestras por símbolo habían sido sospechados de generar errores en la lectura en recepción, y aunque fueran descartados con pruebas en su momento, modificar estos valores exigiría una revisión más profunda en caso de errores. Sin embargo, no hubo que modificar otros parámetros y se mantuvo el alcance la comunicación.

La primera cota que tiene nuestro sistema de comunicación es la potencia de transmisión del USRP B200 mini, que a pesar de estar rozando la ganancia máxima admitida, díficilmente alcanza los 7 metros. Además, la ganancia en recepción aumenta todas las señales que llegan: tanto el ruido como los datos. No conviene aumentar el piso de ruido, no sólo porque no mejora la relación señal a ruido, sino que además la capa MAC puede detectar el canal ocupado cuando esté libre. Sumado a que de ser muy grande la ganancia comienza a distorsionar la salida por efectos no lineales. Una solución posible es la incorporación de un amplificador de ganancia en el transmisor. Otra solución posible es el estudio de la capa física para ver si se puede mejorar el alcance. Para esto se consideró disminuir la frecuencia

de la portadora a la mitad (pasamos de 850 MHz a 400MHz), lo que nos hizo avanzar un poco. También se pueden considerar alternativas en la alimentación para permitir una transmisión con mayor potencia por parte del B200 mini.

Nuestro sistema de comunicación, con los parámetros en los valores elegidos, tiene una capacidad de 133 Kbps. Aunque no es un valor grande, permite cumplir el objetivo planteado inicialmente: se mantiene una comunicación por chat en todo momento del recorrido de los robots móviles. El alcance de nuestro sistema es de alrededor de 6 metros sin repetidor, y de hasta 10 metros con repetidor incorporado.

Para el trabajo en campo, se debe regular la ganancia de la etapa de RF de los SDR entre una cota mínima para la recepción de las señales ajenas y una cota máxima que impida el problema de la recepción de la señal transmitida por el propio nodo. Los resultados obtenidos arrojan que las antenas deben estar entre 40 y 65 dB de ganancia para un B200 en la estación base (parámetros más regulables), y con 75 y 55 dB de potencia en transmisión y recepción respectivamente en los robots móviles (parámetros menos regulables, por estar cerca de los límites del hardware) [26] [61]. Sumado a esto, se debe estimar con particular cuidado los parámetros de ataque y decaimiento del bloque controlador de ganancia automática, así como el tamaño del encabezado de las tramas.

El diagrama de flujos final implementado en cada nodo se puede encontrar en el Apéndice “E”. Estos son prácticamente idénticos entre sí, y la única diferencia con respecto al presentado en el capítulo (figura 3.12) es la utilización de dos bloques de Socket PDU (se utiliza UDP en vez de TCP).

3.11. La medida de la QoS: concepto e implementación

Existen distintas formas de medir lo que se llama calidad de servicio (QoS). Según lo que nos interesa en una comunicación determinada es que elegiremos una u otra de estas métricas.

En redes de computadoras [85], se le dice calidad de servicio a las garantías que una comunicación ofrece. Se distinguen particularmente las referidas a:

- Ancho de banda: la velocidad de transmisión de datos, es muy importante para la descarga de archivos o las conexiones exigentes en velocidad de intercambio de datos
- Retardo: el tiempo de espera para la llegada de datos, por ejemplo una llamada por voz es particularmente sensible a este parámetro
- Fluctuación o jitter: la variación del retardo, si llegan paquetes con tiempo de espera variable puede dificultar por ejemplo la reproducción de audio o

video.

- Confiabilidad: la exigencia de no perder datos en el intercambio, sumamente importante en intercambios de correo o en transferencias de archivos.

Dependiendo de lo que uno necesita del servicio, priorizará alguno(s) de los aspectos mencionados. Por ejemplo, cuando uno realiza una videoconferencia, desea priorizar la comunicación en tiempo real, bajo riesgo de perder calidad en la definición de la imagen o del sonido. Si se trata de intercambiar archivos, desea que no se pierdan datos y la conexión sea estable. En caso de buscar descargarse una película, se deseara que el ancho de banda sea grande, para acelerar la descarga.

Uno de los desafíos planteados para el proyecto era el mantener siempre cierta QoS, que elegimos en la definición del problema vincularla a la pérdida de paquetes. En particular, porque en nuestro esquema de trabajo nos interesa nunca perder conectividad, es decir no quedar incomunicados con los nodos.

Consideramos entonces para la QoS lo que se llama tasa de éxitos: la cantidad de tramas enviadas con éxito frente a la cantidad total de intentos de envío. Si todos los intercambios se realizan con éxito, tendremos que la QoS vale 100 %, mientras que si, por ejemplo, fue necesario repetir todos los envíos una vez, la QoS será de 50 %.

Para medir esto, se incluyó en el script de python de la radio base una función que genera 100 intentos de envío de datos. Esta será estudiada con detalle cuando hablemos de la estación base. Pero en resumidas cuentas, enviamos una ráfaga de datos para observar cuántos de estos envíos fueron exitosos. Ahora, ¿cómo sabemos si los datos que enviamos desde la capa superior son intentos o éxitos?

Esto motivó que en la CSMA se realizaran modificaciones de forma tal de poder medir si los datos fueron enviados, y si fueron envíos exitosos. La solución pasa por generar en la CSMA eventos dirigidos a la capa superior: por cada intento se envía un mensaje de **“Try!”**, y por cada éxito se envía un mensaje de **“Success!”**. Esto permite, contando ambos y dividiendo los **“Success!”** por los **“Try!”**, obtener la medida deseada de la QoS. La medida puede tener un error de 1 %, ya que al cortar la cuenta en el intento 100, podría suceder que el ACK, y por ende el **“Success!”**, no haya llegado.

Por otro lado, cuando el repetidor está en juego, la medida puede verse distorsionada, porque podríamos no estar realizando un sólo intento, sino dos. Si sólo llegaran al receptor en el extremo las tramas que reenvía el repetidor, y sólo llegaran al emisor los ACKs generados por el repetidor, no tendríamos problema. Pero como la comunicación en general falla primero por uno de los sentidos del camino, y no por ambos, es altamente probable que exista en un sentido una superposición de intentos. Esto debería mejorar la calidad de servicio, ya que en la estación base

contaremos como un intento lo que finalmente podrían ser dos.

Al mismo tiempo, la multiplicación de mensajes (por cada mensaje tengo uno más) genera mayor ocupación del canal, y entra en juego la CSMA/CA con su función de sensado. Es decir que puede empeorar la medida de la QoS cuando el repetidor es innecesario (cuando el Explorador está en la zona de buena QoS).

El repetidor tiene su mayor utilidad si las puntas están alejadas, obteniendo buenos resultados: estando a 7 metros el Explorador, su QoS pasa de 23 % a 65 %, por ejemplo.

Otra posibilidad que surgió como forma de medir la QoS era utilizar la relación señal a ruido, o SNR (Signal to Noise Ratio). Cuanto mayor sea ésta, mejor es la QoS, y menor será la probabilidad de perder datos. Para esto se podría usar el mismo bloque que sensa el canal y devuelve una medida del módulo de la potencia al cuadrado en el mismo. Sin embargo, el bloque AGC dificulta la utilización de esta medida. El AGC, recordemos, toma las muestras que le llegan y las amplifica o atenúa según estemos en un pulso o recibiendo ruido, buscando homogeneizar su amplitud en torno a un valor de referencia. La variación tal vez suave de las potencias de señal y ruido no se ve directamente reflejada en el éxito o fracaso de la comunicación, que es mucho más binaria por efecto del AGC.

Esto también afecta nuestra forma de medir la QoS, que es muy alta cuando los nodos están cerca, y que pasada una cierta distancia desciende muy abruptamente. Sin embargo, y volviendo a argumentos ya esgrimidos, preferimos tomar una medida de lo que realmente nos interesa, como es no perder conectividad, frente a estimar la SNR y a partir de ahí buscar su vínculo con la tasa de éxitos.

Finalmente, al no estar optimizada nuestra capa física, e introducir errores propios, la medida de la QoS a partir de la SNR (una medida de capa física) no es conveniente, ya que nosotros buscamos la QoS en capa de enlace.

3.12. Conclusiones parciales: Implementación real de una capa MAC con GWN sobre GNU Radio

En este primer capítulo abordamos el diseño de la capa de enlace de datos de nuestro sistema de comunicación entre nodos móviles.

Esto implicó el desarrollo de la subcapa MAC, así como el ajuste del protocolo ARQ de la capa control lógico. También conllevó a la implementación para depuración de un chat, con posibilidad de comunicación entre más de dos nodos. Se realizó un conjunto de pruebas que permitieran garantizar el funcionamiento de la capa de enlace de datos.

3.12. Conclusiones parciales: Implementación real de una capa MAC con GWN sobre GNU Radio

Finalmente, se probó el sistema de comunicación en la implementación final del proyecto: en el conjunto de estación base y robots móviles. El pasaje a hardware, en particular el trabajo con los USRP B200 y B200 mini, resultó un desafío, pero a través del estudio de la capa física y de la construcción de las tramas se lograron solucionar las dificultades encontradas.

En resumidas cuentas, contamos con una capa de enlace de datos que incluye una CSMA/CA que corre sobre el protocolo Stop and Wait, y con una capa física con modulación digital QPSK y control de errores por redundancia cíclica, además de corrección de errores de fase, frecuencia y sincronización temporal.

Se obtiene con los robots en funcionamiento autónomo (alimentados por las baterías propias) un alcance entre la estación base y el explorador de hasta 6 metros. Incluyendo el repetidor, esta distancia llega a aumentar hasta los 10 metros aproximadamente. Estos valores son en las condiciones planteadas de mantener una QoS de mínima, en este caso consideramos un valor de umbral de mínimo de 25 % de éxito en la transmisión de datos.

Si bien la implementación pudo realizarse y el sistema de comunicación funciona, hay varias restricciones a enfrentarse para mejorar el alcance y pensar en posibles aplicaciones de utilidad. En este sentido, quedan unas cuantas puertas abiertas, que representan posibles caminos a recorrer para mejorar este experimento de combinación entre GWN y robótica móvil.

Entre ellas, resaltamos que nuestras principales restricciones son a nivel de hardware, aunque creemos que se puede avanzar en varios sentidos. En particular, destacamos la posibilidad de explorar la fuerte correlación existente entre la alimentación y el alcance de los SDR. Queda todavía camino por recorrer en el estudio de las antenas más adecuadas, y si es posible incluir amplificadores para mejorar el alcance del sistema (a riesgo de aumentar el consumo energético). También se puede estudiar con mayor profundidad la relación entre lo que GNU Radio considera el 1 digital y la potencia de salida de las antenas. En este sentido un estudio de la relación entre su alimentación y la potencia con que termina transmitiéndose desde el SDR podría dar respuestas. Una idea planteada por el grupo de proyecto, que no llegó a probarse por falta de tiempo, fue desarrollar un sistema realimentado, donde la ganancia de los amplificadores de RF pudiera autorregularse cuando se detectara una disminución en la potencia de las señales recibidas. Lo realizado respecto del sensado del medio (el intercambio de valores de variables entre bloques diferentes) sienta buenas bases para pensar soluciones en este sentido.

Otra línea de trabajo podría ser el estudio de modulaciones que pudieran implicar un menor consumo, aún si fuera a costa de perder velocidad, como ser BPSK. O mejor aún, usar OFDM, que evitaría la aparición del engorroso bloque AGC2. Sin embargo, la exploración de esquemas que puedan tener tasas de intercambio mayores a las logradas es necesaria si se desea aproximarse a una aplicación útil

del sistema. Para esto se puede buscar ensayar diagramas de flujo que sean menos exigentes para el procesador que el planteado. De la misma forma, la capa MAC y GWN pueden ser aún optimizados, disminuyendo sus tiempos de ejecución: por ejemplo en la conversión de los bloques de lenguaje Python a C++, un trabajo actualmente en curso por parte del Grupo ARTES. Varios parámetros de la capa MAC pueden ser ajustados según el interés de futuros desarrolladores: el número de reintentos o los tiempos de espera son parte de debates corrientes [12], y se puede pensar en implementar el intercambio de paquetes para inicio de la comunicación (Request to Send, Clear to Send). La inclusión de mensajes de control para el acceso al medio como RTS/CTS, en un esquema en que además del sensado del canal se solicita permiso para transmitir, no ha sido explorada en esta ocasión, y podría implicar mejoras en el ancho de banda. También se puede pensar en la fragmentación de datos con carga útil mayor a la que un evento GWN posibilita, por ejemplo para probar en el intercambio de archivos. El tamaño máximo de las tramas con que trabajamos es de alrededor de 5000 bytes. Podría ser interesante que el robot Explorador transmitiera imágenes o video de lo que percibe.

Tampoco está cerrado el debate respecto a la construcción de los eventos GWN, qué largo debe tener el encabezado, o cómo se deben incluir las direcciones. Algunas ideas se han planteado en el sentido de buscar reproducir un modelo más similar a las diferentes capas con que cuentan las redes de datos, para lo que puede ser necesario revisar los intercambios entre capas que hoy se realizan (por ejemplo en la asignación de direcciones, o en la construcción del repetidor). Por lo pronto, hoy ocupan una porción demasiado grande de la trama, respecto de la carga útil. En el protocolo 802.11 (wifi), los encabezados de la MAC son de 32 bytes de la trama (que puede contener hasta 8000 bytes en total) [83] [96]. En nuestro esquema actual, el encabezado agregado que no contiene información es un tercio del largo en bytes del mensaje. Todos los encabezados juntos son casi la totalidad del mensaje (800 bytes), ya que nuestras cargas útiles son muy pequeñas. Esto también se da porque necesitamos mensajes con cierto tamaño mínimo. De contar con mensajes de mayor tamaño (por ejemplo transmitiendo archivos) podría reducirse el encabezado, analizando los límites del AGC. Avanzar en reducir el tamaño del encabezado podría mejorar sustancialmente la velocidad útil de la comunicación.

Sin embargo, se entiende que el objetivo de GWN para el que fue diseñado debe proporcionar un conjunto de herramientas con flexibilidad, claridad, facilidad en su uso. Esto se enfrenta hasta cierto punto a la búsqueda de optimizar las velocidades de comunicación, o la eficiencia del sistema. Por ejemplo, se podría pensar en que la construcción de los eventos se hiciera uniendo todos los datos (direcciones, número de secuencia, carga útil) en un string, con cierto cuidado de poder recuperarlos en recepción, y de esta forma ahorrar gran parte del encabezado. Sin embargo esto quita la posibilidad de agregar un diccionario que lleve la etiqueta de si el evento debe ser transmitido por broadcast, o crear etiquetas nuevas si se desea, una función que como “usuarios” de GWN nos fue más que útil (y necesaria).

3.12. Conclusiones parciales: Implementación real de una capa MAC con GWN sobre GNU Radio

Con equipos de mayor capacidad de procesamiento, que el Uruguay a través del grupo ARTES está en proceso de adquisición, no sería poco interesante estudiar cuáles son los cuellos de botella en hardware que tiene nuestro sistema. Pruebas sobre tiempos de procesamiento en la computadora, en las FPGA y los conversores, y la necesaria optimización del tiempo de ejecución de las funciones de capa MAC podrían dar lugar a aplicaciones realmente competitivas. En este sentido existen trabajos que avanzan sobre modelos híbridos en donde parte del procesamiento se pasa a realizar en los DSP o FPGA, y parte se mantiene a nivel de usuario (PC). Si bien esto se aleja de las premisas filosóficas de SDR, ha mostrado buenos resultados [57].

En el proyecto se cumplió con el objetivo de implementar un sistema de comunicación con CSMA/CA sobre robots autónomos móviles, obteniendo resultados moderados en cuanto al alcance en distancia. Se acumuló experiencia respecto del trato con eventos GWN y modulación digital, lo que permite realimentar el trabajo del grupo ARTES que generó el marco de trabajo de GWN en GNU Radio para el desarrollo de este proyecto. Se colaboró de esta manera en la realización de un paper que fue presentado en la conferencia WinnComm2017 sobre GWN, donde la capa MAC con CSMA/CA fue presentada como ejemplo de aplicación [34]. En este sentido se probaron librerías y conceptos que pusieron a prueba los fundamentos de GWN, completando la librería con bloques propios que esperamos puedan ser de utilidad en el porvenir, en su estado actual o modificados. Se encontraron soluciones para los diferentes problemas de implementación sobre hardware, en particular conociendo sutilezas de manejo de los SDR, que sistematizadas pueden prevenir futuros dolores de cabeza.

Esperamos entonces, como decía el poeta Roque Dalton, que los conocimientos adquiridos en nuestra experiencia sean “como una aspirina del tamaño del Sol” para futuros interesados en desarrollar aplicaciones con GWN y SDR.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Navegación de los robots

Existen aplicaciones donde se necesita establecer una comunicación inalámbrica con determinados parámetros de calidad de servicio en algún punto (o conjunto de puntos) donde previamente no existía comunicación, sin que el usuario deba desplazarse de su posición original. Son ejemplos de estas, el espionaje militar y el sensado de terrenos en tiempo real, entre otras. En estos casos se pretende recabar información de distintos tipos de sensores (video, temperatura o humedad) que se encuentren ubicados en el punto deseado y transmitirlos en una estación base alejada.

Una respuesta para estas situaciones es el desarrollo de robots móviles, los cuales pueden ser o no autónomos. Estos últimos permiten mejorar parámetros del sistema para optimizar la comunicación. Cuando la distancia entre punto de origen y destino es mayor al alcance de las antenas, se requieren nodos intermedios repetidores de señal que permitan aumentar el área de cobertura del sistema. Estos nodos también son robots móviles que cooperan para maximizar la calidad de servicio en el destino.

Para la realización del proyecto se desarrollaron dos robots móviles, coordinados mediante lógica cooperativa, capaces de dirigirse de forma autónoma, a una posición de destino indicada por latitud y longitud. Para ello se los dotó de ruedas que les permitieran transportarse, sensores para el posicionamiento, un microcontrolador que se encarga de integrar la información de sus periféricos y de controlar a los actuadores, baterías que le permitan independencia de la red eléctrica y una computadora y un SDR para las comunicaciones.

Los robots resuelven su posicionamiento mediante sucesivas iteraciones. Ejecutan un loop en el que inicialmente esperan una posición de destino válida recibida inalámbricamente mediante GNU Radio. Si se les envía una posición, los robots calibran sus sensores y se orientan en dirección a la posición de destino. Se desplazan 3 metros[1], se detienen y vuelven a enviar su posición.

[1]Esta distancia está determinada por la incertidumbre de los sensores usados para el

En este capítulo consideraremos a los componentes mecánicos y electrónicos encargados del desplazamiento autónomo de los robots.

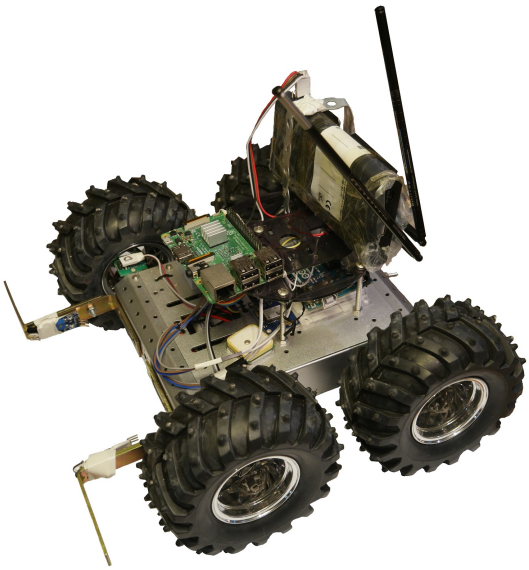


Figura 4.1: Robot Explorador

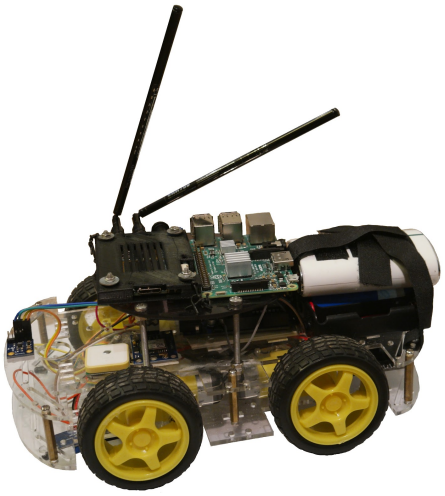


Figura 4.2: Robot Repetidor

posicionamiento y porque la calidad de servicio de la comunicación decae abruptamente en distancias del orden de los 3 metros.

4.1. Componentes de los robots

Los robots tienen cinco componentes fundamentales:

- **Estructura mecánica**, que le permita avanzar en línea recta y girar sobre su propio eje, soportando toda la carga del robot.

  Consta de:

  - **chasis**, que sostiene las partes y les da rigidez,
  - **cuatro ruedas con cuatro motores**, encargadas de desplazar al robot de una ubicación a otra,

- **Sistema sensorial**, que le permita conocer su posición actual, orientarse hacia su destino, evitar choques y medir su desplazamiento efectivo.

  Consta de:

  - **sensor GPS GY-NEO6MV2** [55], utilizado para el posicionamiento del robot en exteriores,
  - **compás digital CJ-M49** [37], utilizado para orientar al robot en la dirección de la posición de destino,
  - **dos sensores IR** [5], ubicados en el frente que detectan obstáculos,
  - **dos encoders para ruedas** [77], colocados en las ruedas frontales para medir su avance efectivo,

- **Sistema de control**, como centro encargado de gestionar la navegación automática de los robots.

  Consta de:

  - **microcontrolador Arduino UNO [75]**, encargado de la integración de la información de todos los sensores y del comando de los motores,
  - **driver L298N para motores** [25], interfaz necesaria para controlar y alimentar a los motores desde Arduino,

- **Sistema de comunicación**[2], encargado de establecer la comunicación inalámbrica entre robots y estación base.

  Consta de:

  - **Rasbperry Pi 3 Modelo B [64]**, que envía y recibe posiciones desde Arduino y GNU Radio,

[2]Se aborda en detalle en el Capítulo 3. Sistema de comunicación

    - **USRP B200 mini con antenas [26]**, le brinda conectividad inalámbrica a los robots utilizando GNU Radio,

- **Alimentación**, que les brinda a los robot autonomía de la red eléctrica.

  Consta de:

    - **pilas recargables**, que alimentan al driver de motores,
    - **baterías recargables USB**, que alimentan al Arduino, Rasbperry Pi, GPS, magnetómetro, sensores IR, encoders y USRP B200 mini.

- **cables USB y conectores varios**

Ambos robots cuentan con el mismo equipamiento electrónico pero difieren en su chasis, ruedas y alimentación. Estas diferencias se deben al presupuesto y tiempo del que se disponía para la ejecución del proyecto y no responden a elecciones específicas de diseño. El robot Repetidor fue el primero en comprarse, con el cual se pretendía estudiar las prestaciones y limitaciones de ese chasis. Luego de tener en claro los parámetros que se debían modificar para mejorar la navegación de los robots, se compró el segundo, que es el Explorador. Es por esto que uno es más grande y tiene más potencia en sus motores que el otro.

En las imágenes 4.3 y 4.4 se muestra la ubicación de los componentes en el robot **Explorador**.

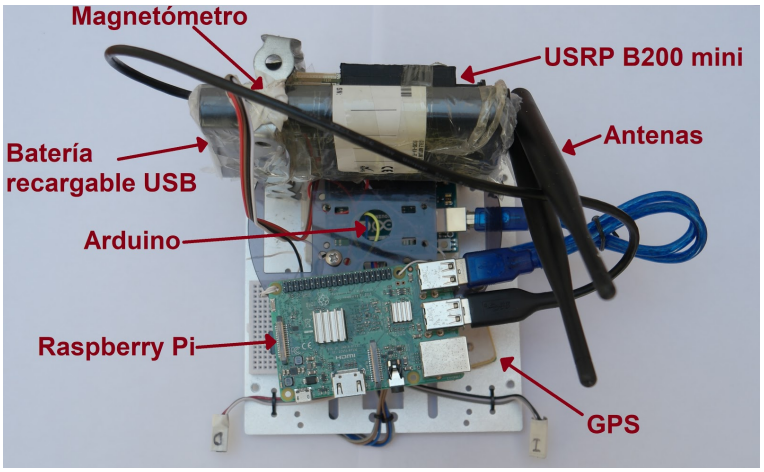


Figura 4.3: Componentes del Explorador (visto de arriba)

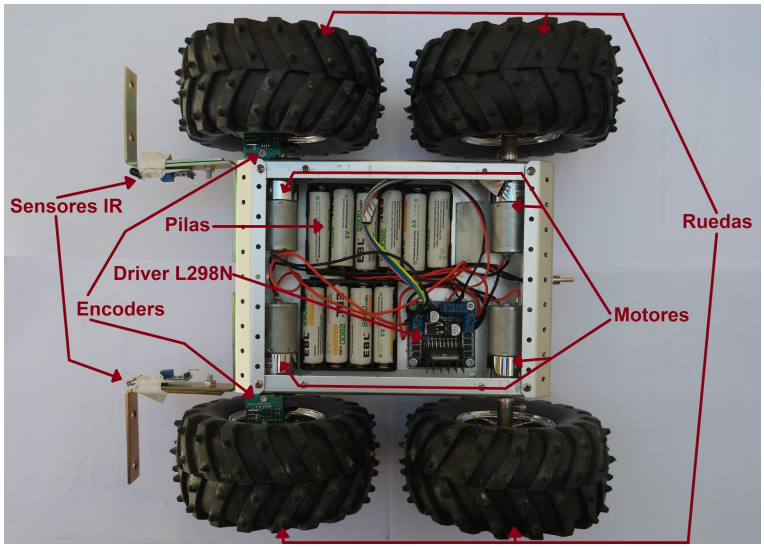


Figura 4.4: Componentes del Explorador (visto de adentro)

En las imágenes 4.5 y 4.6 se muestra la ubicación de los componentes en el robot **Repetidor**.

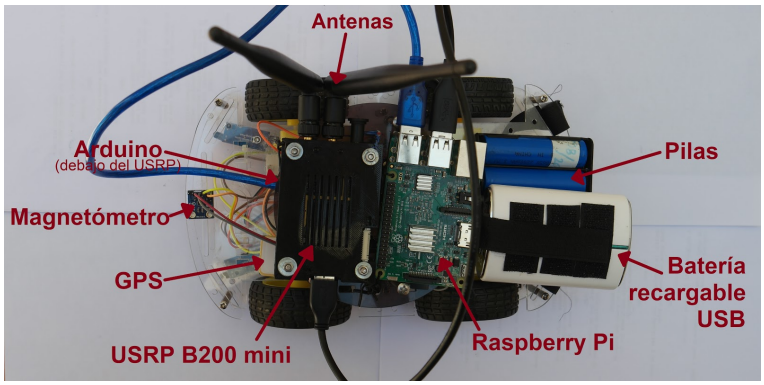


Figura 4.5: Componentes del Repetidor (visto de arriba)

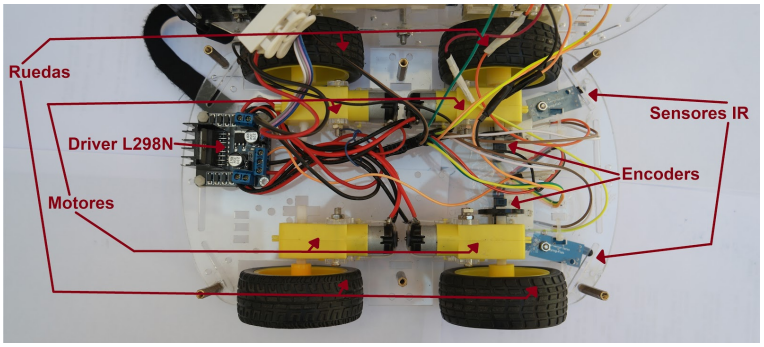


Figura 4.6: Componentes del Repetidor (visto de adentro)

4.2. Estructura mecánica

Desde un comienzo del proyecto se pensó en trabajar con robots de cuatro ruedas, cada una con un motor, para que fuesen capaces de soportar todo el peso que deben transportar. Los motores se conectan en paralelo y se ubican en los laterales de los robots.

Cuando se comenzó a trabajar con el Repetidor, se constató que sus ruedas deslizaban sobre la superficie y bajo algunas condiciones de carga, tenía dificultades para hacer girar a sus motores. Por esto se consideró que para la segunda compra se debía buscar un robot con más potencia en sus motores y mayor fricción de sus ruedas. Como se verá más adelante, fue un error pensar que es conveniente aumentar su rugosidad.

4.2.1. Dimensiones del chasis, motores y peso

Robot Repetidor

- Masa total del robot: 1,24 kg
- Masa de chasis, ruedas y motores: 0,52 kg
- Material del chasis: Acrílico (6 mm de espesor)
- Tamaño:
- Motores [19]:
  - Tensión de operación: 8V
  - Torque de salida: 107mN.m
  - Corriente de carga: 250mA (550mA MAX)

Robot Explorador

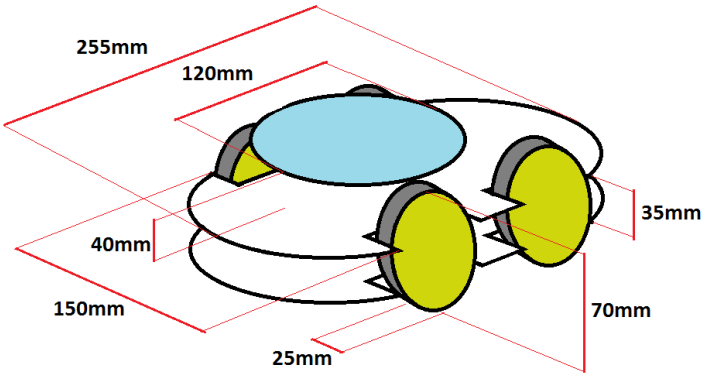


Figura 4.7: Dimensiones del Repetidor

Figura 4.8: Motor del Repetidor. Foto obtenida de página web del fabricante [19].

- Masa total del robot: 2,71 kg
- Masa de chasis, ruedas y motores: 1,28 kg
- Material del chasis: Estructura central de aluminio y tercer piso de acrílico
- Tamaño:

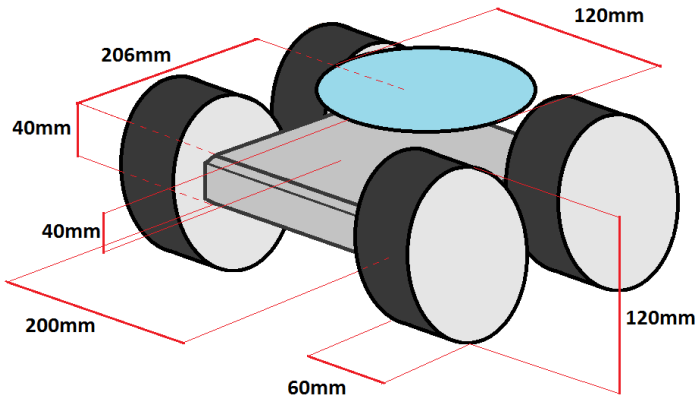


Figura 4.9: Dimensiones del Explorador

- Motores [6]:

- Tensión de operación: 8V
- Torque de salida: 158mN.m
- Corriente de carga: 370mA (760mA MAX)

Figura 4.10: Motor del Explorador. Foto obtenida de la página del vendedor [6]

4.2.2. Problemas de diseño

Al comienzo del proyecto se propuso un diseño que no contemplaba todos los factores implicados en el movimiento, lo que determinó que el robot no fuese capaz de girar. Fue necesario identificar el problema para estudiar diferentes soluciones[3] y finalmente elegir alguna de ellas.

A continuación se propone estudiar, desde el punto de vista mecánico, las variables involucradas en el giro de los robots. Se plantean las condiciones que debe cumplir el torque de los motores para que se pueda producir el giro. Con esto se pretende comprender el motivo por el que los robots no giran y facilitar la toma de decisiones para solucionar el problema.

Diagrama de Newton

Considérese que el robot tiene masa neta $Mnet$ con el centro de masa en su punto medio. Las ruedas son de radio r. El coeficiente de rozamiento entre la superficie y las ruedas es μ. Las ruedas tocan el suelo en su punto medio. El versor e es paralelo a la recta que une el punto medio de la rueda con el centro de masa del robot. El versor x es paralelo al eje transversal del robot. El ángulo formado entre los versores e y $\mathbf{x}$ es θ. Los motores ejercen un torque T.

Si se quiere que el robot gire sobre su propio eje, se encienden los motores de la derecha hacia atrás y los motores de la izquierda hacia adelante. La condición para que el robot no gire partiendo del reposo es que la fuerza neta y el momento, sean nulos sobre cada parte de él; es decir, cada rueda por separado y el chasis del robot en sí.

De esta manera, los motores ejercen un par sobre el eje de la rueda y aparece una fricción ejercida por el piso que actúa en el punto de contacto de la rueda con el piso. El par sobre la rueda derecha delantera es de módulo T y apunta según x. La componente según y de la fricción cancela al par motor T, que actúa sobre la rueda, de forma de garantizar el equilibrio de la misma.

[3]Más información sobre las diferentes soluciones se puede encontrar en el apéndice D.1

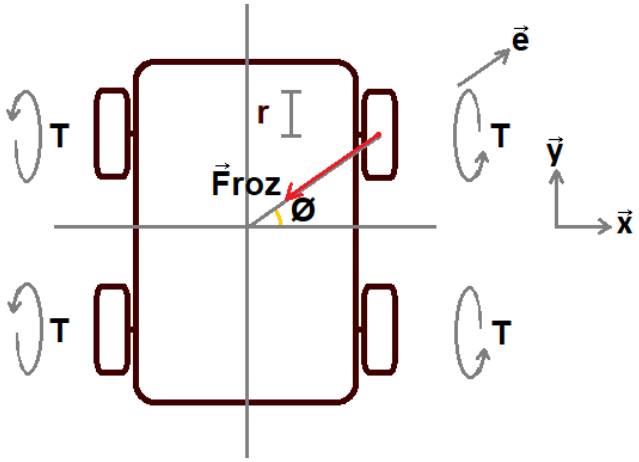


Figura 4.11: Diagrama de Newton

Es decir que, como el par motor aplicado en la rueda es compensado por el de la fricción, se tiene que:

$$T = r.Froz_y \tag{4.1}$$

En cada punto de apoyo de las ruedas, el piso ejerce una reacción normal de valor:

$$N = Mnet.g/4 \tag{4.2}$$

Del equilibrio del conjunto del robot (ruedas más chasis), surge que las fuerzas externas (como la fricción en cada rueda, fuerzas normales y peso) deben dar momento nulo visto, por ejemplo, desde el centro de masas del robot. Como el momento del peso es cero, el de todas las fuerzas normales iguales también es cero, y como el rol de cada rueda es equivalente por simetría de la distribución de masa, no queda otra que el momento de la fricción neta de cada rueda sea nulo visto desde el centro de masas del robot. Esto se cumple si la fuerza es según el versor e, es decir, si está en la línea que une el punto de aplicación con la proyección del centro de masa en el piso. por lo que se tiene que:

$$Froz.sen(\emptyset) = T/r \tag{4.3}$$

Por otro lado, la fuerza de rozamiento cumple la ley de Coulomb:

$$|Froz| \leq \mu|N| \tag{4.4}$$

De las ecuaciones (4.2), (4.3) y (4.4) resulta que el robot no se mueve si el par motor cumple que:

$$T \leq \frac{1}{4}\mu.r.Mnet.g.sen(\emptyset) \tag{4.5}$$

De la inecuación (4.5) se obtiene una cota mínima para que el torque ejercido por los motores haga girar al robot, para el caso en que el centro de masa se encuentra en el centro del robot. Se observa que el movimiento depende de: T (torque de los motores), μ (coeficiente de rozamiento entre las ruedas y la superficie), $Mnet$ (masa total del robot), Ø (ángulo que forman las ruedas con el centro del robot) y r (radio de las ruedas).

Observar que el caso en que el ángulo Ø = 0, que es el de un robot con sólo dos ruedas, no hay par máximo que evite el movimiento y el robot seguro arranca. Por esto los robots de dos ruedas son más eficientes para girar alrededor de su centro de masa.

El primer diseño de los robots tenía el centro de masa en el medio de cada robot. Esto no tenía en cuenta que, cuando se quiere girar, los motores deben ejercer una fuerza extra para hacer deslizar las ruedas del robot. Si los motores no son lo suficientemente potentes como para vencer ese rozamiento, el robot no girará.

4.2.3. Solución propuesta

Teniendo en cuenta el modelo anterior y habiendo explicado los factores que determinan que el robot no pueda girar, se proponen las soluciones detalladas a continuación.

Se redujo la fricción entre las ruedas y la superficie. Para ello se consiguió una locación que cumpla con los requerimientos que plantean los demás componentes del proyecto (GPS, antenas)[4], pero que a su vez, tenga una superficie con un coeficiente de rugosidad tal que le permita a las ruedas deslizar. También se redujo la rugosidad de las ruedas del robot Explorador cubriéndolas con un plástico liso.

Se redujo la masa del robot Repetidor. Se buscó que todos los componentes agregados a la estructura del robot fuesen lo más livianos posibles. En el robot Repetidor, cuyos motores son menos potentes (menor torque máximo) y su giro estaba más comprometido, se cambiaron las pilas de NiMh por pilas de Litio, que son más livianas. También se intentó utilizar a las pilas como única fuente de alimentación, pero esto no fue posible y finalmente tuvimos que agregar una batería recargable USB para alimentar a los componentes electrónicos.

Se cambió el centro de masa del robot Repetidor. Basados en los robots de dos ruedas, que no tienen problema para girar sobre su propio eje de forma eficiente, se le cambió al robot Repetidor el centro de masa para que su apoyo sea,

[4]Las limitaciones del alcance de las antenas se discuten en la sección 3.10 y las del GPS en 4.5.4

fundamentalmente, sobre las ruedas traseras. De esta manera, las ruedas delanteras se encuentran menos apoyadas sobre la superficie y tienen menos rozamiento que las traseras. Cuando el robot gira, las ruedas delanteras deslizan y las traseras funcionan como apoyo.

4.2.4. Conclusiones

Interesados en diseñar robots que fuesen capaces de trasladarse con un peso importante, se eligió aumentar la potencia colocando 4 motores, pero en esta elección no se tuvo en cuenta los movimientos sobre su propio eje. Durante la ejecución del proyecto nos enfrentamos a la dificultad de combinar estas dos variables. Identificado el problema, se exploraron las soluciones existentes en el mercado, que se detallan en el apéndice D.1. Luego de evaluar los costos económicos y temporales de su incorporación, debimos descartarlas. Se propone una solución poco eficiente desde el punto de vista energético y que plantea limitantes en la topología del terreno sobre el que se desplaza, pero que le permite al robot avanzar y girar.

4.3. Alimentación

Los robots cuentan con dos fuentes de alimentación independientes. Un banco de pilas recargables para los motores y una batería USB de Litio recargable para los componentes electrónicos.

4.3.1. Conexionado y consumo

Como se muestra en el diagrama 4.12 y en la figura 4.13, la batería USB alimenta directamente al Raspberry Pi. Éste, a su vez, alimenta al Arduino y al USRP B200 mediante el cable USB. Arduino tiene una salida de 5V desde dónde se alimentan los sensores, excepto el GPS que se alimenta desde otra salida de 3,5V del Arduino.

Las pilas se encuentran conectadas a la entrada de tensión del driver L298N. En el caso del robot Repetidor, se usan 3 pilas de Litio. Para el Explorador se usan 10 pilas de níquel-metal hidruro (NiMh).

El driver de motores y Arduino necesitan compartir tierra para referenciar a las señales de control.

- **Robot Repetidor:**

Pilas: En el robot Repetidor, las pilas son de Litio de 3,7V nominales y se conectan tres en serie, por lo que se obtiene una fuente de 11,1 V. Cada una tiene una capacidad de 4.200 mAh. Estas pilas alimentan a los motores que tienen un consumo nominal de 250 mA cada uno. Se midió que el consumo de los motores en vacío

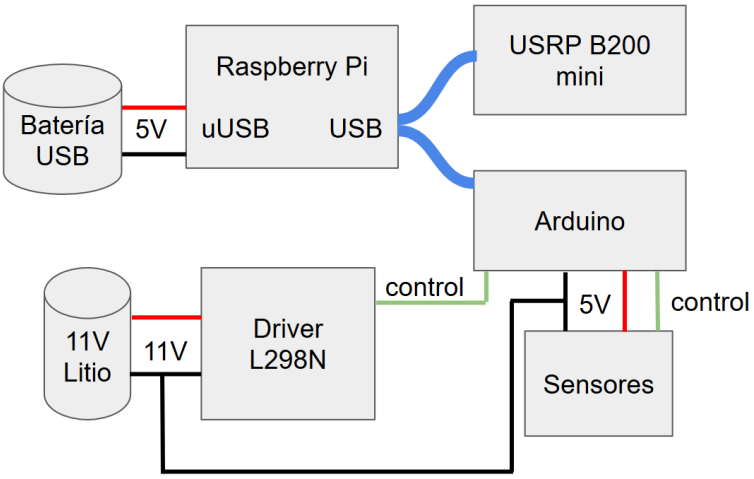


Figura 4.12: Diagrama definitivo de alimentación del robot Repetidor

es igual al nominal. Cuando estos están bloqueados, consumen 550mA cada uno. A su vez, la electrónica del driver de motores L298N consume 36mA. Por lo tanto, las pilas deben entregar una corriente mínima de 36mA y una máxima de 2.236mA.

Batería USB: Puede entregar 5V, una corriente máxima de 2100mA y tiene una capacidad de 4000mAh. Alimenta al USRP B200 mini que consume 750mA en promedio, con máximos de 850mA, al Arduino y todos sus periféricos que consumen 100mA y al Raspberry Pi que consume 250mA. Esto significa que el consumo de todos los dispositivos conectados a la batería recargable USB es de 1200mA.

- **Robot Explorador:**

Pilas: En el robot Explorador, las pilas son de NiMh de 1,2V nominales y se conectan 10 en serie, por lo que se obtiene una fuente de 12V. Cada una tiene una capacidad de 2.800 mAh. Los motores alimentados por estas pilas tienen un consumo nominal de 370 mA cada uno. Cuando están bloqueados, consumen 760mA cada uno. El driver de motores L298N consume 36mA. En este robot, las pilas deben entregar una corriente máxima de 3.076mA.

Batería USB: Puede entregar 5V, a una corriente máxima de 2000mA y tiene una capacidad de 5000mAh. El consumo de los componentes electrónicos es igual que en caso del robot Repetidor.

Según los experimentos realizados, los robot tienen una autonomía de 2 horas. Las baterías USB son las primeras en agotarse ya que entregan en el orden de 1000mA desde que se encienden los robot. La capacidad indicada por su fabricante es un poco mayor a la real y por eso se agotan antes de lo que deberían según esas indicaciones. Por otro lado, las pilas prácticamente no se agotan ya que los

motores están la mayor parte del tiempo apagados.

En el diagrama 4.13 se detalla el **conexionado** de cada uno de los componentes, que es común en ambos robots.

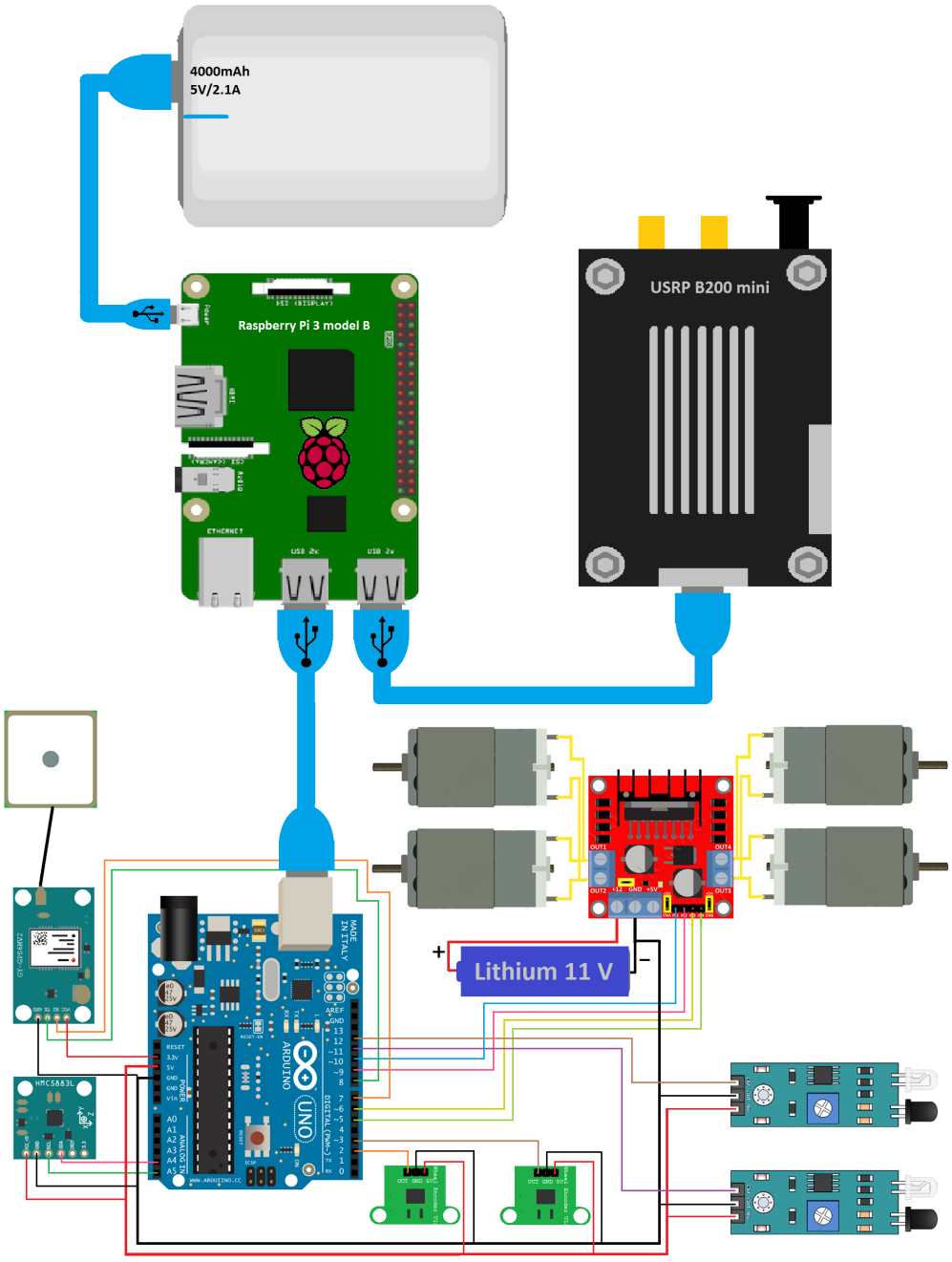


Figura 4.13: Conexionado

4.3.2. Dificultades encontradas

En el diseño original de los robots se pensaba utilizar una sola fuente de alimentación para todos los componentes. Además, como se mencionó en el capítulo 4.2 Estructura Mecánica, se quiso reducir el peso de los robots para que pudiesen girar sobre su propio eje y para eso resultaba fundamental eliminar una de las dos fuentes de alimentación. En función de estos requerimientos, en un principio, intentamos armar los robots con una sola fuente pero los resultados fueron insatisfactorios.

Pilas de NiMh como única fuente de alimentación

Al comienzo del proyecto se compraron 16 pilas de NiMh recargables de la marca EBL [10]. Luego de identificado el problema con el peso del robot, se comenzó a trabajar en el desarrollo de una alimentación única con este tipo de pilas. La primera meta consistió en generar una fuente de 5V con estas pilas, que permitiera alimentar directamente al Raspberry Pi y éste, a su vez, al USRP B200 mini.

Se conectaron 6 pilas de NiMh en serie, lo que totaliza una tensión de 7,2V. Se utilizó este número de pilas por ser las que utiliza el Explorador, que es el más comprometido de los dos. Se conectaron las pilas a la entrada de tensión de un conversor DC/DC que coloca 5V a la salida. Se creó un cable microUSB especial que permitiese conectar los 5V del DC/DC a la entrada de tensión del Raspberry Pi. A éste se le conectó el USRP B200 mini y se ejecutó el programa de reconocimiento del periférico.

El USRP B200 mini es un dispositivo que requiere de una alimentación muy estable y levemente por encima de los 5V para poder funcionar sin interrupciones. Se observó que este esquema de conexión no permitía que el USRP encendiera. Se colocó un voltímetro a la salida de las pilas y se observó que la tensión variaba abruptamente para distintas condiciones de carga. Estas pilas mostraron no ser capaces de mantener la tensión estable.

Al no poder generar una fuente con pilas de NiMh que alimente al Raspberry Pi y al USRP B200 mini, se descartó esta iniciativa.

Pilas de Litio como única fuente de alimentación

Las baterías de Litio son una de las tecnologías más avanzadas, que últimamente está sufriendo una gran expansión gracias a tecnologías móviles de otra índoles como ordenadores portátiles y teléfonos móviles. La diferencia de rendimiento de unos tipos de pilas respecto a otros es notable, así como su tamaño y precio.

Se decidió continuar trabajando con baterías de Litio prestadas por el grupo de TALLERINE Robótico de la FING. Éstas mostraron ser capaces de mantener la tensión constante para distintos tipos de carga, además de tener más capacidad

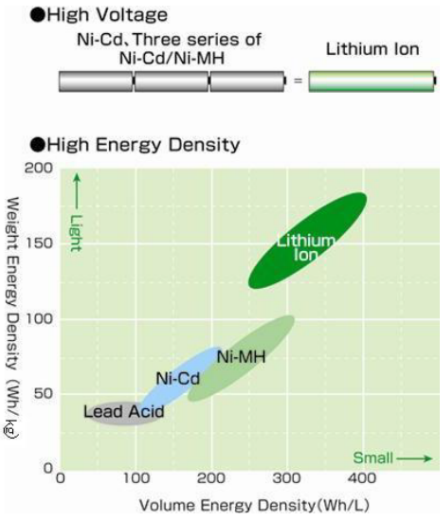


Figura 4.14: Comparación en rendimiento de distintas tecnologías de baterías. Figura obtenida de proyecto de fin de carrera de Vicent Mayans [78].

de almacenamiento y ser sensiblemente menos pesadas que las de NiMh.

El diseño de la batería consistió en 3 de estas pilas conectadas en serie y luego a un conversor DC/DC, al igual que en el caso anterior. Los resultados fueron positivos, ya que se consiguió encender al USRP B200 mini y enviar y recibir información. Luego se intentó integrar la batería, el conversor DC/DC, el Raspberry Pi y el USRP B200 mini al resto del robot, como se muestra en la figura 4.15.

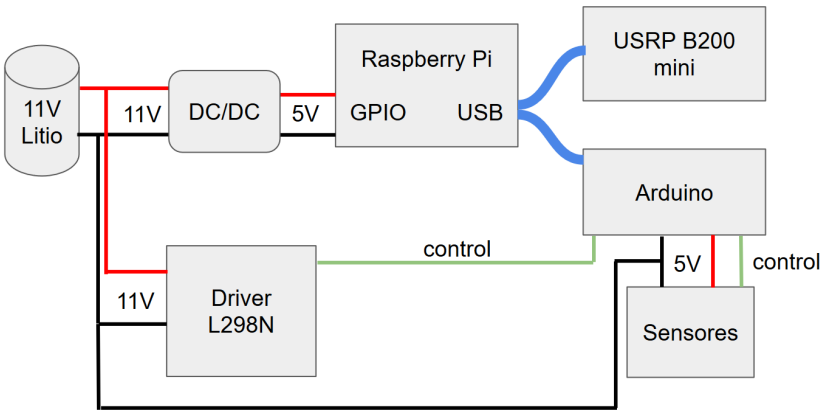


Figura 4.15: Diagrama de conexionado para una única fuente de alimentación de litio

Se observó que con éste esquema de conexión, el microcontrolador Arduino no lograba funcionar correctamente. En este diseño no se tiene en cuenta que, cuando todo está encendido y funcionando, el DC/DC eleva la tensión de la tierra en 0,69V y el conjunto Rasberry Pi + Arduino la elevan en 0,12V. Esto genera una

diferencia de 0,81V entre el polo negativo de las pilas y la tierra de Arduino, como se ve en la figura 4.16. Estos dos puntos se deben cortocircuitar para referenciar las señales de control del driver de motores L298N, pero al hacer eso se genera una corriente de circulación que descarga las pilas e inutiliza al Arduino.

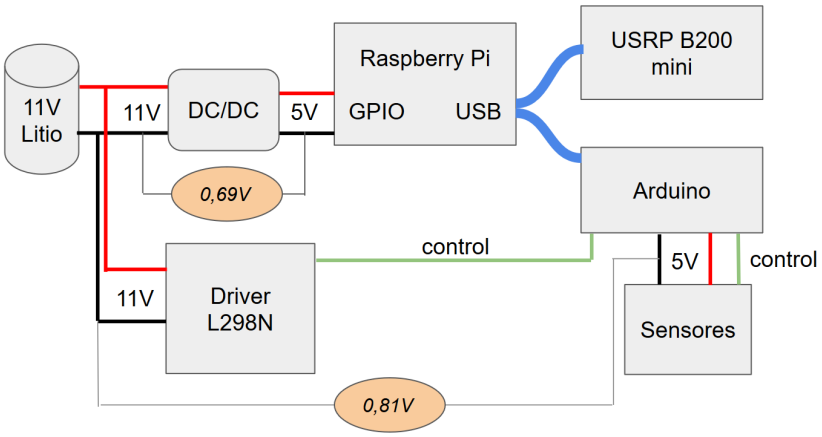


Figura 4.16: Voltajes medidos en sistema con una única batería de litio (sin el cable de tierra entre Arduino y el driver de motores)

Por estos motivos se debió incorporar una segunda batería que permita desconectar la tierra de las pilas a la tierra del driver de motores. Se buscó que ésta fuese de 5V a la salida para eliminar al conversor DC/DC y se resolvió trabajar con una batería recargable USB.

4.4. Sistema de control

4.4.1. Driver de motores

Se utiliza un driver dual L298N para controlar a los motores. Se lo conecta a la alimentación proveniente directamente de las pilas, a Arduino, mediante las señales de control, y a los motores.

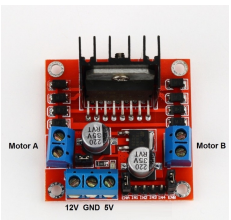


Figura 4.17: Driver dual L298N. Foto tomada de página web del vendedor [25].

Tabla 4.1: Señales de control para manejo de motores

(IN1,3 , IN2,4)	(OFF , OFF)	(ON , OFF)	(OFF , ON)
Acción	Motores no giran	Motores giran	Motores giro invertido

El driver tiene cuatro entradas digitales: IN1, IN2 para controlar el Motor A; IN3 y IN4 para el Motor B. En la siguiente tabla se muestra la lógica con la que se deben controlar.

Si se colocan 5V entre las entradas digitales, los motores giran con la máxima potencia. Si se quiere que las ruedas funcionen con la mitad de la potencia, se debería bajar la tensión a la mitad. Como Arduino solamente tiene salidas digitales que pueden tomar valores de 0V o 5V, se debe utilizar la Modulación por Ancho de Pulso (Pulse Width Modulation - PWM) [92]. Esta es una técnica que alterna el valor de las salidas digitales para simular valores entre 0V y 5V, cambiando la proporción de tiempo que una señal está ON respecto al tiempo que está OFF. El tiempo que la señal está en ON se le conoce como el "ancho del pulso". Los motores tienen inercia mecánica al girar, lo que hace que parezcan estar siempre encendidos cuando se utiliza PWM. No todos los pines de salida de Arduino soportan PWM, por lo que para su utilización se debe chequear la hoja de datos.

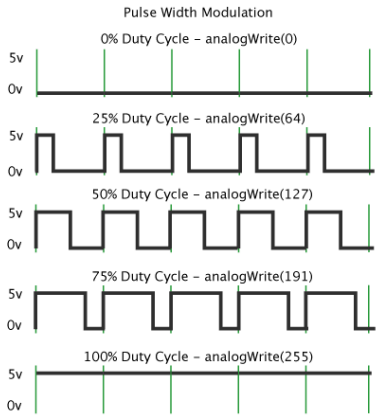


Figura 4.18: PWM - Pulse Width Modulation. Figura obtenida de página web de Arduino [92]

En el robot Repetidor, cada motor consume nominalmente 250mA y con el rotor bloqueado llega a 550mA. Esto implica un consumo nominal de 500mA y uno máximo de 1100mA por canal del driver de motores.

En el robot Explorador, cada motor consume 370mA en condiciones nominales y puede llegar a 760mA. Esto implica un consumo de nominal de 740mA y uno máximo de 1520mA por canal del driver de motores.

Estos valores de corriente son adecuados ya que el L298N puede entregar hasta 2000mA por canal y una tensión de 8V. [25]

4.4.2. Arduino

Arduino es una compañía que se especializa en el desarrollo y producción de microcontroladores de hardware libre. Sus productos son conocidos por su fácil utilización, su bajo costo y por ser de código abierto tanto el hardware como el software. Existe una gran comunidad de usuarios que constantemente generan contenido, y de hecho en este proyecto se utilizan librerías de terceros. Como además, en el equipo de proyecto ya se contaba con experiencia previa de haber trabajado con este microcontrolador, se encontró que esta era una elección adecuada.

El microcontrolador es utilizado para recibir la información de los distintos sensores periféricos y procesarla, con el fin de controlar a los motores para que trasladen al robot a la posición indicada por Raspberry Pi. Para esto, también establece una comunicación serial bidireccional con Raspberry Pi.

Existen varios modelos de Arduino, los cuáles varían fundamentalmente en su capacidad de procesamiento, cantidad de entradas y salidas e interfaces de comunicación. En este proyecto se eligió trabajar con Arduino UNO, un microcontrolador basado en ATmega328P. Tiene 14 entradas/salidas digitales (de los cuales 6 pueden ser utilizados para salidas PWM), 6 entradas analógicas, un reloj de cristal de cuarzo de 16MHz, conector USB, conector Power Jack, una entrada ICSP y un botón para resetear.

Figura 4.19: Arduino UNO. Tomada de la página de Arduino [75]

Programación

El programa tiene un loop general que se ejecuta cada vez que el robot avanza 3mts. El robot debe detenerse, y no puede avanzar en línea recta desde su origen a su destino, porque la medida de calidad de servicio de GNU Radio requiere que el robot permanezca estático durante cierto tiempo. Cada vez que el robot se detiene, realiza las mediciones de QoS y actualiza su posición para reorientarse y seguir avanzando. Quisimos maximizar la distancia de avance para minimizar la cantidad

de iteraciones que el robot debe hacer hasta llegar a su destino. No pudimos aumentar más la distancia ya que la QoS decrece abruptamente en algunos sectores y queremos que siempre haya una QoS mínima en todo instante. Por otra parte, el GPS tiene una incertidumbre de 3mts, por lo que no tenía sentido hacerlo avanzar una distancia menor a esa.

La programación subida a Arduino utiliza 21992 bytes (68 %) del espacio de almacenamiento de programa siendo el máximo de 32256 bytes. Las variables globales usan 1807 bytes (88 %) de la memoria dinámica, dejando 241 bytes para las variables locales. El máximo es de 2048 bytes.

Bibliotecas

Para la integración del sensor de campo magnético y el GPS se debió integrar bibliotecas de terceros conseguidas en Internet. Estas son:

- "TinyGPS", de Mikal Hart [89]. Esta biblioteca es utilizada para establecer la comunicación con el GPS, pedirle una medida de latitud y longitud, calcular el ángulo existente entre la recta que une dos posiciones distintas y la recta que apunta al norte geográfico, y para calcular la distancia que hay entre dos posiciones.

- "I2Cdev" de Jeff Rowberg [39], se utiliza para establecer la comunicación I2C con el sensor de campo magnético.

- "compass" de Omer Ikram ul Haq [95], se utiliza para calibrar el magnetómetro y para obtener la medida del ángulo entre el eje 'X' del sensor y el norte magnético.

También se utilizan bibliotecas ya existentes en el entorno de Arduino como son:

- "Arduino": contiene definiciones de funciones y constantes matemáticas.

- "Wire": permite la comunicación I2C

- "SoftwareSerial": permite comunicaciones seriales utilizando la UART

Se crearon desde cero dos bibliotecas:

- "ubicacion"

- "motores"

"ubicacion"

Esta biblioteca es la que incluye las funciones encargadas de establecer la comunicación con el magnetómetro y el GPS, del tratamiento de las medidas que de ellos se obtienen y funciones que permitan la comunicación capas arriba con Raspberry Pi.

Muchas de las funciones de esta biblioteca hacen uso fundamental de funciones definidas en "TinyGPS", "I2Cdev" y "compass", pero se redefinieron nuevamente para hacer el tratamiento correspondiente de las medidas obtenidas y el manejo correcto de las variables globales.

Se preveen funciones para:

- inicializar los sensores
- calibrar el magnetómetro
- obtener la posición del GPS disminuyendo el error al promediar múltiplos de 35 medidas
- obtener el ángulo de orientación del robot respecto del Norte
- calcular el ángulo que debe girar el robot para ir a una posición deseada utilizando la información de su posición actual obtenida del GPS
- idem al anterior pero sin utilizar el GPS
- calcular la distancia en centímetros que separa al robot de la posición deseada
- enviar la posición actual del robot al Raspberry Pi e información de si llegó a la posición deseada o aún está en tránsito, y recibir la posición deseada actualizada e información sobre la calidad de la señal.

A continuación se describe el protocolo diseñado para la comunicación entre Arduino y Raspberry Pi. Cuando el algoritmo en Arduino lo entiende necesario, toma una medida de posición del GPS y calcula la distancia entre ésta y la posición deseada. En caso de que la distancia sea mayor al error del GPS, considera que aún no llegó y coloca un cero en el puerto serial; en caso contrario coloca un uno. Seguido de esto coloca un string de 8 caracteres donde cada carácter es un dígito de la latitud actual del robot. Luego coloca otro string de largo 8 con la longitud. Para finalizar coloca un símbolo de fin de mensaje '%'. El algoritmo aguarda hasta un máximo de 2 segundos para que el Raspberry Pi le conteste con un mensaje de similares características al que recibió. En el primer char que recibe Arduino se informa el estado de la calidad de señal: cero significa mala calidad y uno significa buena. Luego recibe un string de largo 8 con la latitud y a continuación otro con la longitud de destino más el símbolo de fin de mensaje.

- **Ejemplo de mensaje enviado:**

[0][3][4][9][0][1][3][5][9][5][6][1][5][5][2][0][8][%]

0: indica que el robot no llegó (1 indicaría que el robot sí llegó)
-34,901359: es la latitud actual en grados decimales, que se asumen negativos
-56,155208: es la longitud actual en grados decimales

- **Ejemplo de mensaje recibido:**

[1][3][4][9][1][9][0][1][6][5][6][1][6][5][9][5][2][%]

1: indica que la señal del B200 es buena (0 indicaría que es mala y el robot no debe avanzar)
-34,919016: es la latitud destino en grados decimales
-56,165952: es la longitud destino en grados decimales

"motores"

En esta biblioteca se intenta agrupar todas las funciones que involucran directamente el manejo del driver de motores. Al comienzo se definen varias constantes obtenidas experimentalmente. Algunas son velocidades de giro de los motores que funcionan por PWM (Pulse-Width-Modulation), otras son constantes por las que se multiplica la cuenta llevada a cabo por los encoders para estimar la distancia recorrida en línea recta o el ángulo de giro. Existe un juego distinto de constantes para cada uno de los robots debido a sus diferentes dimensiones constructivas.

Aquí se encuentran funciones que permiten al robot:

- inicializar las salidas del microcontrolador para manejo de motores
- avanzar en línea recta
- retroceder en línea recta
- parar
- girar a la izquierda a una velocidad específica
- girar a la izquierda a máxima velocidad un cierto ángulo
- girar a la derecha a una velocidad específica
- girar a la derecha a máxima velocidad un cierto ángulo
- rutina de atención a interrupción de encoder izquierdo
- rutina de atención a interrupción de encoder derecho

- algoritmo para esquivar obstáculos, que detiene al robot si los sensores IR detectan un obstáculo, retrocede 0,4m, gira 90º en la dirección de su destino y avanza 0,8m
- algoritmo de orientación del robot que incluye dos métodos

El primer método con el que se orienta el robot comienza leyendo el ángulo entre el eje 'X' del magnetómetro y la recta que une la posición actual del robot con su destino y los representa entre [-180º, +180º]. Si el ángulo es negativo, invoca a la función de giro hacia la izquierda y le pasa como parámetro los grados que debe girar. Lo mismo en caso que sean positivos pero con la función de giro a la derecha. Realiza este procedimiento hasta un máximo de 10 veces, luego de lo cual determina que el método no le permite orientarse.

El segundo método de orientación es esencialmente pensado para ser utilizado en el robot Explorador que tiene menos resolución en los encoders de sus ruedas y demostró mayores dificultades para orientarse con el primer método. El algoritmo comienza igual que en el caso anterior, leyendo el ángulo hacia donde se encuentra orientado, para determinar a qué lado debe girar. Luego los motores comienzan a girar en el sentido elegido a la mínima velocidad posible, tal que una velocidad menor no permitiría que los motores rompan el rozamiento estático. Mientras se encuentra en movimiento, se realizan lecturas del ángulo del magnetómetro. Una baja velocidad de giro, permite una mayor cantidad de lecturas del sensor mientras el robot se encuentra en movimiento. Cuando el ángulo es igual a cero, el robot se detiene. Si bien en teoría el algoritmo podría converger en el primer intento, el robot tiene inercia mecánica al moverse, que determina que el robot no pueda detenerse en la dirección deseada, pasándose unos pocos grados.

Para resolver este problema se incorporaron dos tipos de soluciones. Por un lado, se estimó que la inercia mecánica determina que el robot continue girando hasta 5º luego de detectar que el ángulo de orientación es 0º. Incorporando esta corrección, compensa el efecto de la inercia en la mayoría de los casos.

A pesar de esta solución, en algunos casos se vio que el robot no lograba converger al ángulo 0º, por lo que se incorporó una segunda condición de parada para cuando el robot está girando. Se agregó un tiempo de salida (timeout), que comienza siendo tal, que le permite al robot girar poco más de 180º en la primera iteración. En las sucesivas iteraciones, el timeout decrece linealmente, hasta que en la última, es casi cero. Esto le permite al robot ir poco a poco convergiendo al ángulo deseado.

El algoritmo global de orientación ejecuta el primer método, pero si no funciona prueba con el segundo y si éste tampoco logra orientar al robot intenta recalibrar el magnetómetro y probar nuevamente con cada uno de los métodos. El robot no se da por vencido hasta quedar orientado y poder avanzar.

Loop principal

Luego de que se inicializan los sensores y se obtiene la primera medida del GPS, el programa se detiene durante dos minutos para que el sensor mejore su precisión. Al cabo de este tiempo, comienza la ejecución del loop principal.

Una vez dentro de él, lo primero es enviar la posición actual del robot a Raspberry Pi utilizando la comunicación serial y luego esperar hasta que éste conteste con una posición de destino. Si no contesta al cabo de 2 segundos, repite el procedimiento de enviar su posición y esperar. En el momento en que Arduino recibe una posición de destino válida (por válida entendemos que debe estar a una distancia menor a 2km del robot), el programa chequea que la calidad de la señal de la comunicación con la radiobase sea buena y que la distancia entre las posiciones sea mayor al error del GPS. En caso que esto se cumpla, se calibra el magnetómetro y se orienta al robot. Si no se logró orientar con ninguno de los dos métodos anteriormente mencionados, se calibra nuevamente el magnetómetro y se ejecuta el algoritmo de orientación. Cuando el robot se logra orientar hacia su destino, avanza 3mts hacia adelante y se detiene.

Cuando el robot termina de avanzar, repite el procedimiento desde el comienzo. El diagrama 4.20 ilustra lo que se acaba de describir.

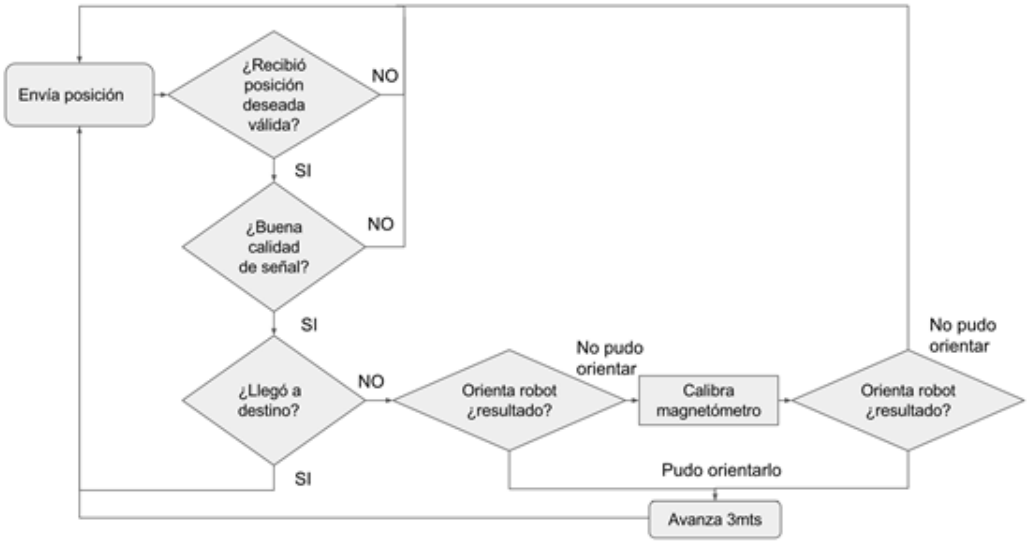


Figura 4.20: Diagrama de flujo del programa principal que corre Arduino

Tres comentarios:

- La calibración del magnetómetro se realiza la primera vez que se ingresa al loop principal desde que se encendió el sistema. Luego no se vuelve a calibrar

a menos que el robot no se pueda orientar y considere que debe recalibrar.

- El robot avanza 3mts en línea recta mientras los sensores IR no detecten un obstáculo. En caso que esto suceda, lo esquiva de acuerdo al algoritmo explicado más arriba y vuelve a ejecutar el loop principal desde el envío de su posición.

- Como forma de reducir el error se realiza el promedio de las medidas de GPS. Éste debe considerar todas las medidas obtenidas con el receptor estático. Cada vez que el robot se desplaza, el promedio comienza desde cero. Por otro lado, Arduino le envía su posición a Raspberry Pi luego de obtener 35 medidas nuevas de GPS y promediarlas todas con el promedio acumulado. Si éste es cero (porque en la itereación anterior el robot se desplazó) sólamente envía el promedio de 35 medias. En los otros casos, se envía el promedio acumulado de múltiplos de 35.

4.5. Sistema sensorial

4.5.1. Magnetómetro

Para orientar a los robots a su destino, se eligió utilizar un sensor de campo magnético modelo HMC5883l. Considerando que en el lugar de trabajo, el campo magnético predominante es el de la Tierra, éste sirve como referencia para calcular el ángulo entre el robot y el Norte geográfico. Además, con el GPS se determina el ángulo entre la posición de origen y destino con respecto al Norte geográfico, y combinando esta información, se determina la dirección en la que se debe orientar el robot.

A continuación se presenta al instrumento, se describe su funcionamiento y se detallan las consideraciones que se deben tener para utilizarlo en aplicaciones de este índole.

Funcionamiento

El magnetómetro HMC5883l es un compás de 3 ejes diseñado para medir con precisión los pequeños campos magnéticos aplicados sobre cada eje. Su conversor ADC de 12 bit le permite tener un rango de escala de 8 Gauss y una resolución de hasta 5 miliGauss, que es equivalente a una resolución en grados de entre 1º y 2º. Utiliza comunicación I2C.

Existen varios modos de funcionamiento, pero el que se utiliza en el proyecto es el “Single Measurement”. Esto implica que el sensor toma una medida cuando

se le solicita y luego vuelve a estar en "Sleep Mode". [37]

Suponiendo que el sensor se encuentra con el eje "Z" orientado según la normal, se utilizan las medidas de campo en el eje "X" y el "Y"(m_x y m_y respectivamente) para calcular el ángulo con respecto al norte magnético, como se muestra en la ecuación (4.6).

$$angulo = \frac{360}{2} arctan(\frac{m_y}{m_x}) \tag{4.6}$$

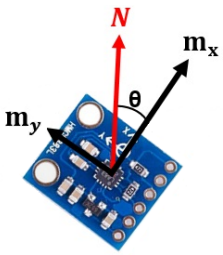


Figura 4.21: Componentes del campo magnético N (norte) en el eje X y en el Y. Figura obtenida del sitio Naylamp Mechatronics [38].

El sensor mide campo magnético y se pretende utilizar para encontrar el Norte geográfico, que es distinto del magnético. Para lograrlo se debe realizar una corrección de 10,58 grados que corresponde con la diferencia que hay en Montevideo entre los dos Nortes.

Calibración

Como una primera aproximación al funcionamiento del sensor se probó la biblioteca "HMC588l" de LoveElectronics. Ésta establece la comunicación utilizando I2C y realiza una lectura de los registros del encapsulado que contienen un valor proporcional al campo magnético existente en cada eje. A la constante de proporcionalidad se la denomina ganancia y puede ser distinta en cada eje. Esta biblioteca utiliza una ganancia única por defecto. De la fórmula (4.6) se ve que, si lo que se quiere es únicamente encontrar el Norte magnético, no importa qué ganancias se apliquen, ya que se buscará que ángulo $= 0$ y eso sólo ocurre cuando $m_y = 0$. Si por el contrario se quiere medir el ángulo, esta librería no demuestra un correcto funcionamiento.

Por otro lado, el robot sobre el que va montado el magnetómetro tiene varios componentes que generan significativos campos magnéticos, como por ejemplo los motores. Si consideramos que estos campos son constantes, entonces el campo variable es el de la tierra más los del entorno (como los generados por los cables de baja tensión) en función del ángulo de orientación del robot. Se realizó medidas de

campo magnético con el eje X apuntando en una dirección y luego en la dirección opuesta y lo que se observó fue que la media de las medidas era distinta de cero, lo que implica la existencia de un offset magnético en el robot.

Estas dificultades justifican que, previa adquisición de medidas, se deba calibrar el sensor. La librería "compass" (Omer Ikram ul Haq, 2014) [95] propone un método de calibración que hace uso del modo "Self Test" del que dispone el HMC588l. Cuando se activa este modo se toman dos medidas: en la primera se mide el valor de campo existente en cada eje, luego para la segunda medida se aplica un campo nominal conocido y se vuelve a medir para dar como resultado la resta entre las dos medidas. Esto permite comparar el valor obtenido con el esperado (que debería ser 1.1 Gauss aprox.) y así ajustar la ganancia en cada eje. Luego el algoritmo solicita que se haga girar al robot en círculos durante medio minuto mientras registra las medidas. Una vez finalizado, obtiene el offset como el promedio entre el valor máximo y el valor mínimo en cada eje.

En la figura 4.22 se ilustra el problema de la calibración del sensor. Cabe destacar que este procedimiento tiene sentido ya que se ajustan las ganancias antes de calcular el offset. Si esto no se hiciera, la distribución de las medidas que se observa en la figura se parecería más a una elipse.

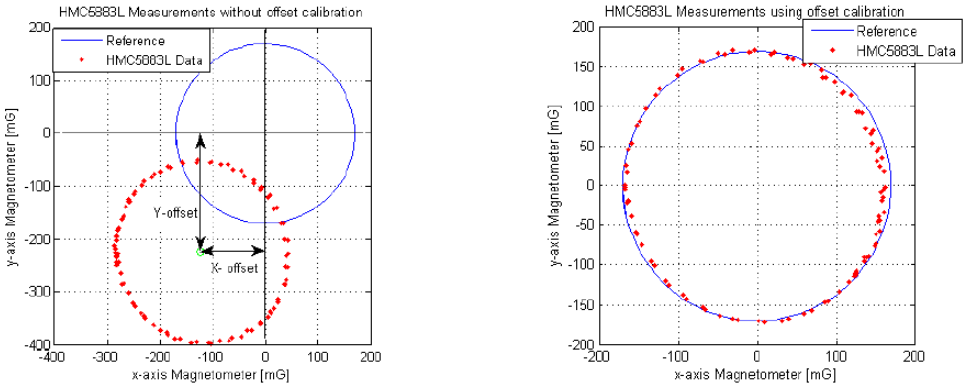


Figura 4.22: Comparación entre un HMC5883l sin calibrar y otro calibrado. Figura obtenida de guía de calibración de Omer Ikram ul Haq [95].

Inducción magnética de los motores

El robot tiene distintos componentes que generan campos magnéticos. En particular, los motores de las ruedas, generan un potente campo magnético cuando se encuentran en funcionamiento.

Originalmente se colocó al magnetómetro cerca de los motores y el robot demostró tener dificultades para orientarse a su destino. Si el robot no está orientado, comienza a girar en la dirección correspondiente mientras mide el ángulo. Cuando

el ángulo se hace cero o cambia de signo, el robot se detiene. Por tanto, las medidas de campo magnético son simultáneas al arranque de los motores, que es cuando éstos consumen más corriente y, por tanto, generan mayor campo magnético.

En la siguiente captura de pantalla se muestra el ángulo medido por el magnetómetro mientras los motores están apagados y las medidas que registra luego de que se encienden. Se puede observar que la medida de ángulo cambia abruptamente al momento que los motores comienzan a arrancar, y rápidamente se vuelve a estabilizar[5].

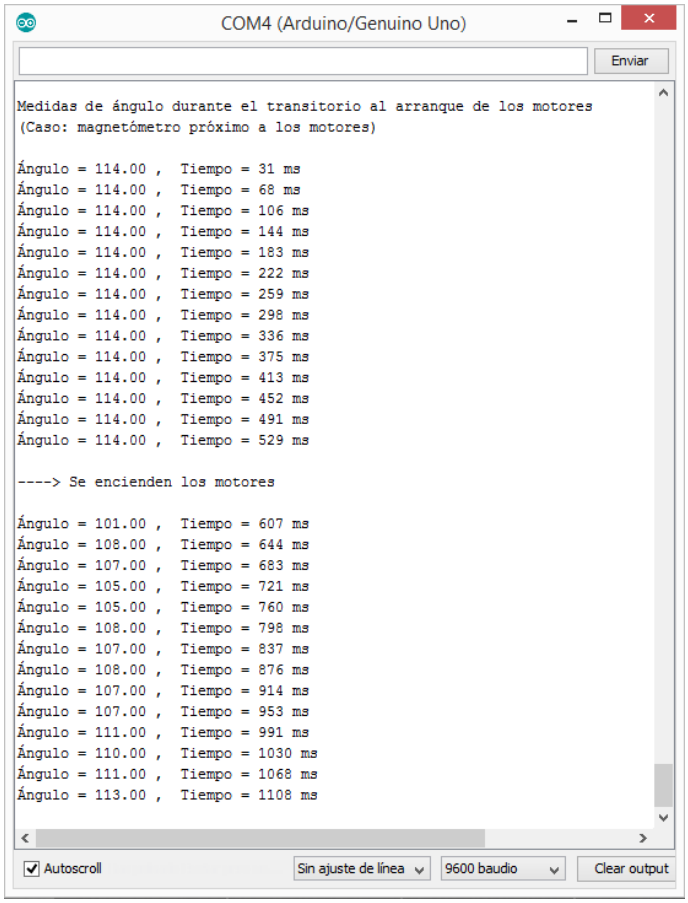


Figura 4.23: Ángulo obtenido mientras los motores están apagados y luego encendidos

La solución consistió en colocar el sensor en una ubicación que no se vea afec-

[5] Este fenómeno es significativo en los casos en que el robot se encuentra, por ejemplo, girado 7 grados con respecto a su destino y comienza a girar para orientarse. En ese mismo momento que se encendieron los motores pero el robot no alcanzó a girar, el ángulo registrado por el magnetómetro es de -6 grados (los motores introducen un error de 13 grados por lo general), por lo que el robot considera que giró demás y debe frenar, a pesar de que en realidad sigue estando a 7 grados.

tada por el campo de los motores.

4.5.2. Encoders

Los encoders ópticos, representados en la figura 4.24, son unos sensor que se utilizan como una de las formas de medir el avance de cada rueda del robot. A los motores se les aplica una tensión en bornes para hacerlos girar. La velocidad de giro no solo depende de la tensión aplicada, sino que también, en nuestro caso que deslizan, depende de la fricción que tengan las ruedas con la superficie. Como una forma de tener mayor control sobre el giro de las ruedas, se coloca sobre su mismo eje un círculo de plástico con hendiduras radiales equiespaciadas. El sensor dispone de un emisor de luz infrarroja y un receptor enfrentados por los que se hace girar el cilindro de plástico, de forma que se permita el pasaje de luz intermitentemente a través de las hendiduras. Cuando pasa luz desde el emisor al receptor, el sensor coloca en la salida digital un uno, y cuando la luz se interrumpe, coloca un cero.

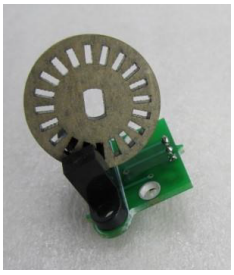


Figura 4.24: Encoder óptico. Figura obtenida de la página del ofertante [77].

Se colocó un sensor en cada una de las ruedas frontales de los robots. Estos se conectaron a las entradas de interrupciones de Arduino Uno. Éste modelo sólo prevé dos entradas digitales que se pueden configurar para interrupciones. Cuando se quiere utilizar la medida de los sensores se habilitan las interrupciones que incrementan el valor de una variable y al dejar de usarlas se deshabilitan nuevamente. Esa variable se toma en cuenta en la función que hace girar al robot en un determinado ángulo, o en la que lo hace avanzar una determinada distancia. Para esto las variables se multiplican por constantes de proporcionalidad obtenidas experimentalmente.

4.5.3. Sensores infrarrojos

Los sensores de luz infrarroja son comúnmente utilizados en aplicaciones de detección de obstáculos. Éstos constan de un emisor y un receptor de luz infrarroja orientados en la misma dirección. El emisor se encuentra constantemente encendido. Si se lo enfrenta a una superficie reflectora, el receptor se enciende y el sensor IR levanta la tensión de su salida.

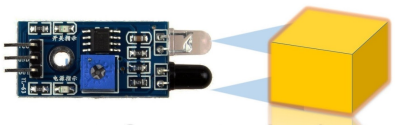


Figura 4.25: Sensor infrarrojo. Figura obtenida de la página del ofertante [5].

Los sensores se pueden utilizar conectados a los pines de interrupciones del microcontrolador. Como en este proyecto estos pines son usados para los encoders, los sensores IR están conectados a entradas digitales. Hay un par de estos sensores en cada robot, colocados en el frente orientados hacia adelante. En el loop principal del programa se chequea constantemente el estado de los sensores mientras se hace avanzar al robot. Si éste se enfrenta a un obstáculo, se detiene y gira.

Originalmente estos sensores se colocaron para evitar que los robots se choquen y se dañen, pero posteriormente se incorporaron como una medida paliativa para solucionar los problemas de orientación generados por el GPS. Gracias a estos sensores y a las barreras que delimitan la Cancha del Club de Bochas (locación elegida para testear el funcionamiento del proyecto) se tiene la seguridad de que los robots no se dirigen hacia destinos incorrectos.

Estos sensores introducen una nueva limitación para el proyecto, ya que detectan la luz infrarroja proveniente del sol. Por este motivo el robot no puede funcionar en entornos donde haya luz solar; están pensados para funcionar **durante la noche**.

4.5.4. GPS

Para el posicionamiento al aire libre se utilizan los sistemas globales de navegación por satélite (Global Navigation Satellite System, GNSS). Éstos cuentan con una extensa constelación de satélites girando en diferentes órbitas alrededor de la tierra mientras transmiten las señales utilizadas para el posicionamiento. Mediante trilateración, estas señales brindan una medida de la distancia desde tres ubicaciones conocidas a un punto que se quiere localizar.

Existen varios sistemas de posicionamiento. El más conocido y utilizado en la actualidad es el Global Positioning System (**GPS**) operativo desde 1978, desarrollado por USA originalmente para fines militares, aunque en la actualidad tiene un uso civil. El Global'naya Navigatsionnaya Sputnikovaya Sistema (**GLONASS**), fue desarrollado por Rusia y está oficialmente operativo desde 1995. **GALILEO** es el homólogo de la Unión Europea, que se estima esté completado para 2020. También para ese año se espera que el sistema **BeiDou-2**, de desarrollo Chino, quede operativo.

Para determinar la posición de los robots se eligió trabajar con un sensor GPS. Se pretende que, conociendo su posición y las coordenadas de origen y destino,

estos puedan conducirse automáticamente. Esto implicó una activa búsqueda de información con el fin de sortear las dificultades con que nos íbamos encontrando.

En el apéndice [3] se realiza una somera presentación de las tecnologías existentes para la localización de objetos. A continuación se describen los fundamentos manejados para la elección del GPS. Luego se describe su funcionamiento, se detallan las limitaciones que plantea trabajar con estos sensores exponiendo, finalmente, las soluciones propuestas para su manejo.

Elección de GPS

A pesar de existir una amplia variedad de tecnologías para el posicionamiento, optamos por el GPS. Los motivos se listan a continuación.

Integrable con Arduino

Los robots de los cuales se quiere conocer su posición, cuentan con un Raspberry Pi y un Arduino. Para modularizar su diseño se determinó que, por un lado, todas las funciones referentes a la navegación automática de los robots fuese controlada por Arduino, mientras que el Raspberry Pi se encarga de gestionar las comunicaciones. Esto requirió de sensores existentes en el mercado que fueran fácilmente integrables con este microcontrolador.

Comunidad y bibliotecas

Arduino cuenta con una gran comunidad de usuarios que constantemente están generando bibliotecas para integrar distinto tipo de periféricos. Para el caso del GPS, Mikal Hart desarrolló, en colaboración con otros usuarios, la librería TinyGPS. [89]. Ésta fue diseñada para proveer la mayoría de las funcionalidades del estándar NMEA[6] [84] para GPS como posición, fecha, hora, altitud, velocidad y dirección, eliminando el largo encabezado que se suele obtener si se lee directamente el receptor. La biblioteca además evita la utilización de punto flotante e ignora algunos campos del encabezado del mensaje del GPS, a fin de consumir pocos recursos. Más adelante se explica las funciones de la biblioteca que son útiles al proyecto.

Escalabilidad del proyecto

Si se pretende que el proyecto se expanda, para aplicaciones en grandes distancias y al aire libre, actualmente el GPS es la tecnología disponible que mejor se ajusta.

[6]La National Marine Electronics Association (NMEA) es una organización de comercio electrónico estadounidense que establece estándares de comunicación entre electrónica marina.

Simplicidad del sistema

Otras tecnologías para posicionamiento requieren que el usuario monte previamente una infraestructura con nodos emisores de señal colocados en ubicaciones conocidas. En cambio, el usuario de GPS sólo debe disponer de un receptor de señal.

Conocimientos de partida

Al comienzo del proyecto el equipo desconocía el alcance de las antenas de los USRP B200 mini, la incertidumbre de las medidas de GPS y las limitaciones para su uso en interiores. Se eligió trabajar con esta tecnología para el posicionamiento como forma de explorar su utilización.

Funcionamiento del GPS

El Sistema de Posicionamiento Global (GPS) es un servicio de los Estados Unidos que mediante señales de radio transfiere información precisa sobre la posición, navegación y el tiempo (PVT) a un número ilimitado de usuarios adecuadamente equipados en tierra, mar, aire o en el espacio. Las correcciones PVT están disponibles en todo el mundo, expresadas en un sistema de coordenadas universal.

El sistema se compone de tres segmentos principales: *Espacial*, *Control* y *Usuario*.

El *Segmento Espacial* consiste en una constelación nominal de 24 satélites. En la actualidad el sistema cuenta con 31 satélites operativos para mejorar la cobertura en algunas regiones. Cada satélite transmite en radiofrecuencia códigos de distancia y un mensaje de navegación.

El *Segmento de Control* consiste en una red de monitoreo y control que se utiliza para administrar la constelación de satélites y actualizar los mensajes de navegación.

El *Segmento de Usuario* consiste en una variedad de receptores de radionavegación específicamente diseñados para recibir, decodificar y procesar los códigos y los mensajes de navegación transmitidos por los satélites.

Los satélites transmiten señales de distancia en dos bandas de frecuencias: Link 1 (L1) a 1575,42MHz y Link 2 (L2) a 1227,6MHz. Las señales satelitales son transmitidas usando técnicas de espectro ensanchado (SS), empleando dos códigos distintos: un código de adquisición (C/A) de 1,023Mbps modulado con L1 y uno código de precisión (P(Y)) a 10,23Mbps sobre L1 y L2. Tanto C/A como P(Y) pueden ser usados para determinar la distancia entre el satélite y el usuario, sin embargo, P(Y) es diez veces más preciso pero está encriptado de forma que sólo

se encuentra disponible para un número limitado de usuarios. A 50Hz se superpone con C/A y P(Y) un mensaje de navegación. El mensaje incluye la hora del reloj satelital al momento de la transmisión, correcciones del reloj, efemérides del satélite transmisor, parámetros para la corrección de la propagación de la señal en la ionosfera y el almanaque de las órbitas de todos los satélites de la constelación.

El receptor usa esta información para determinar su posición mediante trilateración, que básicamente son cálculos similares a los de cualquier otro dispositivo de navegación basado en distancias. Conceptualmente, cada medida de distancia define una esfera centrada en el satélite. El punto de intersección de las esferas sobre o cerca de la superficie de la tierra define la posición del receptor.

Mediante el loop de correlación con el que se analizan los códigos, se determina el tiempo de llegada de la señal del GPS al receptor. El receptor realiza una versión interna del mensaje que espera recibir y al realizar la correlación entre el mensaje recibido y el esperado se determina el bit que maximiza la correlación, el cual se toma como medida del offset temporal local. El tiempo en que el mensaje fue enviado se transmite en el mensaje de navegación, que se decodifica en el receptor luego de realizar la correlación. La diferencia entre el tiempo de transmisión y el tiempo de llegada es el tiempo de transición de la señal desde el satélite al receptor. Éste incluye los offset de los relojes del satélite y el receptor de GPS.

Multiplicando el tiempo de transición recién calculado por la velocidad de transmisión (que es igual a la de la luz), se obtiene una medida de la pseudodistancia, que es prácticamente igual a la distancia a diferencia que en ésta se incluye errores de los relojes, ya que el reloj del receptor es bastante malo.

Con cuatro medidas de pseudodistancia de cuatro satélites distintos, se puede encontrar la posición absoluta y determinar el offset del reloj. Para el posicionamiento por GPS se requiere un mínimo de cuatro satélites estén simultáneamente en "línea vista" con el receptor, cada uno brindando su medida de distancia. Esto le permite al receptor calcular los tres parámetros desconocidos para representar su posición 3D, así como también un cuarto parámetro que representa el error del reloj del usuario. Considerar el error del reloj como un valor desconocido permite que la mayoría de los receptores sean construidos con un oscilador de cristal muy económico en vez de un reloj atómico costoso. Es importante tener estimaciones precisas del tiempo para tener un posicionamiento preciso, ya que un error de unos 3 nanosegundos es aproximadamente equivalente a 1 metro de distancia. Se podría usar menos de cuatro satélites en el caso de que se pueda determinar algún parámetro de forma externa. Por otro lado, si se capta la señal de entre siete y nueve satélites, y si éstos están en una geometría adecuada (están dispersos), pueden obtenerse incertidumbres inferiores a los 2,5 metros el 95 % del tiempo.

Las medidas son afectadas de error. Algunos ejemplos son: que la frecuencia de reloj del receptor no es perfecta, errores en las correcciones que se hacen en la io-

Tabla 4.2: Tabla de fuentes de incertidumbre en el GPS [91]

Fuente de error	**Efecto (metros)**
Arribo de la señal C/A	+- 3
Arribo de la señal P(Y)	+- 0,3
Ionosfera	+- 5
Efemérides	+- 2,5
Reloj Satelital	+- 2
Multipath	+- 1
Troposfera	+- 0,5
Errores numéricos	+- 1

nosfera, y no considerar la dinámica del sistema mientras se adquieren las medidas.

Se deben realizar correcciones en esta medida de pseudodistancia para considerar y modelar los retardos generados por la troposfera y la ionosfera. En los receptores stand-alone que operan en una sola frecuencia, se realizan correcciones obtenidas de modelar a la ionosfera. En los receptores de dos bandas de frecuencias, se calcula directamente el retardo introducido en la ionosfera y se corrige. [71]

Fuentes de error

Existen una serie de influencias físicas que afectan las observaciones del GPS. La posición calculada por un receptor GPS requiere el instante actual, la posición del satélite y el retraso medido de la señal recibida. La principal distorsión es el retardo en la llegada de la señal introducido por la ionosfera, que afecta el cálculo de la distancia. La precisión del GPS suele ser de 15 metros, y si la electrónica de éstos es buena, se llega a 3 metros. En la siguiente tabla se muestran las distintas fuentes de error y el peso que tienen en las medidas.

Arribo de la señal C/A y P(Y)

El tiempo de llegada se mide comparando la secuencia de bits de los códigos recibidos desde el satélite con una versión que el receptor generó internamente. Al medir la comparación de la salida y la llegada de las transiciones de bits, se puede medir el desplazamiento de la señal hasta un 1 % del ancho del pulso del bit, con lo cual la resolución es de 10 nanosegundos aproximadamente en el caso del código C/A, lo que representa 3 metros. En cambio, si se usa el código P(Y) se puede obtener mayor precisión y asumiendo que la electrónica detecta el mismo 1 %, esta señal tiene una precisión de 30 centímetros. [91] [13]

Efemérides

La efemérides es una tabla de valores en los que se almacenan las posiciones de

los satélites. Las órbitas de éstos se van alterando debido a la atracción del sol y la luna, la diferencia de gravedad entre distintas zonas de la corteza terrestre, viento solar, etc. Es por esto que la información cambia frecuentemente y es actualizada por las estaciones de seguimiento de la tierra. Las efemérides transmitidas son calculadas por el sistema oficial de control de GPS en base al rastreo de todos los satélites. El error es la diferencia entre la ubicación actual del satélite y la posición predicha por la información de la órbita del satélite. Normalmente, este error es menor a 8,2 metros el 95 % del tiempo. [91] [71]

Efectos ionosféricos

Este efecto es una fuente importante de error en las medidas de GPS. El retraso de la señal debido a la ionósfera puede variar desde unos pocos metros a más de 20m en un día. Es difícil generalmente modelar a los efectos ionosféricos por las complicadas interacciones físicas que ocurren entre el campo magnético terrestre y la actividad solar. Sin embargo, la ionósfera es un medio dispersivo, por lo que los efectos dependen de la frecuencia. Esta propiedad se utiliza para diseñar sistemas GPS con varias frecuencias de trabajo tales que los efectos se puedan medir o corregir. [101]

Efectos troposféricos

La troposfera es la parte más baja de la atmósfera terrestre y contiene 99 % de vapor de agua. A diferencia de la ionósfera, la troposfera es eléctricamente neutral y es un medio no dispersivo para la radiofrecuencias por debajo de 15GHz. Como resultado, ésta retarda la fase del GPS y el código con las medidas idénticamente. El retardo en la troposfera es mínimo en su zenit y es de casi 2,4m al nivel del mar. [47]

Error del reloj satelital

Este error es la diferencia entre el tiempo actual del satélite GPS y aquel predicho por la información del satélite. El mensaje de navegación contiene correcciones para estos errores y estimaciones del reloj atómico. Sin embargo, esta información se basa en observaciones y puede no indicar el estado actual del reloj. [71]

Multipath

El efecto de caminos múltiples ocurre cuando una señal de GPS llega a la antena del receptor desde más de un camino. En posicionamiento estático y cinemático, el efecto por multipath es una fuente de error que debe ser tomada en cuenta. Es un efecto que depende solamente del lugar donde se encuentra la antena. El receptor puede recibir la señal directamente desde el satélite y la señal reflejada (indirecta). El camino indirecto es claramente dependiente de la superficie reflectora y la posición del satélite. La superficie reflectora es generalmente

estática y está relacionada con el receptor; sin embargo, el satélite se mueve con el tiempo. Por este motivo, el multipath es dependiente del tiempo. [101]

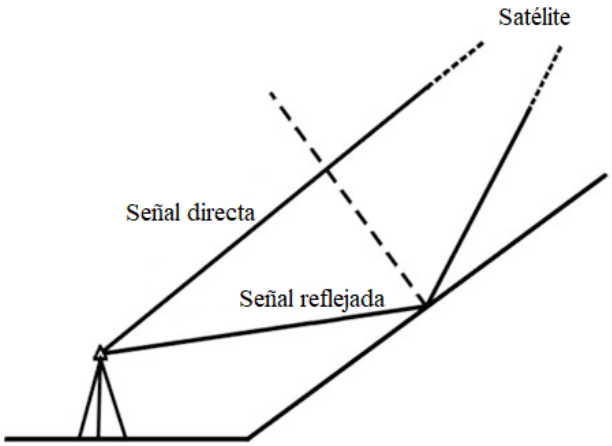


Figura 4.26: Efecto del multipath. Figura obtenida de GPS: theory, algorithms and applications [101].

Debilitamiento Geométrico de la Precisión

Geometrical Dilution of Precision (**GDOP**) es un parámetro que determina la precisión de la posición medida. Es una medida instantánea del error debido a una mala relación geométrica entre los satélites vistos por el receptor. Es un factor multiplicativo sin dimensión que varía debido a que los satélites están en constante movimiento y sus relaciones geométricas están constantemente cambiando. Éste parámetros se calcula fácilmente en el receptor. De forma cuantitativa, cuanto más separados estén los satélites, la medida obtenida tendrá menor incertidumbre. Existen distintos modos de GDOP dependiendo de las dimensiones que se quiera considerar, ya que tienen un significado distinto respecto a los ejes x, y y z en el sistema de coordenadas local. Está el debilitamiento en la precisión de la posición (**PDOP**), el debilitamiento en la precisión horizontal (**HDOP**), y el debilitamiento en la posición vertical (**VDOP**). Para aplicaciones de posicionamiento se suelen tomar en cuenta las medidas con parámetros menores a 10. Las medidas con valores menores a 2 se consideran excelentes para la mayoría de las aplicaciones. [71]

Hardware

Los robots están equipados con el integrado GY-NEO6MV2 de Ublox y su antena. A la salida tiene pines de alimentación Vcc (3,5V) y GND y de comunicación serial Rx y Tx, que por defecto viene configurada en 9600 bits por segundo. El protocolo que utiliza es el NMEA 0183. [84]

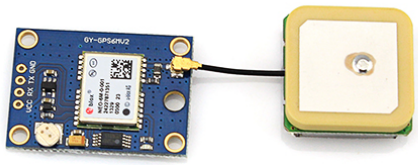


Figura 4.27: GPS GY-NEO6MV2 de Ublox. Figura obtenida del sitio Fritzing [32].

El receptor destaca por la buena performance del GPS. Detecta hasta 50 canales, trabaja en la frecuencia L1 de 1575,42 MHz con código C/A y soporta los sistemas de aumentación (Satellite Based Augmentation System - SBAS) WAAS que cubre el área de Estados Unidos, EGNOS de Europa y MSAS de Japón, por lo que no corrige las señales para el área de Uruguay. Demora como mínimo unos 27 segundos en comenzar a funcionar desde que se lo enciende. Tiene una sensibilidad de -161dBm, y en buenas condiciones de señal el fabricante asegura que el 50 % de las medidas horizontales tienen hasta 2,5 metros de error; si se utiliza con el sistema SPAS esta incertidumbre disminuye a 2,0 metros. Los datos de posicionamiento los actualiza cada 1 segundo. [55]

En cuanto a la resolución de las medidas, la biblioteca TinyGPS expresa los datos en grados decimales con una resolución de 6 dígitos luego de la coma. Esto quiere decir que la resolución del GPS, siendo utilizado a través de Arduino, es de 1×10^{-6} grados decimales, lo que equivale a una resolución en la medida de la latitud de 11,13 cm y en la de la longitud de 9,13cm. Para fijar ideas, podemos decir que la resolución del GPS es del orden de los 10 cm en cada eje. El instrumento en sí mismo tiene una resolución de 1×10^{-8}.

Depuración

Se debió incorporar herramientas que permitieran entender el funcionamiento del GPS antes de comenzar a integrarlo con Arduino. Para esto se utilizó un software desarrollado por Ublox que se llama U-Center. El software es gratuito, funciona para Windows y se puede descargar desde el sitio web [7]. Permite evaluar, testear y configurar los chips de u-blox para posicionamiento por GNSS y visualizar la constelación de satélites visible por el receptor. [93]

Nosotros usamos el software para conocer la cantidad de satélites vistos por el

[7] Sitio web de Ublox: $www.u-blox.com$

receptor, y saber en qué momentos éste perdía la señal. Esto nos ayudó a encontrar una ubicación apropiada para utilizar el GPS al aire libre. Además el programa calculó los valores de PDOP, HDOP, el promedio, máximo y mínimo de las medidas obtenidas y su desviación estándar.

Para poder conectar el sensor a la computadora se utilizó un adaptador USB a Serial/UART/RS232 del modelo PL2303.

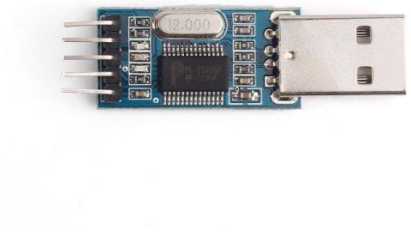


Figura 4.28: Adaptador USB a Serial/UART/RS232. Figura obtenida de página del vendedor [18].

Mejoras del posicionamiento

Motivación

En este proyecto la incertidumbre de las medidas del GPS adquirió una relevancia significativa. Una vez que comenzamos a realizar las pruebas de alcance de las antenas USRP B200 mini utilizando el protocolo CSMA/CA que desarrollamos sobre GNU Radio, vimos que fue posible mantener una buena calidad de servicio hasta los 10 metros de distancia; para distancias mayores la calidad decrece rápidamente. Considerando que en condiciones óptimas, el receptor GPS tiene una incertidumbre de 2,5 metros (para el 50 % de las medidas), esta incertidumbre se hace comparable con la distancia de alcance de las antenas.

El robot Explorador debe dirigirse a una posición de destino calculada con precisión, por lo que el error que cometa a la hora de orientarse a su destino se deberá al error generado por el GPS en el cálculo de su propia posición. Luego, el robot Repetidor se dirigirá a una posición calculada en función de la posición reportada por el Explorador. Es decir que el destino del Explorador ya no es una ubicación precisa (como en el caso anterior), sino que está afectada de error. Por ese motivo se buscó reducir el error de las medidas para que los robots no se orienten hacia destinos incorrectos deteriorando la integridad de la navegación automática. En la imagen 4.29 se ilustra un caso en que las posiciones calculadas por los receptores

GPS generan un error importante en el ángulo con el que se orienta el Repetidor.

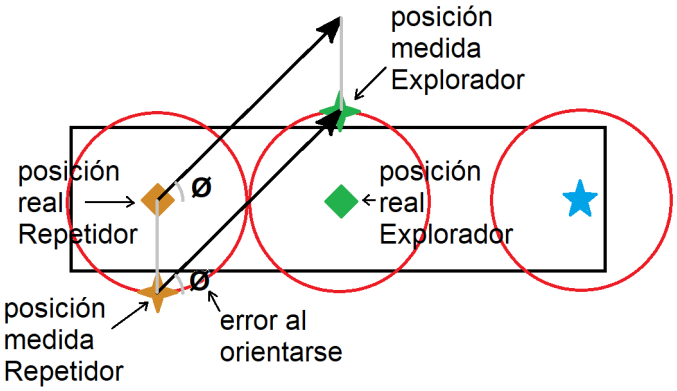


Figura 4.29: Caso en que el error en la orientación debido a error del GPS es importante.

Soluciones existentes

Para reducir la incertidumbre en el posicionamiento, comúnmente se agrega información adicional que ayude a corregir las mediciones. Esta información puede ser la proveniente de otro receptor GPS o de otro tipo de sensores que brinden una estimación de la posición del robot. Las dos técnicas comúnmente utilizadas para reducirla son el GPS Diferencial o los filtros de Kalman.

El **GPS Diferencial** utiliza la información de otro sensor de referencia que experimente los mismos efectos atmosféricos, mide los errores introducidos y los resta de las medidas.

El **filtro de Kalman** es un método recursivo que utiliza la información de varias mediciones para construir un estimador de la posición. Se puede aplicar para estimar la posición utilizando como entradas las medidas de dos o más receptores GPS funcionando al unísono. Otra forma habitual es integrando la información de distintas fuentes sensoriales (como por ejemplo GPS, magnetómetro, acelerómetro, encoders) para calcular el estimador.

En los casos en que se debe trabajar solamente con la información obtenida por un receptor aislado y estático, se puede mejorar la precisión realizando el **promedio** de las medidas obtenidas e intentando reducir las fuentes de error antes descritas. Para eso se debe colocar al receptor en lugares sin interferencias por multipath, y donde los indicadores PDOP y HDOP sean buenos.

En el apéndice D.3 se profundiza sobre cada una de ellas.

Nuestro manejo del error

Área de llegada

Las medidas de posición del robot siempre van a estar afectadas de error. Gracias a mediciones experimentales, se observó que la incertidumbre de las medidas es de 3 metros el 68 % de las veces. Es decir que la ubicación medida por el robot (ubicación virtual), puede estar en cualquier lugar dentro de un círculo de 3 metros de radio centrado en el robot. Puede ocurrir que el robot llegue exactamente a su posición de destino pero que su ubicación virtual esté a cierta distancia de este punto. Es muy improbable que la medida de GPS sea igual a la posición de destino, y hasta que eso no ocurra el robot debería seguir iterando hasta alcanzarlo.

Para determinar que el robot llega al destino, definimos un círculo de 3 metros de radio centrado en esa posición. Si la ubicación virtual del robot está dentro de este círculo, considera que llegó a su destino. Esto implica que el robot podría considerar que llegó estando aún a 6 metros de su destino.

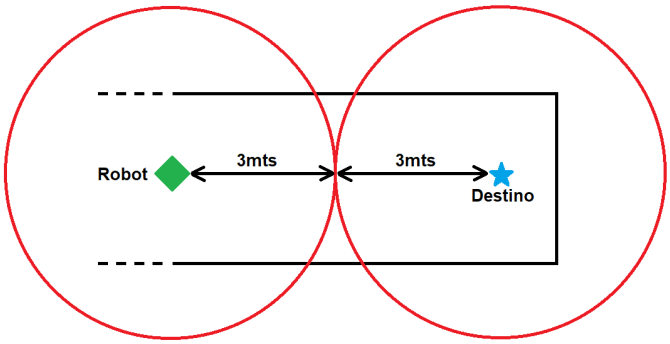


Figura 4.30: Área de llegada

Ubicación

Se eligió un lugar que permitiera un adecuado funcionamiento del GPS y de las antenas, así como los movimientos necesarios de los robot. Estos requerimientos fueron expuestos en la sección 4.2.3.

Se eligió probar el sistema completo en el Club de Bochas del Parque Rodó de Montevideo (Av. Tomás Giribaldi esq. Ing. Carlos Ma. Maggiolo), figura 4.31, porque tiene varias características que se ajustan a las necesidades del sistema.

- **Dimensiones:** La cancha es de 24 x 4 metros. El largo de la cancha es poco mayor que el alcance inalámbrico del sistema completo Explorador + Repetidor. El ancho es comparable con el error del GPS y es lo suficientemente estrecho como para impedirle a los robots que se alejen mucho de la recta que une el punto de origen y el destino.

Figura 4.31: Club de Bochas. Foto obtenida de página de promoción del Parque Rodó [102].

- **Superficie:** Está cubierta con gravilla y piedra caliza pulida compactada, lo que lo convierte en un suelo liso y con bajo rozamiento. En base a experimentos hechos con los robots sobre superficies con distinta rugosidad, se determinó que un suelo con bajo rozamiento le permitía a los robots girar sobre su propio eje. En la sección 4.2.3 se analiza en detalle los motivos por los que esto ocurría.
- **Perímetro cerrado:** El perímetro de la cancha está delimitado por maderas. Esto es útil para que los robots no salgan del área de la cancha a consecuencia de una medida incorrecta del GPS. Cuando éste se aproxima el perímetro, los sensores IR que están colocados en el frente del robot detectan el obstáculo y comienza a ejecutarse el algoritmo de detección de obstáculos anteriormente descrito: el robot se detiene, retrocede, gira 90º y avanza una pequeña distancia.
- **Al aire libre:** La cancha no es techada y tiene pocos edificios altos a los lados. Un punto representativo dentro de la cancha, tiene un ángulo de visión de la bóveda celeste de 32º a 148º en promedio, lo que permite al sensor de GPS estar constantemente observando entre 5 y 8 satélites y agregándole redundancia a las medidas.
- **Reducida interferencia de edificios:** Como se vio anteriormente, el error generado por multipath es imposible de estimar y para reducirlo es conveniente evitar tener edificios cercanos que puedan generar la reflexión de la señal. Si bien en el club de bochas hay construcciones aledañas, los ensayos

realizados en el lugar mostraron una performance similar a la obtenida en entornos sin interferencia por este fenómeno.

Antes de contactar con los administradores del Club de Bochas, habíamos estado realizando pruebas en el garaje de la Facultad de Ingeniería de la Udelar, detrás del Instituto de Ingeniería Eléctrica. Si bien en esa locación hay casi siempre recepción del GPS, las medidas obtenidas tienen una gran desviación estándar. Esto se debe, probablemente, a los efectos de multipath generados por los rebotes de la señal en la fachada de cemento de la facultad. En la figura 4.32 se observa una captura de pantalla del programa u-center realizada durante media hora. La desviación estándar de la latitud es de 3 metros, y la de la longitud es de 10 metros, con medidas extremas de más de 40 metros de distancia de la ubicación promediada. Los indicadores de dilución de la precisión son, en promedio, PDOP=7,9 y HDOP=3,2. Según esta información se puede determinar que esta locación no es adecuada para el proyecto.

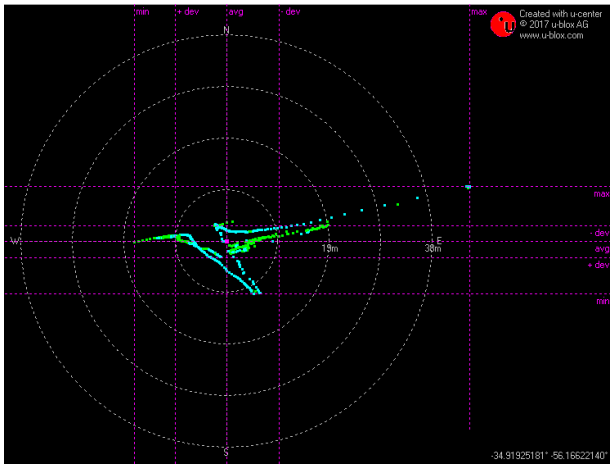


Figura 4.32: Medidas de GPS tomadas durante media hora en el garage de la FING. Desviación estándar de latitud=3m, desviación de longitud=10m. PDOP=7,9 y HDOP=3,2 (en promedio). Captura de pantalla de u-center.

A en la figura 4.33 se muestra la captura de pantalla del software U-Center con el sensor GPS ubicado estáticamente en el Club de Bochas durante una hora.

Los indicadores PDOP, HDOP son más bajos ya que hay más satélites en línea vista con el receptor. La dispersión de las medidas es menor que la que se obtiene en el estacionamiento del IIE, consiguiendo una variación estándar de 3 metros

Promedio

Fue necesario incorporar algún tipo de corrección de las medidas obtenidas por el GPS a fin de no comprometer la navegación automática de los robots por los

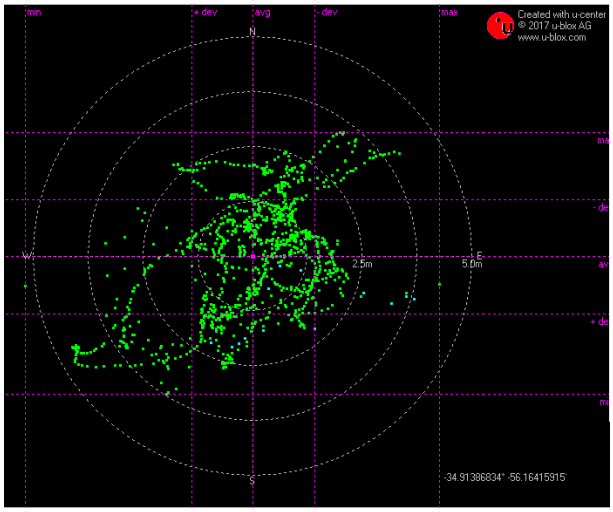


Figura 4.33: Medidas de GPS tomadas durante una hora en el Club de Bochas. Desviación estándar de latitud y longitud=2,9m. PDOP=3,5 y HDOP=1,8 (en promedio). Captura de pantalla de u-center.

errores introducidos por este sensor. Como se detalla en el apéndice D existen soluciones del tipo GPS Diferencial D.3.1 o filtros de Kalman D.3.2. Ambos pueden ser incorporados en robots móviles, pero por la dificultad de su implementación se consideró que escapaban al alcance del proyecto.

Como cada una de las partes de este proyecto plantea limitantes, además de las que se exponen este capítulo dependientes del GPS, se debió buscar una solución de compromiso que las contemple a todas. Dado que el robot debe realizar mediciones en varios puntos distintos y las baterías tienen una duración limitada, se buscó disminuir el error en un tiempo acotado aunque no se pudiera determinar la posición exacta. Se aprovechan los tiempos muertos, en que el robot mide la calidad de servicio con la estación base, para promediar su posición (en el apéndice D.3.3 se analizan los resultados esperados de aplicar éste método).

El Arduino de los robots, realiza el promedio de las medidas de GPS como forma reducir la incertidumbre. Cada vez que el Explorador o el Repetidor tienen que orientarse o reportar su ubicación a la Estación Base, realizan el promedio de un mínimo de 35 medidas, como ya se vio en la sección 4.4.2. A una cadencia de una lectura por segundo, corresponde a unos 35 segundos de promedio. Cuando los robots se detienen, comienzan a promediar su ubicación hasta que se les ordene moverse, por lo que promedian un número de muestras igual a la cantidad de segundos que permanezcan detenidos.

De acuerdo a los planteos del prof. David L. Wilson[8], no se espera que el promedio de 35 medidas cambie sustancialmente la incertidumbre, pero sí que mitigue

[8]Por referencias a los planteos del prof. David L. Wilson, mirar el apéndice D.3.3.

el peso que puedan llegar a tener algunas medidas extremas. Arduino nunca considera una sola medida de forma aislada, sino que reporta su ubicación luego de promediar de un total de 35, o más muestras.

A continuación se hace un somero análisis de la performance de aplicar esta técnica.

Se analizó durante dos horas un total de 4263 medidas de posición en condiciones ideales de recepción del GPS (buenas condiciones atmosféricas y sin interferencias provocadas por edificios cercanos), mientras éste permanecía quieto. Las medidas obtenidas tienen coeficientes PDOP y HDOP menores a 2, lo que confirma que todas las medidas se obtuvieron bajo excelentes condiciones. El promedio de la posición es (-34,90138963 , -56,15497262) con una desviación estándar en las medidas de la latitud de 0,9 metros y 1,1 metros en el caso de la longitud.

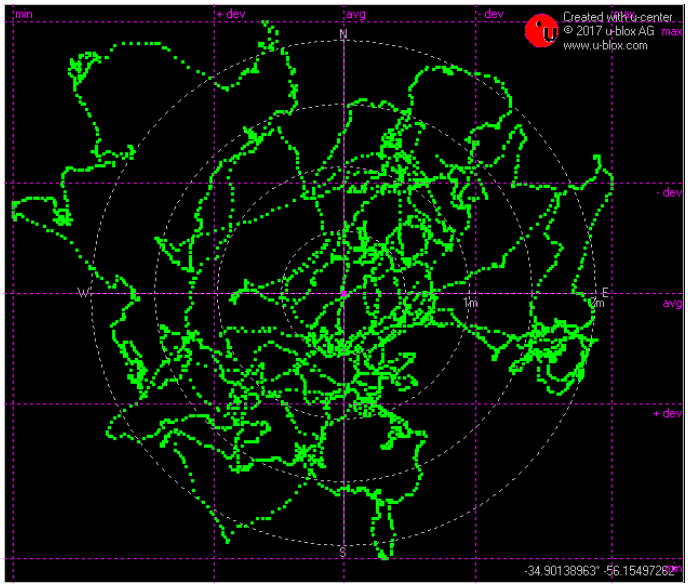


Figura 4.34: Medidas de GPS tomadas durante dos horas. Desviación estándar de latitud= 0,9m, desviación de longitud=1,1m. PDOP<2. Captura de pantalla de u-center.

Luego se forman grupos consecutivos de n muestras consecutivas y se promedian. De esta forma se obtienen $4263/n$ nuevas muestras, donde cada una es el promedio de n muestras consecutivas. En la figura 4.35 se analiza la desviación estándar de las nuevas medidas, cuando n varía desde 1 hasta 1400 y se lo representa en escala logarítmica.

Observar que, a medida que n se hace más grande, las curvas presentan mayores oscilaciones. En un caso ideal, las curvas tendrían pendiente negativa para todo n, que es equivalente a decir que la desviación solamente decrece a medida que se aumenta el número de muestras promediadas. En este ensayo, se trabaja con un

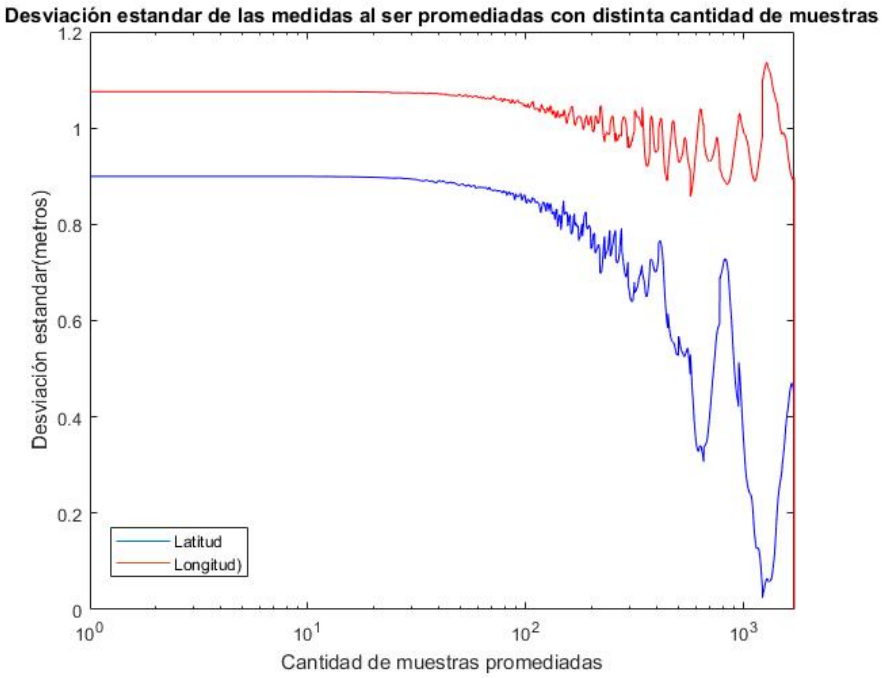


Figura 4.35: Desviación estándar de las medidas al ser promediadas con distinta cantidad de muestras

número finito de muestras, lo que implica que para el caso $n = 1400$, por ejemplo, se tenga un total de 3 muestras con las que se calcula la desviación estándar. Este es un número chico para calcular la desviación estandar, y posiblemente por esto se producen las oscilaciones.

La curva obtenida, tiene similares características a las del gráfico D.6 de David L. Wilson, presentada en el apéndice D.3.3. Se desprende de esto que realizar el promedio es efectivo en la reducción de la incertidumbre, pero para que esta técnica muestre buenos resultados, se debe realizar un promedio de más de 100 muestras. En particular, el promedio de 35 muestras mejora la latitud en un 1,1 % y la longitud en un 1,0 %.

En la gráfica 4.36 se muestra la velocidad con la que converge el promedio en función de la cantidad de muestras promediadas y se lo contrasta con los valores instantáneos de posicionamiento.

Se observa del gráfico 4.36 que para una cantidad pequeña de muestras promediadas, el promedio es bastante similar a las muestras obtenidas en cada instante. Así mismo, se observa que esta técnica suaviza la curva y evita que se puedan considerar medidas extremas para el posicionamiento.

Los resultados obtenidos muestran que realizar el promedio de 35 medidas no

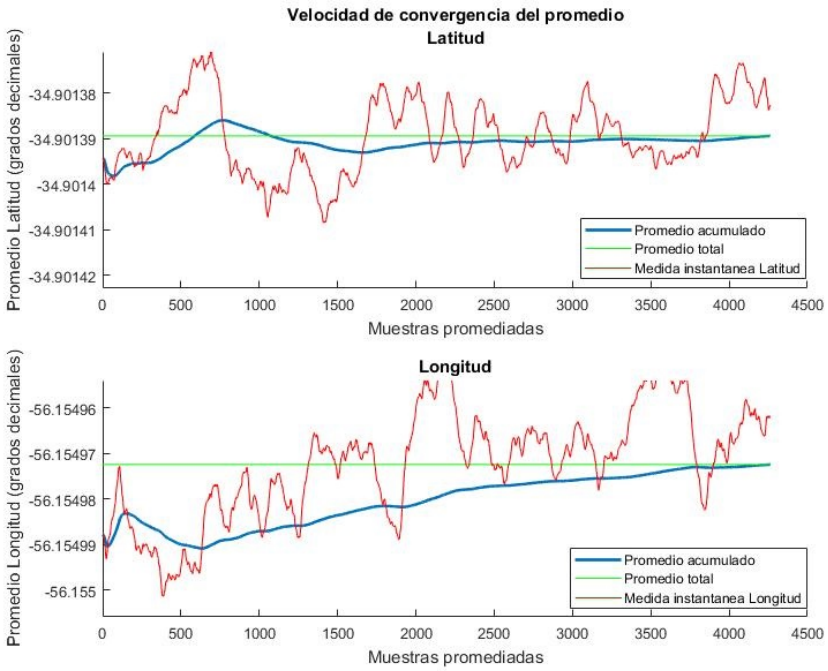


Figura 4.36: Velocidad de convergencia del promedio de medidas de GPS

es una técnica efectiva para disminuir la incertidumbre del GPS. Sin embargo, puede ser una técnica de utilidad si se dispone de más tiempo para realizar mediciones y promediarlas. Es fácil de implementar y no requiere de hardware adicional. En el proyecto igualmente se realiza un promedio de 35 medidas, ya que se aprovecha que los robots permanecen durante ese tiempo detenidos para realizar también otro tipo de mediciones.

El resultado más significativo de este análisis es que se pueden obtener desviaciones estándar de 1 metro, bajo condiciones ideales de calidad de señal. Las mediciones en la cancha del Club de Bochas, que está próximo a edificios, muestran que en ese lugar se obtiene una desviación de 3 metros aproximadamente con el mismo receptor GPS. Esto evidencia el peso que tienen algunas fuentes de incertidumbre, como efectos de multipath y dilución de la precisión. No fue posible conseguir una ubicación que permita una mejor recepción de la señal y que además fuese compatible con las otras limitaciones que presentan los demás componentes del proyecto.

Algoritmo alternativo por coordenadas relativas

Teniendo en cuenta las dificultades que presenta usar GPS, y en particular, la incompatibilidad entre el alcance de las comunicación entre los USRP B200 mini y la precisión del GPS, se ensayó un algoritmo alternativo de navegación. Previo a conseguir el Club de Bochas para realizar los trabajos, hubo una instancia en

la que no se tenía una locación en la que pudiesen coexistir todas las limitaciones que presenta trabajar con este tipo de robots. El nuevo algoritmo surgió como una alternativa al diseño original del robot que se posiciona con GPS. Ambos algoritmos de posicionamiento se desarrollando en paralelo.

Conocemos a este algoritmo con el nombre de "coordenadas relativas", debido a que la posición de los robots se estima a partir de su posición inicial, la cual tomamos como origen de coordenadas. Este usa únicamente las medidas del magnetómetro y los encoders para estimar su posición. Al no utilizar el GPS, es apto para utilizar en entornos cerrados o con mala recepción del GPS.

Este algoritmo surge de la combinación de la navegación por odometría (la cual utiliza la rotación de las ruedas medida por los encoders para estimar el desplazamiento del robot), junto con la información proporcionada por el magnetómetro. Los algoritmos de navegación por odometría tienen la ventaja de ser simples de implementar, de costo reducido y obtienen buena precisión para cortas distancias. Sin embargo, dado que la odometría integra información de movimiento a lo largo del tiempo, se tiene inevitablemente una acumulación de errores. A medida que el robot se desplaza, su posición real se aleja cada vez más de la estimada. Una de las principales fuentes de error del algoritmo por odometría es el deslizamiento de las ruedas debido a suelos resbaladizos, aceleraciones excesivas, derrapes, pérdida de contacto de alguna de las ruedas con el suelo [14]. Este efecto hace que se pierda la correlación entre el desplazamiento lineal de la rueda y el ángulo de giro medido por los encoders. Esta fuente de error afecta las trayectorias rectas y principalmente la medida de los ángulos de giro. Es por este motivo que se introdujo al magnetómetro: se utiliza la este sensor para medir el ángulo que tiene el robot con respecto del Norte, evitando así el uso de los encoders para estas medidas.

El algoritmo alternativo utiliza la mayoría de las funciones implementadas para el algoritmo principal. Se agregó una función para estimar la posición mediante fórmulas trigonométricas y se modificó la función de comunicación con el Raspberry Pi, para adaptarlo a los mensajes que se van a enviar (en lugar de coordenadas de latitud y longitud, se envían coordenadas cartesianas con una resolución de cuatro dígitos cada una).

El algoritmo comienza inicializando los sensores y luego envía la posición inicial del robot al Raspberry Pi (coordenadas (0,0)). Se espera entonces la respuesta del Raspberry Pi con las coordenadas de destino. En caso de que la QoS sea buena, se realiza la calibración del magnetómetro, se orienta al robot en la dirección deseada y comienza el desplazamiento en línea recta. Durante el desplazamiento, se mide la distancia recorrida con los encoders y el ángulo respecto del Norte. Con esta información, se calcula cual es la posición actual del robot. En caso que se detecte una desviación respecto del ángulo deseado de más de 15 grados, el robot se detiene. En ese momento, intercambia su posición con la estación base, recibe la posición de destino y corrige su orientación.

El algoritmo de navegación por coordenadas relativas mostró en la práctica ser una buena alternativa para la navegación por GPS en aplicaciones de corta distancia. Como presenta errores en largas distancias, este esquema está solamente pensado para recorridos cortos, de máximo 10 metros, compatible con el alcance de la comunicación inalámbrica mediante USRP B200 mini.

4.6. Conclusiones

Se construyeron dos robots rodados con autonomía de funcionamiento. Cada robot tiene una estructura mecánica que le permite moverse soportando todo el peso de sus componentes. Los movimientos que son capaces de realizar son de giro sobre su propio eje, avance y retroceso en línea recta. Esto es posible gracias al diseño de sus elementos constructivos, con especial hincapié, en el caso del robot Explorador, que tuviese motores suficientemente potentes y, en el del Repetidor, que tuviese el centro de masa ubicado en su tercio posterior. La incorporación de sensores IR les permiten esquivar obstáculos. Por otra parte, son capaces de conocer su posición utilizando GPS y de orientarse a su ubicación de destino al combinar esta información con las medidas del compás magnético. El esquema de funcionamiento para desplazarse a la posición de destino es de modo iterativo, por lo que repite un algoritmo en el transmite su posición, calcula la calidad de servicio de la comunicación, se orienta a su destino y avanza una cierta distancia. El tiempo de autonomía, determinado por la presencia de dos baterías independientes, es de dos horas, suficiente para que puedan completar sus tareas satisfactoriamente.

Existen condiciones que limitan su funcionamiento. Los robots requieren que la superficie sobre la que se desplazan tenga un coeficiente de rugosidad que le permita a las ruedas deslizar cuando el robot gira sobre su propio eje y rodar sin deslizar, al avanzar. La presencia de sensores IR obliga a un funcionamiento nocturno para evitar la interferencia de la luz infrarroja del sol. La locación debe ser a cielo abierto y con pocas edificaciones cercanas que deterioren la recepción del GPS. El nivel de precisión de la posición de destino está condicionada por la incertidumbre de las medidas del GPS.

En este sentido, se plantean propuestas de solución a los diferentes problemas encontrados: modificaciones a las ruedas o al chasis que posibiliten mayor movilidad -usando ruedas Omni Wheel por ejemplo-, la utilización de GPS diferencial para disminuir el error en la lectura de al posición, incorporar al filtro de Kalman para mejorar la estimación de la posición, o descartar el uso de GPS y plantear formas alternativas de navegación. En el Apéndice D se encuentra un estudio de relativa profundidad sobre las mejoras planteadas, en particular sobre las ruedas Omni Wheel, el Filtro de Kalman y el GPS diferencial.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Integración del sistema completo

En los capítulos anteriores se estudiaron dos grandes bloques que componen el sistema: por un lado, la comunicación sobre SDR, y por otro, el sistema de navegación de los robots. En este capítulo se describe como se integran ambos bloques para poner en funcionamiento el sistema completo.

Recapitulando, el sistema completo está formado por una radiobase y dos robots: el Explorador y el Repetidor. La radiobase está compuesta por una computadora portátil y un SDR. Será la encargada de controlar el loop general: enviar órdenes de movimiento a los robots, medir la QoS y decidir cuando activar el Repetidor. Por su parte, los robots deben ser capaces de recibir las órdenes enviadas por comunicación sobre SDR, procesarlas para que efectivamente se traduzcan en movimiento, y enviarle a la estación base su posición.

Esto impone una serie de requerimientos sobre los robots. Se vuelve necesario que éstos tengan una computadora a bordo, con capacidad suficiente para correr la comunicación sobre GNU Radio, interactuar con el periférico USRP, y comunicarse con Arduino. En la sección 5.1 se describe la elección del hardware y sistema operativo que se adapte a estas necesidades.

A continuación, en la sección 5.2 se trata un tema fundamental para lograr la integración del sistema: la comunicación entre procesos dentro de un mismo sistema anfitrión. Ésta se realiza mediante sockets de dominio Unix, los cuales permiten intercambiar mensajes entre GNU Radio y los restantes scripts de Python.

La sección 5.3 es la central de este capítulo. En ella se describe el funcionamiento del sistema completo, es decir, cómo es el flujo de mensajes entre los nodos y las tareas que realizan en cada momento. Es aquí donde se detalla como se logra la cooperación entre los robots. Se especifican los distintos algoritmos implementados a lo largo de la realización del proyecto, en un orden creciente de dificultad.

En las últimas secciones, 5.4 y 5.5, se estudian dos aspectos que hacen a la autonomía de los robots: monitoreo mediante watchdog e inicio automático de los

procesos.

5.1. Raspberry Pi

Raspberry Pi es una computadora de placa única, de pequeño tamaño y costo reducido. Fue desarrollada por la organización Raspberry Pi [64] con el fin de incentivar el estudio de las ciencias de la computación en los jóvenes. Desde su lanzamiento en el año 2012, ha recibido una gran aceptación, diversificando sus aplicaciones más allá del ámbito educativo, por ejemplo para el automatismo industrial y del hogar [50] [4]. La comunidad activa que se generó en torno al Raspberry Pi lo convierten en una de las opciones más atractivas de computadora de placa única, contando con extensa documentación, foros de discusión y proyectos de código abierto disponibles.

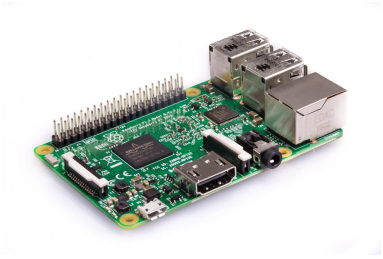


Figura 5.1: Raspberry Pi Modelo 3 B [63]

5.1.1. Hardware

En cuanto a su arquitectura, está compuesta por una única placa que incluye al procesador, la memoria y los puertos de entrada y salida. En la figura 5.2 se observa un esquema de los principales bloques que componen el hardware del Raspberry Pi.

Las distintas generaciones de Raspberry Pi fueron incrementando sus prestaciones en cuanto a capacidad de procesamiento, frecuencia de operación, memoria RAM, entre otras. Para este proyecto se utilizó el modelo más potente disponible a la fecha de realizar la compra, el modelo 3 B. A continuación se resumen sus principales características [36]:

- SoC (System on a Chip): Broadcom BCM2837 que incluye CPU, GPU, DSP, SDRAM y puertos USB.
- CPU: Cuatro núcleos con arquitectura ARMv8, frecuencia de operación 1,2 GHz.
- Memoria: SDRAM de 1 GB.
- Almacenamiento: Mediante tarjeta microSD.

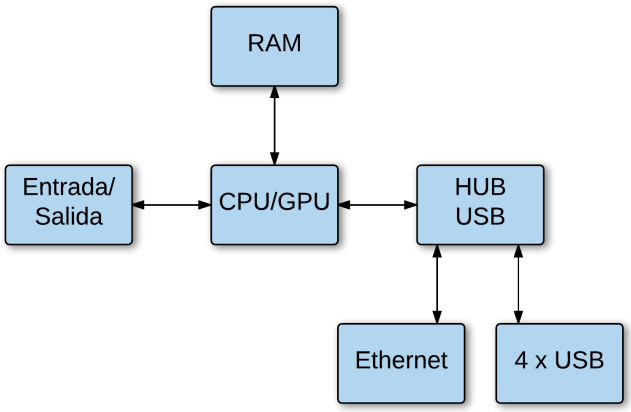


Figura 5.2: Diagrama de bloques del hardware del Raspberry Pi

- Puertos USB: 4 puertos USB 2.0.
- Conectividad: Ethernet 10/100, WiFi 802.11n, Bluetooth 4.1
- GPIO (General Purpose Input/Output pins): 40 pines
- Alimentación: a través del puerto micro USB (5,1 V) o a través de los pines GPIO (5 V o 3,3V).
- Dimensiones: 8,6 cm x 5,7 cm x 1,8 cm
- Peso: 45 gramos

Se destacan la capacidad de procesamiento y memoria RAM suficientes para los procesos que se desean correr. Además, su peso y dimensiones reducidas facilitan el montaje sobre los robots.

5.1.2. Sistema Operativo

El Raspberry Pi no es compatible con los sistemas operativos que se usan habitualmente en las computadoras de propósito general, debido a que utilizan arquitectura ARM[1] y no x86 [2]. No obstante, se han desarrollado sistemas operativos orientados específicamente al Raspberry Pi, basándose en distribuciones de escritorio y servidores tan populares como Debian o Arch Linux.

[1] ARM es un tipo de arquitectura RISC (Reduced Instruction Set Computer u Ordenador con Conjunto Reducido de Instrucciones). Típicamente requieren menos transistores que los necesarios en una computadora con un conjunto complejo de instrucciones, lo que disminuye su costo, consumo energético y disipación de calor.

[2] x86 es una arquitectura basada en el CPU Intel 8086 y el Intel 8088. Es a la fecha la arquitectura más utilizada en los PC de escritorio y laptops.

Entre los sistemas operativos para Raspberry Pi más difundidos se encuentran:

- Raspbian [90]: El sistema operativo que soporta Raspberry Pi de manera oficial, basado en Debian.
- Arch Linux ARM [46]: Distribución basada en Arch Linux con soporte para arquitectura ARM.
- Pidora [72]: Distribución basada en Fedora adaptada para Raspberry Pi.
- Ubuntu MATE [94]: Distribución basada en Ubuntu con el escritorio MATE por defecto.
- CentOS [60]: Distribución para servidores disponible para Raspberry Pi.
- Windows 10 IoT Core [49]: Sistema operativo Windows optimizado para computadoras de placa reducida y dispositivos de Internet de las Cosas.

La gran cantidad de sistemas operativos disponibles que cuentan con distribuciones específicas para Raspberry Pi, lo posicionan nuevamente sobre sus competidores.

La elección del sistema operativo se fundamentó en varios motivos. Por un lado, se buscó seguir por la línea de las tecnologías de código abierto con comunidad activa, entre las que se destacan Raspbian y Arch Linux. Si bien Raspbian es la distribución oficial de Raspberry Pi, se optó por Arch Linux por dos razones. En primer lugar, y como se verá en detalle en la siguiente sección, Arch Linux es un sistema operativo liviano y flexible, que permite adaptarse a las necesidades del usuario. Raspbian en cambio no posee esa flexibilidad y es más pesado. Finalmente, se terminó de descartar Raspbian por un motivo práctico: la versión de GNU Radio disponible es más antigua de la que se puede tener en Arch Linux y genera problemas de permisos para comunicarse con el USRP, entre otros.

5.1.3. Arch Linux ARM

Arch Linux ARM es una distribución del sistema operativo Linux para computadoras ARM. Arch Linux ARM mantiene la filosofía de su predecesor Arch Linux, caracterizado por ser liviano y flexible. Esta distribución se compone de software libre y código abierto. Se sustenta fuertemente en la participación y el apoyo de la comunidad.

La filosofía de Arch Linux se basa en cinco principios [45]:

- Simplicidad: Arch Linux está formado por una estructura ligera de Unix, sin adiciones o modificaciones innecesarias.

- Modernidad: Arch Linux procura mantener las últimas versiones estables liberadas de software, siguiendo el modelo rolling-release (actualización rodante o actualización continua). En lugar de utilizar un sistema de liberaciones estándar, mediante versiones de software que deben reinstalarse sobre la versión anterior, el software rodante está continuamente actualizándose.

- Pragmatismo: Arch Linux es una distribución pragmática antes que idealista. Cada decisión de diseño se analiza individualmente, basándose en evidencia y debates, y no en ideologías.

- Centrado en el usuario: A diferencia de otras distribuciones GNU/Linux que buscan ser amigables con el usuario, Arch Linux está centrada en el usuario, dándole completo control y responsabilidad sobre el sistema. Arch Linux está orientada a usuarios experimentados en GNU/Linux, dispuestos a investigar, leer documentación y resolver ellos mismos sus problemas.

- Versatilidad: Luego de la instalación, Arch Linux proporciona únicamente una terminal del comandos para que el usuario pueda instalar cada paquete que necesite, obteniendo así a un sistema personalizado de acuerdo a sus necesidades.

En el apéndice C se encuentra una guía para la instalación de Arch Linux, GNU Radio, GWN y las imágenes de la FPGA en la tarjeta SD del Raspberry Pi. Esta información se encuentra también en el GitHub del proyecto.

5.2. Comunicación entre procesos

Cuando existen distintos procesos que interactúan entre si se deben satisfacer dos requerimientos fundamentales: sincronización y comunicación. La comunicación entre procesos (o IPC, Inter Process Communications), es una función básica de los sistemas operativos, que provee un mecanismo para intercambiar mensajes y así poder sincronizar los distintos programas [86].

Generalmente la funcionalidad se provee mediante dos primitivas:

enviar(mensaje, destino)
recibir(mensaje, origen)

La primera envía un mensaje a un destino dado y la segunda recibe un mensaje de un origen dado (o de cualquiera, si al receptor no le importa quien es el emisor). Si no hay mensajes disponibles, el emisor puede bloquearse hasta que llegue uno, o bien regresar de inmediato un código de error.

La comunicación entre procesos puede realizarse a través de distintos métodos: sockets, semáforos, tuberías, memoria compartida, puertos, entre otras. No todas

ellas están disponibles en todos los sistemas operativos.

En este caso se optó por los sockets de dominio Unix, también conocidos como sockets IPC, debido a que en el proyecto GWN ya se había experimentado el uso de los mismos sobre Python y GNU Radio [33].

Los sockets de dominio Unix son sockets virtuales, similares a los sockets de red, que permiten la comunicación entre procesos ejecutándose en un mismo sistema operativo. Esta comunicación aparece como un flujo de bytes, al igual que en los sockets de red, pero donde todos los datos se mantienen dentro de la computadora local.

Un socket queda definido por una dirección IP y el puerto en el que escucha. Profundicemos este concepto para el caso de los sockets de red. Supongamos que una persona quiere navegar desde su computadora por una página web. La aplicación cliente (el navegador web en este caso), deberá establecer una comunicación con un proceso remoto (la aplicación web que recibe peticiones en el servidor de la página). Para lograr esta comunicación, debe poder encontrar al servidor dentro de toda la red, y para esto requiere conocer la dirección IP del mismo. Además, necesita saber el puerto donde escucha la aplicación, que para este caso sería el conocido puerto 80, reservado para HTTP (Hypertext Transfer Protocol).

Para el caso de los sockets Unix, el par dirección IP - puerto representa una ruta en el sistema de archivos del host. Como dirección IP se utiliza la de loopback, esto es, una dirección dentro del rango 127.0.0.0/8 que se utiliza para comunicar datos cuyo destino es el propio host. Generalmente se toma la dirección 127.0.0.1 por ser la primera del rango. En cuanto al puerto, se elige un número alto que no esté reservado para aplicaciones conocidas. Generalmente usamos los puertos 50007 o 50008.

Al igual que en las redes de datos, los sockets Unix manejan el modelo de cliente y servidor. El cliente es el que juega el papel activo en la comunicación, iniciando la misma mediante peticiones al servidor y esperando su respuesta. Generalmente el cliente interactúa con el usuario final a través de una interfaz gráfica. En este proyecto, para la validación del funcionamiento de la comunicación sobre SDR, se utilizó una terminal de chat[3], donde el usuario puede escribir mensajes cortos que son enviados a GNU Radio a través de un cliente TCP. También funciona en el camino inverso, recibiendo mensajes desde el servidor en GNU Radio.

El servidor cumple un papel pasivo en la comunicación, se encuentra inicialmente escuchando en un determinado puerto a la espera de una conexión. Cuando un cliente realiza una solicitud, éste la procesa y luego envía la respuesta.

[3]Cortesía de Binarytides
`http://www.binarytides.com/code-chat-application-server-client-sockets-python/`
Se hicieron modificaciones del código original para incorporar la medida de la QoS

Los sockets Unix soportan tanto la transmisión de un flujo confiable de bytes (SOCK_STREAM, en analogía con el protocolo de capa de transporte TCP), como la transmisión de datagramas sin garantía de orden, entrega o mensajes duplicados (SOCK_DGRAM, en analogía con UDP). En este proyecto, se usaron indistintamente los sockets Unix TCP o UDP, pero se encontró una ventaja de los UDP sobre los TCP: al ser menos estricto, UDP permite que se abra un cliente en determinado puerto, sin que previamente haya un servidor escuchando. Esto es una ventaja a la hora de realizar pruebas, no hay necesidad de preocuparse si se inicia antes el servidor o no.

Python posee una biblioteca que permite trabajar con sockets. Los pasos a seguir para crear un servidor y un cliente UDP son los siguientes:

Servidor:

1. Crear un socket
2. Enlazar (bind) a un puerto y una dirección
3. Recibir/Enviar información

Cliente:

1. Crear un socket
2. Conectarse a un servidor
3. Enviar/Recibir información

Por su parte GNU Radio ya cuenta con un bloque (el Socket PDU) que permite crear clientes o servidores con protocolo TCP o UDP.

5.3. Descripción del loop principal

Los sistemas formados por múltiples robots tienen por objetivo desarrollar una tarea de forma más eficiente que si fuera ejecutada por un único robot. En nuestro caso, la suma de los esfuerzos de ambos robots permite cubrir una distancia mayor asegurando cierta QoS mínima.

La arquitectura de un sistema multi-robot se puede caracterizar mediante tres aspectos [54]:

1. Control: Existe una gama de estrategias de control, que van desde el control centralizado, donde los robots se comunican con una computadora central encargada de tomar las decisiones, al distribuido, donde cada robot toma sus propias decisiones y actúa independientemente. En nuestro caso gran

parte del control lo ejerce la estación base: ésta se encarga de recibir la posición donde se encuentran los robots, enviar a donde deben dirigirse, medir periódicamente la QoS y activar al Repetidor en caso de ser necesario, indicándole donde posicionarse de forma tal de mejorar la señal. Sin embargo, el sistema no es completamente centralizado, ya que una vez que los robots reciben la orden de a donde deben dirigirse, éstos son capaces de tomar todas las decisiones necesarias para llegar a ese punto.

2. Objetivo de los robots: Para lograr la meta final del sistema, o bien cada robot actúa siguiendo su propio objetivo individual, o todos comparten el mismo objetivo. En este proyecto, cada robot tiene una meta bien diferenciada:

   - Robot Explorador: Su objetivo es acercarse lo más posible a una cierta posición de destino, indicada por la estación base, manteniendo en todo momento una QoS mínima con ésta.
   - Robot Repetidor: Su objetivo es ser un intermediario en la comunicación entre el Explorador y la estación base: captura la señal, la filtra y la amplifica. Para esto debe posicionarse en un lugar adecuado, indicado por la estación base. Éste robot se encuentra inicialmente en estado de latencia, a la espera que la estación base lo active en caso de que la QoS sea lo suficientemente baja.

3. Cooperación: La cooperación se refiere a cómo interactúan los robots en busca de lograr la meta final. En este caso, se da una cooperación implícita: los robots no intercambian información entre ellos "directamente", sino que se comunican únicamente como forma de que el Explorador pueda lograr una buena QoS con la radiobase. Ni el Repetidor ni el Explorador generan información específica para su compañero, ellos no conocen donde se encuentra el otro robot.

Sentadas las bases de la arquitectura del sistema, procedemos a presentar cómo se implementó en la práctica lo que conocemos como "loop principal", esto es, el protocolo de intercambio de mensajes y las acciones que realizan los nodos del sistema en cada momento.

Para llegar al esquema final de funcionamiento, se pasó previamente por dos esquemas de menor dificultad, ambos compuestos únicamente por la estación base y el robot Explorador. A continuación se describen las tres etapas por las que se pasaron.

5.3.1. Primer esquema: Órdenes directas de movimiento

La idea de este esquema es ingresar órdenes directas de movimiento (adelante, atrás, izquierda, derecha, parar) mediante el teclado de la radiobase. Este modelo implica una comunicación unidireccional, donde sólo se envía información desde la

estación base al robot (más allá de los ACKs del protocolo CSMA/CA enviados en el sentido inverso). Además, el sistema de navegación de los robots implica una autonomía mucho menor que en el sistema final, porque los mensajes recibidos se transforman inequívocamente en órdenes de movimiento a los motores, casi sin necesidad de procesamiento por parte de Arduino.

Sin embargo, la sencillez de este esquema permitió probar una serie de conceptos (comunicación por GNU Radio, comunicación entre procesos mediante sockets, interacción entre Raspberry Pi y Arduino a través del puerto serial, manejo de motores y encoders) y sentar las bases para las siguientes etapas. Éste fue el entregable que se presentó en el primer hito del proyecto y se utilizó como demostración en Ingeniería de Muestra.

La figura 5.3 muestra esquemáticamente los principales componentes del sistema.

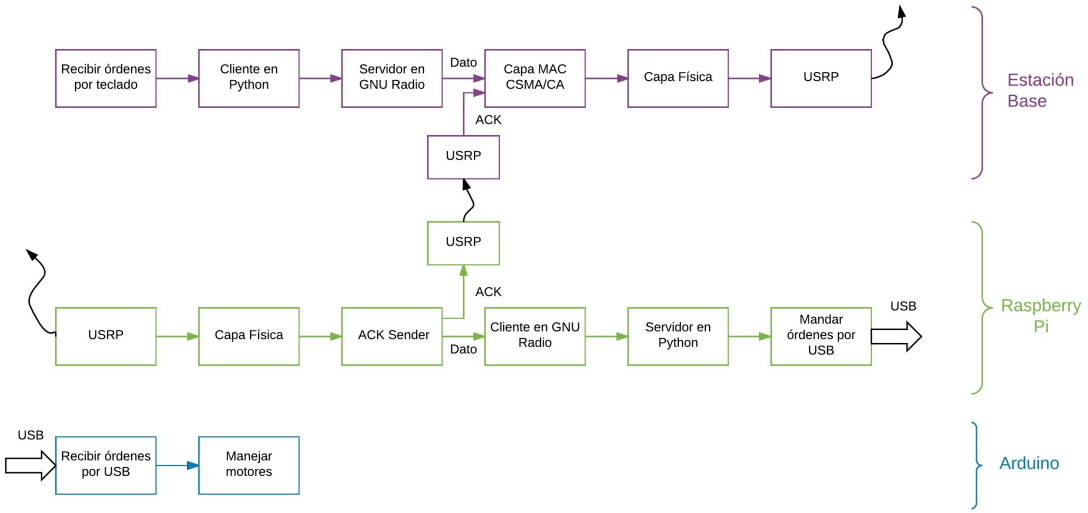


Figura 5.3: Esquema de órdenes directas de movimiento

Para terminar de comprender la inteligencia que implementa cada nodo, se hará una breve descripción de los scripts que corren en cada uno. En líneas generales, se tienen dos scripts de Python en cada nodo: uno que ejecuta la comunicación por GNU Radio, y otro que realiza el resto de las tareas necesarias. Ambos scripts intercambian mensajes mediante sockets. La comunicación sobre GNU Radio fue cubierta en el capítulo 3, por lo que no se reiterarán aquí los conceptos vistos. Para leer la versión completa de los códigos, referirse al GitHub del proyecto.

1. Estación base: la inteligencia de la estación base se encuentra en el script **estacion_base.py**. Éste corre un loop en el que se reciben órdenes a través

del teclado, arma el mensaje a enviar (haciendo una correspondencia entre la tecla ingresada y un número, y agregando un carácter especial que delimita el fin de la trama), y lo envía por un socket cliente UDP. Para clarificar, se presenta el pseudocódigo del script:

While (1):

Esperar pulsación de tecla

Armar mensaje a enviar

Enviar mensaje por cliente UDP

Para hacer el ingreso de las órdenes por teclado de forma más amena, sin necesidad de que el usuario presione Enter luego de cada tecla (como se hace con la función de entrada estándar), se usó el paquete Tkinter de Python [67] (estándar para la Interfaz Gráfica de Usuario (GUI)), el cual contiene funciones para capturar pulsaciones del teclado, movimiento en el mouse, entre otras.

2. Explorador: el Raspberry Pi del robot Explorador corre el script **com_serial.py**, que funciona como intermediario entre GNU Radio y Arduino. Corre un loop en el cual recibe los mensajes a través de un servidor UDP y los envía por el puerto serial a Arduino:

   While (1):

   Recibir mensaje por servidor UDP

   Enviar mensaje por puerto serial

   Un párrafo aparte merece la comunicación serial. Para tener acceso al puerto USB, se utiliza la biblioteca PySerial de Python [66]: Esta permite abrir la comunicación serial, indicando el nombre del puerto y el baudrate a utilizar, mediante la sentencia: ser = serial.Serial('/dev/ttyACM0', 9600). Para averiguar en que puerto Raspberry Pi monta al Arduino, se utiliza el comando ls /dev/tty*. Antes de conectar el Arduino, se corre este comando. Luego de conectarlo se lo vuelve a correr y se verifica qué dispositivo se agregó a la lista. Es importe modificar los permisos de acceso del Raspberry Pi al puerto serial, en caso contrario se tendrá una excepción que no permitirá continuar. Mediante la sentecia sudo -S chmod a+rw /dev/ttyACM0, se le da permisos de lectura y escritura. Una vez abierta la comunicación serial con Arduino, mediante las funciones read() y write(), se lee y se escriben mensajes hasta el carácter de fin de línea.

La inteligencia termina en Arduino, que recibe las órdenes a través del puerto USB, y las traduce en movimiento de los motores.

Para validar el funcionamiento de este esquema se efectuaron una serie de pruebas mediante simulaciones, primero individuales y luego de integración. Se deja para el capítulo "Pruebas" los test hechos utilizando el medio físico y los resultados obtenidos.

El objetivo es probar cada proceso individualmente e identificar y corregir problemas internos, para luego agregar gradualmente la comunicación entre procesos. En cada paso, se agrega un elemento nuevo a verificar, sabiendo que todo lo anterior ya fue verificado.

Las pruebas individuales consistieron en probar estacion_base.py y com_serial.py por separado, anulando la comunicación mediante sockets. Es decir, en las líneas del código donde se escribía o recibía un mensaje a través de un socket, se sustituyó por una escritura o lectura de terminal. Para el caso de estacion_base.py, se probó que al presionar una flecha del teclado, se imprimía en pantalla el mensaje correspondiente con su caracter de fin de trama. Por su parte, la prueba de com_serial.py consistió en ingresar un mensaje de movimiento a través de la terminal, y verificar que se traduzca en un movimiento correcto del robot (aquí ya se tenía probada la librería de movimiento del robot mediante órdenes directas, faltaba probar la comunicación serial, justamente el objetivo del script com_serial.py).

Luego, la primera prueba de integración consistió en probar el par cliente-servidor programados en Python. Para esto se corrieron los procesos estacion_base.py y com_serial.py en paralelo dentro de una misma máquina, haciendo que ambos sockets usaran el mismo puerto para que se conectaran entre ellos. Se observó que al ingresar órdenes por el teclado, el robot se movía acorde a ellas.

Finalmente se involucró también a GNU Radio y sus sockets. Esta vez, corriendo dentro de una misma máquina estacion_base.py, com_serial.py y un top_block.py conteniendo los bloques de comunicación de la estación base, un canal simulado, y los bloques de comunicación del Raspberry Pi. Con esta prueba se verificó la correcta interacción entre los sockets programados por nosotros en Python, y el bloque de sockets utilizado en GNU Radio.

5.3.2. Segundo esquema: Órdenes de traslado a nueva posición

En la siguiente etapa del proyecto se aumentó la complejidad en varios aspectos. Por un lado, la comunicación pasó a ser bidireccional: el robot envía a la estación base cuales son sus coordenadas actuales, y la estación base le responde a donde debe dirigirse. Esto implica un grado de coordinación mayor en los loops de ambos nodos. Además, se agregó la medida de la QoS y la orden de parada al robot en caso de mala señal. Por otro lado, se aumentó la autonomía de movimiento del robot: ya no responde a órdenes directas, sino que él mismo debe tomar las decisiones necesarias, esto es, valerse de la información del GPS, magnetómetro, encoders e infrarrojos para llegar al destino.

Los mensajes intercambiados siguen la sintaxis descrita en el capítulo anterior. Recordando, los mensajes que envía el robot comienzan con un 0 o un 1 indicando si llegó a destino, continúa con las coordenadas actuales y termina con el carácter de fin de trama. En el caso de los mensajes que envía la radiobase, empiezan con 0 o 1 para señalizar si la QoS es mayor al umbral mínimo, luego tiene las coordenadas a donde se quiere enviar al robot, terminando con el carácter final.

La figura 5.4 ilustra el protocolo de intercambio de mensajes, mediante un diagrama de tiempos.

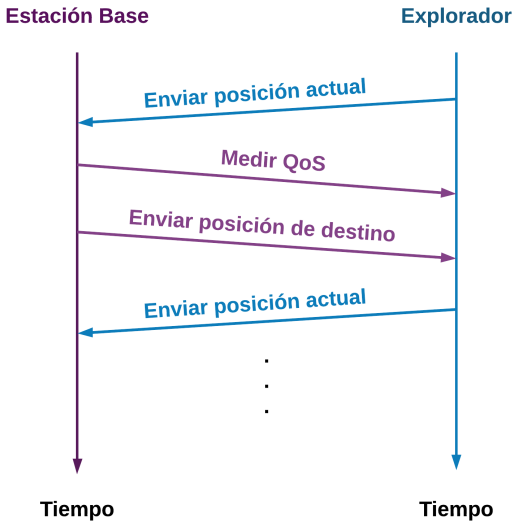


Figura 5.4: Diagrama de tiempos del segundo esquema de funcionamiento

La lógica que corre en cada nodo se puede describir de la siguiente forma:

1. Estacion base: Realiza un loop en el que se ejecutan las siguientes tareas:

   While (1):

   Recibir posición del robot por servidor UDP

   Medir QoS

   Actualizar dígito de avance

   Enviar posición de destino por cliente UDP

   Recordar que el dígito de avance se actualiza en base a la medida de la QoS; si ésta es menor a cierto umbral, el dígito queda en 0. .

2. Explorador: La lógica en el Raspberry Pi es un espejo de la implementada en la estación base. Naturalmente, los loops de ambos nodos deben estar sincronizados para poder implementar el intercambio de mensajes observado en la figura anterior. Para fijar ideas, en el primer paso de la comunicación, el Raspberry Pi envía la posición del robot, por lo que la radiobase se encuentra escuchando en el servidor de estacion_base.py a la espera de recibir ese mensaje desde GNU Radio. Bajando estas ideas a pseudocódigo, el script **com_serial.py** ejecuta el siguiente loop de tareas:

   While (1):

   Recibir posición del robot por puerto serial

   Enviar posición del robot por cliente UDP

   Recibir posición de destino por servidor UDP

   Enviar posición de destino por puerto serial

   Vale aclarar que cada vez que se reciben datos (ya sea a través de sockets o del puerto serial) se hace un parseo de los mensajes para chequear que tengan un formato válido y no sean basura. Además, en el caso de los mensajes recibidos por sockets, hay que eliminar la primera parte del mensaje, en la que se indica la dirección de destino.

Este modelo de funcionamiento es básico para pasar a la siguiente etapa. Para el andamiaje del sistema completo, es esencial que la estación base pueda dirigir a cada robot por separado a una determinada posición. No se puede pensar en controlar a ambos robots, si previamente no se controló a uno solo.

5.3.3. Tercer esquema: Cooperación entre robots

Vayamos entonces a los detalles del tercer esquema propuesto. La complejidad de este modelo aumenta del lado de la estación base, manteniéndose la lógica de

los robots prácticamente intacta con respecto al caso anterior.

En cuanto a la estrategia para asegurar una QoS mínima, se implementó un control mediante un umbral. Recordemos que, tal como se estudió en el capítulo "Sistema de Comunicación", decidimos medir la QoS a través de la tasa de éxitos. En la práctica se da que a medida que se alejan los nodos de la comunicación, se llega a una cierta distancia en donde la QoS empieza a decaer abruptamente. Se fijó entonces un umbral para la QoS bajo el cual la calidad comienza a descender rápidamente.

Inicialmente, y mientras la QoS sea mayor al umbral, se enviará al Explorador a su destino. Si se detecta que la QoS está por debajo del umbral, se activará al Repetidor (mediante un mensaje especial para despertarlo, y habilitarlo para que éste envié su posición) y se lo enviará a la posición anterior por la que pasó el Explorador, es decir, la última posición en la que se tuvo una QoS aceptable.

En la etapa en la que el Repetidor se aleja para llegar al destino indicado, también se realiza un control periódico de la QoS entre él y la radiobase para evitar que se pierda la comunicación. Si la calidad baja de cierto umbral establecido para el Repetidor, se manda la orden de que se detenga.

Una vez que el Repetidor llegó a su destino, se vuelve a medir la QoS entre la estación base y el Explorador. En caso que la QoS haya aumentado lo suficiente por efecto del Repetidor, el Explorador puede seguir avanzando hacia su destino.

A continuación se detallan los scripts que corren en cada nodo de la comunicación.

1. Estación base: El loop principal se divide en tres bloques diferenciados según el valor de la QoS. Para simplificar, en el siguiente pseudocódigo se omiten los detalles de los sockets y el armado de los mensajes:

```
While (1):
  Recibir posición del Explorador
  Medir QoS_Explorador
  If (QoS_Explorador <= umbral) and (Repetidor no activado):
    Enviar "despertate!" al Repetidor
    Recibir posición del Repetidor
    Medir QoS_Repetidor
    While (Repetidor no llegó a destino):
      Enviar al Repetidor a destino
      Recibir posición del Repetidor
```

Medir QoS_Repetidor

Medir QoS_Explorador

If (QoS_Explorador > umbral):

Enviar al Explorador a destino

Else:

Enviar al Explorador a destino

2. Explorador: El script **com_serial.py** es igual al del segundo esquema de funcionamiento.

3. Repetidor: Corre el mismo script que el Explorador, con el agregado de que inicialmente espera un mensaje de "despertate!" antes de enviar el primer mensaje.

Para clarificar, la figura 5.5 muestra el diagrama de tiempos con el intercambio de mensajes entre los tres nodos.

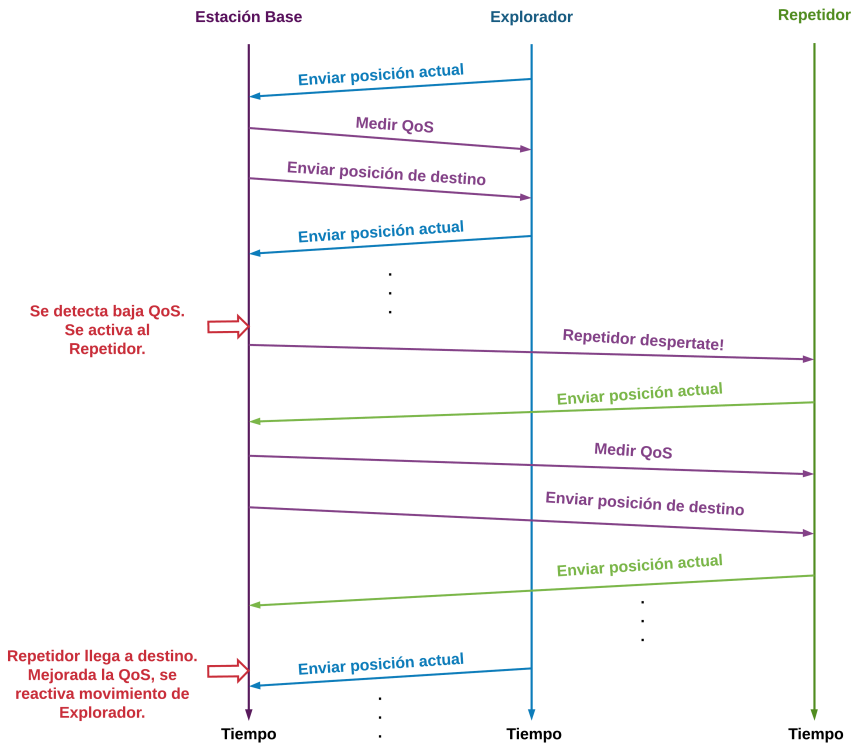


Figura 5.5: Diagrama de tiempos del tercer esquema de funcionamiento

Para validar este esquema de funcionamiento, se corrieron dentro de una misma máquina los scripts estacion_base.py, dos com_serial.py (uno simulando ser el

Explorador y otro el Repetidor) y un top_block.py simulando la comunicación entre todos los nodos usando un canal virtual y una serie de sockets. Dado que las variables de control del loop son la QoS y las posiciones de los robots (específicamente si llegaron a destino o no), se anuló la comunicación de las mismas, y se estableció el ingreso de esos valores mediante la entrada estándar. De esta forma, se tiene un control total de los escenarios de prueba, y se puede hacer pasar al código por todas las situaciones posibles. Mediante el agregado de mensajes de debuggueo, se verificó que la lógica implementada fuera la correcta.

5.4. Watchdog

Un watchdog es un mecanismo para detectar fallas en un sistema y actuar frente a ellas. Son esenciales para el correcto funcionamiento de sistemas embebidos y sistemas computacionales autónomos. Usualmente no es aceptable esperar que un usuario resetee manualmente un sistema autónomo luego que este se bloquea. En algunas situaciones incluso los sistemas no son humanamente accesibles, como el caso de las sondas espaciales.

Generalmente un watchdog se implementa a través de un temporizador. Durante la operación normal del sistema, el procesador reinicia periódicamente este temporizador antes de que llegue a dar su señal de time out. Si el temporizador llega a cero sin ser reiniciado previamente, se presume que hubo un problema, ya sea a nivel de hardware o software, que hizo bloquear el sistema. En ese caso, se envía una señal de reseteo al procesador. [27]

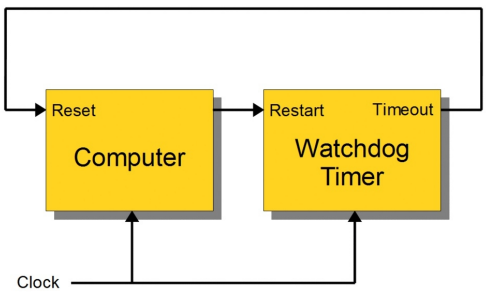


Figura 5.6: Funcionamiento general de un watchdog [27]

Tomando este concepto como base, se implementó un watchdog adaptado a las necesidades de los robots. El objetivo es brindarle a los robots la capacidad de auto recuperarse frente a distintos problemas que puedan aparecer.

A lo largo de la realización del proyecto, se encontraron dos situaciones que merecen ser monitoreadas por el watchdog: cortes en los procesos que corre Raspberry Pi y sobrecalentamiento de la placa. Por otro lado, no se dieron situaciones

donde el Raspberry Pi se bloqueara, por lo que se entendió que no era necesario que el watchdog implementado cumpliera la función "tradicional" de reseteo frente a un bloqueo.

5.4.1. Recuperación frente a cortes

Existen diversas razones que pueden generar que los procesos se detengan. La principal es que la máquina de estados que implementa la capa CSMA/CA llegue al estado de parada, previsto cuando se supera la cantidad de reintentos máxima o se llena la capacidad del buffer. Surge entonces la necesidad de contar con un mecanismo que restablezca la comunicación una vez llegado al ese estado.

Sin embargo, éste no es el único proceso que puede detenerse. Por ejemplo, se vio que bajo determinadas circunstancias la comunicación serial puede lanzar excepciones que deben ser atendidas.

Se implementó entonces el script de Python watchdog.py, el cual verifica periódicamente el estado de ejecución de los procesos. Para esto utiliza de la biblioteca OS de Python, la cual posee funciones para interactuar con el sistema operativo. En particular, la función os.system() permite enviar comandos directamente al Shell de Arch Linux. Utilizando el comando ps (process status), elabora una lista con los procesos que se encuentran activos en determinado momento. Luego chequea si falta algún proceso. En ese caso, lo reinicia mediante comandos al Shell.

Auxiliarmente al script de Python que implementa el watchdog, fue necesario forzar el cierre de los programas luego de un problema. En el caso de top_block.py, se agregó el envío de un mensaje de alerta a través del socket cuando se llega al estado de Stop. El script com_serial.py, que lee los mensajes que se transmiten por el socket, se encarga de cerrar a top_block.py en ese caso.

Por su parte, para el script com_serial.py se agregó código para el manejo de excepciones. En caso de que se lance una excepción (de cualquier tipo, sea proveniente del puerto serial o no), se cierra el script, nuevamente mediante comandos al Shell. Luego, el watchdog se encargará de reiniciarlo.

5.4.2. Control de temperatura

Si bien no está previsto que un Rasperry Pi se sobrecaliente en condiciones normales de uso, sí sucedió que como consecuencia de un descuido en la alimentación, se provocó un cortocircuito en una de las placas. Luego de este incidente, se decidió que podía ser útil agregar un control de temperatura en el watchdog, de forma que si detecta un aumento de la temperatura, que puede ser producida por una falla en la alimentación, se manda un comando al Shell para apagar el Rasperry Pi. El concepto es el de actuar frente a un síntoma lo más rápido posible para reducir daños. La medida de la temperatura se obtiene a través del comando

del Shell /opt/vc/bin/vcgencmd measure_temp.

5.5. Inicio automático

Para lograr autonomía completa de los robots, es necesario que los procesos se ejecuten automáticamente luego de la etapa de booteo, evitando así que el usuario tenga que iniciar manualmente los mismos.

Arch Linux soporta diversas formas para ejecutar programas automáticamente. Durante la realización del proyecto, se estudiaron dos métodos distintos, cada uno respondiendo a las necesidades que fueron surgiendo. Concretamente, los métodos implementados fueron los siguientes:

1. Daemons del Systemd
2. Script de Python

El primer método probado fue incluir los procesos como daemons del Systemd. En los sistemas operativos multitarea, un daemon es un proceso que corre en segundo plano, en lugar de ser controlado directamente por un usuario de forma interactiva. Inicialmente el daemon se encuentra en estado de latencia a la espera de que determinada condición ocurra y lo active.

En Arch Linux, los daemons son administrados por el Systemd [43]. Systemd es un gestor de servicios y sistemas para Linux. Es el encargado de inicializar todos los procesos necesarios para que Linux arranque [44]. Adicionalmente tiene la opción de arrancar procesos creados por el usuario.

En Systemd los demonios se definen y configuran en los llamados archivos de servicio. Estos archivos tienen que seguir una determinada estructura de tags (etiquetas). Por ejemplo, para el caso de top_block.service:

```
[Unit]
Description = Comunicación por GNU Radio
After=syslog.target
[Service]
ExecStart=/usr/bin/python2 /home/pi/top_block.py
```

La etiqueta After indica en que momento debe iniciarse el servicio. En este caso, se le está indicando que debe arrancar luego del booteo. La etiqueta ExecStart describe la ruta absoluta donde se encuentra el archivo a ejecutar y su intérprete.

Los archivos de servicio se alojan en el directorio /etc/systemd/system/. Una vez creado el servicio, se utiliza las herramientas del controlador de Systemd, Systemctl, para verificar el correcto funcionamiento de los daemons y luego ponerlos

en funcionamiento, mediante los comandos:

```
sudo systemctl start pi_arduino.service
sudo systemctl enable pi_arduino.service
```

Este método de arranque fue el usado en la primera etapa del proyecto, donde se buscaba mover el robot a través de órdenes directas ingresadas por el teclado de la estación base. Se configuraron dos daemons, top_block.service y com_serial.service. Ambos daemons eran ejecutados simultáneamente luego del booteo del Raspberry Pi.

En la siguiente etapa del proyecto, cuyo objetivo era enviarle al robot las coordenadas a las que debe dirigirse, se vio la necesidad de iniciar los procesos uno a continuación del otro para no perder los mensajes iniciales de conversación. El problema es el siguiente: top_block.py tarda unos 45 segundos en iniciarse (carga de las imágenes de la FPGA, etc), en cambio com_serial.py es prácticamente automático. En este escenario, la comunicación serial se abrirá mucho antes que la comunicación por SDR. Si Arduino tiene datos prontos para enviar, comenzará la comunicación por el puerto serial. Pero GNU Radio no estará pronto para recibir esos mensajes, con lo que se generará una excepción de la biblioteca de sockets y el mensaje finalmente se perderá.

Para evitar este problema, se ideó un método que permita el arranque de dos procesos, uno luego que termina de cargar el otro. Se utilizaron las herramientas que da el lenguaje Python, en particular, la biblioteca Time para incluir temporizadores, y la OS, para enviar comandos al Shell.

El script inicio.py sigue la siguiente lógica:

```
iniciar top_block.py
esperar 45 segundos
iniciar com_serial.py
esperar 5 segundos
iniciar watchdog.py
```

Luego de que los tres procesos se inician, deben seguir corriendo en paralelo. Para esto se utiliza el comando ampersand (&) al iniciar cada proceso. Este comando permite iniciar los procesos en background, por lo que se puede continuar usando el Shell sin tener que esperar que el script termine su ejecución.

Para que el script de inicio arranque luego del booteo, se editó el archivo etc/profile del usuario Pi. De esta forma, cada vez que se inicia sesión en el usuario Pi, el archivo inicio.py se iniciará automáticamente.

Complementariamente fue necesario establecer el login automático del usuario

Pi, sin necesidad de la intervención manual del usuario.

5.6. Conclusiones

En este capítulo se abordó el tercer pilar del proyecto: la integración del sistema completo. Coordinar aspectos tan diversos como las comunicaciones inalámbricas y la robótica no es un tema menor.

Se fundamentó la elección de la infraestructura a montar sobre los robots para cumplir con este objetivo. Luego se estudió un mecanismo para la comunicación entre procesos: los sockets de dominio Unix. Se estudiaron también aspectos complementarios que hacen a la autonomía del sistema, como el inicio automático y el control mediante watchdog.

El aspecto central del capítulo, fue la descripción de los tres esquemas de funcionamiento propuestos. Con el último esquema, se logró establecer una red de robots comunicados que aseguren una QoS mínima en todo momento. Destacamos no sólo el producto final al que se llegó, sino todo el camino recorrido. Realizamos un desarrollo iterativo, en el que en cada etapa se fueron agregando funcionalidades. En cada iteración se pasó por una fase de diseño, mediante diagramas de flujo o pseudocódigos. Luego por una fase de codificación, donde se pasaron esas ideas a código. Por último, se validaron los esquemas de funcionamiento propuestos mediante simulaciones que permitieran plantear escenarios de prueba, donde se conozca de antemano cuál es la salida esperada para dar por válido el funcionamiento.

Resta entonces validar el funcionamiento del sistema en la "vida real", aspecto que será abordado en el siguiente capítulo.

5.6.1. Trabajo a futuro

Si bien inicialmente se planteó lograr un algoritmo que no sólo asegurara una calidad de servicio mínima, sino que la optimizara, la propia realización del proyecto y las dificultades encontradas fueron dejando de lado la optimización, priorizando contar con un prototipo funcional que asegure conectividad en todo momento. Lógicamente, lograr operar en condiciones de optimalidad requiere un algoritmo más sofisticado, amparado en una sólida base matemática. De todas formas, se realizó un estudio de algunos papers vinculados al tema y se hizo el ejercicio de pensar qué modificaciones habría que hacer al algoritmo propuesto para lograr esta optimización.

En particular, el paper *"Network Integrity in Mobile Robotic Networks"* [105] propone un algoritmo para restringir el movimiento colectivo de los robots de forma de asegurar la integridad de la red, definiendo la integridad como la habilidad de soportar los parámetros de comunicación deseados. Se utiliza la teoría de grafos

como abstracción de la red, donde cada arco del grafo tiene un peso que representa la probabilidad de error de paquetes de cada enlace. La idea central del paper se basa en proponer un algoritmo de optimización para encontrar los puntos óptimos de operación de la red inalámbrica. Mantener la integridad de la red impone no sólo restricciones físicas (en la trayectoria de los robots), sino también restricciones en cuanto a las variables de la comunicación (en la elección del siguiente salto en el ruteo de los paquetes y en las tasas de transferencia). Se utiliza entonces un enfoque híbrido, descomponiendo el control del sistema en el dominio físico y en el de las comunicaciones. Las variables de comunicaciones se actualizan en tiempo discreto utilizando un descenso por gradiente distribuido en la función dual, mientras que el movimiento de los robots es controlado en tiempo continuo utilizando una función potencial apropiada para mantener la integridad de la red.

En concreto, llevar a la práctica el algoritmo descripto en el paper implicaría cuatro cambios importantes (pero realizables) en nuestra sistema de cooperación entre robots:

1. Los robots deben conocer en todo momento (tiempo continuo) su posición. Esto requiere afinar los métodos de localización utilizados, por ejemplo mediante filtros de Kalman o GPS diferencial. Además, cada cierto tiempo, los robots deben intercambiar entre ellos su posición.

2. Los robots deben calcular periódicamente la probabilidad de pérdida de paquetes que tienen con cada nodo del sistema. En nuestro sistema, solamente se calcula la QoS entre cada robot y la estación base, pero no entre ellos.

3. El esquema de cooperación entre los robots pasa a ser explícito, debido a que necesitan intercambiar periódicamente información (posición, tasa de transferencia de mensajes, probabilidad de pérdida y probabilidad de ruteo entre cada nodo). Éste punto, junto con el anterior, implican un uso mucho más intenso del canal para poder transmitir todos estos mensajes.

4. El control del sistema pasa de ser centralizado en la radiobase, a distribuido en los robots. Éstos deben correr el algoritmo de optimización para decidir hacia dónde moverse, y elegir la tasa de transmisión y las variables de ruteo, en base a la información obtenida por cuenta propia y la recabada a partir del intercambio con los demás agentes.

La optimización es un tema vasto, que involucra distintas ramas de la ingeniería y de las ciencias. Hay un gran interés en el estudio de la optimización de trayectorias de robots móviles para lograr cierto objetivo, y en particular, manteniendo determinado nivel de comunicación entre los agentes. Existen no pocas investigaciones sobre el tema, basta ver por ejemplo los enfoques estrictos propuestos en [22], [40]y [103], que mantienen en todo momento la calidad de los enlaces, seguidos de otros menos restrictivos que bajo ciertas condiciones admiten pérdidas de enlace [82], [104]. Queda planteada una línea interesante de trabajo a futuro, que mediante la utilización de la infraestructura propuesta en este proyecto implemente algún algoritmo para lograr operar en condiciones de optimalidad.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Pruebas

Como se ha planteado, el proyecto incluyó varios aspectos bien diferenciados: el desarrollo de un sistema de comunicación con subcapa MAC, la construcción de robots autónomos con capacidad de navegación y alimentación para la comunicación y la inteligencia, y el desarrollo de programas de control general que permitan el funcionamiento del sistema completo.

A su vez, el objetivo del proyecto es justamente la integración de las partes. La mayor parte de los desafíos encontrados en cada una de estas áreas se debió precisamente a incompatibilidades que surgieron al momento de integrarlas. Por ejemplo, la comunicación funcionaba con los USRP B100, y fue al momento de usar los B200 y B200 mini que comenzaron problemas y se tuvo que analizar la capa física y la construcción de los eventos. O la alimentación, que inicialmente permitía el funcionamiento del Raspberry Pi y el Arduino, pero que no fue suficiente cuando se agregaron los USRP. Por supuesto que el loop principal del sistema, que depende directamente del funcionamiento de cada parte, tuvo que ser ajustado en función de éstas: ajustando tiempos del Watchdog, del inicio automático, ordenando las comunicaciones serial y por SDR.

Si bien se realizaron comentarios sobre los tests de cada una de las partes en sus respectivos capítulos, esta sección se centrará en las pruebas y los resultados sobre el sistema completo. Se detallarán entonces a continuación algunas de las pruebas que realizamos, analizando límites y restricciones de nuestro sistema.

Manteniendo el criterio de que a cada paso que se daba en alguna dirección correspondió una prueba que garantizara el funcionamiento de lo realizado, se dividió el sistema, generando pruebas con aumento gradual de dificultad. El objetivo es confirmar lo realizado antes de aumentar la dificultad del conjunto.

En primer lugar, se trabajó sobre un esquema sencillo con un robot y la estación base. El objetivo fue lograr el establecimiento de la comunicación, y la indicación mediante órdenes de acciones a realizar por el robot. En un primer escenario, se le indicó al robot desde la radiobase hacia dónde dirigirse mediante comandos en

tiempo real (adelante, izquierda, derecha, atrás, parar).

La primera prueba permitió confirmar el funcionamiento de las capas de enlace y física, aunque no de la función de sensado del medio, por haber sólo dos nodos en comunicación directa. Es improbable que uno quiera transmitir cuando el otro está ocupando el canal, ya que uno de los nodos (el robot) solamente envía acuses de recibo, y estos siempre son inmediatos a la recepción de una trama, es decir durante la cuenta regresiva del temporizador de reintentos (y por ende la radiobase no está enviando nada).

Los resultados fueron buenos en el sentido que validaron el funcionamiento de las interfaces de comunicación del sistema. También confirmó la implementación del sistema de comunicación, de las funciones de movimiento y de los sensores IR del robot. Esta prueba permitió calibrar las constantes de movimiento de los robots, asociadas a la velocidad de los motores y a la lectura de los encoders.

La segunda prueba aumenta en dificultad: ya no alcanza con seguir órdenes directas, sino que se exige del robot la navegación hasta la ubicación destino. Esto implica testear el correcto funcionamiento de las funciones de Arduino: orientación, cálculo de trayectoria, avance, esquivamiento de obstáculos. Al mismo tiempo, depende de la comunicación en el sentido de que el robot envía su posición cuando va avanzando, y recibe confirmación de continuar de tener buena calidad de servicio. Al aumentarse la distancia, la comunicación empeora, principalmente debido a la baja potencia en transmisión del USRP B200 mini. No se contó en esta instancia con la participación del Repetidor, ya que esto implicaba que el test fuera mucho más amplio, incluyendo un loop que controlara ambos robots desde la radiobase, y se cambiaba el paradigma de ir confirmando cada paso dado.

En esta ocasión, y para evitar la pérdida de calidad de servicio que detuviera la prueba sin haber llegado el robot a la ubicación destino, se optó por acercar la radiobase al robot cuando se empezaba a perder calidad de servicio.

El resultado de esta prueba no fue alentador respecto de la utilidad del GPS, ya que en distancias pequeñas, el robot realizó más iteraciones de las necesarias (por errores en la resolución del GPS). Esto nos planteó el desafío de explorar alternativas en la navegación que lograran independencia del GPS, como es el caso de orientación por magnetómetro y avance por odometría, lo que se ensayó con éxito (sin incluir la comunicación por SDR). Esto fue presentado en el capítulo 4, y aunque no fue incorporado al loop principal por falta de tiempo, puede ser una alternativa interesante en trabajos a futuro.

Luego se realizó una prueba intermedia que incluyó el posicionamiento manual del Repetidor entre la estación base y el Explorador, para verificar el funcionamiento del mismo. El resultado fue bueno en el sentido en que la función de repetidor fue transparente para el sistema de comunicación, y permitió evaluar la existencia

de una mejora en la comunicación. Sin embargo, de estar muy cerca de la estación base el Explorador, el Repetidor es inútil y representa un incremento importante en la ocupación del canal (duplicando todos los mensajes, aún si estos fueron bien recibidos). Esto representa un problema, ya que entra en juego la función de sensado del medio producto de los mensajes del repetidor, y la calidad de servicio se ve seriamente afectada a cortas distancias, es decir, cuando es relativamente buena la QoS entre las puntas.

En la tabla 6.1 se resumen los resultados de calidad de servicio en relación con la distancia entre un robot y la radiobase. Cada medida en la tabla es un promedio de diez medidas de calidad de servicio.

Vale la pena realizar alguna observación sobre estos resultados. En primer lugar, es conveniente antes de testear el sistema setear la ganancia de la etapa de RF en la estación base para lograr una buena QoS a la distancia deseada. Es decir que no hay un valor dado que se mantenga siempre como el correcto, sino que varía entre 55dB-65dB para la transmisión y 50dB-60dB para la recepción. Una vez seteado este valor, los resultados son consistentes, pero pueden variar ligeramente frente a pruebas realizadas en otro momento, o con condiciones ligeramente distintas: desde cambios en la posición de las antenas hasta nodos a diferentes alturas han generado cambios no despreciables en los resultados de la comunicación. De ahí alguna diferencia que pueda surgir entre los resultados de las tablas 6.1 y 6.2.

Tabla 6.1: Tabla de la QoS en función de la distancia entre estación base y Explorador. Observar el período de buena QoS entre 0 y 6 metros, y el descenso abrupto pasados los 6 metros.

distancia entre Explorador y EB	**QoS (sin repetidor)**
(metros)	**(%)**
1	93
2	87
3	92
4	91
5	65
6	54
7	12
8	6
9	4
10	7

También se buscó estimar la distancia máxima a la que puede llegar el sistema de comunicación con el hardware utilizado, y las mejoras que introduce la utilización del repetidor. Los resultados aparecen en la tabla 6.2. El repetidor se ubicó

a 4 metros de la estación base, ya que esta fue la última medida por encima del umbral elegido (70 %), y corresponde a la que sería su ubicación destino del loop final. Luego se lo alejó a 5 y 6 metros para ver el alcance máximo del sistema, y porque en algunas medidas a 5 metros se tenía también muy buena QoS, lo que plantea que el umbral varía, estando ubicado a una distancia de entre 5 y 7 metros de la estación base.

Tabla 6.2: Tabla de QoS con y sin repetidor. Notar que a 5 y 6 metros hay una variación muy grande respecto de otras medidas. Esto se debe a que es una distancia cercana al umbral donde cae rápidamente la QoS, y presenta cierta inconsistencia en los resultados de una prueba a la otra.

	sin Repetidor	**Rep a 4 mts**	**Rep a 5 mts**	**Rep a 6 mts**
QoS entre EB y Rep	-	91	86	64
distancia entre EB y Exp	QoS (%)	QoS (%)	QoS (%)	QoS (%)
5	55	58	-	-
6	39	35	-	-
7	23	65	-	-
8	6	29	24	-
10	3	-	-	25

El resultado es que la mayor distancia alcanzable en situación de línea de vista (estación base y Explorador no tienen obstáculos entre sí) es de 10 mts para una QoS mayor a 20 %. El Repetidor funciona bien cuando la distancia es tal que la QoS entre las puntas es muy baja, pero cuando ésta no es tan baja, el efecto positivo del repetidor se cancela con la aparición de muchos mensajes y la mayor ocupación del canal. Este es el caso a 5 y 6 metros, en que se observa que la QoS no varía mucho o incluso empeora (los resultados de la tabla son promedios de diez medidas). Es decir: el Repetidor mejora la comunicación cuando el Explorador se encuentra pasado el umbral en que desciende abruptamente la calidad de servicio. Este umbral se ubica entre los 5 y los 7 metros. Este resultado confirma lo que se plantea en el loop final: el Repetidor debe entrar en juego una vez que el Explorador sale de la zona de buena QoS, y debe ubicarse dentro de la zona de buena QoS.

La última prueba incluyó al Repetidor, y consistió en probar al sistema completo. Es decir, enviar al Explorador hacia una ubicación destino, y cuando la QoS descendiera por debajo de un cierto umbral, enviar al Repetidor a una posición intermedia entre éste y la radiobase para mejorar la QoS. Mejorada la QoS, el Explorador puede avanzar hacia el objetivo, ya sea llegando o acercándose al mismo.

Esto ya es el esquema completo de funcionamiento del sistema. Se buscó testear en esta ocasión la estimación de la ubicación del Repetidor, así como el loop

principal de funcionamiento y la mejora de la comunicación por la incorporación del repetidor.

Los resultados obtenidos en el conjunto de las pruebas realizadas establecen una serie de restricciones para el funcionamiento de nuestro sistema. Siendo éste un trabajo de exploración en torno a la viabilidad de una implementación de SDR y robótica móvil, las restricciones halladas presentan una fuente interesante de problemas. En el próximo capítulo veremos que para varios de estos problemas proponemos soluciones, y a su vez planteamos líneas de trabajo a futuro.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Conclusiones

Queremos empezar por destacar el cumplimiento del objetivo general del proyecto: se dispone hoy de un acumulado de conocimientos en torno a la combinación entre SDR y robótica móvil, y se logró desarrollar un sistema funcional de radiobase y nodos móviles trabajando en una red inalámbrica construida sobre SDR.

Este proyecto, como ya fue mencionado, cuenta con un componente transversal a las partes, que es la implementación de una solución al problema planteado:

Que un robot se acerque lo más posible a una ubicación dada manteniendo una QoS mínima en un sistema comunicado a través de SDR, contando con capa MAC, incorporando un repetidor móvil de ser necesario.

Esta consigna, y por ende este proyecto, planteó como tarea intrínseca para su desarrollo la familiarización con el hardware y software necesario para su realización, que es heterogéneo, y en este sentido el grupo de proyecto decidió concentrar sus esfuerzos acorde a la división planteada de las áreas de trabajo. Es decir que hubo recursos particularmente concentrados en el sistema de comunicación, en la navegación y en la inteligencia general del sistema.

Esto trajo muchas consecuencias: por un lado la especialización permite a priori un enfoque con mayor profundidad, presentando diversas soluciones posibles para los problemas particulares que van surgiendo en cada área; sin embargo la integración puede resultar difícil, ya que sin un diseño original adecuado, cada parte corre riesgo de converger a soluciones incompatibles entre sí.

La principal conclusión del proyecto en cuanto a los aprendizajes adquiridos, es la importancia de la etapa de planificación y diseño, así como la necesidad de establecer ciertas reglas (porque no, protocolos) para el trabajo en equipo.

Este capítulo continúa presentando las conclusiones respecto de la gestión del proyecto.

Le sigue el desarrollo de los problemas y las limitaciones de cada una de las partes, así como las soluciones halladas (o pensadas). También se analizan las restricciones globales del sistema y posibles mejoras.

Finalmente se analizan posibles líneas de trabajo y aplicaciones que vayan en el sentido del proyecto, así como se desarrollan ideas respecto de lo planteado en la motivación, vinculadas a la utilidad y las posibilidades de SDR y robótica cooperativa.

7.1. Gestión del proyecto

El proyecto tuvo un desarrollo bastante diferente al planteado en el plan de proyecto original.

Se mantuvo la lógica del desarrollo modular de cada parte para luego pasar a su etapa de integración. Se dieron todos los riesgos enunciados en el plan de proyecto, lo que generó atrasos y reasignación de recursos. En particular, el desarrollo de la CSMA/CA y su implementación en los USRP B200, la solución del problema de alimentación para la autonomía de los robots y el funcionamiento del GPS llevaron tiempos mayores a los previstos.

Esto consumió recursos asignados al desarrollo del algoritmo de optimización, el cual sólo pudo estudiarse y pensar en posibles implementaciones.

Contrapuesto a nuestras previsiones originales, planteamos la realidad de la gestión del proyecto, centrándonos en aspectos que afectaron la planificación original.

7.1.1. Supuestos

- *El grupo ARTES suministra una implementación CSMA/CA de la capa MAC.*
  El desarrollo de la CSMA/CA se incluyó como parte del proyecto y debió ser asumido por el grupo. Esto permitió un conocimiento con mayor profundidad del funcionamiento de la CSMA y de GWN, GNU Radio y SDR en general. Por otro lado, la capa física que también se consideraba funcional a priori, terminó resultando insuficiente para los USRP B200, lo que también conllevó a la necesidad de estudiarla con detenimiento y modificarla para el desarrollo del proyecto.

- *Se cuenta con el hardware apropiado para la realización del proyecto, bajo verificación de parte del grupo ARTES y tutores.*
  El hardware fue apropiado para el desarrollo del proyecto. Esto no impide que la etapa de familiarización con el hardware (muy heterogéneo) exigiera

tiempo y generara dificultades. Por ejemplo se asumió un diseño dado que no se revisó cabalmente al inicio del proyecto, lo que generó incompatibilidades entre soluciones propuestas. En particular, la resolución del GPS es prácticamente incompatible con el alcance de los USRP B200, siendo del mismo orden de magnitud.

- *El presupuesto asignado para desarrollar el proyecto es suficiente.*
  Este supuesto se cumplió. Se contó en todo momento con apoyo de los tutores para la realización del proyecto, en particular, como veremos en el análisis de costos, aún en escenarios de sobrecosto se pudo remediar la situación.

7.1.2. Riesgos

En el plan de proyecto se planteaban los 5 riesgos más relevantes del Proyecto como:

1. *Los insumos necesarios para el montaje del segundo robot no llegan en el tiempo previsto, considerado poco probable y con un nivel de impacto leve.*

   Si bien se tuvo este problema, las demoras en el hardware del segundo robot dieron también la oportunidad de mejorar la compra, buscando un chasis más robusto y motores más poderosos, para evitar problemas de peso y movilidad del robot. El desarrollo del proyecto no se vió demasiado afectado por esta situación, a pesar de que se tuvo que intercambiar tareas: el repetidor se probó entre computadoras personales, y los ajustes para el manejo del hardware del segundo robot tuvieron que postergarse.

2. *Los insumos adquiridos son insuficientes para el funcionamiento del hardware debido a incompatibilidad entre los componentes, fallas técnicas, roturas, pérdida. Este riesgo se estimó como muy probable y con un impacto leve.*

   Se dió el problema previsto. En particular, dos de los USRP B200 mini adquiridos llegaron rotos. Esto atrasó las pruebas de la comunicación sobre los USRP B200, ya que se continuó trabajando sobre USRP B100, sin prever que surgirían dificultades propias de los B200.

   Al inicio del proyecto (Junio 2016), se realizó la compra de un B200 mini, previendo demoras en su llegada y la necesidad de realizar pruebas para confirmar su funcionamiento, y así poder evaluar la compra de un segundo B200 mini. El B200 mini llegó sobre Agosto de 2016, pero demoró hasta principios de Octubre la adquisición del mismo por parte del grupo de proyecto, por diferentes trabas y problemas administrativos/burocráticos.

   Al momento de probar su funcionamiento, se detectó el desprendimiento de un transformador. Esto motivó el reclamo frente al distribuidor local (Coasin Instrumentos), y ante respuestas evasivas por su parte, frente a Ettus. Finalmente, se recibieron dos USRP B200 mini en reemplazo del original, que permaneció en poder del grupo de proyecto. Uno de los dos B200 mini que llegaron también vino con el desprendimiento del mismo

transformador. Luego de estudiar los esquemáticos del USRP B200 mini, se procedió a solicitar ayuda a Pablo Pérez, docente del Departamento de Electrónica del IIE, que colaboró soldando dichos transformadores. Esto permitió tener 3 USRP B200 mini funcionando, a los cuales se les realizaron pruebas de transmisión y recepción de distintas señales analógicas y digitales para verificar su correcto desempeño.

También se quemó un Raspberry Pi: probando con usar la alimentación externa del USRP B200 y estando este conectado por USB al Pi (alimentado desde una PC), el Pi comenzó a aumentar la temperatura y terminó quemándose. Esto provocó la implementación del control de temperatura en el watchdog, y exigió un mayor cuidado con el trato del hardware, incorporando disipadores de calor en los Raspberry Pi.

3. *La implementación de la capa CSMA/CA se atrasa o no funciona, representando un riesgo de alto impacto y probabilidad moderada.*

   Como se comentó en los supuestos, la CSMA/CA no existía al inicio del proyecto, si bien se contaba con material de trabajo (bitácoras, pruebas) de integrantes del grupo ARTES en tareas similares. El proyecto entonces implicó el desarrollo de la CSMA/CA, lo que fue la primera tarea asumida respecto de la comunicación, y también generó atrasos respecto de las pruebas en hardware y la integración del sistema completo.

4. *La estimación de la duración de las tareas del WBS resulta incorrecta y se requiere de mayor tiempo para finalizar el proyecto, lo cual se estimó con probabilidad de ocurrencia moderada e impacto alto.*

   Este es un riesgo muy general, y se dió con razonable ocurrencia para muchas de las tareas llevadas adelante. Es difícil predecir al inicio de un proyecto cuánto tiempo va a requerir cada tarea si no se tiene experiencia en el área de trabajo. En nuestro caso, la inexperiencia del grupo respecto a trabajar con hardware tan heterogéneo implicó mayores tiempos para su familiarización, así como para el desarrollo en sí de las funciones a ejecutar en cada parte. Tuvo como consecuencia el redimensionamiento del proyecto, excluyendo la implementación del algoritmo de optimización para la determinación de la posición del repetidor.

5. *El algoritmo de maximización propuesto no ofrece resultados aceptables en su implementación. Este riesgo se consideró poco probable y con un impacto medio.*

   Como se dijo, no se implementó el algoritmo de optimización. Sin embargo, se realizó un estudio de papers vinculados a la temática, y se consideró que es viable implementar una versión sencilla del algoritmo presentado en un paper en particular, realizando relativamente pocas modificaciones a nuestro sistema.

7.1.3. Costos

El análisis de costos se centra en las compras de hardware. Es decir que, a diferencia del presupuesto planteado en el plan de proyecto, no se considera el gasto en recursos humanos.

También se realizaron compras más adelante en el proyecto, por ejemplo de un segundo regulador DC-DC, disipadores, GPS. Éstas no tuvieron grandes costos de importación, y se encuentran incluidas en la tabla de gastos que figura en el Anexo titulado "Análisis de costos". En dicha planilla se puede observar la ejecución de los gastos para la realización del proyecto. En particular, las principales diferencias con la previsión se dan en que la cantidad de Raspberry Pi terminó siendo 3 (prevista en 2), así como la cantidad de USRP B200 mini adquiridos fue de 3 (al costo de 1).

El proyecto pudo realizarse con el presupuesto disponible, con un sobrecosto de un 16 %. Este se debió principalmente al costo asociado a la mejora en la elección del chasis del segundo robot, en el tercer Raspberry Pi y en costos de importación.

7.2. Conclusiones respecto de los problemas encontrados

Repasemos algunas de las conclusiones a las que se había llegado para cada parte.

Por un lado, el sistema de comunicación implementado cumple con los objetivos específicos y el alcance planteados: se cuenta con una capa MAC con CSMA/CA y se realiza una medida de la calidad de servicio considerándola como la tasa de éxitos. Se logra la comunicación entre los tres nodos, así como el funcionamiento a través de la inclusión de un repetidor.

Sin embargo, como se vió en el capítulo 3, la comunicación tiene varias limitaciones. A continuación profundizaremos sobre algunas de éstas y propondremos soluciones a explorar.

La primera se debe a la potencia de los USRP B200. Principalmente en transmisión, ya que en recepción aumentar la potencia también aumenta el piso de ruido y la potencia recibida del pulso que el propio USRP envía, y como vimos, esto puede dificultar la recepción de pulsos ajenos. En este sentido, se llegó casi a los umbrales máximos de ganancia posible para los SDR. Las ganancias de los robots y la radiobase resultaron como indica la tabla .

Con respecto a las ganancias, vale la pena destacar que la potencia real de salida tiene un vínculo directo con la alimentación. Es conveniente ajustar la ganancia por ejemplo de la estación base, que es la más versátil por estar alimentada desde una computadora portátil, y tiene mayor margen respecto de los límites planteados por el equipo, puede verse que estando en 60dB como valor de ganancia podemos

Equipo	Ganancia en transmisión dB	Ganancia en recepción dB
Estación Base (B200)	60	55
Explorador (B200 mini)	75	55
Repetidor (B200 mini)	75	55

Tabla 7.1: Ganancias de los amplificadores de las antenas para cada equipo.

aumentar 15dB de ser necesario. Claro que aumentar demasiado puede generar la autoescucha, en cuyo caso uno debería reducir la ganancia de la antena de recepción. Recomendamos usar valores entre 50 y 70 en transmisión, y entre 40 y 60 en recepción.

Sin repetidor y con estas ganancias, se llegó a que el alcance con QoS buena (mayor a 70 %) es de 4 mts. Entre 5 y 7 mts se tiene una QoS que varía entre 65 % y 11 % . Con distancias mayores a 7 mts, la comunicación ya se ve seriamente comprometida, teniendo una QoS del 10 % o menor.

Con el repetidor en funcionamiento, posicionado entre la estación base y el Explorador se obtiene que la QoS buena se mantiene hasta 7 mts. Entre 7 y 10 mts se tiene una QoS que varía entre 65 % y 25 % . La distancia máxima a la que se puede llegar con una QoS mayor al 20 % es de 10 mts.

Una posibilidad es entonces mejorar el sistema de comunicación entrando en la modulación, es decir analizando si es razonable cambiar la modulación para tener mayor fidelidad con menor potencia, o si es posible eliminar el problema de la "autodetección" de los pulsos. Un estudio a fondo del AGC2, analizando los cálculos que realiza y las tasas de muestreo en juego, podría arrojar luz sobre este aspecto. Pensar en usar OFDM, u otras modulaciones, también puede ser útil en este sentido.

Otra posibilidad es utilizar amplificadores para aumentar la salida de las antenas, algo que no se exploró, pero podría aumentar el alcance. En este caso hay que analizar si no se aumenta excesivamente el consumo energético, lo que podría resultar en menor potencia para la transmisión, o en una alimentación insuficiente del sistema completo. En este último caso, puede verse si no es posible alimentar los amplificadores externamente (que compartan alimentación con los motores, por ejemplo). Otra posibilidad es utilizar antenas cuyo patrón de radiación esté más dirigido hacia el plano en que está operando nuestro sistema. También se podría pensar en un sistema que regule la ganancia de transmisión y recepción a medida que varía la distancia, buscando mantener la QoS.

Por otro lado, la modificación en la construcción de las tramas para aumentar el encabezado inútil empeora la capacidad útil de nuestro sistema. La trama está

compuesta por encabezados con información y otros que no aportan información, sino que aumentan el tamaño del paquete para evitar que se dañe el contenido útil en el pasaje por el AGC. Dentro de los encabezados con información, tenemos el encabezado que podríamos llamar de capa de enlace: unos 465 bytes, que incluyen direcciones de origen y destino, número de secuencia, tipo de dato, nombre del dato, y opciones varias (podrían pensarse como equivalentes a las "banderas"). Por otro lado tenemos el encabezado de capa física, que incluye al código de redundancia cíclica (uno 20 bytes) y unos 10 bytes al final de la trama, así como el encabezado agregado para que el mensaje sea bien recibido por el AGC. Este encabezado (que no contiene información) representa una porción demasiado grande de la trama (300 bytes sobre los 800 de encabezados totales). Un trabajo de exploración del AGC2 podría ayudar a achicar su tamaño, obteniendo mejores resultados. La posibilidad de fragmentar archivos y transmitirlos podría ser una idea interesante para ver si con cargas útiles mayores no son necesarios encabezados "inútiles" tan largos. Por otro lado, de buscar una aplicación con mayor velocidad, se podría cambiar la forma en que se construye el evento, sin agregar diccionarios sino que simplemente concatenando en un string el conjunto de parámetros necesarios y la carga útil (tal vez codificando parte de la información).

La modulación puede ser mejorada para obtener mayores anchos de banda: aumentar la tasa de muestreo ya es un avance, para lo que se debería estudiar la capacidad de procesamiento en el Raspberry Pi. De usarse en computadoras con mayor capacidad, este problema está más resuelto, aunque no se buscó el límite de la velocidad de nuestro sistema en estas condiciones. Otra alternativa es pensar en revisar a fondo la capa física, por ejemplo para usar OFDM, que ahorra la utilización del AGC, y podría resolver entonces varios problemas.

Sobre la CSMA/CA, se pueden ensayar mejoras como la solicitud de envío y su respuesta, con mecanismos como RTS/CTS. Esto sólo tiene valor si se logra disminuir el encabezado, ya que de lo contrario resultará mucho más en saturación del medio que en optimización de los intercambios. Otra posibilidad es concentrarse en mejorar la selección de los parámetros: tiempos de reintentos y backoff, cantidad de reintentos admisibles.

Más allá de estas limitantes, el sistema demostró ser implementable. Esto valida la idea detrás de GWN: el conjunto de herramientas que provee GWN es aplicable a escenarios de simulación, prueba y desarrollo para redes de datos. Con la llegada de equipos más poderosos al Uruguay y el desarrollo de GWN en C++ se puede pensar en explorar varios límites, como se ha hecho en otros países, y donde se han obtenido resultados interesantes: una MAC sobre SDR puede llegar a tiempos similares a una MAC de 802.11. [57]

También tiene valor en sí la construcción de bloques, que permite colaborar en el trabajo de desarrollo de GWN y de intercambio en una comunidad dinámica y abierta como es la de GNU Radio.

El conjunto de CSMA/CA y medida de la calidad de servicio puede ser útil para pruebas o simulaciones, más allá de aplicaciones a la robótica móvil. Se puede pensar en usos hacia el análisis de redes de nodos móviles, por ejemplo.

Respecto de los robots y su sistema de navegación, podemos decir que se cumplió el objetivo planteado. Se conoce la posición de los robots en todo momento, con una resolución de 3 metros. Tienen autonomía energética para un funcionamiento de hasta 2 horas. Los robots se orientan y trazan un camino hasta la ubicación destino, esquivando obstáculos de ser necesario.

Sin embargo, surgieron no pocas restricciones en torno a la combinación de SDR y robots móviles. Por un lado, para el alcance posible del sistema de comunicación (del orden de 6 metros), el GPS es un instrumento que no conviene utilizar, o al menos no sólo. De contar con sistemas realimentados que estimen la posición en base a varios sensores (encoders, magnetómetro, GPS) y realicen correcciones de los errores introducidos, se podría utilizar el GPS. Pero el propio error en su resolución (del orden de 3 metros en los mejores casos) es similar a la distancia que mejora la introducción del repetidor, por lo que no hay mucha compatibilidad en este sentido.

Para resolver esto, surgió un plan B, considerando los problemas que el GPS introducía. La solución propuesta esquiva la utilización de GPS, proponiendo un sistema que puede utilizarse en espacios cerrados, o con mala calidad de señal de GPS. El esquema de funcionamiento en este caso es de cálculo de la posición relativa al inicio, considerado como las coordenadas [0, 0]. El movimiento del robot se realiza a partir de la orientación dada por el magnetómetro y las cuentas que realizan los encoders. En distancias pequeñas, este sistema mostró funcionar bien. Su principal problema reside en que el error de las medidas de los encoders es acumulativo, lo que se buscó corregir con limitar el apartamiento posible de la dirección trazada como camino. Esta diferencia se mide con el magnetómetro, y de ser mayor a un cierto umbral, el robot se detiene. Durante el movimiento del robot, se acumulan las medidas y se estima el error cada cierto tiempo, lo que permite corregir las desviaciones. En distancias como las que manejamos en el proyecto esta solución tuvo buenos resultados.

Otra solución planteada fue la compra de GPS diferenciales, que mejoran mucho la precisión, y permitiría, sin grandes modificaciones excepto en Arduino y la adquisición de datos del GPS, utilizar nuestro sistema ya implementado.

También se deben explorar alternativas respecto de la alimentación. Como dijimos, este es un componente fundamental para aumentar el alcance de la comunicación. Se realizaron respecto de este punto numerosas pruebas, que incluyeron alimentación del Raspberry Pi desde el regulador DC-DC a través de los pines y de un conector USB artesanal. Otra posibilidad es mejorar la alimentación usando

fuentes mejores. De usar una fuente más potente y estable se podría buscar unificar la alimentación de todo el sistema, que en nuestro caso no fue posible.

Para la movilidad de los robots, considerando el peso que deben soportar (entre 1 y 2 kg), se proponen diferentes soluciones. Una es la utilización de ruedas Omni wheel que disminuyen el rozamiento, facilitando el giro de los robots. Otra posibilidad es considerar robots con mejores características de chasis y motores, capaces de moverse con el peso planteado.

En conclusión es posible usar SDR para robótica móvil, pero tenemos aún obstáculos y limitantes que sortear.

Con respecto a la inteligencia, se encuentran pocas restricciones que podemos enfrentar. Ya se mencionó que se tuvo que disminuir la tasa de muestreo de la comunicación por no ser soportada por el procesador de tener valores más altos. Una computadora con las características de tamaño y peso como Raspberry Pi y de rendimiento superior podría considerarse. No se realizaron búsqueda extensas al respecto, dada la popularidad y las ventajas (ya planteadas en el capítulo 5) del Raspberry Pi.

Lo que sí tiene mucho margen para mejorar es la inteligencia del sistema. En este vértice no se pudo, por falta de tiempos, implementar el algoritmo de optimización que se estudió. Si se logró analizar cómo debería ser esta implementación. Esta es una línea muy interesante de trabajo, y permitiría estimar la ubicación del repetidor para mejorar la QoS. Incluso en escenarios de contar con más nodos, se pueden determinar las posiciones de estos para tener una red de nodos móviles con cierta QoS mínima.

Las modificaciones necesarias del sistema para implementar el algoritmo son:

- Todos los nodos deben conocer su posición y la del resto de los nodos: en nuestro caso sólo la estación base conoce todas las posiciones.

- Todos deben tener medidas de la QoS entre sí: en la solución propuesta sólo la estación base conoce la QoS, y esta se halla entre la estación base y los dos nodos restantes.

- Todos los nodos deben comunicarse periódicamente para intercambiar información (posición, tasa de transferencia de mensajes, probabilidad de pérdida y probabilidad de ruteo entre cada nodo): en nuestro sistema actual, los robots no intercambian información “explícitamente” , sólo se comunican en un esquema de repetición de mensajes.

- Cada uno deberá realizar parte del algoritmo de optimización para hallar su nueva posición: en nuestro sistema sólo la estación base realiza cálculos de ubicaciones de destino.

En este sentido, una ventaja que tiene nuestra solución es que no es difícil incorporar más nodos a nuestro sistema. Necesitarán de dirección de capa de enlace, el mismo hardware (o similar) al usado en los dos robots ya construidos, y una tarjeta SD clonada y ajustada de las que hoy se utilizan para el Raspberry Pi de cualquiera de los dos robots. Las modificaciones en la capa de enlace y el trabajo sobre hardware y software de propósito general permiten que nuestro esquema sea escalable en este sentido.

Esto se alinea con la idea de ensayar redes con más nodos, ya sea para el estudio de redes móviles o de robótica cooperativa.

Finalmente, queremos realizar una serie de consideraciones sobre el trabajo desarrollado en torno a SDR y robótica móvil.

Creemos que las SDR son una tecnología pujante, que tarde o temprano se abrirá camino por sobre el hardware especializado. En este sentido es destacable el esfuerzo que se ha realizado de estudio y desarrollo en torno a una tecnología de punta, ampliando para el Uruguay y la comunidad internacional las posibilidades existentes en torno a su utilización hacia redes de datos. Este proyecto demuestra la viabilidad de GWN como conjunto de herramientas a seguir desarrollando, y a su vez permite pensar en futuros trabajos sobre SDR en el Uruguay. Todavía queda mucho por mejorar de la capa de enlace implementada, del estudio de los límites de SDR en estas funciones frente a las soluciones comerciales.

Aún queda planteada la interrogante sobre el futuro de las tecnologías de las telecomunicaciones. Sin dudas, este futuro dependerá en gran medida de decisiones ajenas a las que se pueden tomar en Uruguay: las tomará el puñado de multinacionales vinculadas a estas industrias y sus Estados asociados. Pero es claro que prepararse desde la academia, planteando soluciones concretas a problemas reales (donde el ejemplo más claro es el receptor de televisión digital [35]), es un camino a transitar para estar preparados ante diferentes escenarios, y para poder proveer soluciones autóctonas a los problemas del mañana.

Creemos que la ruptura de la dependencia en que están sumidos los países y pueblos pobres necesariamente va de la mano con el desarrollo de capacidades locales técnicas y tecnológicas que permitan trazarse caminos propios. En esta línea, el trabajo sobre software libre y colaborativo nos permite acceder a la resolución de problemas de avanzada y participar en la construcción de ciencia y conocimiento. Los debates respecto de la apropiación del conocimiento y de las consecuencias del desarrollo de la humanidad no están cerrados, y ni la academia ni el país pueden tomar una posición neutral: el impulso de la contribución colectiva y el intercambio de conocimiento a través de herramientas como el software libre son una necesidad para la participación del Uruguay en proyectos de alto nivel y tecnología de punta.

Es por esto que queremos que nuestro esfuerzo hacia la implementación de

varias funciones de GNU Radio, Arch Linux, GWN, Arduino, Python, sea útil para el resto de la humanidad. Para esto, se desarrolló un Git `https://github.com/MartinRandallC/RoCo/wiki`, en donde se encuentran subidos los Anexos que corresponden a soluciones encontradas y links de utilidad que se relacionan con el mundo de software. Esta pequeña contribución a comunidades que nos permitieron llevar adelante el proyecto puede tener más o menos valor, eso no podemos preverlo. Pero el trabajo de sistematizar nuestras soluciones y ponerlas a disposición nos parece una forma de retribución y, sobre todo, pasamos a ser parte activa y ya no sólo pasiva de estas comunidades.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Plan de proyecto

A.1. Resumen

A.1.1. Título del proyecto

Robots Comunicados

A.1.2. Nombre corto

RoCo

A.1.3. Integrantes

- Gabriel Bibbó
- Mariana Gelós
- Martín Randall

A.1.4. Tutores

- Dr. Ing. Pablo Belzarena
- Dr. Ing. Federico La Rocca
- Dr. Ing. Leonardo Steinfeld

A.1.5. Cliente

Grupo ARTES, IIE – FING.

A.1.6. Fuente de financiación

Proyecto Grupos CSIC – Grupo ARTES.

A.1.7. Fecha estimada de finalización y horas de trabajo previstas

Se prevee que el proyecto termine el 30 de Setiembre del año 2017.

Esta predicción se basa en que el proyecto se realizará con una dedicación total de 2700 horas.

A.1.8. Entregables intermedios

Hito 1, 15/02/2017
Objetivos:

- Tener un robot que responda a órdenes básicas, comunicadas a través de GNU Radio desde la PC.
- Tener avanzado el estudio teórico previo del algoritmo de optimización, y seleccionada en conjunto con los tutores la variable a maximizar.
- Haber realizado la compra, montaje y prueba del 2do Robot.

Alcance:

- Montaje y prueba del 1er Robot.
- Familiarización con GNU Radio y Arduino, montaje de comunicación entre Arduino y Raspberry Pi.
- Del Bloque Integración Robots, la integración del GPS y Magnetómetro.
- Módulo de órdenes básicas y Comunicación GNU Radio - Raspberry PI, del Bloque de Comunicaciones.
- Estudio Teórico Previo y Selección de Variable a Maximizar del Bloque Algoritmo de Maximización de la QoS.
- Compra, montaje y verificación del funcionamiento del 2do Robot.

Hito 2, 15/06/2017
Objetivo:

- Tener los dos robots funcionando con el módulo de traslado a nueva posición.
- Tener implementado el Algoritmo de Maximización.

Alcance:

- Módulo de Traslado a Nueva Posición y Módulo de Repetidor, del Bloque de Comunicaciones.
- Módulo de Estimación de QoS y Algormitmo de Maximización de la QoS terminados.

- La tarea de Integración del Módulo de Órdenes Básicas a los robots (por ejemplo, indicando nueva posición a la que deban llegar), del Bloque de Integración Del Sistema Completo.

Dentro de las tareas a realizar previo a cada hito se incluyen: documentación de lo realizado; análisis y ajuste del diagrama de Gantt (calendario, recursos, etc); preparación de cada hito (o defensa).

A.2. Descripción del proyecto

Se le encomendará la siguiente tarea a dos robots y una estación base que se encuentran comunicados a través de SDRs. La tarea consistirá en que uno de los robots deba acercarse lo más posible a una coordenada (latitud, longitud) manteniendo un nivel mínimo en la calidad de comunicación. La calidad de las comunicaciones deberá ser la mejor posible pero tendrá un mínimo especificado de antemano (medido por ejemplo a través de un SNR mínimo o una probabilidad de error de paquetes máxima). Si la coordenada objetivo es cercana a la base, el segundo robot no tendrá porqué intervenir en la tarea, pero si la coordenada es más lejana, el segundo robot deberá buscar su mejor posición para actuar como repetidor y brindar una mejor calidad de comunicaciones y al menos asegurar el mínimo especificado.

Una posible solución para el problema es asumir que la calidad de las comunicaciones tiene un modelo que decrece con la distancia pero que no hay multipath ni fading y por tanto la calidad depende solo de la distancia. Esto no es cierto en un entorno con fading y multipath y en ese caso más real lo que hay que hacer es: o tener un modelo previo del territorio, o en cada posición hacer una exploración del canal en direcciones aleatorias antes de decidir la dirección en la cual moverse.

La estación base será un PC con GNU Radio y una placa Universal Software Radio Peripheral (de aquí en más USRP) B100.

Cada robot contará con:

1. 4 motores con encoders
2. Driver de dos canales
3. GPS
4. Magnetómetro
5. Microcontrolador Arduino que de acuerdo a las coordenadas que reciba se dirigirá hacia ellas utilizando el GPS y el magnetómetro.
6. Deberá tener también por ejemplo un sensor de ultrasonido para detectar obstáculos.

7. Un USRP B200

8. Un Raspberry PI comunicado con el Arduino y el B200 a través de USB. El PI será encargado de comunicarse con el otro robot y la base a través del B200. Será el encargado de decidir hacia donde moverse y se lo informará al Arduino para que lo haga.

El Raspberry PI correrá un Linux con distribución ArchLinux. Sobre el ArchLinux correrá GNU Radio.

Para la comunicación en capa física y MAC se partirá de una capa física básica funcionando y de una MAC con una implementación simple con CSMA/CA. En el transcurso del proyecto es muy probable que se deba mejorar o modificar la capa física y la MAC de acuerdo a necesidades que surjan del mismo.

Como antecedentes contamos con la experiencia del grupo ARTES del departamento de Telecomunicaciones en el proyecto CSIC - Grupos. En particular una de las líneas que se ha trabajado es el desarrollo de librerías e implementaciones de comunicaciones inalámbricas para SDR a través de GNU Radio.

En este sentido ya se han desarrollado proyectos de fin de carrera en el marco de SDR, aunque no necesariamente referidos a la temática de comunicaciones entre robots. Como ejemplo se puede mencionar el proyecto “Implementación en un FPGA de la etapa de sincronismo de un receptor OFDM para recepción de señales de DTV del estándar ISDB-T”, realizado en el período 2015-2016 por Florencia Ferrer y Daniel Contrera.

Por otro lado se pueden mencionar las pruebas realizadas por uno de nuestros tutores (Pablo Belzarena), acerca del funcionamiento de GNU Radio sobre el Raspberry Pi corriendo con ArchLinux; así como las pruebas realizadas de envío de órdenes a robots a través de SDR, o el proyecto Yocto, entre otros.

A.3. Objetivo General

Se buscará establecer una comunicación entre dos robots y una estación base a través de GNU Radio, utilizando un algoritmo para la optimización de la QoS según la posición de los robots y el entorno. Los robots deberán llegar a cierta posición, la cual será indicada por la estación base, y la comunicación deberá mantener una QoS mínima en todo momento.

A.4. Alcance

Está comprendido dentro del alcance del proyecto:

- Lograr que al ubicar la radio base y determinar una posición razonablemente cercana, los robots se posicionen de forma tal de garantizar la QoS.
- Medición y estimación de la QoS según SNR o throughput.
- Desarrollar un algoritmo que indique la posición del robot que actúe como repetidor, de forma tal de mejorar la QoS.
- Establecer mecanismos de comunicación que permitan controlar los movimientos del robot y obtener datos del mismo (posición, calidad de servicio).

A.5. Criterios de éxito

- Se logra cierta calidad de servicio (QoS) en la ubicación de interés, que permita la transmisión de mensajes instantáneos (chat). La radiobase debe poder recibir los mensajes, aún cuando ambos robots participen del enlace.
- Se logra un algoritmo competente para la maximización de la QoS.
- Se prueba la implementación de capa física y MAC proporcionada por el grupo ARTES.
- Se logra que los robots se comuniquen entre ellos.
- Se logra que un robot vaya a la posición indicada.
- Se logra que el segundo robot calcule la posición a la que debe ir para actuar como repetidor para la calidad de servicio pedida.
- Se logra completar satisfactoriamente el curso de Proyecto de Fin de Carrera por parte de los integrantes del grupo RoCo.

A.6. Actores

- Grupo ARTES.
- Tutores.
- Equipo de proyecto.
- Responsables del curso de proyecto.
- Aduana y Departamento de Contaduría de FING.
- Empresas proveedoras de equipos.
- Empresa importadora.

A.7. Supuestos

- El grupo ARTES suministra una implementación CSMA/CA de una capa MAC.
- Se cuenta con el hardware apropiado para la realización del proyecto, bajo verificación de parte del grupo ARTES y tutores.
- El presupuesto asignado para desarrollar el proyecto es suficiente.

A.8. Restricciones

- Limitantes del hardware (Arduino, Raspberry Pi, motores, etc).
- Se deberá entregar el proyecto terminado el 30 de Setiembre de 2017.
- Los integrantes del grupo disponen de 20 horas semanales.
- Se deberá trabajar con sistemas de comunicación basados en SDR.
- Se deberá trabajar en el marco de GNU Radio.

A.9. Especificación funcional del Proyecto

A continuación se presenta un diagrama de bloques del hardware a utilizar:

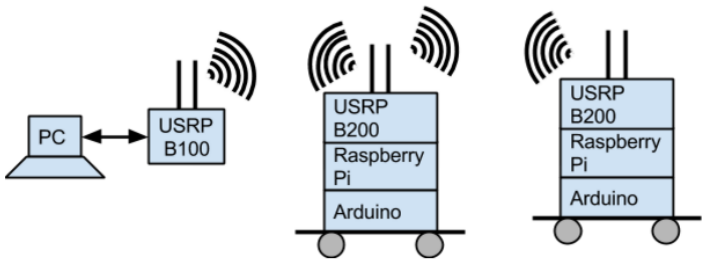


Figura A.1: Diagrama de bloques del hardware a utilizar

La siguiente tabla presenta un listado tentativo con las tareas y sistemas operativos de cada actor:

A.10. Objetivos Específicos

- Comprender el funcionamiento de GNU Radio.

Características	Estación Base	Robot 1 (Repetidor)	Robot 2 (Explorador)
Sistema Operativo	Linux (probablemente Ubuntu 16)	ArchLinux	ArchLinux
Tareas a desarrollar	Enviar coordenadas de nueva posición, o indicar acciones a realizar	Ir a destino indicado	Ir a destino indicado
	Recibir datos de posiciones y QoS	Función de repetidor	Medir QoS
	Comunicarse con el Explorador (y de ser necesario con el Repetidor)		Comunicarse con la Estación Base (y de ser necesario con el Repetidor)
	Ejecutar parte del algoritmo de optimización, calcular nueva posición		Ejecutar parte del algoritmo de optimización, calcular nueva posición

Figura A.2: Tareas a realizar por robots y estación base

- Integrar en cada robot el control por Arduino, dirigido por un Raspberry Pi sobre el que corra GNU Radio.
- Estimar correctamente la posición de cada robot en todo momento.
- Controlar a distancia la posición de los robots, desde la estación base mediante órdenes directas o mediante un algoritmo que maximice la QoS.
- Estimar la QoS entre robots y con la estación base.
- Identificar el entorno mediante un modelo sencillo ya sea precargado o explorando aleatoriamente.
- Establecer un algoritmo sencillo que permita identificar la dirección hacia la que hay mayor QoS.
- Implementar un protocolo de comunicación utilizando uno de los robots como repetidor.
- Documentar el proyecto.

A.11. Tareas

En la siguiente figura se detalla la Estructura de División de Tareas (WBS):

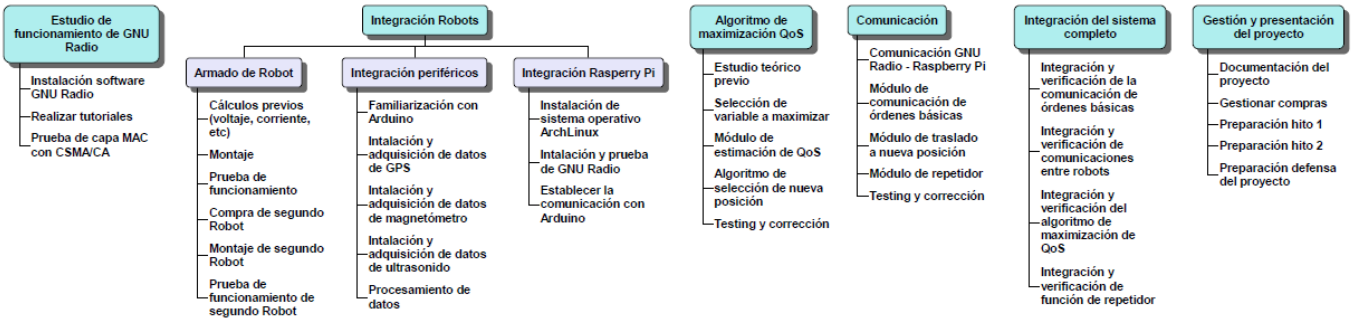


Figura A.3: WBS

A.12. Cronograma detallado del Proyecto

En la figura A.4 se observa en diagrama de Gantt del proyecto. En el mismo se detalla la fecha de incio y finalización estimada de cada tarea, junto con los recursos asignados. Además se observan las relaciones de dependencia ente las tareas.

En todo momento se respetó la disonibilidad horaria de cada recurso, sin generar sobreasignación de los mismos, tal como se observa en la siguiente figura:

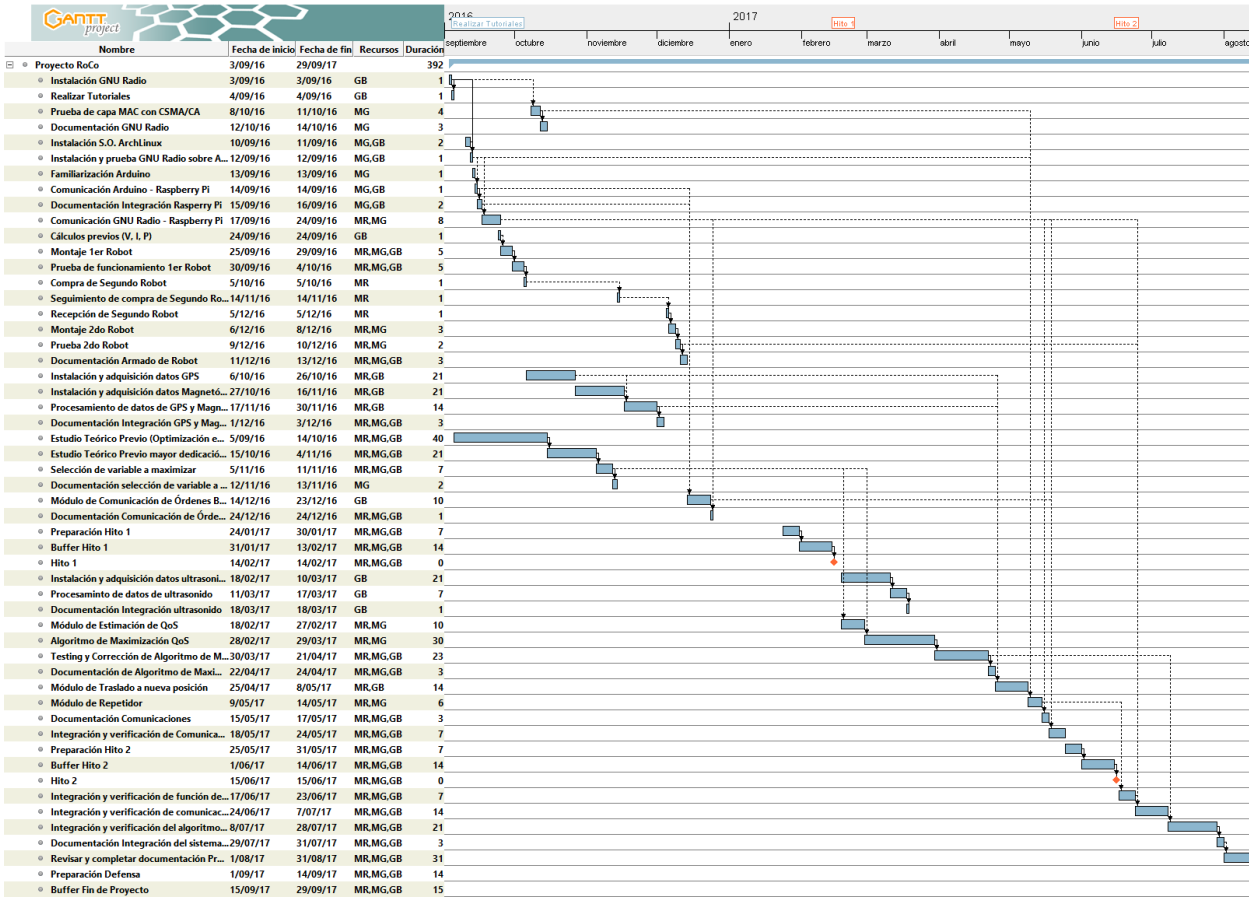
A.13. Análisis de Costos

A.13.1. Estimación del presupuesto

Para la estimación del presupuesto de proyecto debemos tener en cuenta dos factores: los sueldos a pagar por la cantidad de horas hombre trabajadas y el costo de los equipos a comprar.

En lo que refiere a la estimación de sueldos tomamos como base el sueldo de un ayudante grado 1 de la Universidad de la República. Teniendo en cuenta que el curso de Proyecto de Fin de Carrera otorga 35 créditos, y que la duración total del proyecto es de 13 meses, se obtiene como resultado un promedio de 11 horas semanales de trabajo. Sin embargo, nosotros estimamos en el Plan de Trabajo una dedicación de 20 horas semanales. Esta dedicación fue la que se tomó en cuenta para el cálculo de los sueldos. No se incluyeron aportes, sino que se utilizó el salario neto acorde a la escala salarial de la Universidad, ajustada en Enero de 2016 (tampoco se consideraron los incrementos de Julio 2016 y Enero 2017). Es decir que a nivel de salario, el gasto es la cantidad de horas estimadas de trabajo a precio de una hora de Grado 1 - 20hs. Como aclaración, para la estimación del gasto en salarios no tomamos en cuenta a los tutores.

Por otro lado, en cuanto a los equipos a comprar, se calcularon en base de la compra (ya realizada) del 1er Robot, y los costos de envío asociados a esta.



Nombre	Fecha de inicio	Fecha de fin	Recursos	Duración
Proyecto RoCo	3/09/16	29/09/17		392
Instalación GNU Radio	3/09/16	3/09/16	GB	1
Realizar Tutoriales	4/09/16	4/09/16	GB	1
Prueba de capa MAC con CSMA/CA	8/10/16	11/10/16	MG	4
Documentación GNU Radio	12/10/16	14/10/16	MG	3
Instalación S.O. ArchLinux	10/09/16	11/09/16	MG,GB	2
Instalación y prueba GNU Radio sobre A...	12/09/16	12/09/16	MG,GB	1
Familiarización Arduino	13/09/16	13/09/16	MG	1
Comunicación Arduino - Raspberry Pi	14/09/16	14/09/16	MG,GB	1
Documentación Integración Rasperry Pi	15/09/16	16/09/16	MG,GB	2
Comunicación GNU Radio - Raspberry Pi	17/09/16	24/09/16	MR,MG	8
Cálculos previos (V, I, P)	24/09/16	24/09/16	GB	1
Montaje 1er Robot	25/09/16	29/09/16	MR,MG,GB	5
Prueba de funcionamiento 1er Robot	30/09/16	4/10/16	MR,MG,GB	5
Compra de Segundo Robot	5/10/16	5/10/16	MR	1
Seguimiento de compra de Segundo Ro...	14/11/16	14/11/16	MR	1
Recepción de Segundo Robot	5/12/16	5/12/16	MR	1
Montaje 2do Robot	6/12/16	8/12/16	MR,MG	3
Prueba 2do Robot	9/12/16	10/12/16	MR,MG	2
Documentación Armado de Robot	11/12/16	13/12/16	MR,MG,GB	3
Instalación y adquisición datos GPS	6/10/16	26/10/16	MR,GB	21
Instalación y adquisición datos Magnetó...	27/10/16	16/11/16	MR,GB	21
Procesamiento de datos de GPS y Magn...	17/11/16	30/11/16	MR,GB	14
Documentación Integración GPS y Mag...	1/12/16	3/12/16	MR,MG,GB	3
Estudio Teórico Previo (Optimización e...	5/09/16	14/10/16	MR,MG,GB	40
Estudio Teórico Previo mayor dedicació...	15/10/16	4/11/16	MR,MG,GB	21
Selección de variable a maximizar	5/11/16	11/11/16	MR,MG,GB	7
Documentación selección de variable a ...	12/11/16	13/11/16	MG	2
Módulo de Comunicación de Órdenes B...	14/12/16	23/12/16	GB	10
Documentación Comunicación de Órde...	24/12/16	24/12/16	MR,MG,GB	1
Preparación Hito 1	24/01/17	30/01/17	MR,MG,GB	7
Buffer Hito 1	31/01/17	13/02/17	MR,MG,GB	14
Hito 1	14/02/17	14/02/17	MR,MG,GB	0
Instalación y adquisición datos ultrasoni...	18/02/17	10/03/17	GB	21
Procesaminto de datos de ultrasonido	11/03/17	17/03/17	GB	7
Documentación Integración ultrasonido	18/03/17	18/03/17	GB	1
Módulo de Estimación de QoS	18/02/17	27/02/17	MR,MG	10
Algoritmo de Maximización QoS	28/02/17	29/03/17	MR,MG	30
Testing y Corrección de Algoritmo de M...	30/03/17	21/04/17	MR,MG,GB	23
Documentación de Algoritmo de Maxi...	22/04/17	24/04/17	MR,MG,GB	3
Módulo de Traslado a nueva posición	25/04/17	8/05/17	MR,GB	14
Módulo de Repetidor	9/05/17	14/05/17	MR,MG	6
Documentación Comunicaciones	15/05/17	17/05/17	MR,MG,GB	3
Integración y verificación de Comunica...	18/05/17	24/05/17	MR,MG,GB	7
Preparación Hito 2	25/05/17	31/05/17	MR,MG,GB	7
Buffer Hito 2	1/06/17	14/06/17	MR,MG,GB	14
Hito 2	15/06/17	15/06/17	MR,MG,GB	0
Integración y verificación de función de...	17/06/17	23/06/17	MR,MG,GB	7
Integración y verificación de comunicac...	24/06/17	7/07/17	MR,MG,GB	14
Integración y verificación del algoritmo...	8/07/17	28/07/17	MR,MG,GB	21
Documentación Integración del sistema...	29/07/17	31/07/17	MR,MG,GB	3
Revisar y completar documentación Pr...	1/08/17	31/08/17	MR,MG,GB	31
Preparación Defensa	1/09/17	14/09/17	MR,MG,GB	14
Buffer Fin de Proyecto	15/09/17	29/09/17	MR,MG,GB	15

Figura A.4: Diagrama de Gantt

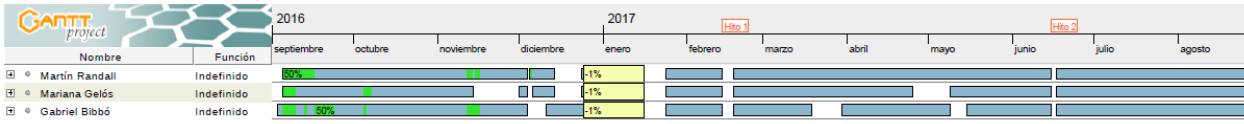


Figura A.5: Diagrama de asignación de recursos

Como nota agregada, conviene aclarar que el precio del dólar se consideró a 29.8, es decir el precio en la compra al día 7 de Setiembre de 2016. Si bien gran parte de los insumos ya fueron adquiridos con un mejor precio, en Octubre se realizarán más gastos, por lo que resolvimos encontrar un término medio que fue utilizar la cotización en la fecha actual.

En la siguiente tabla se podrá ver el presupuesto estimado del Proyecto RoCo:

Para finalizar, resta calcular el porcentaje para imprevistos. Siguiendo las recomendaciones para proyectos con altos niveles de incertidumbre, corresponde asignar un 10 % del presupuesto para imprevistos, es decir $39.404 .

Concepto	Costo unitario	Cantidad	Costo por insumo
Insumos (dólares americanos)			
B200 mini - Ettus	$820,00	1	$820,00
Estructura robot con 4 ruedas y encoder	$23,00	2	$46,00
Sensor del encoder	$11,00	2	$22,00
Pilas de níquel de alta capacidad	$15,00	2	$30,00
Portapilas	$6,00	2	$12,00
Cargador de pilas	$10,00	2	$20,00
Arduino	$22,00	2	$44,00
Regulador DC/DC	$17,00	2	$34,00
Cables	$8,99	2	$17,98
Driver motores	$10,00	2	$20,00
Raspberry PI model 3	$40,00	2	$80,00
		SUB TOTAL (dólares)	$1.145,98
		SUB TOTAL (pesos)	$34.150,20
Salarios (pesos uruguayos)	Costo diario	Días totales	
Trabajador 20 hs semanales	$361,37	964	$348.360,68
		SUB TOTAL (pesos)	$348.360,68
Otros			
Costos de importación Robot (dólares americanos)	$50,00	2	$100,00
Costo Aduana Robot (pesos)	$2.040,00	2	$4.080,00
Costos de importación B200 (dólares americanos)	$150,00	1	$150,00
		SUB TOTAL (pesos)	$11.530,00
		TOTAL	$394.040,88

Figura A.6: Estimación del presupuesto

A.13.2. Flujo de caja

Como se podrá observar a continuación (figura A.7), los gastos estimados del proyecto (adquisición de hardware) serán cubiertos gracias a presupuesto asignado de parte del Grupo ARTES (fondos del Proyecto Grupos CSIC "Grupo ARTES", nro 147).

Esto representa una ventaja importante al contar ya con recursos establecidos, pero también un riesgo importante al no contar hasta la fecha con recursos adicionales, que de ser necesarios deberíamos obtener.

Fecha	Concepto	Egreso	Ingreso	Flujo Efectivo	Saldo en Caja
25/04/2016	Ingreso de capitales, grupo ARTES	0	1720	1720	1720
04/07/2016	Compra de Robot 1 y B200 mini	1345	0	-1345	375
15/10/2016	Compra de Robot 2	375	0	375	0
				Total Saldo	**0**

Figura A.7: Flujo de caja

A.14. Análisis de Riesgos

A continuación se enueran los 5 riesgos más relevantes del Proyecto:

1. Los insumos necesarios para el montaje del segundo robot no llegan en el tiempo previsto.
2. Los insumos adquiridos son insuficientes para el funcionamiento del hardware debido a incompatibilidad entre los componentes, fallas técnicas, roturas, pérdida.
3. La implementación de la capa CSMA-CA se atrasa o no funciona.
4. La estimación de la duración de las tareas del WBS resulta incorrecta y se requiere de mayor tiempo para finalizar el proyecto.
5. El algoritmo de maximización propuesto no ofrece resultados aceptables en su implementación.

En la siguiente tabla se puede observar la estimación y evaluación inicial de los 5 riesgos identificados:

		Probabilidad de Ocurrencia		
		Poco probable	Moderado	Muy probable
Nivel de Impacto	Ninguno			
	Leve			
	Medio	R1, R5		R2
	Alto		R3, R4	
	Extremo			

Figura A.8: Matriz de probabilidades e impacto de riesgos

A.14.1. Plan de respuesta para cada riesgo

1. Se reordenan las tareas en el diagrama de Gantt para utilizar el tiempo de la mejor forma posible.
2. Se utiliza el remanente de fondos asignados al proyecto. En caso de que estos sean insuficientes, se hablará con los tutores para obtener fondos adicionales o se buscarán insumos de reemplazo a través de la colaboración de otras áreas del instituto de ingeniería eléctrica. Pueden ser por ejemplo el TEL, Tallerine (que utiliza robots similares), el laboratorio de control, etc. Si bien esto puede ser riesgoso e implicar tiempos de ajuste de tareas, también contamos con que se han realizado experiencias por parte de uno de los tutores usando los robots de Tallerine (Proyecto Yocto, Belzarena).

3. Debido a que esta tarea es impostergable, se realizará la capa CSMA-CA como parte del proyecto, ajustando el Gantt y la WBS en consecuencia.

4. Revisión periódica de que el proyecto se ajusta a los tiempos previstos en el WBS. En caso de ser necesario se pedirá una prórroga, aún sabiendo que el plazo que se utiliza como prórroga es relativamente corto. Lo más importante es poder detectar esto a tiempo, para incrementar de ser posible la carga de trabajo de los recursos humanos asignados al proyecto.

5. Probar algoritmos alternativos y eventualmente modificar el alcance del proyecto especificando los entornos en los cuales el algoritmo sea aplicable.

Identificación de nuevas tareas asociadas al Plan:

3. Se incorporaría una nueva tarea en el diagrama de Gantt que implique la implementación en GWN de una MAC tipo CSMA-CA simple.

A.14.2. Estimación y evaluación final de riesgos

En la gestión de riesgos identificamos situaciones que pudieran ser perjudiciales para el proyecto. En función de estimaciones, fundamentalmente basados en nuestra experiencia previa y conocimientos del tema, realizamos un croquis con los niveles de impacto y las probabilidades de ocurrencia de cada riesgo.

El grueso de los riesgos se generan a partir de errores en la estimación de tiempos, por lo que afectan directamente al diagrama de Gantt y la WBS planteada inicialmente. Como ventaja, tenemos margen de tiempo entre las tareas con riesgos y las subsiguientes, lo que nos da la oportunidad de solucionarlos con trabajo propio. Nuestro plan de trabajo incluye la previsión de este tipo de problemas, motivo por el cual la carga asignada a los recursos se encuentra distribuida a lo largo del año. El incremento de la carga en recursos (humanos) provocada por la ocurrencia de un suceso hoy riesgoso podría ser susceptible de provocar sobreasignación de recursos y nos forzaría a una reevaluación del diagrama de Gantt y la WBS.

La principal forma de estar, entonces, protegido frente a los riesgos planteados es mediante el control sistemático del cumplimiento de las tareas, la corrección periódica del diagrama de Gantt y la WBS, y el análisis justo de la dimensión de las tareas restantes, a medida que la experiencia nos vaya aportando datos.

El plan de respuesta propuesto para cada riesgo nos permitirá disminuir la probabilidad de ocurrencia y/o el nivel de impacto de de los mismos. Por lo tanto corresponde actualizar la matriz de riesgos:

		Probabilidad de Ocurrencia		
		Poco probable	Moderado	Muy probable
Nivel de Impacto	Ninguno			
	Leve	R1,R5		R2
	Medio		R3, R4	
	Alto			
	Extremo			

Figura A.9: Actualización de la matriz de probabilidades e impacto de riesgos

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Análisis de costos

En la tabla B.1 se pueden observar la ejecución de los gastos para la realización del proyecto. En particular, obsérvese que la cantidad de Raspberry PI terminó siendo 3 (prevista en 2), así como la cantidad de USRP B200 mini adquiridos fue de 3 (al costo de 1).

También se realizaron compras más adelante en el proyecto, por ejemplo de un segundo regulador DC-DC, disipadores, GPS. Estas no tuvieron grandes costos de importación, y se encuentran incluidas en la tabla de gastos.

Se considera que para un proyecto exploratorio y de esta envergadura: con bastante hardware, y heterogéneo (desde encoders a antenas, pasando por GPS y SDR...), el sobrecosto de 15 % está dentro de lo estimado (el sobrecosto fue menor al previsto originalmente en el plan de proyecto).

Concepto	Costo unitario	Cantidad	Costo por insumo
Insumos (dólares americanos)			
B200 mini - Ettus	$820.00	1	$820.00
Estructura robot con 4 ruedas y encoder	$23.00	1	$23.00
Estructura 2do robot	$103.88	1	$103.88
Sensor del encoder	$11.00	2	$22.00
Pilas de níquel de alta capacidad	$15.00	2	$30.00
Portapilas	$6.00	2	$12.00
Cargador de pilas	$10.00	1	$10.00
Arduino	$22.00	2	$44.00
Regulador DC/DC	$17.00	2	$34.00
Cables	$8.99	2	$17.98
Driver motores	$10.00	2	$20.00
Raspberry PI model 3	$40.00	2	$80.00
Sensores Infrarrojos	$9.99	2	$19.98
GPS	$9.99	2	$19.98
Magnetómetro	$6.22	2	$12.44
Disipadores	$16.98	-	$16.98
Antenas	$38.00	2	$76.00
	SUB TOTAL (dólares)		$1,362.24
	SUB TOTAL (pesos)		$40,594.75
Otros			
Costos de importación Robot (dólares)	$50.00	2	$100.00
Costo Aduana Robot (pesos)	$2,040.00	2	$4,080.00
Costos de importación B200 (dólares)	$150.00	1	$150.00
	SUB TOTAL (pesos)		$12,250.00
	TOTAL (pesos)		$52,844.75

Figura B.1: Tabla final de gastos asociados al proyecto RoCo.

Apéndice C

Software y Pruebas

C.1. Instalación de Arch Linux, GNU Radio y GWN

Para comenzar, se debe contar con una computadora con lector de tarjetas SD. Se recomienda usar alguna distribución de Linux como sistema operativo para trabajar sobre la tarjeta SD, o en su defecto, instalar un emulador de terminal Linux (para el caso de Windows, el emulador más conocido es Cygwin https://www.cygwin.com/).

1. Insertar la tarjeta SD en el lector de la computadora. Verificar el nombre con el que es reconocida la tarjeta. De aquí en más, se la mencionará como sdX.

2. La partición de la tarjeta SD se realiza usando el software fdisk. Para iniciar la partición, ejecutar el siguiente comando: `fdisk /dev/sdX`

3. En el prompt de fdisk, eliminar las particiones existentes y crear una nueva:

   a) Tipear `o`. Con esto se borrarán todas las particiones existentes en la tarjeta.

   b) Tipear `p` para listar las particiones. No se debería ver ninguna.

   c) Tipear `n`, luego `p` por primario, `1` para la primera partición, apretar ENTER para aceptar el primer sector por defecto, después tipear `+100M` para el último sector.

   d) Tipear `t`, luego `c` para setear la primera partición al tipo W95 FAT32 (LBA).

   e) Tipear `n`, luego `p` por primario, `2` para la segunda partición, apretar ENTER dos veces para aceptar el primer y último sectores por defecto.

   f) Escribir la tabla de particiones y salir tipeando `w`.

C.2. Instalación de GR-GWN

Para poder trabajar con las herramientas implementadas por el proyecto GR-GWN, es necesario instalar el repositorio. Para esto se ejecutaron los siguientes comandos indicados en la wiki del proyecto (https://github.com/vagonbar/gr-gwn/wiki/Installation):

```
git clone https://github.com/vagonbar/gr-gwn.git
cd gr-gwn
mkdir build
cd build
cmake ../
make && sudo make install
sudo ldconfig
```

C.3. Creación del bloque

El script de GNU Radio `gr_modtool` realiza todas las tareas necesarias para crear un nuevo bloque y mantenerlo integrado al módulo `gr-gwn`. Se ejecuta con el siguiente comando:

```
gr_modtool add -t sync -l python
```

Con esto se agrega un bloque del tipo `sync` usando lenguaje Python.

Al crear el bloque se generan dos archivos Python: uno corresponde al bloque propiamente dicho (en este caso se encuentra bajo el nombre `CSMA_FSM.py`) y el otro al código de prueba (`qa_CSMA_FSM.py`), el cual permite testear el bloque para los casos de prueba que se definan.

C.4. Testeo del bloque CSMA/CA

Como fue mencionado anteriormente, al crear el bloque de GNU Radio se genera un programa Python para realizar el testeo. El mismo permite definir un flowgraph que sea adecuado a las pruebas que se pretende realizar. En este caso se implementó el siguiente flowgraph:

Para validar el funcionamiento del bloque en todas las situaciones posibles, fue necesario simular distintos escenarios que permitan que la máquina pase por todas las transiciones. Para emular estas situaciones, en lugar de usar el bloque real de detección de ocupación del medio (bloque `Probe Medium`), se utilizó una función que devuelve True o False en base a un vector booleano recibido por la función

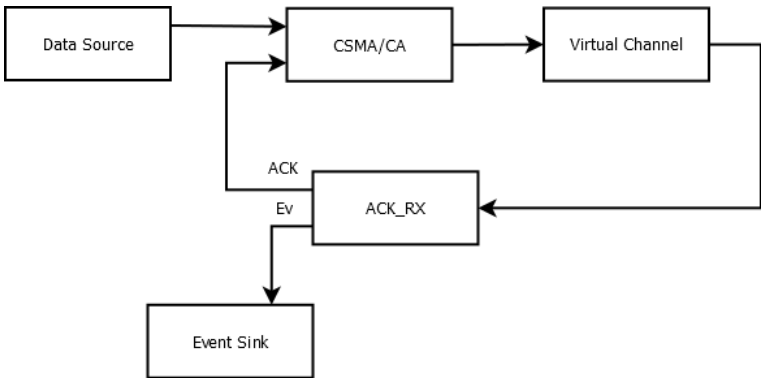


Figura C.1: Flowgraph de prueba

como parámetro, en cada invocación sucesiva devuelve el siguiente item del vector.

A continuación se detalla la serie de pruebas realizadas. Estas pruebas abarcan todas las transiciones de la máquina de estados. A mediada que se fue realizando las pruebas, se encontraron errores en la máquina de estados, principalmente en la definición de las funciones. Luego de depurar los errores encontrados se logró que la máquina pasara correctamente todos los tests.

Evento	Dato1	ACK	Dato2	TimeOut	ACK	Dato3
Estado de partida	Idle	WaitACK	Idle	BackOff	WaitACK	Idle
Función	Send	ack_ok	hold	sendformbackoff	ack_ok	hold
Canal libre	True	True	False	Tue	True	False

Tabla C.1: Test 1 - prob_pérdida = 0

Evento	TimeOut	ACK	ACK	
Estado de partida	BackOff	WaitACK	WaitACK	Idle
Función	sendfrombackoff	send_from_buffer	ack_ok	
Canal libre	True	True	True	

Tabla C.2: Test 1 (continuación) - prob_pérdida = 0

Evento	Dato1	TimeACK	Dato2	TimeACK	TimeOut	TimeOut	...	Time
Estado de partida	Idle	WaitACK	WaitACK	WaitACK	BackOff	BackOff	...	Back
Función	Send	resend	push	rehold	rehold	rehold	...	sto
Canal libre	True	True	True	False	False	False	...	Fal

Tabla C.3: Test 2 - prob_pérd = 1

Evento	Dato1	TimeACK	TimeACK	...	TimeACK	
Estado de partida	Idle	WaitACK	WaitACK	...	WaitACK	stop
Función	Send	resend	resend	...	stop	
Canal libre	True	True	True	...	False	

Tabla C.4: Test 3 - prob_pérd = 1

Evento	Dato1	TimeACK	Dato2	TimeACK	Dato3		TimeACK	
Estado de partida	Idle	WaitACK	WaitACK	WaitACK	WaitACK	...	WaitACK	sto
Función	Send	resend	push	resend	push		stop	
Canal libre	True	True	True	True	True		True	

Tabla C.5: Test 4 - prob_pérd = 1

Evento	Dato1	Dato2	TimeACK	TimeOut	ACK	ACK
Estado de partida	Idle	WaitACK	WaitACK	BackOff	WaitACK	WaitAC
Función	Send	push	hold_ack	sendformbackoff	send_from_buffer	ack_ok
Canal libre	True	True	True	True	True	True

Tabla C.6: Test 5 - prob_pérd = 0,8

Apéndice D

Mejoras a considerar sobre la navegación de los robots en futuros proyectos

Durante la ejecución del proyecto nos enfrentamos a distintas dificultades para poder crear dos robots móviles autónomos que puedan navegar correctamente y soporten todo el peso requerido. Para dar con la solución definitiva, previamente se realizó una búsqueda bibliográfica de las técnicas generalmente utilizadas para este tipo de aplicaciones. Las soluciones que se implementaron sobre los robots, ya fueron expuestas en el Capítulo 4 Navegación de los robots. Las que no se implementaron, por distintos motivos, se exponen a continuación.

En este anexo se proponen alternativas a las ruedas utilizadas en los robots para mejorar el giro sobre su propio eje; se exponen distintas tecnologías existentes para el posicionamiento de objetos y se presentan tres técnicas que se utilizan para disminuir la incertidumbre de las medidas del GPS.

D.1. Giro del robot sobre su propio eje

Es buena idea utilizar Mecanum Wheels u Omni Wheels en robots pesados, que generan mucha fricción con la superficie y es preferible que las ruedas no deslicen.

Las **OmniWheels** [98] son ruedas con pequeños rodillos a lo largo de la circunferencia dispuestos perpendicularmente al eje de giro de la rueda. De esta manera la rueda puede girar hacia adelante, hacia atrás y sobre sí misma como las ruedas convencionales. Pero, además, pueden desplazarse transversalmente sin deslizar, gracias al giro de los rodillos que están oficiando de apoyo.

Las **Mecanum Wheels** [2] son ruedas con pequeños rodillos sobre su circunferencia orientados a 45º. Una de estas ruedas tienen tres grados de libertad que son: la rotación de la rueda, la rotación del rodillo y la rotación sobre el eje vertical

Figura D.1: Rueda OmniWheel. Figura tomada de artículo de Wikipedia [98].

que pasa a través del punto de apoyo.

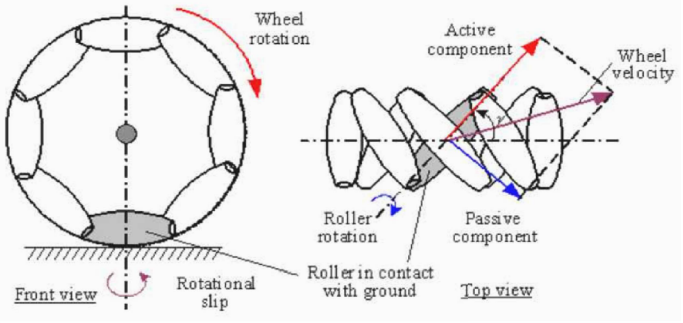


Figura D.2: Rueda Mecanum Wheel. Figura tomada del artículo de Taha Bin Mohamed [2].

El objetivo de utilizar estas ruedas es permitir que los robots puedan girar sobre su eje de forma eficiente y que además puedan desplazarse en cualquier dirección sin la necesidad de orientación previa. Para ello, algunas de las configuraciones posibles y más conocidas son:

- Kiwi Drive: Se utilizan tres OmniWheels dispuestas en triángulo.
- Holonomic Drive: Se utilizan cuatro OmniWheels dispuestas en cuadrado.
- Mecanum Drive: Se utilizan cuatro MecanumWheels colocadas en los laterales en formato tanque.

También existen configuraciones donde se colocan cuatro OmniWheels en los laterales en formato tanque. Esta configuración no permite realizar movimientos omnidireccionales, solamente puede avanzar, retroceder y girar sobre su propio eje, pero soluciona el problema del deslizamiento de las ruedas sobre la superficie, por lo que son capaces de cargar mucho peso.

Estas ruedas tienen la ventaja de tener un diseño compacto, permitir transportar mucha carga, ser fáciles de controlar, poca velocidad y poca fuerza de empuje

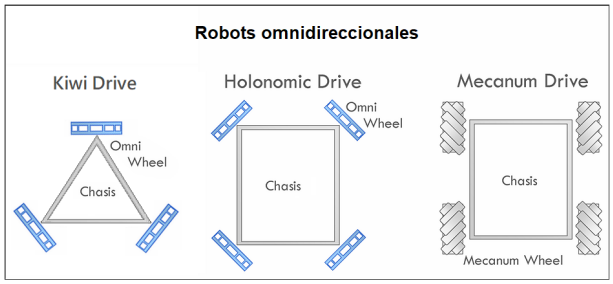


Figura D.3: Configuraciones posibles

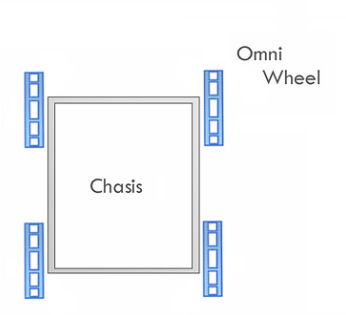


Figura D.4: OmniWheels en los laterales en formato tanque

cuando se mueven diagonalmente. Como desventajas, se pueden mencionar el no tener un contacto continuo con la superficie, ser muy sensibles a las irregularidades del terreno y tiener un diseño complejo. El movimiento omnidireccional del robot depende de que las cuatro ruedas estén en contacto con la superficie, por lo que algunos robots suelen tener suspensión en sus cuatro ruedas. Sobre superficies inclinadas se puede perder el control del robot.

Debido a la falta de fuentes de financiamiento para la compra de cualquiera de estas ruedas, sumada la falta de tiempo para realizar la compra, esperar la llegada del envío y realizar posteriormente las pruebas de desempeño de un producto con el que nunca se había trabajado, se decidió mantener las ruedas originales de los robots y buscar una solución alternativa a pesar de que esta no fuese la más eficiente.

D.2. Tecnologías para el posicionamiento de objetos [3]

La localización se refiere a la ubicación, más o menos precisa, de un objeto, persona u otras entidades mediante la utilización de datos sensoriales. Estos pueden ser ondas magnéticas, ondas acústicas e incluso luz. La idea es utilizar una infraestructura dedicada, o algunos medios adicionales, para localizar con precisión una posición. La localización se puede dividir en sistemas al **aire libre** (Global Navigation Satellite System - GNSS) o de **interiores** (Indoor Positioning Systems - IPS).

La mayoría de las tecnologías para la localización que utilizan RF toman como base al tiempo. Éstos usan el retardo en la propagación de la señal desde un transmisor a un receptor e incluyen el Tiempo de llegada (Time of Arrival - ToA), la Diferencia de tiempo de llegada (Time Difference of Arrival - TDoA) o la telemetría bidireccional (Two Way Ranging - TWR).

D.2.1. Localización al aire libre

GNSS

Para el posicionamiento al aire libre se utilizan los sistemas globales de navegación por satélite (Global Navigation Satellite System, GNSS). Éstos cuentan con una extensa constelación de satélites girando en diferentes órbitas alrededor de la tierra mientras transmiten las señales utilizadas para el posicionamiento. Mediante trilateración, estas señales brindan una medida de la distancia desde tres ubicaciones conocidas a un punto que se quiere localizar.

Existen varios sistemas de posicionamiento. El más conocido y utilizado en la actualidad es el Global Positioning System (**GPS**) operativo desde 1978, desarrollado por USA originalmente para fines militares, aunque en la actualidad tiene un uso civil. El Globalnaya Navigatsionnaya Sputnikovaya Sistema (**GLONASS**), fue desarrollado por Rusia y está oficialmente operativo desde 1995. **GALILEO** es el homólogo de la Unión Europea, que se estima esté completado para 2020. También para ese año se espera que el sistema **BeiDou-2**, de desarrollo Chino, quede operativo.

D.2.2. Localización en interiores

Los sistemas anteriores no se pueden utilizar para IPS debido a que las construcciones obstruyen la señal y generan otros efectos de trayectos múltiples (multipath). Por lo tanto, existen varias metodologías diferentes para la localización en interiores, aunque cada una de ellas tiene sus limitaciones.

Visible Light Communication

La comunicación de luz visible (Visible Light Communication - **VLC**), que aparentemente es un nuevo método de transferencia de información, es en realidad anterior a la radio moderna. Alexander Graham Bell usó luz modulada para transmitir el habla (fotófono). La comunicación de luz visible requiere lámparas LED que pueden encenderse a frecuencias tan altas que no pueden detectarse a simple vista. Esta frecuencia de conmutación puede usarse para enviar información a un receptor. Las lámparas fluorescentes e incandescentes no se pueden encender

a velocidades tan altas, por lo que se necesitan lámparas LED. Philips ofrece una solución integrada que utiliza esta tecnología para IPS. [62]

Received Signal Strength

Los indicadores de intensidad de la señal recibida (Received Signal Strength - **RSS**) utiliza la disminución de la intensidad de la señal a medida que una señal se propaga para medir la ubicación correspondiente. El Wi-Fi (2.4 GHz) es un buen candidato para la localización basada en RSS ya que la infraestructura para redes inalámbricas está universalmente extendida. La intensidad de la señal recibida desde el router se caracteriza y compara con un valor teórico en función de lo cual se estima la distancia. Cabe destacar que las señales de Wi-Fi están sujetas a pérdidas y sombreado por multipath, lo que puede conducir a una significativa degradación de la precisión. Otra tecnología para IPS que está ganando mucha fuerza en este rubro es el Bluetooth Low-Energy (**BLE**) que utiliza la trilateración y fingerprinting como técnicas para estimar la posición en base a la RSS y la LQI (Link Quality Indicator). Para su utilización se deben instalar BLE beacons en las paredes que emiten la señal a medir. Esta solución tiene una incertidumbre razonable (1-2 m), es económica y sirve para utilizarse con pequeñas baterías. [106]

Ultrasonido

El ultrasonido (Ultrasound – **US**) se usa popularmente para la localización de alta precisión debido a su bajo costo ya que se requiere solamente de sensores ultrasonido ubicados en el punto que se quiere medir, pero sus desventajas incluyen una corta distancia operativa (de hasta 13 metros), demasiada sensibilidad a obstáculos, efectos de la reflexión y el multipath, y dificultades para distinguir entre varios objetos.

Ultra Wideband

La tecnología de banda ultra ancha (Ultra Wideband - **UWB**) es de reciente utilización, en comparación con otras tecnologías de RF para la localización. Ésta utiliza un gran ancho de banda y evita interferencias al trabajar con baja densidad de potencia espectral. Entre sus ventajas se encuentra su bajo consumo (como emisor de ondas de radio), bajo costo (se puede implementar con tecnología CMOS) y alta tasa de transferencia de información.

D.3. Técnicas para mejorar la precisión del GPS

Para reducir la incertidumbre en el posicionamiento, comúnmente se agrega información adicional que ayude a corregir las mediciones. Esta información puede ser la proveniente de otro receptor GPS o de otro tipo de sensores que brinden una

estimación de la posición del robot. Las dos técnicas comúnmente utilizadas para reducirla son el **GPS Diferencial** o los **filtros de Kalman**.

En los casos en que se debe trabajar solamente con la información obtenida por un receptor aislado y estático, se puede mejorar la precisión realizando el **promedio** de las medidas obtenidas e intentando reducir las fuentes de error antes descritas. Para eso se debe colocar al receptor en lugares sin interferencias por multipath, y donde los indicadores PDOP y HDOP sean buenos.

D.3.1. GPS Diferencial

El GPS Diferencial (DGPS) fue desarrollado para satisfacer la necesidad de reducir la incertidumbre del GPS estándar en aplicaciones de posicionamiento y medida de distancia. Una arquitectura típica de DGPS consiste en un receptor de referencia ubicado en una posición conocida y uno o más usuarios receptores de DGPS. El usuario receptor es comúnmente llamado receptor móvil, por no estar confinados a ubicarse en una posición conocida como es el caso del receptor de referencia. La antena del receptor de referencia, el sistema de procesamiento para la corrección diferencial y el link de datos (en caso de que se utilice), es lo que se conoce como estación de referencia. Ambos receptores realizan medidas de posicionamiento para almacenarlas y procesarlas luego, o para enviarlas a un lugar específico en tiempo real mediante el link de datos.

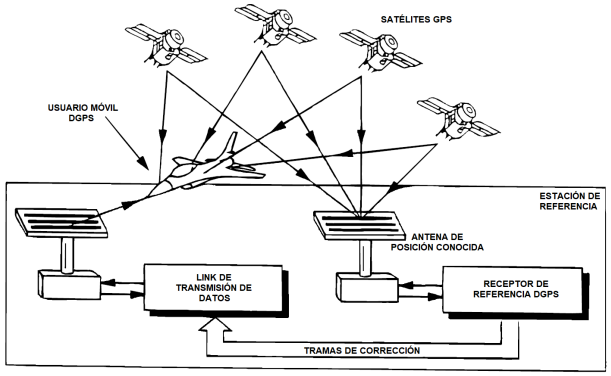


Figura D.5: Esquema de funcionamiento de GPS Diferencial. Figura obtenida de NAVSTAR Introduction [71].

El principio de DGPS se basa en que receptores que se encuentren relativamente cerca entre sí van a experimentar simultáneamente los errores habituales de este tipo de medidas de distancia. En general, el usuario móvil usa las medidas de la estación de referencia para eliminar estos errores. Para esto, el usuario móvil debe posicionarse utilizando los mismos satélites que la estación de referencia. Los errores habituales incluyen los retardos de la señal a través de la atmósfera, y errores en el reloj satelital y el cálculo de las efemérides. Los errores que afectan

únicamente a cada receptor, como es el caso del ruido o el multipath, no se pueden eliminar si no es con un procesamiento recursivo adicional (ya sea por el receptor de referencia, por el móvil o por ambos) para brindar una solución promediada, suavizada y filtrada.

El estándar aceptado para DGPS, que define el formato de la información que se utiliza entre la estación de referencia y el usuario, fue desarrollado por la Radio Technical Commission for Maritime Services (RTCM) Special Committee-104 (SC-104). El estándar está en principio enfocado a utilizarse en aplicaciones de tiempo real y para cubrir una amplia gama de tipos de medición DGPS.

Se emplean varias técnicas DGPS dependiendo de la precisión que se quiera obtener, de dónde se procesarán los datos, y si se utilizarán en tiempo real o no. Si se necesitan los resultados en tiempo real (DGPS/RKT) entonces se deberá utilizar algún tipo de link de datos. En el caso de aplicaciones que no sean en tiempo real, la información se puede almacenar para post procesarla (DGPS/PP). [71]

Dada la naturaleza de este proyecto, solamente hubiera sido aplicable la corrección en tiempo real, para la cual el Servicio Geográfico Militar y el Instituto de Agrimensura de la Facultad de Ingeniería de la Universidad de la República (UDELAR) han llevado adelante un proyecto conjunto para el desarrollo e implementación de un servicio de Corrección Diferencial en Tiempo Real (DGPS/RTK) a través de Internet. Esta metodología permite que un receptor GPS pueda recibir las correcciones diferenciales de forma automática y prácticamente al instante a través de protocolo Ntrip (Red de Transporte de datos en formato RTCM a través Protocolo de Internet), simplemente conectándolo por internet al servidor de la Red Geodésica Nacional Activa de la República Oriental del Uruguay (REGNA-ROU). En Montevideo, las correcciones se realizan en base a las medidas obtenidas por una estación de referencia ubicada en la Fortaleza del Cerro de Montevideo. [23]

En este proyecto, una forma en que se podría haber realizado la comunicación entre el robot y el servidor REGNA-ROU, hubiera sido comunicando a la Estación Base con el servidor directamente a través de internet y ésta comunicada al robot utilizando la infraestructura de GNURadio. Esta opción se descartó porque escapaba al alcance del proyecto. Otra forma podría haber sido conectar la Raspberry Pi de los robots directamente al WiFi local para que éste se comunique con el servidor REGNA-ROU, pero esta opción también se descartó porque depende de la existencia previa de una red WiFi y el proyecto busca generar una red de comunicación inalámbrica en lugares donde puede no haber comunicación.

D.3.2. Filtro de Kalman

El filtro de Kalman (FK a partir de aquí) es un algoritmo de estimación. La técnica inicial fue inventada por Rudolf E. Kalman en 1960 y desarrollada por

numerosos autores posteriormente. El FK sirve como base para un gran número de algoritmos de estimación utilizados en navegación. Sus funciones incluyen la de mantener una solución de navegación por satélite suavizada, el alineamiento y calibración de sistemas de navegación inercial (INS en inglés) y la integración de INS con equipamiento de usuario de GNSS. Esta integración es clave en la obtención de una solución de navegación óptima a partir de las mediciones disponibles en el sistema. En la actualidad es utilizado en la navegación para lograr la integración de la información del GPS y sensores externos (como por ejemplo INS), en numerosas aplicaciones a saber: guiado de misiles, guiado y navegación de los transbordadores espaciales de la NASA o el control de la orientación y navegación de la Estación Espacial Internacional.

El FK es un estimador recursivo lineal que proporciona una forma computacionalmente eficiente de estimar el estado de un proceso de una forma que minimiza la media de los errores al cuadrado. El FK realiza estimaciones en tiempo real de distintos parámetros de un sistema, como por ejemplo posición y velocidad, que pueden cambiar continuamente. Las estimaciones son actualizadas a partir de distintas mediciones realizadas sobre el sistema, que pueden contener ruidos. Estas mediciones son realizadas por sensores, y han de ser función de los parámetros estimados.

El FK utiliza el conocimiento de las propiedades determinísticas y estadísticas de los parámetros del sistema y de las mediciones para obtener estimaciones óptimas de sus estados a partir de la información disponible. El FK incluye una incertidumbre sobre las estimaciones realizadas y una medida de las correlaciones entre los errores en las estimaciones de los distintos parámetros.

El FK es una técnica de estimación Bayesiana. Se le proporciona un grupo de estimaciones iniciales y este opera de forma recursiva, actualizando las estimaciones en función de los valores previos y de los nuevos obtenidos de las últimas mediciones. Este asume que el sistema en estudio es lineal. Para aplicaciones en tiempo real, como es el caso de la navegación, el enfoque recursivo de este algoritmo es más eficiente, dado que solo se deben procesar los datos de las nuevas mediciones en cada iteración.

El FK está compuesto por 5 elementos principales: el vector de estados y su covarianza, el modelo del sistema, el vector de mediciones y su covarianza, el modelo de mediciones y el algoritmo.

Intuitivamente, el FK ordena la información y pondera las contribuciones relativas de las mediciones y del comportamiento dinámico del vector de estado. Las medidas y el vector de estado se ponderan por sus matrices de covarianza respectivas. Si las mediciones son inexactas (variaciones grandes) en comparación con la estimación del vector de estado, entonces el filtro reducirá el peso de las mediciones. Por otro lado, si las mediciones son muy precisas (pequeñas variaciones)

cuando se comparan con la estimación del estado, entonces el filtro tenderá a ponderar las mediciones en gran medida con la consecuencia de que la estimación del estado previamente calculado contribuirá poco a la última estimación del estado.

Cuando al receptor de GPS se le integran otros sensores de navegación (como por ejemplo INS, relojes, altímetros o AHRS), entonces el FK puede ser extendido para incluir las medidas de estos sensores. De hecho, una implementación típica para sistemas integrados sería tener un FK central incorporando las medidas de todas las fuentes disponibles.

Un sistema de navegación inercial (INS) está compuesto por una unidad de medición inercial (IMU) y un procesador de navegación. La IMU está compuesta por acelerómetros y giróscopos, y va montada directamente sobre el robot móvil. Los acelerómetros miden la fuerza específica, mientras que los giróscopos miden la velocidad angular. Esto permite determinar la aceleración, la orientación y la velocidad del móvil en el que se implementa el sistema. Sin embargo, su principal inconveniente es la degradación de la precisión en la determinación de la posición en la navegación con el tiempo, si se lo utiliza como único instrumento de medida del posicionamiento. Un sensor de este tipo, por ejemplo el MPU 9255, que es integrable con Arduino, incluye magnetómetro, giroscopio y acelerómetro, se puede conseguir en el mercado por U$S 10 aproximadamente.

Alternativamente, en el caso de los robots de este proyecto, se puede utilizar la información proveniente de los sensores ya existentes: magnetómetro y encoders. Estos últimos permiten conocer la velocidad y la aceleración del robot al tener información sobre sus dimensiones constructivas.

El GPS proporciona una solución fiable a largo plazo, pero con una precisión de varios metros. Además, el GPS no proporciona mediciones sobre la orientación. Otra desventaja de estos sistemas es que sus señales son susceptibles a interferencias, por lo que su solución de navegación no es válida si se necesita que sea continua.

Por estos motivos, se puede ver que los distintos sensores aportan información complementaria para la navegación y la combinación de todos ellos permite obtener una solución de navegación continua, en la que se conozca la orientación, velocidad angular y aceleración, con una precisión elevada tanto a corto como a largo plazo. En un sistema que integre GPS y otros sensores (INS), evita que la solución inercial derive, mientras que ésta suaviza la solución del GPS y cubre los momentos en los cuales se pierde la señal de éste.

Si bien los FK tienen probados beneficios para el posicionamiento, escapan del alcance del proyecto por el tiempo que implica su puesta en marcha. Se exponen en esta sección por ser una herramienta muy potente y utilizada en este rubro, la cual hay que tener en consideración a la hora de llevar a cabo proyectos de robótica

móvil. [53]

D.3.3. Promedio

Promediar las medidas de ubicación de un punto fijo, obtenidas durante un período prolongado de tiempo, es una técnica utilizada para mejorar la precisión de la medición. Consiste en sumar las medidas obtenidas y dividir el total sobre el número de mediciones. Se puede realizar en tiempo real, agregando al promedio cada medida nueva que se obtenga. Al igual que los filtros de Kalman, el promedio busca mitigar las fuentes de incertidumbre propias del receptor, como errores generados por los relojes o el multipath. Cada vez que la posición cambia, se debe comenzar a promediar desde cero.

El Ing. Juan Manuel Toloza [71] estudió y comparó los resultados de promediar y aplicar los filtros de Kalman a un conjunto de medidas de posicionamiento estáticas obtenidas por dos receptores GPS convencionales, a lo largo de 10 minutos. Concluye que si bien el promedio mejora la estimación de la posición, "cuando existen valores extremos se torna una medida no representativa de la población y depende de la dispersión de las muestras", sugiriendo que es mejor aplicar filtros de Kalman.

El prof. David L. Wilson [65] y Dan Charrois [59] en el año 1999 se propusieron estudiar cuánto mejora la precisión el promedio de las medidas a lo largo del tiempo. En ese entonces, las señales de GPS de uso público estaban degradadas intencionalmente por EEUU, para que el común de la gente no tuviera la misma precisión que los militares. En el año 2000 se discontinuó el uso de esta práctica, mejorando la precisión de las medidas. Corresponde aclararlo, porque la gráfica D.6 fue hecha con medidas del año 1999, afectadas de mayor error que las que podemos obtener hoy en día. Por este motivo no debe considerarse el valor de la desviación estándar en términos absolutos, sino en relativos.

El gráfico D.6 al que nos estamos refiriendo pretende contestar la pregunta de "cuánto tiempo es el suficiente" para obtener una precisión determinada, cuando se promedian las medidas de GPS. Éste muestra la desviación estándar de la información en función del tiempo de promediado (en escala logarítmica). Se distingue que la desviación Norte-Sur es mayor que la Este-Oeste. Un fenómeno similar hemos podido observar nosotros con nuestras mediciones.

Se superponen seis curvas:

1. Desviación Este-Oeste (East-West), usando todas las medidas.
2. Desviación Norte-Sur (North-South), usando todas las medidas.
3. Desviación de la altitud, usando todas las medidas.

4. Desviación Este-Oeste (East-West), sólo para medidas con GDOP < 2
5. Desviación Norte-Sur (North-South), sólo para medidas con GDOP < 2
6. Desviación de la altitud, sólo para medidas con GDOP < 2

En este análisis, las medidas con GDOP < 2 representan el 26 % del total de medidas. Como se explicó anteriormente, este indicador establece qué medidas son más precisas en función de la distribución de los satélites en el espacio. En la gráfica se observa cómo mejoran la precisión al sólo considerar estas medidas.

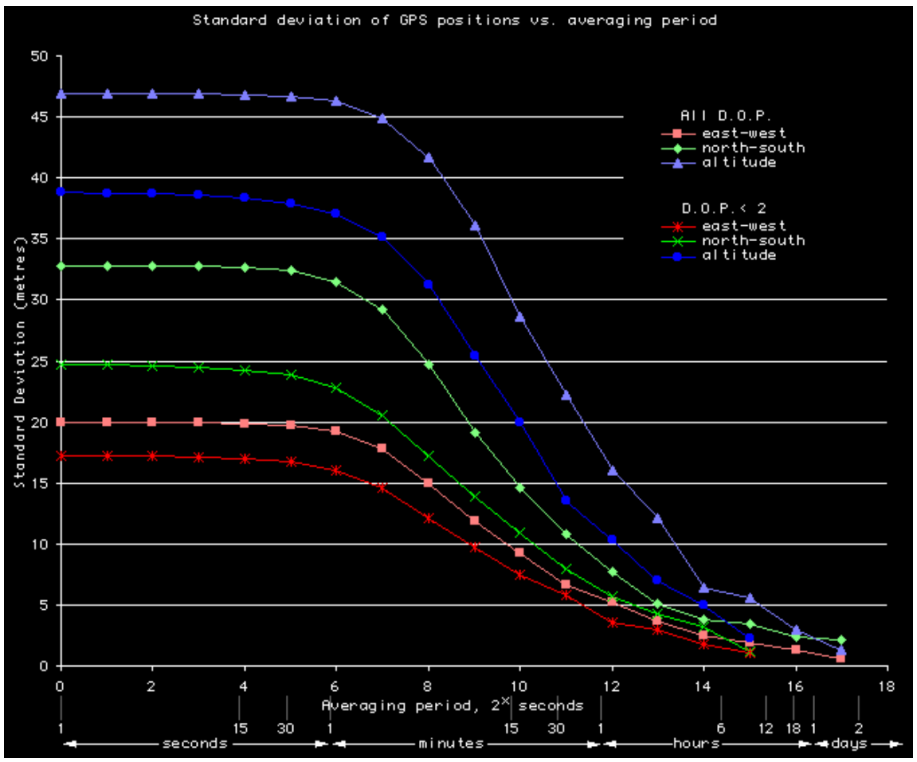


Figura D.6: Desviación estándar en función del período de promediado. Figura obtenida de la página del prof David L. Wilson [65].

Para promedios efectuados en menos de un minuto, prácticamente no hay mejoras en la precisión. Para períodos de entre un minuto y una hora o dos, se mejora considerablemente la precisión obtenida. Promediar por mayores períodos de tiempo naturalmente sigue mejorando la precisión, pero la velocidad con la que esto sucede es menor.

El gráfico se lee de la siguiente manera. Imaginemos que se quiere duplicar la precisión en la dirección Este-Oeste para todas las medidas obtenidas (no necesariamente para aquellas con un GDOP óptimo). Esto es equivalente a reducir a la mitad el valor de la desviación estándar del gráfico rosado. Como el valor en el origen es de 20 metros, el tiempo en el que la desviación se reduce a la mitad

(alcanza los 10 metros) es de 15 minutos. Recordar que valores de la desviación estándar deben ser considerados de forma relativa.

Las ventajas de esta técnica son ser de fácil implementación y que consume pocos recursos informáticos. La desventaja es que, si bien reduce la incertidumbre, lo hace de forma lenta y cada vez que el punto se mueve, el promedio debe comenzar desde cero.

Apéndice E

Diagramas de flujo finales para cada nodo

Diagramas de flujos para la Estación Base (figura E.1), el Explorador (figura E.2) y el Repetidor (figura E.3).

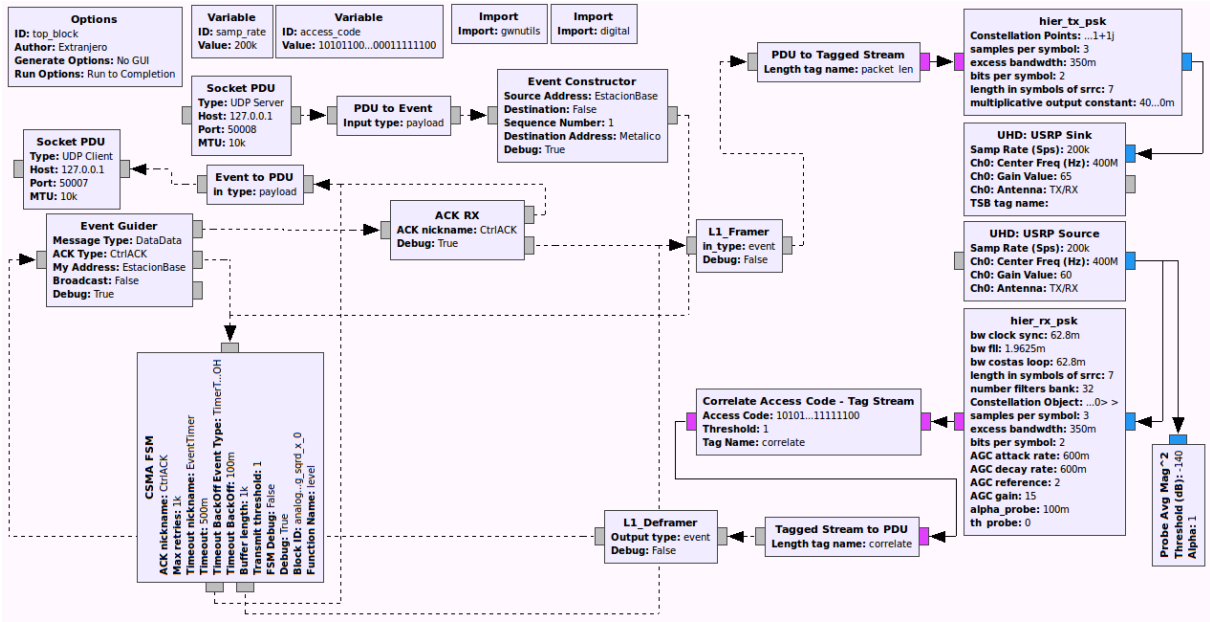


Figura E.1: Flow graph de la Estación Base.

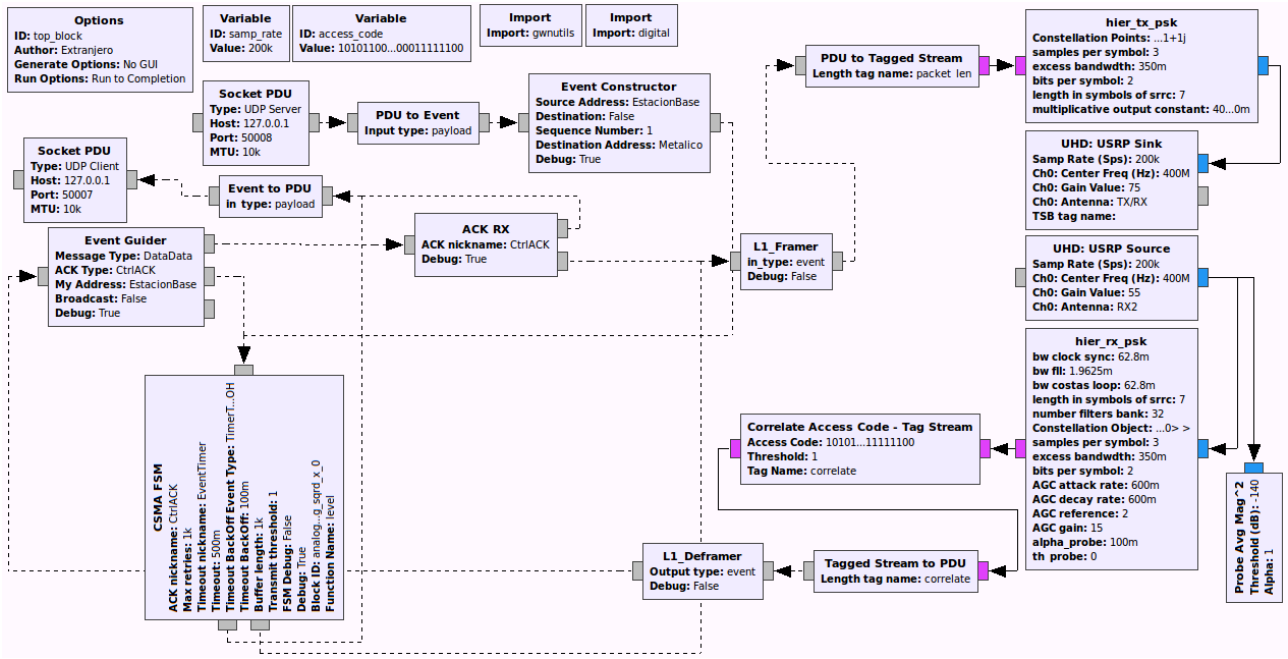


Figura E.2: Flow graph del Explorador.

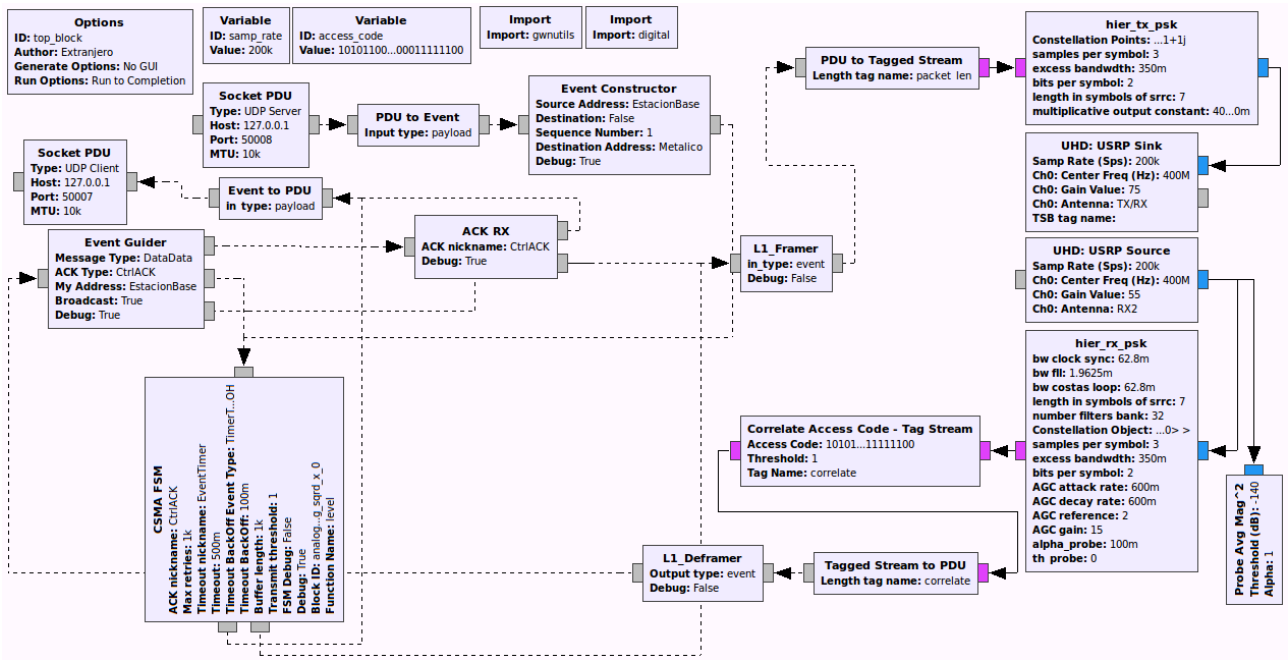


Figura E.3: Flow graph del Repetidor. Notar la opción Broadcast en True en el bloque Event Guider y la conexión correspondiente (de este bloque al L1_Framer).

Referencias

[1] *La Política: Aristoteles*. Biblioteca económica filosófica. Sociedad Española de Librería, 1885.

[2] Development of mobile robot drive system using mecanum wheels. *2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES), Advances in Electrical, Electronic and Systems Engineering (ICAEES), International Conference on*, page 582, 2016.

[3] High accuracy indoor localization for robot-based fine-grain inspection of smart buildings. *2016 IEEE International Conference on Industrial Technology (ICIT), Industrial Technology (ICIT), 2016 IEEE International Conference on*, page 2010, 2016.

[4] Raspberry Pi Foundation. Annual Review 2016. `https://static.raspberrypi.org/files/about/RaspberryPiFoundationReview2016.pdf`. 2016.

[5] 2pcs IR Infrared Obstacle Avoidance Sensor Module f Arduino. `https://www.ebay.com/itm/2pcs-IR-Infrared-Obstacle-Avoidance-Sensor-Module-f-Arduino/Smart-Car-3-wire-N59-/401063075418?_trksid=p2349526.m4383.l4275.c10`. 2017.

[6] Sensors Amazon. 4WD Aluminum Mobile Robot Platform/This Platform Can Be Equipped With A Variety Of Controllers, Drives and Etc. Wireless Radio Frequency Modules. `https://www.amazon.com/Aluminum-Platform-Equipped-Controllers-Frequency/dp/B01GXBNYCA/ref=sr_1_55?ie=UTF8&qid=1490802463&sr=8-55&keywords=4wd+chassis`. 2017.

[7] Diamond Antenna. Srh789 telescoping handheld antenna, `http://www.diamondantenna.net/srh789.html`. 2015.

[8] Grupo ARTES. Detalle de la propuesta del grupo artes (análisis de redes, tráfico y estadísticas de servicio) al proyecto grupos – csic. 2014.

[9] Constantine A. Balanis. Antenna theory: Analysis and design, 4th edition. February 2016.

[10] EBL. Batteries. http://www.eblmall.com/batteries/. 2017.

[11] Pablo Belzarena and Federico Larroca. Comunicaciones inalámbricas. notas del curso. https://eva.fing.edu.uy/pluginfile.php/178754/mod_resource/content/1/sinctemp2017.pdf. 2017.

[12] Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on selected areas in communications*, 18(3):535–547, 2000.

[13] Geoffrey Blewitt. Basics of the gps technique: observation equations. *Geodetic applications of GPS*, pages 10–54, 1997.

[14] Johann Borenstein and Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on robotics and automation*, 12(6):869–880, 1996.

[15] Jean L. Broge. The benefits and advantages of digital signal processing, http://articles.sae.org/14409/. 2015.

[16] USRP B100 bus series. https://www.ettus.com/content/files/kb/07495_Ettus_B100_DS_Flyer_HR_4.pdf. 2012.

[17] Mathias Coinchon. Simple fm transmitter using gnu radio, http://wiki.opendigitalradio.org/Simple_FM_transmitter_using_gnuradio. 2017.

[18] Mercado Libre. Adaptador Conversor Usb A Uart Rs232 Ttl 6 Pines Puerto Com. https://articulo.mercadolibre.com.ar/MLA-627741377-adaptador-conversor-usb-a-uart-rs232-ttl-6-pines/-puerto-com-_JM. 2017.

[19] ELECTRONILAB. Motorreductor con caja reductora 6V 1:48. https://electronilab.co/tienda/motorreductor-con-caja-reductora-6v-1-48/. 2017.

[20] Florencia Ferrer Daniel Contrera. Implementación en un fpga de la etapa de sincronismo de un receptor ofdm para recepción de señales de dtv del estándar isdb-t. 2016.

[21] Instituto de Computación. Inteligencia artifical, robótica cooperativa, https://www.fing.edu.uy/inco/cursos/robotica/teorico/IAR10_Clase07_Cooperacion.pdf. 2017.

[22] Maria Carmela De Gennaro and Ali Jadbabaie. Decentralized control of connectivity for multi-agent systems. In *Decision and Control, 2006 45th IEEE Conference on*, pages 3628–3633. IEEE, 2006.

[23] Red Geodésica Nacional Activa de la República Oriental del Uruguay (REGNA-ROU) Norbertino Suárez. http://www.sgm.gub.uy/?mdocs-file=466. 2011.

[24] Universitat Politècnica de València. Densidad de potencia radiada, http://www.upv.es/antenas/Tema_3/Densidad_potencia_dipolo.htm. 2017.

[25] L298. DUAL FULL-BRIDGE DRIVER. http://www.st.com/content/ccc/resource/technical/document/datasheet/82/cc/3f/39/0a/29/4d/f0/CD00000240.pdf/files/CD00000240.pdf/jcr:content/translations/en.CD00000240.pdf. 2000.

[26] USRP Hardware Driver and USRP Manual. http://files.ettus.com/manual/page_usrp_b200.html. 2017.

[27] embedded.com. Introduction to watchdog timers. 2017.

[28] Carolina Fortuna and Mihael Mohorcic. Trends in the development of communication networks: Cognitive networks. *Computer networks*, 53(9):1354–1376, 2009.

[29] Wireless Innovation Forum. Partnerships and memberships, http://www.wirelessinnovation.org/partners. 2017.

[30] Wireless Innovation Forum. Software defined radio, http://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf. 2017.

[31] Linux Foundation. Press release: Networking industry leaders join forces to expand new open source community to drive development of the dpdk project, https://www.linuxfoundation.org/press-release/networking-industry-leaders-join-forces-to-expand-new-open-source-community- 2017.

[32] fritzing. NEO6MV2 GPS Module. http://fritzing.org/projects/neo6mv2-gps-module. 2017.

[33] Victor González. Gr-gwn wiki. socket example. https://github.com/vagonbar/gr-gwn/tree/master/examples/socket_test.

[34] Víctor González-Barbone, Pablo Belzarena, Federico Larroca, Martín Randall, Paola Romero, and Mariana Gelós. Gwn: A framework for packet radio and medium access control in gnu radio. 2017.

[35] Pablo Flores Guridi. La norma isdb-t y un receptor implementado en sdr. 2016.

[36] Raspberry Pi. Raspberry Pi Hardware. https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md. 2017.

[37] Honeywell. 3-Axis Digital Compass IC HMC5883L. 2013.

[38] Naylamp Mechatronics. Tutorial Magnetómetro HMC5883L. http://www.naylampmechatronics.com/blog/49_tutorial-magnetometro-hmc5883l.html. 2017.

[39] Jeff Rowberg. I2Cdev. https://www.i2cdevlib.com. 2017.

[40] Meng Ji and Magnus Egerstedt. Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4):693–703, 2007.

[41] Leonard Kleinrock and Fouad Tobagi. Packet switching in radio channels: Part i–carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE transactions on Communications*, 23(12):1400–1416, 1975.

[42] Federico Larroca, Pablo Flores Guridi, G Sena Gomez, V Gonzalez-Barbone, and Pablo Belzarena. An open and free isdb-t full-seg receiver implemented in gnu radio. *WInnComm*, 2016.

[43] Arch Linux. Daemons (español), https://wiki.archlinux.org/index.php/Daemons_(Espa%C3%B1ol). 2017.

[44] Arch Linux. Systemd (español), https://wiki.archlinux.org/index.php/Systemd_(Espa~nol). 2017.

[45] Arch Linux. https://wiki.archlinux.org/index.php/Arch_Linux. 2017.

[46] Arch LinuxARM. https://archlinuxarm.org/. 2017.

[47] X. Luo. *GPS Stochastic Modelling: Signal Quality Measures and ARMA Processes*. Springer Theses. Springer Berlin Heidelberg, 2013.

[48] Raquel G Machado and Alexander M Wyglinski. Software-defined radio: Bridging the analog–digital divide. *Proceedings of the IEEE*, 103(3):409–423, 2015.

[49] the operating system built for your Internet of Things Microsoft. Windows 10 IoT Core. https://developer.microsoft.com/es-es/windows/iot. 2017.

[50] Raspberry Pi. Ten millionth Raspberry Pi and a new kit. https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/. 2016.

[51] USRP B200 mini. https://www.ettus.com/product/details/USRP-B200mini-i-board. 2017.

[52] Joseph Mitola. Cognitive radio—an integrated agent architecture for software defined radio. 2000.

[53] Vicent Mayans Roca. Desarrollo multiplataforma de aplicaciones de control y comunicación para robots móviles. https://riunet.upv.es/bitstream/handle/10251/17140/Memoria.pdf?sequence=1. 2012.

[54] Robin R. Murphy. Introduction to ai robotics, http://www.profesaulosuna.com/data/files/ROBOTICA/ROBOTICS%20EBOOKS/Introduction%20to%20AI%20Robotics.pdf. 2000.

[55] u-blox 6 GPS Modules NEO-6. https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/GY-NEO6MV2-GPS-Module-Datasheet.pdf. 2011.

[56] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634, 2014.

[57] George Nychis, Thibaud Hottelier, Zhuocheng Yang, Srinivasan Seshan, and Peter Steenkiste. Enabling mac protocol implementations on software-defined radios. 9:91–105, 2009.

[58] A Small List of FPGA Boards. http://gfix.dk/2013/12/a-small-list-of-fpga-boards/. 2017.

[59] Dan Charrois. Study of the Accuracy of Averaged Non-Differential GPS Measurements. https://www.syz.com/gps/gpsaveraging.html. 1999.

[60] CentOS. CentOS Linux on the Raspberry Pi 3. https://wiki.centos.org/SpecialInterestGroup/AltArch/Arm32/RaspberryPi3. 2016.

[61] opendigitalradio.org. Usrp b200 measurments, http://wiki.opendigitalradio.org/USRP_B200_Measurements.

[62] precise location PHILIPS. Perfect light. http://www.lighting.philips.com/main/systems/themes/led-based-indoor-positioning. 2017.

[63] Raspberry Pi. Raspberry pi 3 model b https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. 2016.

[64] Raspberry Pi. Raspberry pi, about ushttps://www.raspberrypi.org/about/. 2017.

[65] prof David L. Wilson. GPS ERROR WHEN AVERAGING POSITION. https://web.archive.org/web/20111118142719/http://users.erols.com/dlwilson/gpsavg.htm. 1999.

[66] pyserial. Welcome to pyserial's documentation https://pythonhosted.org/pyserial/. 2017.

[67] python. TkInter. https://wiki.python.org/moin/TkInter. 2017.

[68] GNU Radio. Gnu radio manual and c++ api reference 3.7.10.1, https://gnuradio.org/doc/doxygen/page_pmt.html. 2017.

[69] GNU Radio. Main page, https://wiki.gnuradio.org/index.php/Main_Page. 2017.

[70] GNU Radio. Pmt page, https://gnuradio.org/doc/doxygen/page_pmt.html. 2017.

[71] NAVSTAR GPS USER EQUIPMENT INTRODUCTION. Public release version. https://www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf. 1996.

[72] Pidora. Fedora Remix. http://pidora.ca/. 2017.

[73] Ettus Research. Ettus research, about us, https://www.ettus.com/about. 2017.

[74] Ettus Research. http://olifantasia.com/gnuradio/usrp/files/datasheets/USRP_B100_datasheet.pdfhttps://kb.ettus.com/B100. 2017.

[75] Arduino Uno Rev3. https://store.arduino.cc/usa/arduino-uno-rev3. 2017.

[76] Michael Rice. Digital communications: A discrete-time approach. 2009.

[77] Sensor De Velocidad Encoder Optico Arduino Robotica. https://articulo.mercadolibre.com.co/MCO-454060091-sensor-de-velocidad-encoder-optico-arduino/-robotica-_JM. 2017.

[78] Vicent Mayans Roca. Desarrollo multiplataforma de aplicaciones de control y comunicación para robots móviles. *Escuela Técnica Superior de Ingeniería Informática Universitat Politècnica de València*, 2012.

[79] Tom Rondeau. https://github.com/gnuradio/gnuradio/blob/master/gr-analog/include/gnuradio/analog/agc2.h. 2017.

[80] Iulian Rosu. Automatic gain control (agc) in receivers. *PDF document], http://www. qsl. net/va3iul*, 2014.

[81] Ylan Archimowicz Santiago Martínez. Edro - un enjambre de robots. 2011.

[82] Michael Schuresko and Jorge Cortés. Distributed motion constraints for algebraic connectivity of robotic networks. *Journal of Intelligent and Robotic Systems*, 56(1-2):99–126, 2009.

[83] IEEE Computer Society. Ieee standard for information technology— telecommunications and information exchange between systems local and metropolitan area networks— specific requirements. 2012.

[84] NMEA 0183 Standard. http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp. 2008.

[85] Andrew S. Tanenbaum. Redes de computadoras, 4ta edición. 2003.

[86] Andrew S Tanenbaum. Sistemas operativos modernos. 2009.

[87] M Taylor. A guide to nfv and sdn. white paper by metaswitch networks. *http://www.metaswitch.com/sites/default/files/Metaswitch White-Paper NFVSDN final rs. pdf*, 2014.

[88] Ilenia Tinnirello, Giuseppe Bianchi, Pierluigi Gallo, Domenico Garlisi, Francesco Giuliano, and Francesco Gringoli. Wireless mac processors: Programming mac protocols on commodity hardware. pages 1269–1277, 2012.

[89] Mikal Hart. TinyGPS. `http://arduiniana.org/libraries/tinygps/`. 2017.

[90] Raspbian. Welcome to Raspbian. `https://www.raspbian.org/`. 2017.

[91] Juan Manuel Toloza. *Algoritmos y t'ecnicas de tiempo real para el incremento de la precisi'on posicional relativa usando receptores GPS est'andar*. PhD thesis, Facultad de Inform'atica, 2013.

[92] Arduino. PWM Tutorial. `https://www.arduino.cc/en/Tutorial/PWM`. 2017.

[93] GNSS evaluation software for Windows u center. `https://www.u-blox.com/sites/default/files/u-center_UserGuide_%28UBX-13005250%29.pdf`. 2017.

[94] ubuntu-mate. Ubuntu MATE for Raspberry Pi 3. `https://ubuntu-mate.org/blog/ubuntu-mate-for-raspberry-pi-3/`. 2017.

[95] Omer Ikram ul Haq. Compass. `https://github.com/jdn-aau/sketchbook/blob/master/sketchbook/libraries/compass/compass.cpp`. 2014.

[96] David A Westcott, David D Coleman, Ben Miller, and Peter Mackenzie. *CWAP Certified Wireless Analysis Professional Official Study Guide: Exam PW0-270*. John Wiley & Sons, 2011.

[97] Wikipedia. Kleroterion, `https://en.wikipedia.org/wiki/Kleroterion`. 2017.

[98] Wikipedia. Omni wheel, `https://en.wikipedia.org/wiki/Omni_wheel`. 2017.

[99] Wikipedia. `https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_desplazamiento_de_fase`. 2017.

[100] Interview: Wireless Bay Area Adventure with USRP B200 Software Defined Radio. `https://www.ettus.com/blog/2013/09/interview-wireless-bay-area-adventure-with-usrp-b200-software-/defined-radio`. 2017.

[101] Guochang Xu and Yan Xu. *GPS: theory, algorithms and applications*. Springer, 2016.

[102] Matthias y Alvaro Ferrando. Paseando por Montevideo. Club de bochas. `http://www.vincealongi.com/montevideo/mvd73.html`. 2017.

[103] Michael M Zavlanos and George J Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on robotics*, 23(4):812–816, 2007.

[104] Michael M Zavlanos and George J Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, 2008.

[105] Michael M Zavlanos, Alejandro Ribeiro, and George J Pappas. Network integrity in mobile robotic networks. *IEEE Transactions on Automatic Control*, 58(1):3–18, 2013.

[106] D. Čabarkapa, I. Grujić, and P. Pavlović. Comparative analysis of the bluetooth low-energy indoor positioning systems. In *2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, pages 76–79, Oct 2015.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el lunes 19 febrero, 2018.
http://iie.fing.edu.uy/

Robots Móviles Autónomos Comunicados con Radio Definida por Software*

Gabriel Bibbó[1] , Mariana Gelós[2] y Martín Randall[3]

***Abstract*— This paper shows the result of combining Software Defined Radio (SDR) with robots that cooperate to achieve a mission. A system composed by two autonomous mobile robots and a base station was built. They work together to direct one of the robots to a location, defined by latitude and longitude. The exchange of data between the actors is achieved using SDR. Carrier-sense multiple access with collision avoidance (CSMA/CA) for wireless networks over GNU Radio was developed to allow this communication. The whole system is built with open-source hardware and software. Conclusions, results obtained, and possible steps to follow in similar projects are established.**

Resumen: En este trabajo se muestra el resultado de combinar Radio Definida por Software (SDR) con robots que cooperan para cumplir con una misión. Se confeccionó un sistema compuesto por dos robots móviles autónomos y una estación base que trabajan en conjunto para lograr que uno de ellos se dirija a una ubicación conocida, definida por latitud y longitud. El intercambio de datos entre los distintos actores se realiza utilizando SDR. Se desarrolló un protocolo de acceso al medio (CSMA/CA) para redes inalámbricas en GNU Radio que permita esta comunicación. Todo el sistema está construido con hardware y software de código abierto. Se establecen conclusiones, resultados obtenidos, y posibles pasos a seguir en proyectos vinculados.

Palabras clave: SDR, GNU Radio, CSMA/CA, Raspberry Pi, cooperative robotics, Arduino, GPS

I. INTRODUCCIÓN

Existen aplicaciones donde se necesita establecer una comunicación inalámbrica con determinados parámetros de calidad de servicio (QoS) en algún punto (o conjunto de puntos) donde previamente no existía comunicación, sin que el usuario deba desplazarse de su posición original. Son ejemplos de éstas, el espionaje militar y el sensado de terrenos en tiempo real, entre otras. En estos casos se pretende recabar información de distintos tipos de sensores (video, temperatura o humedad) que se encuentren ubicados en el punto deseado y transmitirlos en una estación base alejada.

Cuando la distancia entre punto de origen y destino es mayor al alcance del sistema de comunicación, se requieren nodos intermedios repetidores de señal que permitan aumentar el área de cobertura del sistema completo. Una respuesta para estas situaciones es el desarrollo de robots móviles comunicados inalámbricamente, los cuales pueden ser o no autónomos. Estos últimos, al trabajar con una lógica cooperativa, permiten maximizar la calidad de servicio en el destino.

Otro aspecto importante en el desarrollo de aplicaciones tecnológicas, que actualmente se aborda tanto teórica como experimentalmente, es la tecnología de Radio Definida por Software (SDR); que tiene como característica principal obtener sistemas de comunicación completamente reconfigurables. La tecnología SDR, junto con hardware y software de código abierto, permite construir sistemas de comunicación muy flexibles que se pueden desarrollar y usar en muchas aplicaciones.

En este trabajo se desarrolló un sistema que posibilita la interacción entre robots móviles autónomos utilizando SDR. La implementación se hace buscando dirigir un robot (denominado Explorador, ver Figura 1) a una posición deseada, a través de una comunicación inalámbrica usando protocolos sencillos desarrollados por el grupo del proyecto sobre redes inalambricas en GNU Radio (GWN). Se desea mantener una QoS mínima en todo momento entre este robot y una Estación Base. Para esto, puede ser necesaria la participación de otro robot que actúa como repetidor (denominado Repetidor, ver Figura 2) de señal para mejorar la QoS en destino.

II. RADIO DEFINIDA POR SOFTWARE

La tecnología SDR es una arquitectura flexible y versátil en la que las funciones relacionadas con el hardware de comunicaciones se pueden ejecutar a nivel de software, que se puede configurar o programar según las necesidades. Este concepto fue adoptado por Joseph Mitola a principios de la década de 1990. En sus inicios, esta tecnología estaba vinculada a aplicaciones militares, pero con el paso de los años y los avances tecnológicos, se crearon proyectos dirigidos a redes de comunicación y aplicaciones de radio amateur.

El Wireless Innovation Forum [1], en colaboración con el grupo P1900.1 del Instituto de Ingenieros Eléctricos y

*En cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista. Diciembre de 2017.

[1]Gabriel Bibbó es Ingeniero Eléctrico, Universidad de la República, Facultad de Ingeniería, Ave Julio Herrera y Reissig 565, 11200 Montevideo, Uruguay. gabriel.bibbo en eva.fing.edu.uy

[2]Mariana Gelós es estudiante de Ingeniería Eléctrica, Universidad de la República, Facultad de Ingeniería, Ave Julio Herrera y Reissig 565, 11200 Montevideo, Uruguay. mariana.gelos en eva.fing.edu.uy

[3]Martín Randall es Ingeniero Eléctrico, Universidad de la República, Facultad de Ingeniería, Ave Julio Herrera y Reissig 565, 11200 Montevideo, Uruguay. martin.randall en eva.fing.edu.uy

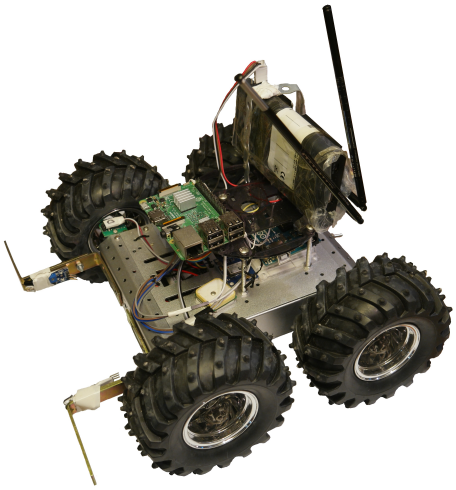


Fig. 1: Foto del robot que se dirige a la posición de destino

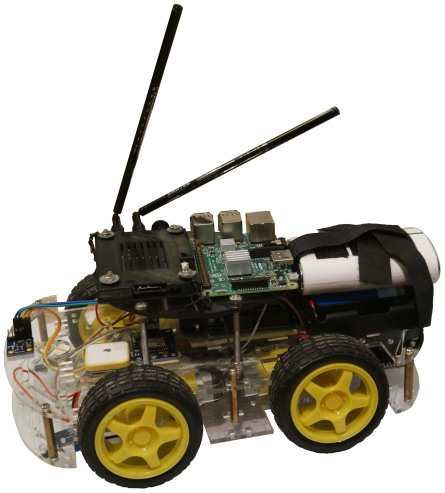


Fig. 2: Foto del robot que oficia de repetidor de señal

Electrónicos (IEEE), estableció la siguiente definición para SDR: “Radio en la que parte o la totalidad de las funciones de la capa física están definidas por software”.

GNU Radio es un conjunto de diferentes archivos y aplicaciones agrupados en bibliotecas para el procesamiento digital de señales, es un software de distribución gratuita bajo la licencia de GNU General Public License (GPL) versión 3, y se puede usar junto con dispositivos SDR. GNU Radio usa el lenguaje de programación Python, que es un lenguaje multiparadigma, que se ejecuta directamente en el sistema operativo. Para construir un sistema de radio (sistema de comunicaciones) se debe crear un gráfo, donde los bloques utilizados representan las etapas de procesamiento y sus conexiones representan el flujo de datos.

En el trabajo actual se usa la aplicación GNU Radio Companion (GRC), que es el entorno gráfico de GNU Radio. Esta aplicación es una herramienta gráfica basada en bloques que permite crear gráficos de flujos de señal y generar códigos fuente a partir del flujo gráfico.

III. Sistema de comunicación

El sistema de comunicación está compuesto por el software y el hardware de los dos robots y la estación base. La comunicación inalámbrica entre los actores fue implementada utilizando el entorno de GNU Radio, guiándonos por el protocolo de acceso múltiple por detección de portadora y prevención de colisiones (CSMA/CA) definido en el estándar IEEE 802.11.

Parte de las bibliotecas utilizadas fueron desarrolladas por el grupo ARTES [2] del IIE, que ha venido trabajando en SDR en el proyecto GWN[12]. El objetivo de GWN es permitir y facilitar el trabajo con SDR para redes de datos. A diferencia de la mayoría de las aplicaciones que utilizan SDR, que trabajan con flujos continuos de datos, las redes de datos necesitan de mensajes encapsulados, con ciertas características y estructura para su manejo a diferentes niveles.

La implementación llevada a cabo consta de una capa de enlace de datos que incluye una CSMA/CA (subcapa MAC) que corre sobre el método Stop and Wait (un tipo de protocolo ARQ para el control de errores en la comunicación, de la subcapa LLC). Estos protocolos se programaron como máquinas de estados finitas, usando las facilidades que brinda GWN. La máquina de estados resultante se puede apreciar en la Figura 3. La capa física utiliza modulación digital QPSK y control de errores por redundancia cíclica, además de correccion de errores de fase, frecuencia y sincronizacion temporal.

La QoS se calcula como la tasa de éxitos: la cantidad de tramas recibidas con éxito frente a la cantidad total de tramas enviadas. Si todos los intercambios se realizan con éxito, se tiene que la QoS vale 100 %.

El hardware elegido para SDR es de bajo peso, pensado para poder ser transportado por los robots móviles. Es así que ellos utilizan un Raspberry Pi con el sistema operativo Arch Linux para correr GNU Radio y el periferico Ettus USRP B200 mini para la transmisión y recepción de la señal. La estación base corre GNU Radio en un laptop con Linux y utiliza un Ettus USRP B200.

IV. Componentes de los robots

Se construyeron dos robots rodados con autonomía de funcionamiento, coordinados mediante lógica cooperativa, capaces de dirigirse de forma autónoma, a una posición de destino indicada por latitud y longitud. Para poder conseguirlo, cuentan con cinco componentes de hardware fundamentales:

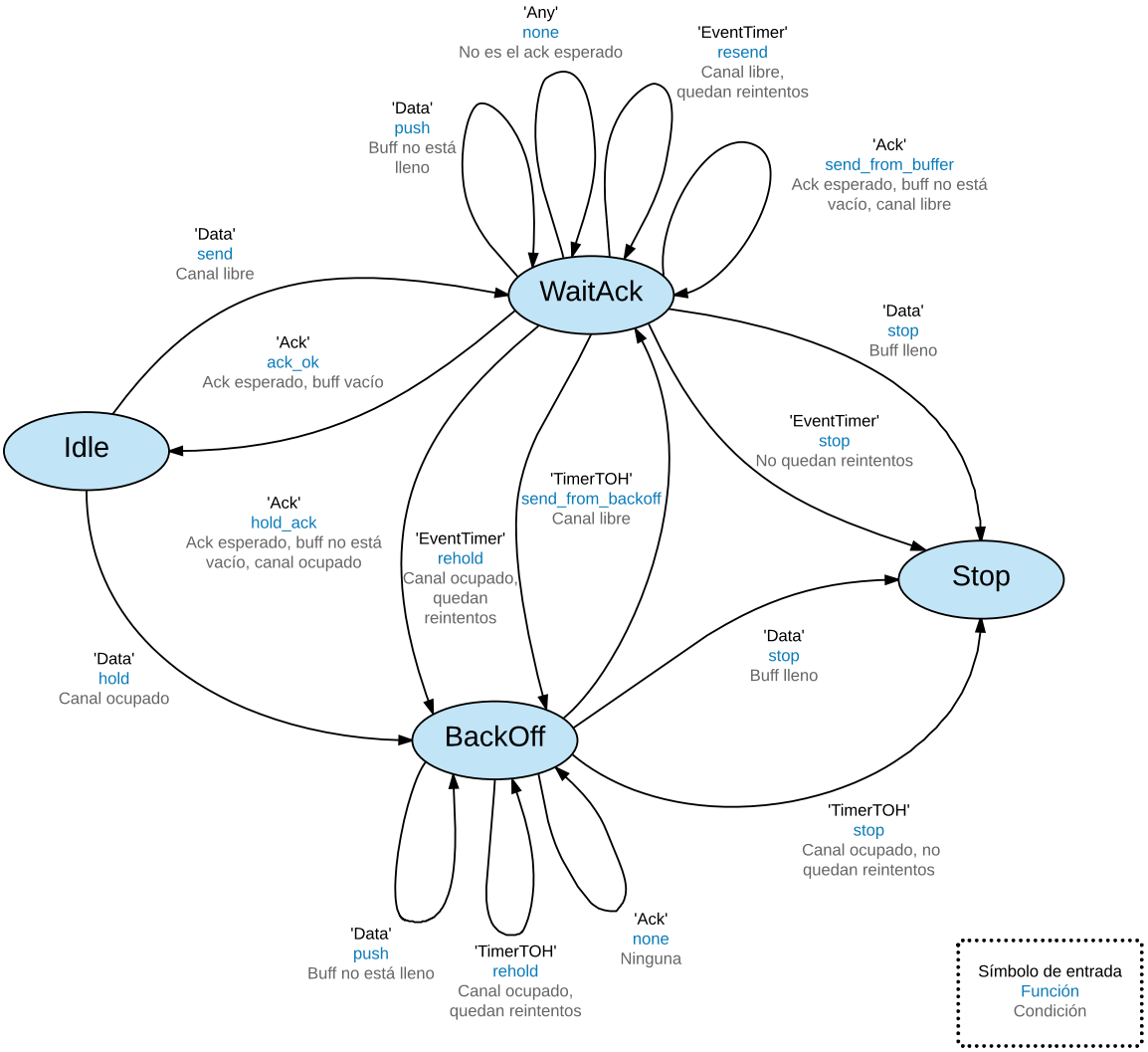


Fig. 3: Máquina de estados implementada del protocolo CSMA/CA

- **Estructura mecánica**, que le permita avanzar en línea recta y girar sobre su propio eje, soportando toda la carga del robot. Consta de:
  - **chasis**, que sostiene las partes y les da rigidez,
  - **cuatro ruedas con cuatro motores**, encargadas de desplazar al robot de una ubicación a otra,
- **Sistema sensorial**, que le permita conocer su posición actual, orientarse hacia su destino, evitar choques y medir su desplazamiento efectivo. Consta de:
  - **sensor GPS GY-NEO6MV2** [3], utilizado para el posicionamiento del robot en exteriores,
  - **compás digital CJ-M49** [4], es un sensor de cámpo magnético que, al igual que una brújula, se guía por el Norte magnético, y es utilizado para orientar al robot a la posición de destino,
  - **dos sensores IR**, ubicados en el frente que detectan obstáculos,
  - **dos encoders ópticos para ruedas**, colocados en las ruedas frontales para medir su avance efectivo,
- **Sistema de control**, como centro encargado de gestionar la navegación automática de los robots. Consta de:
  - **microcontrolador Arduino UNO** [5], encargado de la integración de la información de todos los sensores y del comando de los motores,
  - **driver L298N para motores** [6], interfaz necesaria para controlar y alimentar a los motores desde Arduino,
- **Sistema de comunicación**, encargado de establecer la comunicación inalámbrica entre robots y estación base. Consta de:
  - **Rasbperry Pi 3 Modelo B** [7], que envía y recibe posiciones desde Arduino y GNU Radio,
  - **USRP B200 mini con antenas** [8], le brinda

conectividad inalámbrica a los robots utilizando GNU Radio,

- **Alimentación**, que les brinda a los robot autonomía de la red eléctrica. Consta de:
  - **pilas recargables**, que alimentan al driver de motores,
  - **baterías recargables USB**, que alimentan al Raspberry Pi, Arduino, sensor GPS, compás digital, sensores IR, encoders y USRP B200 mini.
- **cables USB y conectores varios**

Toda la programación del automatismo de los robots está hecha sobre Arduino, para lo que se utilizó el propio software de Arduino (IDE) [9].

Los robots son autónomos en tanto ellos mismos son capaces de digirirse a la posición indicada por la Estación Base. Esto lo logran mediante sucesivas iteraciones; ejecutan un loop principal en el que inicialmente esperan una posición de destino válida recibida inalámbricamente mediante GNU Radio. En el momento que se les envía una posición, los robots realizan una lectura del sensor GPS para conocer su ubicación actual. Mide el ángulo entre su direción actual y el Norte utilizando el compás y combina esta información con la del GPS para saber exactamente cuánto debe girar para orientarse hacia su destino. Una vez orientados correctamente, se desplazan 2 metros[1], se detienen y vuelven a enviar su posición. El robot llega a su destino luego de haber realizado varias veces este loop.

Se utilizan algunas bibliotecas de terceros para la integración del compás y el GPS. Para el compás digital se utilizó la biblioteca desarrollada por Omer Ikram ul Haq [14], que permite calibrar el magnetómetro previo a su utilización y obtener la medida del ángulo entre el eje 'X' del sensor y el Norte magnético. La comunicación con este perisférico es utilziando el protocólo I2C, para lo que se utilizó la biblioteca de Jeff Rowberg[13]. Para el GPS se utilizó la biblioteca de Mikal Hart[15].

Se creó la biblioteca *ubicacion.cpp* (ver Figura 5) que incluye las funciones encargadas de establecer la comunicación con el magnetómetro y el GPS, del tratamiento de las medidas que de ellos se obtienen y funciones que permitan la comunicación capas arriba con Raspberry Pi. Se creó tambien la biblioteca *motores.cpp* que agrupa todas las funciones que involucran directamente el manejo del driver de motores.

V. Integración del sistema completo

Para poner en funcionamiento el sistema de robótica cooperativa, se crearon programas en Python que permitan integrar el sistema de comunicación y de navegación.

[1]Esta distancia está determinada por la incertidumbre de los sensores usados para el posicionamiento y porque la calidad de servicio de la comunicación decae abruptamente en distancias del orden de los 2 metros.

El sistema completo es controlado centralmente por la Estación Base. En la Figura 4 se ve cómo el programa que implementa la inteligencia de la Estación Base (*estacion_base.py*), se encuentra en comunicación con GNU Radio (*Full_CSMA_EB.py*) a través de sockets de dominio Unix. Por su parte, los Raspberry Pi de los robots ejecutan el programa *com_serial.py*, el cual actúa como intermediario entre GNU Radio y Arduino, comunicándose con éste a través del puerto USB, tal como se observa en la Figura 5.

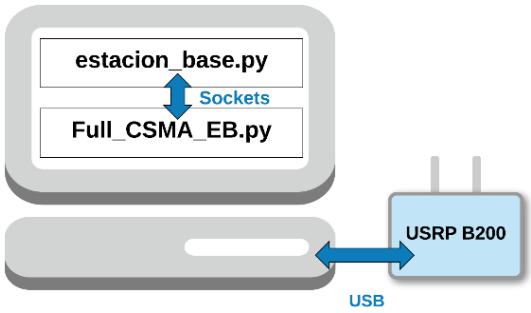


Fig. 4: Diagrama de bloques de la Estación Base

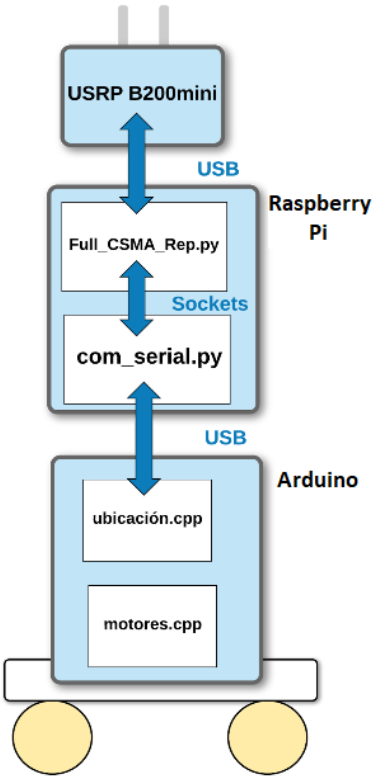


Fig. 5: Diagrama de bloques de los robots

El sistema cooperativo funciona a través del intercambio ordenado de mensajes entre la Estación Base y los robots. Inicialmente, el Explorador obtiene su posición mediante GPS. Esa posición es procesada por Arduino y transmitida a Raspberry Pi. El programa *com_serial.py* recibe esa

posición y arma un mensaje cuya dirección de destino es la Estación Base. GNU Radio recibe este mensaje mediante sockets, lo procesa y lo transmite al medio a través del USRP del robot.

El camino del mensaje en la Estación Base es simétrico: es recibido por el USRP, procesado por GNU Radio y finalmente decodificado en el programa $estacion\_base.py$, el cual verifica que se trate de una combinación latitud-longitud válida. Luego, éste programa se encarga de ejecutar el algoritmo de medida de la QoS y, si ésta es mayor a cierto umbral, arma un mensaje con la posición a la que debe dirigirse el Explorador. Cuando el robot recibe la posición, comienza a dirigirse a ella mediante las directivas de Arduino.

Este ciclo se repite mientras que la QoS sea buena (mayor al umbral). Cuando la Estación Base detecta que la QoS está por debajo del umbral, le indica al robot Repetidor que se dirija a la última posición donde el Explorador tuvo buena calidad de señal. Una vez que el Repetidor llega a su destino, se vuelve a medir la QoS entre el Explorador y la Estación Base. En caso que la QoS haya aumentado lo suficiente por efecto del Repetidor, el Explorador puede seguir avanzando hacia su destino.

Para que los robots puedan funcionar autónomamente, sin la necesidad de intervención de un usuario, se crearon dos programas adicionales que corren en Raspberry Pi. Uno implementa un $watchdog$, que verifica periódicamente el estado de ejecución de los programas, y en caso de encontrar que alguno se haya cortado, lo reinicia. El otro se encarga de ejecutar automáticamente todos los procesos en el orden correcto luego de la etapa de booteo.

VI. RESULTADOS OBTENIDOS

Se implementó un sistema autónomo de dos robots cooperando para establecer una comunicación inalámbrica en un punto determinado, con un alcance de hasta 10 metros de la Estación Base utilizando un Repetidor.

VI-A. SISTEMA DE COMUNICACIÓN

El sistema de comunicación utiliza la banda de 400MHz y tiene una capacidad de 133 Kbps.

Como se ve en la tabla I, el alcance del sistema sin Repetidor manteniendo una buena QoS (mayor a 70 %) es de 4 metros. Entre 5 y 7 mts se tiene una QoS que varía entre 65 % y 11 %. Con distancias mayores a 7 mts, la comunicación ya se ve seriamente comprometida, teniendo una QoS del 10 % o menor.

Con el Repetidor en funcionamiento, posicionado entre la Estación Base y el Explorador, se obtiene que la QoS se mantiene por encima del umbral hasta los 7 mts. Entre 7 y 10 mts se tiene una QoS que varía entre 65 % y 25 %. La distancia máxima a la que se puede llegar con una QoS mayor al 20 % es de 10 mts.

TABLE I: Tabla de QoS en función de la distancia. Observar el descenso abrupto a partir de los 6 metros.

Distancia entre nodos (metros)	**QoS (sin repetidor) (%)**
1	93
2	87
3	92
4	91
5	65
6	54
7	12
8	6
9	4
10	7

La primera limitante que plantea esta implementación se debe a la potencia de los USRP B200, principalmente en transmisión, ya que en recepción aumentar la potencia también aumenta el piso de ruido y la potencia recibida del pulso que el propio USRP envía, lo que puede dificultar la recepción de pulsos ajenos.

Con respecto a las ganancias, vale la pena destacar que la potencia real de salida tiene un vículo directo con la alimentación. Es conveniente ajustar la ganancia por ejemplo de la Estación Base, que es la más versátil por estar alimentada desde una computadora portátil, y tiene mayor margen respecto de los límites planteados por el equipo.

Por otra parte, la medida de QoS es un parámetro importante a la hora de tomar desiciones sobre el movimiento o no de los robots. Por esto, la medida debe ser confiable y reflejar, con la mayor fidelidad posible, el empobrecimiento de la señal a medida que aumenta la distancia. Se propone, como trabajo a futuro, mejorar el algoritmo de medida de la QoS.

VI-B. ROBOTS

Cada robot tiene una estructura mecánica que le permite moverse soportando todo el peso de sus componentes. Los movimientos que son capaces de realizar son de giro sobre su propio eje, avance y retroceso en línea recta. Esto es posible gracias al diseño de sus elementos constructivos. La incorporación de sensores IR les permiten esquivar obstáculos. Por otra parte, son capaces de conocer su posición utilizando GPS y de orientarse a su ubicación de destino al combinar esta información con las medidas del compás magnético. El esquema de funcionamiento para desplazarse a la posición de destino es de modo iterativo, por lo que repite un algoritmo en el transmite su posición, calcula la calidad de servicio de la comunicación, se orienta a su destino y avanza una cierta distancia. El tiempo de autonomía, determinado por la presencia de dos baterías independientes, es de dos horas, suficiente para que puedan

completar sus tareas satisfactoriamente.

Por otra parte, existen condiciones que limitan el funcionamiento autónomo de los robots. Estos requieren que la superficie sobre la que se desplazan tenga un coeficiente de rugosidad que le permita a las ruedas deslizar cuando el robot gira sobre su propio eje y rodar sin deslizar, al avanzar. La presencia de sensores IR obliga a un funcionamiento nocturno para evitar la interferencia de la luz infrarroja del sol. La locación debe ser a cielo abierto y con pocas edificaciones cercanas que deterioren la recepción del GPS. El nivel de precisión de la posición de destino está condicionada por la incertidumbre de las medidas del GPS.

En este sentido, se plantean propuestas de solución a los diferentes problemas encontrados: modificaciones a las ruedas o al chasis que posibiliten mayor movilidad -usando ruedas Omni Wheel por ejemplo-, la utilización de GPS diferencial para disminuir el error en la lectura de la posición, incorporar al filtro de Kalman para mejorar la estimación de la posición, o descartar el uso de GPS y plantear formas alternativas de navegación, como ser la odometría.

VI-C. INTEGRACIÓN

La integración de robotica cooperativa con SDR para el establecimiento de una red multinodos plantea ciertas complicaciones a tener en cuenta:

- El bajo alcance del sistema de comunicación determina que sea muy difícil la utilización exclusiva del GPS para el posicionamiento, dado que éste tiene una incertidumbre en sus medidas de 3 mts aproximadamente.
- Los sistemas de SDR consumen mucha potencia, por lo que se necesita una batería pesada. Los motores del robot requieren una fuente de alimentación independiente, lo que genera que los robots sean pesados dificultando su movilidad.
- Los robots, al desplazarse, generan deslices o sacudidas que empeoran la comunicación al cambiar de posición a las antenas.
- La independencia del GPS exige mejor movilidad: se prioriza la odometría y/o el compás.

VII. CONCLUSIONES

Es posible la puesta en marcha de un proyecto de vinculación entre robótica cooperativa y redes de datos en SDR para el establecimiento de una red multinodos en proyectos experimentales de bajo costo. La performance del sistema implementado tiene asociadas limitantes que se deben solucionar en trabajos futuros si se quiere escalar el proyecto a aplicaciones industriales o militares.

Quedan planteadas posibles mejoras en el alcance del sistema de comunicación, el algoritmo de medida de la QoS y en la precisión con la que los robots resuelven su posicionamiento. Para la lógica del sistema se estudió la posibilidad de determinar la posición deseable de los robots a partir de un algoritmo de optimización como el presentado en el artículo de Zavlanos y otros[11], que finalmente no se llegó a implementar en la práctica.

Siguiendo con la filosofía de hacer crecer a las comunidades de software libre, creamos un GitHub[10] donde está subida la documentación, en la que se analiza el proyecto con mayor profundidad, y se colcan a disposición todos los programas desarrollados y demás información relevante: 1) guías de instalación de Arch Linux y GNU Radio e imágenes de los USRP, 2) códigos de GNU Radio para el establecimiento de la comunicación, 3) diagrama de conexionado de robots, programa de Arduino para navegación automática, 4) script para pruebas de la comunicación, 5) archivos Python que brindan la inteligencia general del sistema.

REFERENCES

[1] Wireless Innovation Forum, 2017. Disponible: $www.wirelessinnovation.org/$.

[2] Grupo ARTES (Analisis de Redes, Traco y Estadsticas de Servicio): grupo academico de la Universidad de la Republica vinculado al estudio de las redes de datos, 2018. Disponible: $iie.fing.edu.uy/investigacion/grupos/artes/es/inicio/$.

[3] NEO-6, u-blox 6 GPS Modules, 2011. Disponible: $www.openimpulse.com/blog/wp - content/uploads/wpsc/downloadables/GY - NEO6MV2 - GPS - Module - Datasheet.pdf$

[4] Honeywell. "3-Axis Digital Compass IC HMC5883L", 2013.

[5] Arduino Uno Rev3, 2017. Disponible: $store.arduino.cc/usa/arduino - uno - rev3$

[6] STMicroelectronics, "L298. DUAL FULL-BRIDGE DRIVER", 2000.

[7] Raspberry Pi, 2017. Disponible: $www.raspberrypi.org/about/$

[8] Ettus, "USRP Hardware Driver and USRP Manual", 2017. Disponible: $files.ettus.com/manual/page_usrp_b200.html$

[9] Arduino - Software, "Access the Online IDE", 2018. Disponible: $www.arduino.cc/en/Main/Software$

[10] Gabriel Bibbó, Mariana Gelós y Martín Randall, "GitHub Proyecto RoCo: Robots Comunicados", 2017. Disponible: $github.com/MartinRandallC/RoCo/wiki$

[11] "Network integrity in mobile robotic networks", Zavlanos, Michael M and Ribeiro, Alejandro and Pappas, George J, 2013. IEEE Transactions on Automatic Control, volume 58, number 1, pages 3–18, publisher IEEE.

[12] "GWN: A framework for packet radio and medium access control in GNU Radio", González-Barbone, Víctor and Belzarena, Pablo and Larroca, Federico and Randall, Martín and Romero, Paola and Gelós, Mariana, 2017, WinnComm 2017.

[13] Je Rowberg, "I2Cdev", 2017. Disponibie: $www.i2cdevlib.com$.

[14] Omer Ikram ul Haq. "Compass", 2014. Disponible: $github.com/jdn - aau/sketchbook/blob/master/sketchbook/libraries/compass/compass.cpp$

[15] Mikal Hart, "TinyGPS", 2017. Disponible: $arduiniana.org/libraries/tinygps/$