Universidad de la República
Facultad de Ingeniería

WSNvision

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

Florencia Arbío, Federico López, Miguel Pereyra

en cumplimiento parcial de los requerimientos
para la obtención del título de ingeniero electricista.

Tutor Leonardo Barboni ..... Universidad de la República

Tribunal

Leonardo Barboni ..... Universidad de la República
Miguel Barreto ..... Universidad de la República
Claudina Rattaro ..... Universidad de la República
Conrado Rossi ..... Universidad de la República

Montevideo
jueves 27 agosto, 2015

WSNvision, Florencia Arbío, Federico López, Miguel Pereyra.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 143 páginas.
Compilada el jueves 27 agosto, 2015.
http://iie.fing.edu.uy/

En la vida hay tres clases de personas: las que saben contar, y las que no.

Saber popular

Agradecimientos

Muchas personas contribuyeron al desarrollo de este proyecto. Expresamos nuestro particular agradecimiento a:

Leonardo Barboni, por la continua disponibilidad y apoyo.

José Luis Vila, por el soporte en las actividades relacionadas al diseño e implementación de la mecánica.

Mariana Siniscalchi, por este último mes de constante ayuda.

Los docentes de las asignaturas SISEM y RSI del Instituto de Ingeniería Eléctrica.

Nuestras familias y amigos, por el apoyo incondicional.

Ale, Xime y la Chiqui, por su apoyo moral, psicológico y afectivo durante este largo año de proyecto.

Y a Julieta, que viene en camino.

Resumen

El presente documento describe el trabajo realizado en marco del proyecto *WSNvision, Wireless Sensor Network with Vision*. En el mismo, se implementó una red de sensores inalámbricos (RSI) con capacidad visual a fin de detectar de forma temprana plagas en cultivos frutales. *WSNvision* se enmarca en el proyecto FPTA-INIA 313 "GERVASIO: Generalización de las redes de sensores inalámbricos como herramienta de valorización en sistemas vegetales intensivos", presentado por el grupo de Microelectrónica del Instituto de Ingeniería Eléctrica (Facultad de Ingeniería, UdelaR) en abril de 2014. El proyecto de investigación tiene como contraparte al INIA y a la cooperativa agraria JUMECAL. Ésta última tiene sus campos en la cuenca del río Santa Lucía.

El monitoreo del nivel de población de polillas *Cydia Pomonella* [1] y *Cydia Molesta* [2] que infectan y afectan los cultivos de manzanos, membrilleros, nogales y perales se realiza mediante el uso de trampas. Éstas contienen sustancias adhesivas y feromonas en sus pisos que atraen a los machos de la especie, dejándolos inmovilizados. Las trampas se colocan en primavera (época de reproducción de las polillas), de forma distribuida en el terreno, colgadas en el tercio más alto de los árboles.

Existen trabajadores encargados de recorrer periódicamente los cultivos, reemplazando el piso de cada trampa por uno nuevo. Los pisos extraídos son entregados a personal calificado que realiza el conteo de polillas, para determinar si existe plaga.

El traslado de pisos, conteo manual de polillas y posterior registro de conteo, trae limitaciones y errores asociados a la intervención de operadores humanos. Del mismo modo que no facilita la centralización de información a nivel regional.

Motivándose en esta problemática, se implementó un sistema de transmisión y registro de imágenes de poblaciones de insectos en las trampas, operado por una RSI conectada a un PC. El sistema permite evaluar la evolución diaria de insectos, generar alertas tempranas y centralizar la información de manera automática y libre de errores humanos. A través de un banco de imágenes es posible construir mapas de las poblaciones de plagas, identificando los predios y cultivos más sensibles a los ataques y valorando qué técnicas de control pueden ser más efectivas. Todo esto contribuye a un uso más eficiente de pesticidas y por consiguiente a una menor contaminación de la cuenca del río Santa Lucía.

La red de sensores implementada consta de cuatro nodos colocados en sus respectivas trampas (nodos *WimSN*) y un nodo *sink*, responsable de actuar como interfaz entre los nodos y el PC, y de gobernar el funcionamiento de la red. Los enlaces de red son lineales, pudiendo existir una separación máxima entre nodos de 255 m si se tiene visibilidad entre

[1] También denominada Carpocapsa

[2] También denominada Grafolita

antenas [3]. El sistema es capaz de tomar imágenes del piso de cada trampa tantas veces al día como el usuario desee y enviarlas al nodo *sink* para su posterior visualización en el PC.

Cada nodo *WimSN* se encuentra conectado a un circuito interfaz, a través del cual comanda la cámara de fotos y el sistema de iluminación LED para tomar las imágenes. El sistema nodo-cámara-iluminación se alimenta con tres baterías AA. El nodo sink se comunica con el PC a través de un circuito interfaz necesario para la comunicación UART. Al recibir una imagen, el PC la almacena con nombre del nodo emisor y fecha y hora de recepción, enviándola además a casillas de correo electrónico configuradas por el programador.

Se implementó un protocolo de transmisión de información por RF que asegura la recepción correcta de la imagen en el nodo *sink*.

Adicionalmente, se elaboró un banco de imágenes con diferentes opciones de iluminación, compresión y resolución, y se caracterizó el alcance de la radio de los nodos en estudio.

El conteo automático de polillas se excluyó del alcance debido a inconvenientes surgidos en el transcurso del proyecto, que implicaron agregar al alcance otras tareas que originalmente estaban por fuera. Esto se explica en la sección 7.2. Si bien no se implementó el conteo automático, sí se implementó el almacenamiento de las imágenes en la memoria flash de cada nodo para facilitar el procesamiento de las mismas en etapas futuras.

[3]Esta distancia se reduce a 60 m si no existe visibilidad entre antenas.

Glosario

ACK Del inglés (Acknowledgment), reconocimiento de datos en comunicaciones digitales.

DUT Del inglés (Device Under Test), dispositivo bajo prueba.

GPIO Del inglés (*General Purpose Input Output*), pines de entrada/salida de propósito general.

IIE Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República.

INIA Instituto Nacional de Investigación Agropecuaria.

J-TAG Del inglés (Joint Test Action Group), es el nombre comúnmente utilizado para la norma IEEE 1149.1 utilizada para testear PCB e IC.

JUMECAL *Jumecal Melilla Cooperativa Agraria*, cooperativa agraria.

PFC Proyecto de Fin de Carrera.

RDC Del inglés (Radio Duty Cycling), Ciclo de trabajo de la radio.

RF Del inglés (*Radio Frecquency*), comunicación por radio frecuencia.

RSI Red de Sensores Inalámbricos.

RTOS Del inglés (Real-time Operating System), Sistema Operativo de Tiempo Real.

SMA Del inglés (SubMiniature version A), es un tipo de conector coaxial utilizado en radiofrecuencia.

SoC Del inglés (*System on Chip*), sistema completo embebido en un chip.

UART Del inglés (*Universal Asynchronous Receiver/Transmitter*), receptor-transmisor asíncrono universal.

WimSN Del inglés (*Wireless Image Sensor Node*), nodo sensor con capacidad de procesamiento y envío de imágenes.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Resumen

Este capítulo consiste en una introducción al contenido del proyecto. Se describe el problema que motiva la realización de *WSNvision* y el objetivo general del mismo. Se presenta un diagrama funcional del sistema propuesto, junto con su descripción. Se presentan los diferentes actores involucrados, así como las restricciones, alcance y criterios de éxito que se definieron junto con el tutor para enmarcar el trabajo realizado. Se muestra un diagrama de bloques del sistema y se realiza un breve resumen del contenido de los capítulos de este documento.

1.2. Definición del problema y motivación del proyecto

El monitoreo del nivel de población de polillas *Cydia Pomonella* [19, 20] [1] y *Cydia Molesta* [24] [2] que dañan cultivos de manzanos, perales, membrilleros y nogales se realiza mediante el uso de trampas tipo Wing [23]. Éstas contienen sustancias adhesivas y feromonas ISOMATE C Plus [18] en sus pisos que atraen a los machos de la especie, dejándolos inmovilizados. Insectos adultos de *Cydia Pomonella* y *Cydia Molesta* se pueden observar en las figuras 1.1a y 1.1b respectivamente. En la figura 1.2 se puede observar una manzana dañada por estas polillas.

[1] También denominada Carpocapsa
[2] También denominada Grafolita

(a) (b)

Figura 1.1: (a) Polilla *Cydia Pomonella*, tomada de [19] (b) Polilla *Cydia Molesta*, tomada de [19]

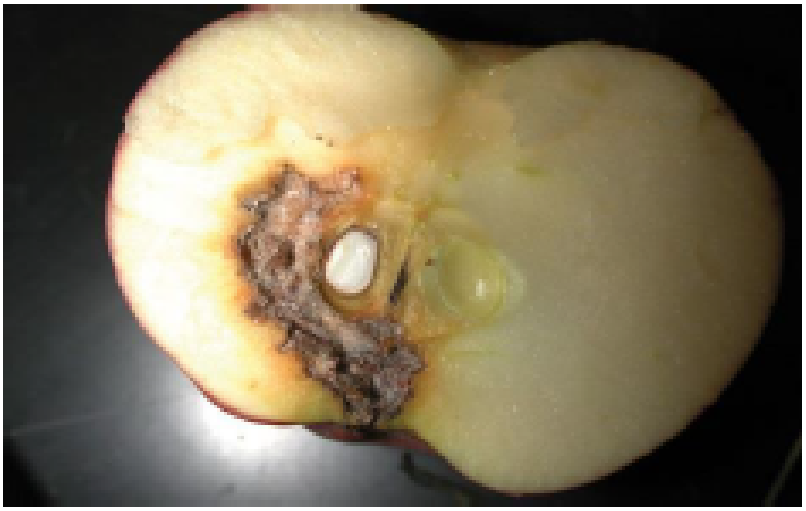


Figura 1.2: Manzana afectada por polillas Cydia Pomonella y Cydia Molesta, tomada de [19]

Las trampas se colocan en primavera (época de reproducción de las polillas), de forma distribuida en el terreno, colgadas en el tercio más alto de los árboles, como se observa en la figura 1.3. Esta imagen fue tomada de un manzano en épocas donde ya estaba perdiendo su follaje.

Figura 1.3: Trampa tipo Wing colocada para control de plagas.

El piso de esta trampa se muestra en la figura 1.4. Se puede ver que el mismo tiene insectos, hojas y tierra adherida. Puede verse sobre la derecha el dispositivo que contiene las feromonas.

Usualmente cada dos semanas un trabajador recorre los cultivos, reemplazando el piso de cada trampa por uno nuevo. Los pisos extraídos son entregados a personal calificado que realiza el conteo de polillas, para determinar si existe plaga. El sistema actual es propenso a errores dados por la intervención de operadores humanos. Además, dificulta la centralización de la información para realizar estadísticas y procesamientos que permitan caracterizar zonas o detectar plagas de forma temprana.

Figura 1.4: Piso de una trampa tipo Delta.

1.3. Descripción del Proyecto y antecedentes

WSNvision refiere al diseño e implementación de una red de sensores inalámbricos con capacidad de transmisión de información visual, utilizada para la detección temprana de plagas en cultivos frutales. Se parte del prototipo de nodo diseñado por el Proyecto de Fin de Carrera Plagavision [27], con el fin de mejorarlo en los siguientes aspectos:

- *Mayor alcance de radio.* El alcance teórico máximo con visibilidad entre antenas del nodo TelosB utilizado en Plagavision es de 120 m. Cabe aclarar que en Plagavision no se realizaron pruebas en campo que relevaran el alcance en un escenario real.
- *Mayor velocidad de procesamiento.* El nodo TelosB brinda una velocidad de procesamiento de 8 MHz.
- *Mayor capacidad de almacenamiento en nodos.* El nodo TelosB provee de una memoria RAM de tamaño 10 kB.
- *Imágenes de mayor resolución que permitan a los especialistas identificar rápidamente a los insectos de interés.* La cámara utilizada en Plagavision permite una resolución máxima de 640x480 pixeles.
- *Trampa tipo Wing de menores dimensiones y aprobada por la cooperativa agraria JUMECAL (contraparte del proyecto).*

Adicionalmente, *WSNvision* implementa la red de sensores inalámbricos con su propio protocolo de transmisión de información.

La topología de la red es lineal, es decir, un nodo determinado sólo intercambiará datos con su antecesor y predecesor, con ruteo estático. Un diagrama funcional del sistema se

muestra en la figura 1.5. La red se instalará en cultivos frutales de la cooperatica JUMECAL, ubicados en la cuenca del Río Santa Lucía.

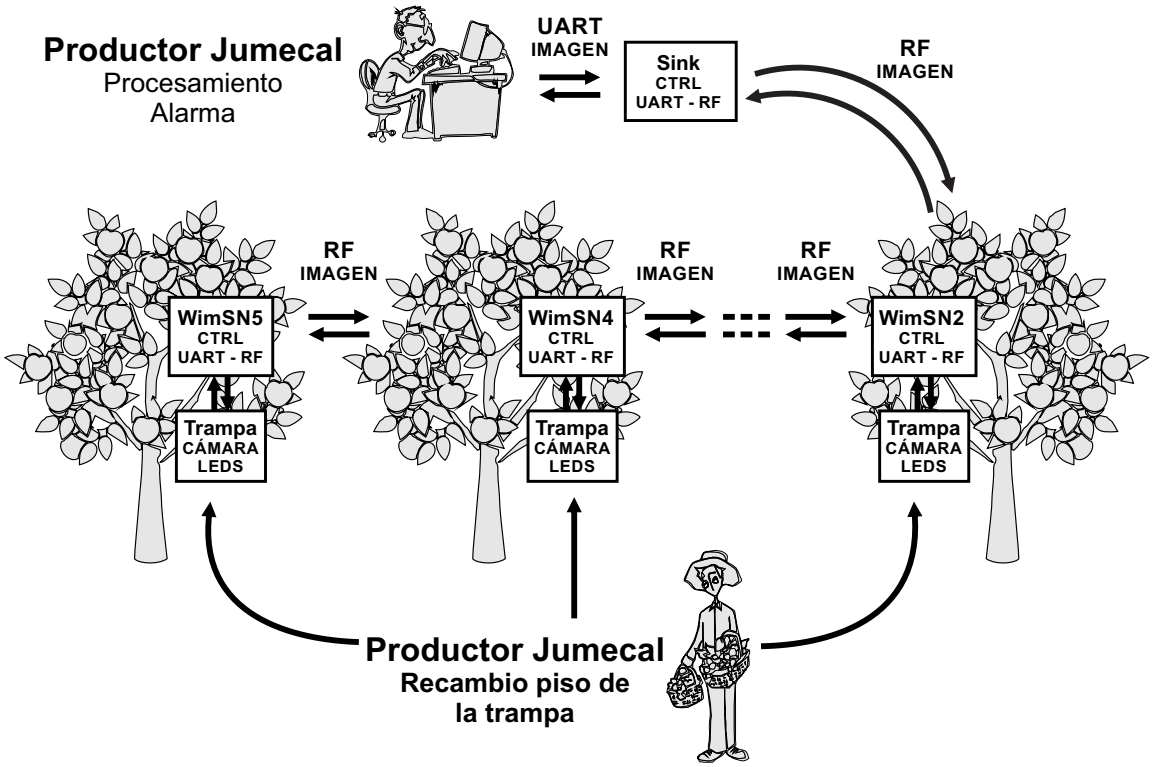


Figura 1.5: Diagrama funcional del sistema implementado.

La red está compuesta por cinco nodos, pudiendo ser extendida a la cantidad de nodos deseada de forma sencilla y transparente. Cuatro de estos nodos (llamados *WimSN* a partir de aquí) están embebidos con su cámara de fotos y su sistema de iluminación en trampas colgadas de los árboles. El nodo restante, al que se lo denomina *sink* en este documento, se encuentra conectado a un PC vía UART, y es el encargado de solicitar el envío de imágenes a los nodos *WimSN* con la periodicidad y horario que el usuario desee.

Los cinco nodos se comunican entre sí por RF, mientras que la comunicación entre los nodos *WimSN* y su respectiva cámara se realiza vía UART. El nodo *sink* se comunica con el PC vía UART. Cada nodo *WimSN* es alimentado por tres baterías AA, mientras que el *sink* se alimenta por USB desde el PC. El PC almacena las imágenes de los pisos de las trampas con la información del nodo emisor y fecha y hora de recepción, generando así un banco de imágenes. Además, cada imagen recibida es enviada por correo electrónico a direcciones configuradas por el programador.

El sistema planteado permite evaluar la evolución diaria de insectos, generar alertas tempranas y centralizar la información de manera automática y libre de errores humanos. A través del banco de imágenes es posible construir mapas de las poblaciones de plagas, identificando los predios y cultivos más sensibles a los ataques y valorando qué técnicas de control pueden ser más efectivas. Todo esto contribuye al uso eficiente de pesticidas y por

lo tanto una menor contaminación de la cuenca del río Santa Lucía.

WSNvision se desarrolla en el marco del proyecto FPTA-INIA 313 "Gervasio: Generalización de las redes de sensores inalámbricos como herramienta de valorización en sistemas vegetales intensivos", presentado por el grupo de Microelectrónica el cual comenzó a ejecutarse en junio de 2014.

1.4. Objetivo General

El objetivo general del proyecto es el diseño e implementación de una red de sensores inalámbricos con capacidad visual que permita la transmisión de grandes volúmenes de datos, basándose en el prototipo de nodo implementado en el Proyecto de Fin de Carrera Plagavision. La red estará compuesta de 5 nodos, cuatro de ellos distribuidos en campo (*WimSN*) y un nodo conectado al PC vía UART (*sink*). Los nodos *WimSN* estarán embebidos en trampas similares a las tipo Wing, y conectados a una cámara de fotos y LEDS de iluminación.

1.5. Actores del Proyecto

Los actores involucrados en el proyecto son:

- Juventud Melilla Cooperativa Agraria [6] e INIA [5] contraparte del proyecto de investigación.
- Dr. Ing. Leonardo Barboni: Tutor del Proyecto
- Ing. José Vila: Diseño de trampas, mecánica necesaria en el proyecto y ajuste de lentes de cámaras.
- Estudiantes Florencia Arbío, Federico López, Miguel Pereyra: equipo de *WSNvision*.
- Grupo de Microelectrónica del IIE-FING-UdelaR.

1.6. Alcance

- Se estudiará el prototipo de nodo desarrollado en *Plagavision*, implementando mejoras en cuanto a capacidad de procesamiento y calidad de imagen.
- Implementación de una red de 5 nodos operativa en laboratorio y campo para el procesamiento y transmisión de imágenes con posibilidad de conteo automático.
- Se implementará el protocolo de transmisión de información por RF entre nodos, teniéndose una topología de red lineal.
- Se realizará la transmisión de datos al PC.

1.7. Restricciones

- El sistema operativo a utilizar será Contiki OS.
- Los nodos *WimSN* deberán ser alimentados por baterías AA, siendo sencilla la adquisición de las mismas al momento de reemplazo.
- Los nodos a utilizar contendrán el SoC cc2538 que incluye el procesador ARM Cortex-M3 .
- La cámara de fotos será LS-Y201 de 2.0 MP, cuya alimentación es de 3 Vdc. Esta cámara devuelve las imágenes ya comprimidas en formato JPEG.
- No se realizará procesamiento de imágenes a menos que el grupo de Procesamiento de Imágenes del IIE brinde un algoritmo determinado para implementar.

1.8. Criterios de éxito

- Implementación de la red de 5 nodos en laboratorio, tomando imágenes al menos dos veces al día y enviándolas al *sink* por RF sin pérdidas.
- Implementación de la red en campo pudiendo transmitir una imagen sin pérdidas a través de los cinco nodos.
- Lograr una cobertura de un nodo *WimSN* por hectárea. Para esto, se impone una distancia mínima entre dos nodos de al menos 100 m en campo.
- Obtener imágenes del piso de la trampa que sean validadas por los entomólogos de JUMECAL.

1.9. Diagrama de bloques del sistema

En la figura 1.6 se muestra el diagrama de bloques del sistema, indicando cuáles son los componentes de los dos tipos de nodo utilizados para la implementación de la red.

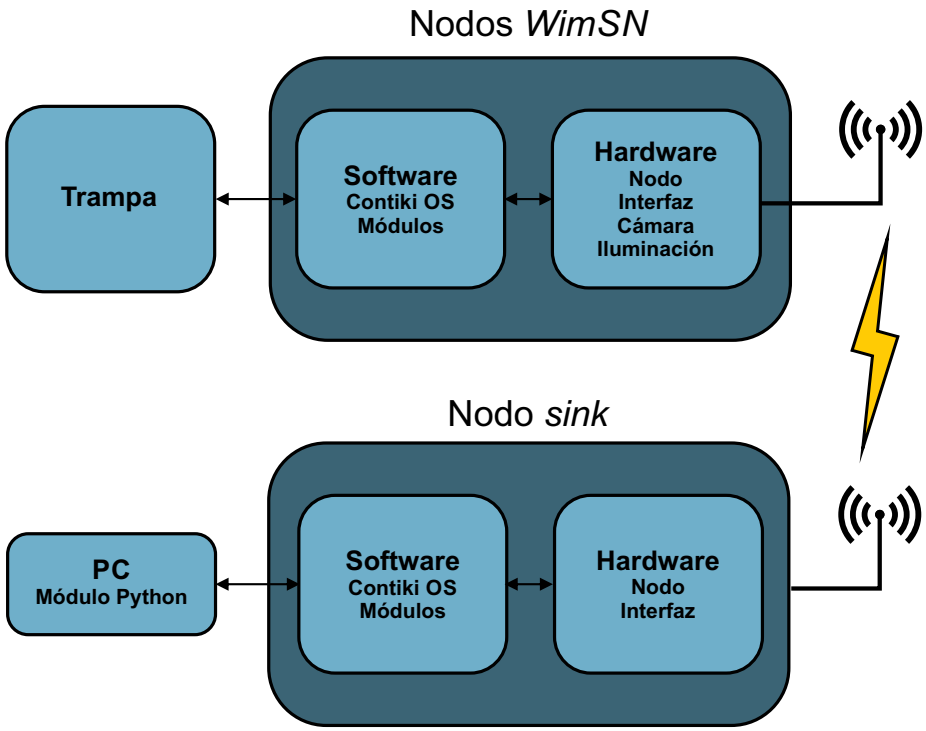


Figura 1.6: Diagrama de bloques del sistema

1.10. Descripción de Capítulos

Capítulo 1 - Introducción

Este capítulo consiste en una introducción al contenido del proyecto. Se describe el problema que motiva la realización de *WSNvision* y el objetivo general del mismo. Se presenta un diagrama funcional del sistema propuesto, junto con su descripción. Se presentan los diferentes actores involucrados, así como las restricciones, alcance y criterios de éxito que se definieron junto con el tutor para enmarcar el trabajo realizado. Se muestra un diagrama de bloques del sistema y se realiza un breve resumen del contenido de los capítulos de este documento.

Capítulo 2 - Hardware

En este capítulo se describe el hardware utilizado en el proyecto. Se presenta el SoC CC2538 y las plataformas CC2538EM y CC2538-CC2592EM que lo utilizan. Se detallan

las características de la cámara y los lentes utilizados por los nodos *WimSN*. Se describe el sistema de alimentación y la interfaz diseñada para la integración de los diferentes componentes de los nodos *WimSN*. Se presenta el diseño de placas para caracterización de consumo de carga de los nodos.

El hardware se compone por: i) Cuatro nodos *WimSN* (CC2538-CC2592EM), ii) Un nodo *sink* (CC2538EM), iii) Cámara fotográfica (Linksprite LS-Y201-2MP, con lente LS-20150), iv) Iluminación (LEDs blancos de alto brillo), v) Interfaz (Nodo-Cámara-Alimentación), vi) Cable USB-TTL 3 Vdc (interfaz *sink*-PC).

Capítulo 3 - Estudio del sistema visual: geometría de la trampa, iluminación e imágenes.

Este capítulo presenta las pruebas realizadas y cómo se logró obtener la configuración que asegura la identificación eficaz de las polillas. Ésta consiste en: lente (LS-20150), 4 LEDs (colocados en los extremos del techo de la trampa, apuntando hacia arriba de la trampa, blancos, de alta luminosidad), altura cámara - piso = 20,5 cm y piso de la trampa inclinado 20^o respecto a la horizontal . A través de este estudio, se desencadenó la aprobación del modelo de trampa por parte de la cooperativa JUMECAL. Sin embargo, no fue posible optimizar el sistema de iluminación ya que al momento de realizar los ensayos no se contaba con la trampa final del sistema. Por esto el capítulo concluye presentando propuestas a futuras mejoras.

Capítulo 4 - Diseño de red y protocolo de comunicación

En este capítulo se presenta el proceso de diseño del protocolo de comunicaciones propuesto. Se realiza una breve introducción a las redes de sensores inalámbricos destacando los requerimientos particulares que tiene que cumplir el sistema para la aplicación en estudio. Se analiza la implementación del *stack* de red en Contiki OS y se evalúa qué protocolos pueden ser reutilizados en *WNSvision*. Finalmente se presenta la implementación de un *stack* propio, y se detalla la topología de la red y el protocolo de nivel de aplicación implementado.

Capítulo 5 - Software

Este capítulo consta de tres secciones. En la primer sección se exponen conceptos básicos del sistema operativo Contiki OS. Luego se presenta el diseño de los módulos de software para el manejo de los nodos de la red y por último la implementación de estos módulos. Dentro de la tercer sección se especifica el procedimiento para la extensión del número de nodos de la red.

Capítulo 6 - Caracterización y ensayos

En este capítulo se presentan los ensayos realizados a fin de caracterizar el alcance de la radio de los nodos *WimSN* y *sink*, estableciendo si se logra un área de cobertura de un nodo *WimSN* por hectárea. Adicionalmente, se presentan las pruebas realizadas al protocolo de transmisión de información en campo de cultivos frutales.

Se tuvo que con visibilidad entre antenas se puede llegar a distancias superiores a los 250 m entre nodos. La situación cambia si las antenas no tienen visibilidad, llegando a pérdidas mayores al 10 % a 60 m de distancia. Se logró recibir imágenes íntegras a través de la red distribuida en campo.

Capítulo 7 - Conclusiones y trabajo futuro

Este capítulo finaliza la documentación aportando conclusiones y sugerencias para aportar mejoras al sistema. Estas sugerencias serán tomadas por otro grupo de PFC que continuará con el desarrollo en una etapa 3. Esta etapa consiste en: aumentar a diez la cantidad de nodos de la red, optimizar el funcionamiento del sistema para disminuir consumo de carga de los nodos e implementar el conteo automático de polillas en nodos *WimSN*.

Capítulo 2

Hardware

2.1. Resumen

En este capítulo se describe el hardware utilizado en el proyecto. Se presenta el SoC CC2538 y las plataformas CC2538EM y CC2538-CC2592EM que lo utilizan. Se detallan las características de la cámara y los lentes utilizados por los nodos *WimSN*. Se describe el sistema de alimentación y la interfaz diseñada para la integración de los diferentes componentes de los nodos *WimSN*. Se presenta el diseño de placas para caracterización de consumo de carga de los nodos.

El hardware se compone por: i) Cuatro nodos *WimSN* (CC2538-CC2592EM), ii) Un nodo *sink* (CC2538EM), iii) Cámara fotográfica (Linksprite LS-Y201-2MP, con lente LS-20150), iv) Iluminación (LEDs blancos de alto brillo), v) Interfaz (Nodo-Cámara-Alimentación), vi) Cable USB-TTL 3 Vdc (interfaz *sink*-PC).

2.2. SoC CC2538

Una de las restricciones del proyecto, es la utilización de plataformas que utilicen el SoC CC2538. A continuación se presentan las características de este SoC de relevancia para el proyecto.

El CC2538 [35,36] es un SoC de Texas Instruments con un microcontrolador Cortex-M3 de 32 bits con un radio transciever integrado. El microcontrolador presenta una frecuencia de funcionamiento de 32 MHz e incluye 32 kB de memoria SRAM y 512 kB de memoria flash, así como los periféricos habituales (GPIO, ADC, Timers, etc.). La radio funciona en la banda de 2,4 GHz y es completamente compatible con el estándar IEEE 802.15.4 [4]. Esto le permite manejar *stacks* de red complejos con seguridad y aplicaciones exigentes. Posee 32 GPIOs que permiten conexiones de manera simple con distintos periféricos. Se tienen 4 modos de bajo consumo (PM0, PM1, PM2 y PM3) que permiten el arranque rápido del modo *sleep*. La figura 2.1 muestra un diagrama de bloques del CC2538.

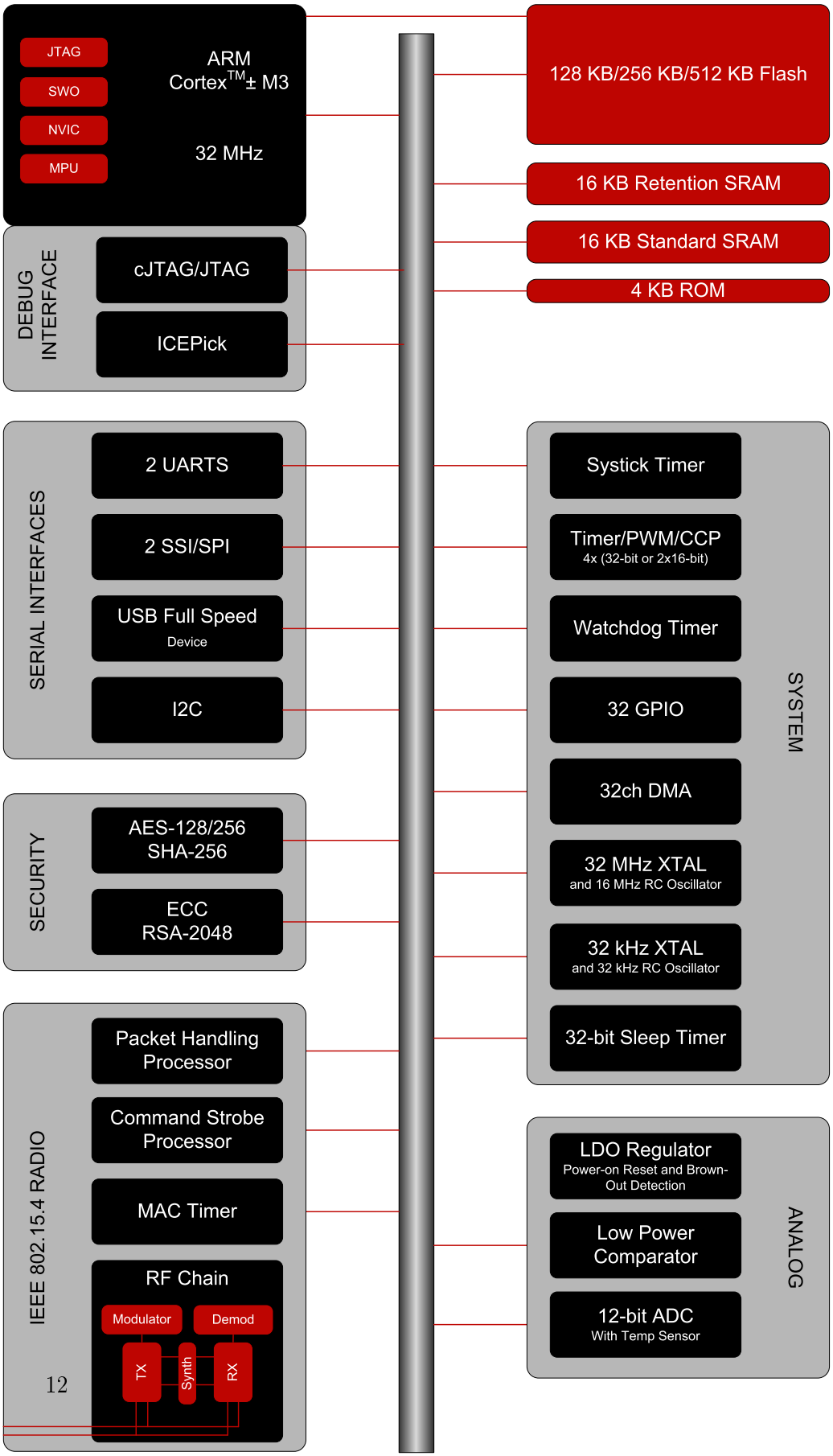


Figura 2.1: SoC cc2538, basado en cortex M3, tomada de [36]

Como se mencionó anteriormente, este SoC posee 32 kB de SRAM divididos en dos bloques de 16 kB. Cuando se ingresa en los modos de bajo consumo quedan útiles 16 kB de RAM con retención. La radio tiene una sensibilidad en recepción típica de -97 dBm y una potencia de transmisión de 7 dBm.

A continuación se detallan las características del cc2538 que resultan de mayor importancia para la aplicación en este proyecto.

Memoria Flash: Este SoC cuenta con 512 kB de memoria flash estructurada en 256 páginas de tamaño 2 kB. Cada página cuenta con un bit de bloqueo que permite impedir la escritura y borrado de la misma. Estos bits se encuentran guardados en la página 255 por lo que es importante no sobrescribir dicha zona. Las escrituras se realizan de a 4 bytes y demoran un mínimo de 20 us. La memoria flash soporta 20000 ciclos de borrado. Cada ciclo de borrado consiste en setear los bits de la sección de interés en '1' lógico, siendo la mínima unidad de borrado una página. El tiempo mínimo en ejecutar esta acción es de 20 ms para una página y 1 s si se trata de la totalidad de la memoria.

GPIO: Cuenta con 32 GPIO distribuidos en 4 bloques físicos (Puerto A, B, C, D). Cuatro de estos GPIO pueden manejar hasta 20 mA y los 28 restantes hasta 4 mA. El voltaje máximo que soportan es de 3.9 Vdc (Vdd + 0.3 V, con Vdd la tensión de alimentación del SoC). En caso de ser usados como puertos de entrada o salida digitales, una tensión superior a 2.5 Vdc se detecta como un '1' lógico, mientras que si la tensión es inferior a 0.5 Vdc se interpretan como un '0' lógico.

UART: Se dispone de dos UARTs: UART_0 y UART_1, con *baudrate* programable hasta 4 Mbps y *buffers* de transmisión y recepción FIFO's de 16x8 bits, pudiendo seleccionar un modo de 9 bits para configuración multipunto. Las UART generan interrupciones enmascaradas al microprocesador. La diferencia entra ambas es que la UART_1 tiene soporte *full handshake* y control de flujo. Los niveles de tensión no deben superar Vdd + 0.3 V, siendo Vdd la tensión de alimentación del SoC. En este proyecto se utiliza la UART_0 por no requerirse de las funcionalidades de UART_1.

Modos de bajo consumo: Se puede operar en modo activo o modo *sleep*. Dentro del modo *sleep* existen 4 modos de bajo consumo: PM0, PM1, PM2, PM3, ordenados de mayor a menor consumo de corriente. Es importante tener en cuenta que Contiki OS no permite el ingreso al modo PM3. Los consumos típicos del SoC declarados por el fabricante en los distintos modos son los siguientes:

- PM0- 24 mA
- PM1- 0.6 mA
- PM2- 1.3 μA
- PM3- 0.4 μA

A continuación se presentan las plataformas CC2538EM y CC2538-CC2592EM utilizadas en los nodos *sink* y *WimSN* respectivamente. Vale aclarar que el nodo *sink* usa una

plataforma diferente de los nodos *WimSN* por no disponer de cinco plataformas CC2538-CC2592EM durante el desarrollo del proyecto, y no por motivos de diseño.

2.2.1. Plataforma CC2538EM

La plataforma CC2538EM [34] es la utilizada en el nodo *sink*. Se trata de un módulo de evaluación del SoC CC2538 de Texas Instruments (TI). Los módulos de evaluación se pueden utilizar para verificación de prototipos de hardware y del rendimiento de la plataforma.

A continuación se enumeran las principales características de la plataforma.

- CC2538: SoC de Texas Instruments.
- TPS76933: Se trata de un conversor DC/DC de Texas Instruments con corriente en estado inactivo de 360 nA.
- ABM8G: Es un cristal de 32 MHz que se utiliza como reloj para el MCU y el radio transceiver.
- ABS07: Es un cristal de 32,768 kHz que se utiliza para el reloj del MCU, RTC (Real Time Clock).
- Antena impresa (PCB) del tipo F invertida de ganancia 3 dB.
- Conector Micro USB.

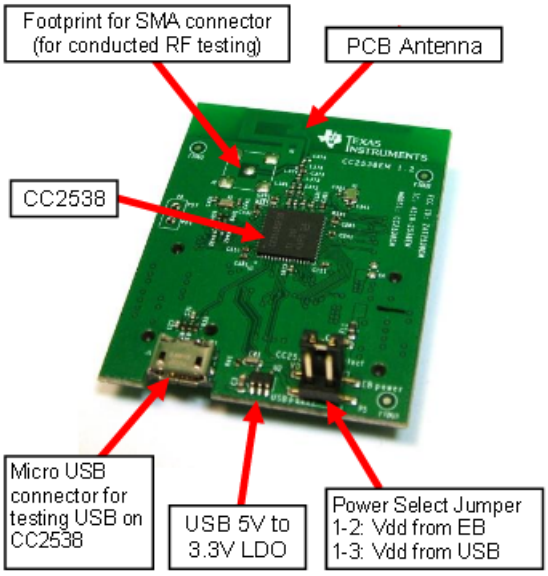


Figura 2.2: Plataforma CC2538EM, tomada de [34].

2.2.2. Plataforma CC2538-CC2592EM [38]

Es la plataforma utilizada en los nodos *WimSN*. Posee características similares al nodo CC2538EM, aumentando la potencia de transmisión de la radio y mejorando la sensibilidad del receptor a través del extensor de rango CC2592 [12, 39]. La hoja de datos de este integrado declara una potencia de transmisión Tx máxima de 22 dBm. El receptor mejora la sensibilidad en 3dB. Se recomienda mantener una distancia mínima de 20 cm entre el usuario y la antena debido a la radiación de esta última.

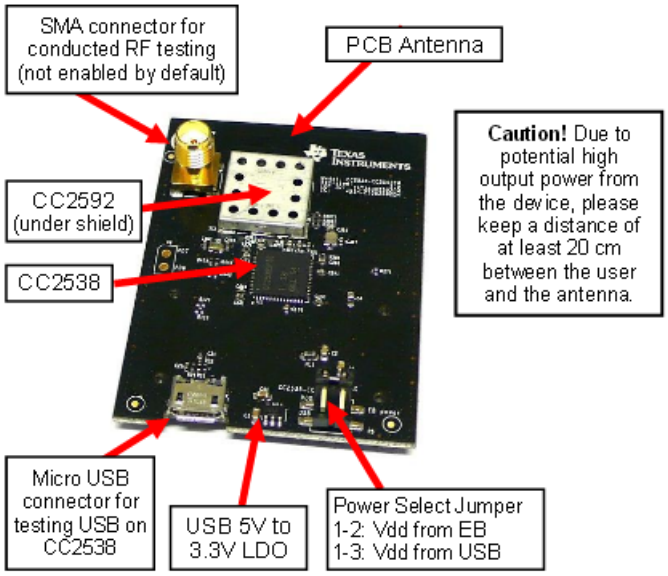


Figura 2.3: Plataforma CC2538-CC2592EM, tomada de [38].

A continuación se enumeran sus principales características:

- CC2538: SoC de Texas Instruments.
- TLV70233: Conversor DC/DC de Texas Instruments de mayor potencia que el TPS76933.
- Un cristal de 32 MHz que se utiliza como reloj para el MCU y radio transceiver.
- Un cristal de 32,768 kHz que se utiliza para el reloj del MCU, RTC (Real Time Clock).
- Antena impresa (PCB) tipo F invertida de ganancia 3 dB.
- Conector para Antena. Permite conectar una antena externa (en este proyecto se utiliza una antena de ganancia 5 dB).
- Conector Micro USB.
- CC2592 PA/LNA RF: Amplifica la salida del transmisor de RF y aumenta la sensibilidad de recepción.

Esta plataforma tiene integrado un conector SMA para utilizar una antena externa, además de traer una antena impresa del tipo $F - invertida$. Sólo es posible tener habilitada una antena por vez. Por defecto tiene habilitada la antena impresa. El cambio de antena requiere de una modificación simple en el hardware de la plataforma. Esto se especifica en el apéndice D.

2.3. Cámara

La cámara seleccionada para el proyecto es LinkSprite LS−Y201−2MP [30]. Se trata de un dispositivo de bajo costo (U$S 49) y pequeñas dimensiones, con compresión JPEG [16].

(a) (b)

Figura 2.4: *(a)cámara LS−Y201−2MP(Top) (b) cámara LS−Y201−2MP(Bottom)*

Sus características principales son:

- Alimentación: 5 Vdc.
- Dimensiones: 32 mm x 32 mm.
- Soporte para lentes M12xP0.5.
- Sensor CMOS OV2643
- Resoluciones: 160x120 pixeles, 320x240 pixeles, 640x480 pixeles, 800x600 pixeles, 1024x768 pixeles, 1280x960 pixeles, 1600x1200 pixeles.
- Factor de compresión configurable entre 00h y FFh.
- Velocidad máxima de fotogramas 15 fps.
- Baudrate configurable en 9600 bps, 38400 bps, 57600 bps, 115200 bps, 128000 bps o 256000 bps.
- Comunicación UART.

El fabricante declara un valor de alimentación de 5 Vdc, pero experimentalmente se observa que la cámara funciona con tensiones a partir de los 2.6 Vdc. En el PFC Plagavision se hizo un estudio comparativo entre diferentes cámaras, concluyendo que la más adecuada era Linksprite $LS-Y201$ de 0.3 MP. Siguiendo los mismos criterios y con el objetivo de mejorar la calidad[1] de imagen se optó por utilizar una versión más reciente, con mayor resolución.

$LS-Y201-2MP$ tiene como ventaja que el procesamiento JPEG y el almacenamiento de la imagen se realiza directamente en la cámara disponiendo de un *chip* de procesamiento y una memoria para estos propósitos. El hecho de que el formato de salida sea JPEG evita tener que transmitir la imagen en formato crudo[2] para luego procesarla en el nodo. La comunicación UART facilita altas tasas de transferencia de datos de hasta 256000 bps. Otra ventaja importante de esta cámara es que no necesita de hardware adicional para conectarla al nodo.

Durante la utilización de la cámara, se detectó un defecto de fabricación en el soporte donde se enrosca el lente. El sensor CMOS no se encuentra centrado respecto del soporte, y por ende tampoco respecto del lente, lo que deriva en problemas para tomar imágenes completamente centradas respecto al piso de la trampa. Únicamente se corrigió este defecto en una de las cámaras para probar su funcionamiento, en el apéndice B se detallan las modificaciones necesarias para repetir el procedimiento.

2.4. Lentes

En el capítulo 3 se presenta el proceso de selección del lente, optando entre los siguientes modelos:

- Hx-1820, distancia focal de 1.8 mm.
- LS-40166, distancia focal de 2.6 mm.
- LS-20150, distancia focal de 2.8 mm.
- Lente CL4022IR, distancia focal de 3.6mm.

El lente elegido es LS-20150. En la figura 2.5 se observa su hoja de datos.

[1] Mejorar la calidad de imagen se refiere a lograr una imagen que permita distinguir con mayor detalle los insectos que se encuentren en la trampa. Para ello es necesario mejorar iluminación, resolución y compresión.

[2] Formato crudo implica tener el formato de la imagen en el dominio del pixel, formato matricial.

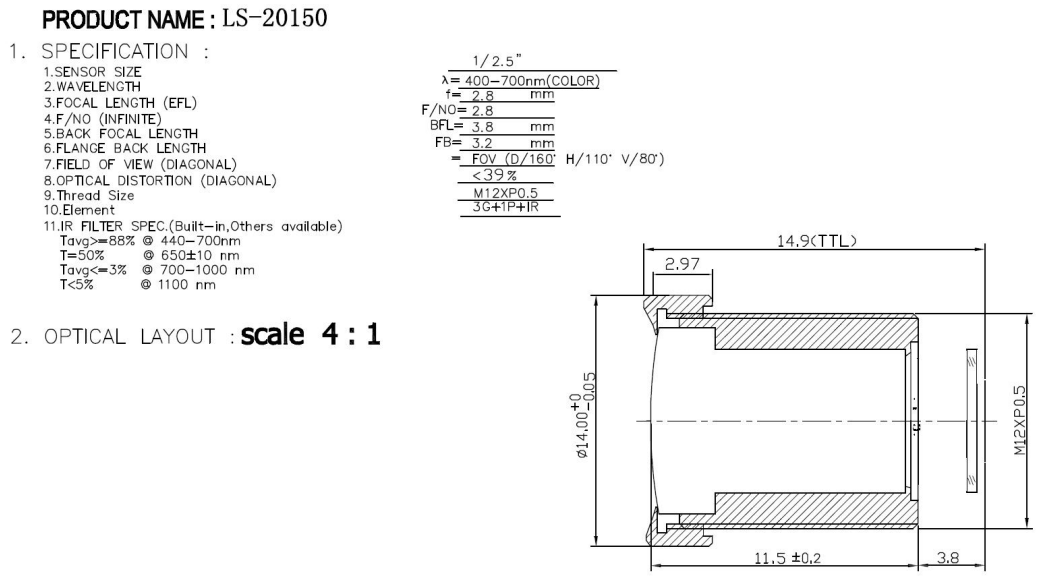


Figura 2.5: *Características del lente LS-20150*, tomada de [7].

2.5. Interfaz Nodo-Cámara-Alimentación

Se diseña e implementa una interfaz que permita la interconexión entre la cámara LS-Y201-MP, el nodo *WimSN*, los LEDS de iluminación y la alimentación de todos los componentes. Un diagrama de bloques se presenta en la figura 2.6.

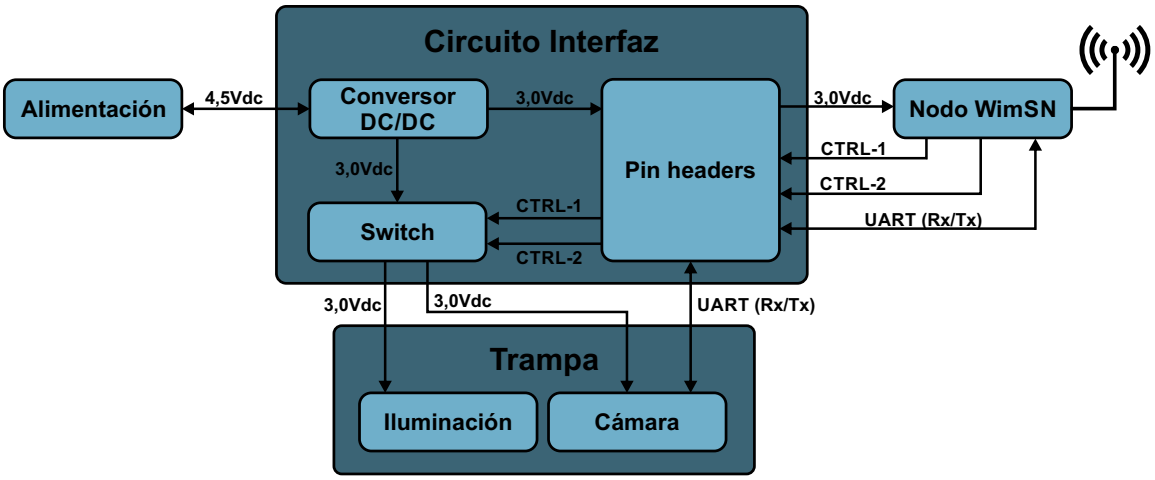


Figura 2.6: Diagrama de bloques del sistema Nodo-PC-Cámara-Alimentación.

Una de las restricciones es el uso de baterías AA por ser sencilla su adquisición al momento del reemplazo. Experimentalmente se tiene que el sistema cámara−nodo−iluminación,

funciona correctamente con una tensión de alimentación mínima de 2.6 Vdc. Una primera opción es alimentar el sistema con 2 baterías AA en serie, teniéndose una tensión de alimentación de 3 Vdc.

La ecuación 2.1 refiere a la corriente máxima que puede consumir un nodo *WimSN* dependiendo del estado en que se encuentra (inicialización de la red, toma de imagen, transmisión de imagen por UART y almacenamiento en flash, transmisión de imagen por RF y modo de bajo consumo). I_{Cam} es la corriente de la cámara declarada por el fabricante, I_{Ilum} es la corriente del sistema de iluminación estimado en 40mA e I_{Nodo} es la corriente del nodo CC2538-CC2592EM declarada por el fabricante en los diferentes estados (modo de bajo consumo, estado de la radio, almacenamiento en flash, etc).

De la ecuación 2.1 y la tabla 2.1 se tiene que el consumo total del sistema por momentos podría sobrepasar los 200mA. Esto, junto al desgaste natural de las baterías, puede ocasionar que el voltaje de alimentación caiga por debajo del nivel mínimo que garantiza el funcionamiento del sistema. Para minimizar este efecto, se opta por alimentar el sistema con 3 baterías AA en serie (con una tensión de salida de 4.5 Vdc). Para asegurar una tensión de alimentación de los dispositivos de 3 Vdc, se utiliza un conversor DC/DC. De esta manera cada pila podría bajar su voltaje hasta 1.02 Vdc sin afectar al sistema.

$$I_{max} = I_{Cam} + I_{Ilum} + I_{Nodo} \tag{2.1}$$

Datos de consumo de corriente extraídos de las hojas de datos:

- **CC2592:** 155 mA con P_{Tx} =22 dBm, 4 mA en Rx (*high gain*) y 100 nA en modo inactivo.
- **CC2538-CC2592EM:** 34 mA con P_{Tx} =7 dBm, 0,6 mA en modo PM1.
- **Cámara LS-Y201-2MP:** 120 mA.
- **Leds:** se estima en 40 mA.

Estado sistema	Imax (mA)	Tiempo(s)
Instante inicial (luego de un reset)	198	1
Inicialización de la red	189	4
Nodo durmiendo	0,6	85698
Tomar foto	194	3
Transmisión foto UART	158	14
Transmisión foto RF	189	680

Tabla 2.1: Peso estimado de los distintos estados del sistema en tiempo de ejecución.

La tabla 2.1 presenta los valores de Imax junto con los tiempos de permanencia en cada estado del sistema, para el nodo de ID 2 transmitiéndose imágenes de 70 kB por la red de 5 nodos sin pérdidas. Teniendo en cuenta estas estimaciones se puede ver gráficamente como influye cada proceso en el consumo del sistema como lo muestra la figura 2.7.

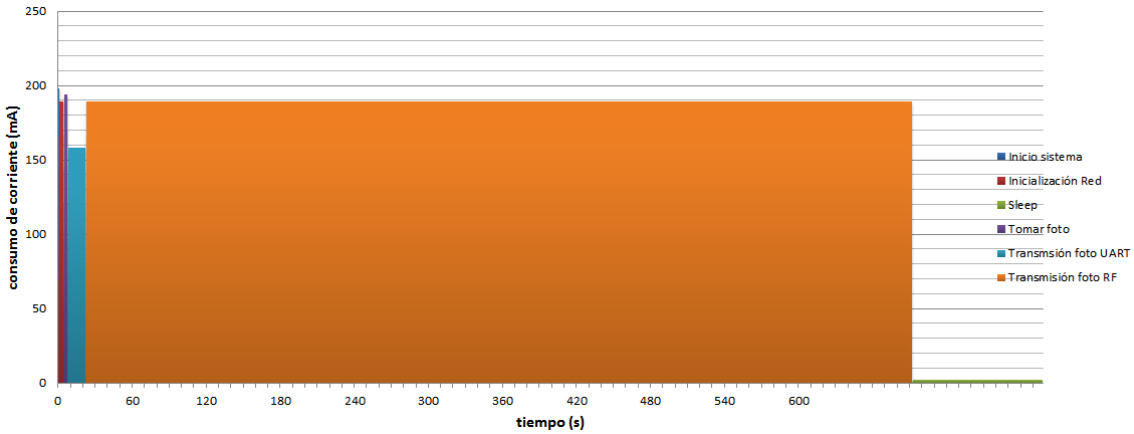


Figura 2.7: Gráfico ilustrativo del peso de cada proceso por ciclo de trabajo.

El Conversor DC/DC elegido es el TPS62740 de TI. Es un conversor de alta eficiencia (90 % con I_{out}=10μA)y sus principales características son:

- Bajo consumo en estado inactivo (360 nA).
- Corriente máxima de salida de 300 mA.
- Tensión de alimentación máxima 5.5 Vdc.
- Tensión de salida regulable de 1.8 a 3.3 Vdc configurable en pasos de 100 mV. Para esto cuenta con los pines de entrada (VSEL1,VSEL2,VSEL3,VSEL4).

Al habilitarse la entrada enable (EN), el dispositivo ingresa en modo *soft start* (arranque suave) que limita la corriente de salida a 1/4 del limite de corriente nominal y se incrementa progresivamente durante un tiempo $t_{softsart} \simeq 400\mu$s. La hoja de datos asegura el buen funcionamiento del dispositivo si $V_{in} \geq V_{out} + 0.7$V, siendo V_{in} y V_{out} los voltajes de entrada y salida respectivamente.

Los requerimientos de bajo consumo de este proyecto hacen necesario que, tanto a la cámara como a la iluminación, se le deba quitar la alimentación mientras no se encuentran en uso. Esto se debe a que la cámara tiene un consumo fijo de unos 100 mA aunque no se esté adquiriendo una imagen y a que la corriente consumida por los LEDs de iluminación superan la corriente máxima que puede brindar un GPIO del nodo.

Para resolver este problema es necesario utilizar switches que permitan la conexión y desconexión de dispositivos. Los mismos deben cumplir los siguientes requerimientos:

- Corriente de salida mayor o igual a 100 mA.
- Menor Corriente de reposo (*quiescent*) posible.
- Alimentación 3 Vdc.
- Baja resistencia de encendido.
- Manejar al menos 2 salidas. independientemente.

Se propone la utilización del *switch* FPF2300, cuyo diagrama de bloques se presenta en la figura 2.8. Es un dispositivo de manejo de carga que se compone de dos transistores MOSFET con su *gate* conectado a un circuito de control que recibe una señal externa para conmutar entre sus estados, actuando como llaves. El circuito de control también incluye una realimentación de corriente y temperatura para evitar sobrecargas que puedan causar daños.

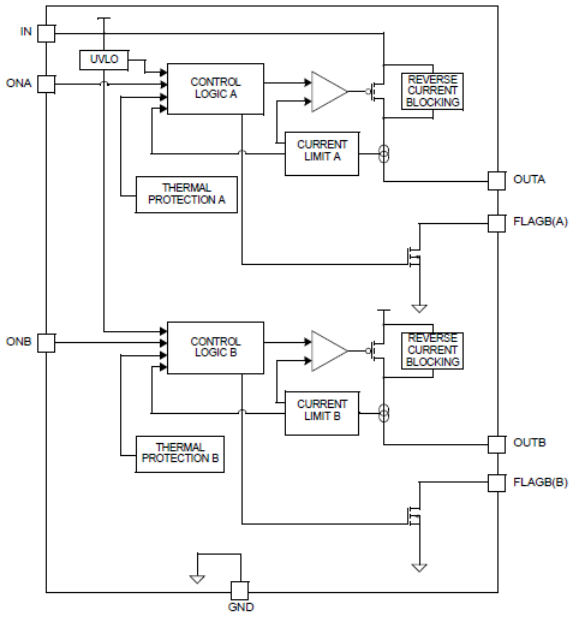


Figura 2.8: Diagrama de bloques del FPF2300, tomada de [22].

Para el diseño de circuitos se utilizó la herramienta de software EAGLE. En la figura 2.10 se presenta el circuito esquemático interfaz. Descripción de pines de la interfaz:

- Vin: Entrada para alimentación del sistema (5.5 Vdc Máx).
- CAM: Conexión de alimentación y comunicación UART de la cámara (3V,0V,Tx,Rx).
- LEDS: Conexión para iluminación (3V,3V,0V,0V).
- button_user: Conexión directa a GPIO del nodo (PA3, Vcc).
- button_reset: Conexión directa a GPIO del nodo (nRESET, Vcc).
- DI_SENSOR: Conexión directa a GPIO del nodo (PA2, Vcc, GND).
- AI_SENS: Conexión directa a GPIO del nodo (PA6, PA7).
- PAD′s: se dejan accesibles los pines (VSEL1, VSEL2).

Los pines del grupo DI_SENSOR y AI_SENS, se dejan previstos en caso de que surja la necesidad de integrar algún sensor de luz u otro tipo de utilidad que pueda surgir en trabajos futuros. De forma de poder modificar el voltaje de salida del conversor DC/DC se dejan accesibles los pines VSEL1 y VSEL2.

Una consideración importante y que aporta para bajar el consumo del sistema es que el integrado FPF2300 activa los switches por nivel bajo (contacto normalmente cerrado), siendo necesario para apagar la cámara o LEDS de iluminación un ‘1’ lógico a la salida del puerto correspondiente. Para mejorar esto se recomienda utilizar switches activos por nivel alto.

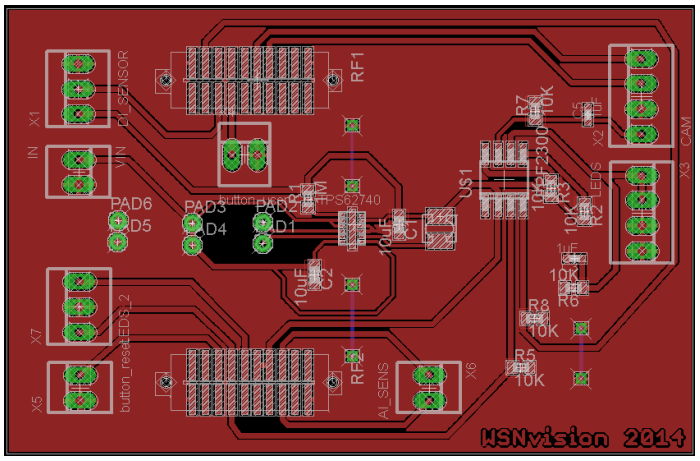


Figura 2.9: PCB de Interfaz Nodo-Cámara-Alimentación (bottom view).

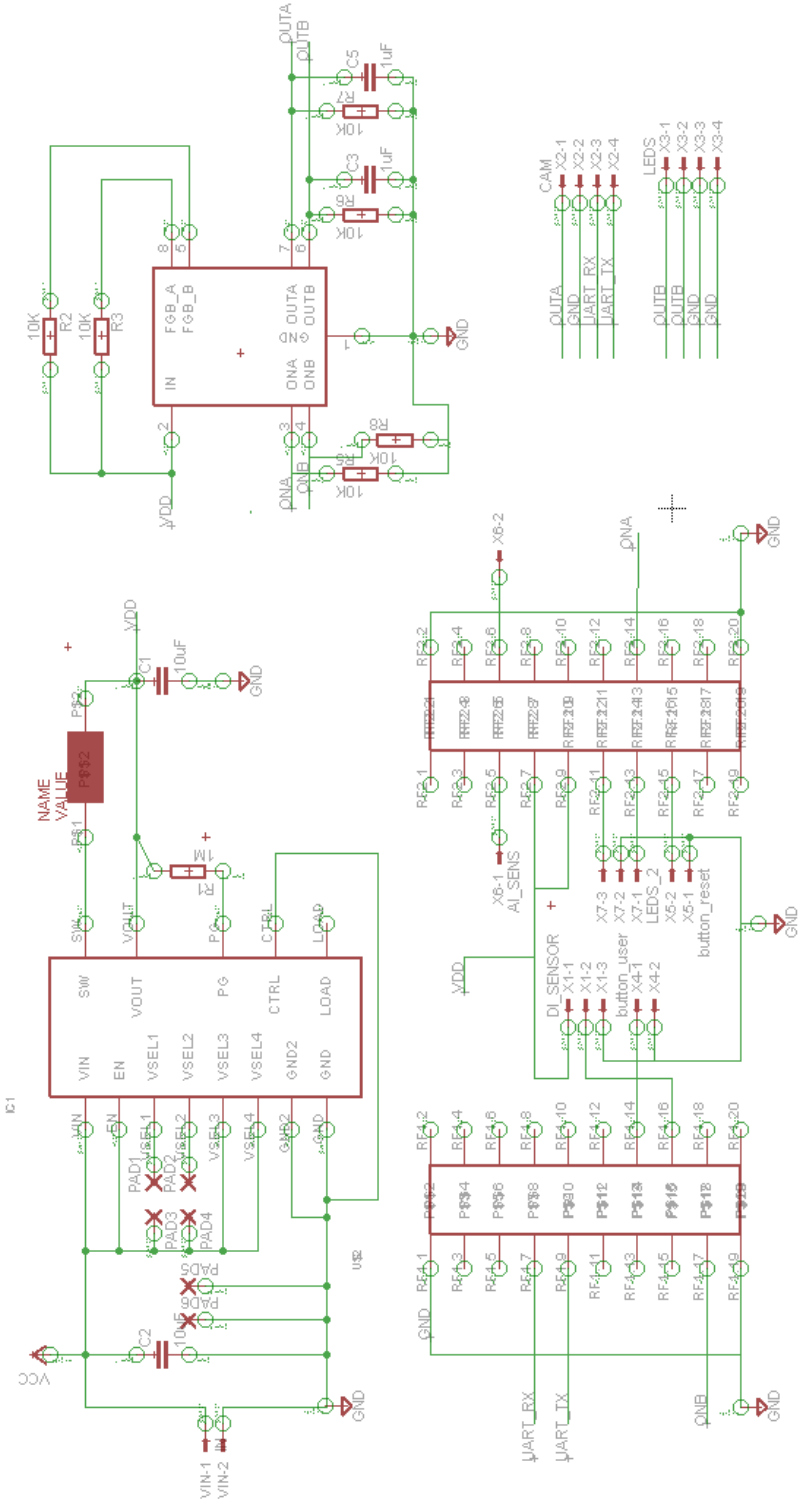
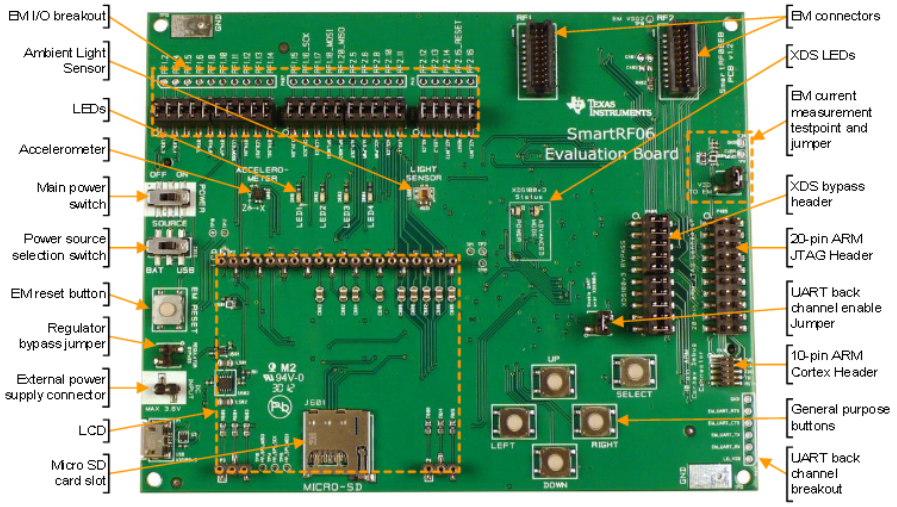


Figura 2.10: Circuito esquemático de Interfaz Nodo-Cámara-Alimentación

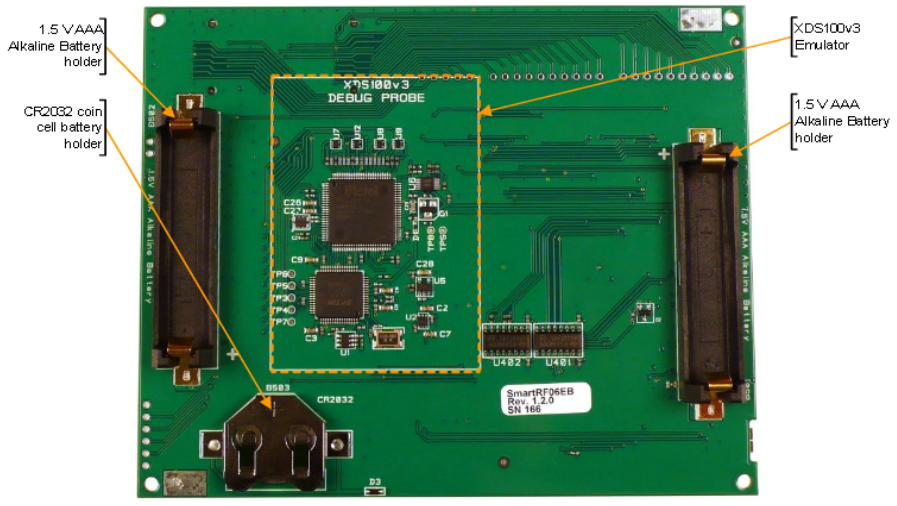
2.6. Hardware adicional

2.6.1. Placa Programadora SmartRF06EBK

SmartRF06EBK es la placa programadora para las plataformas CC2538EM y CC2538-CC2592EM, que incluye todo lo necesario para interactuar y programar el nodo. La misma cuenta con varios periféricos y pines para acceder a los GPIO del SoC CC2538, conector para J-TAG y circuito para medir consumo, entre otros. Sus características se pueden observar en la figura 2.11.



(a)

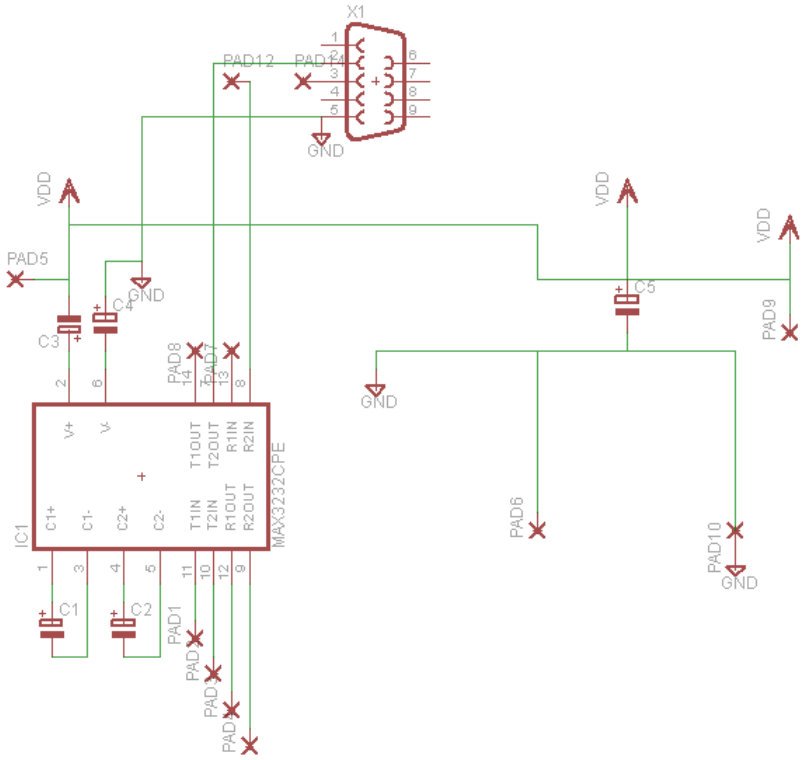


(b)

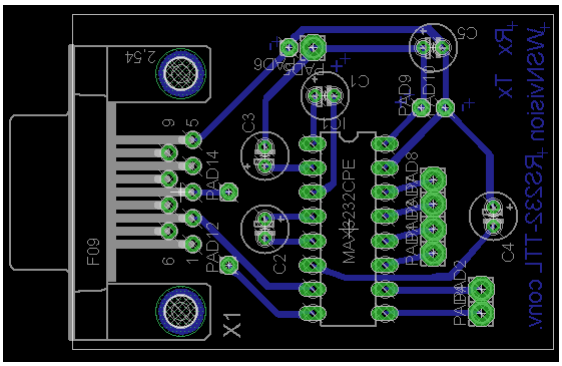
Figura 2.11: *(a)Placa programadora SmartRF06EBK (Top) (b) Placa programadora SmartRF06EBK (Bottom)*, tomadas de [37]

2.6.2. Interfaz Nodo-PC y/o Cámara-PC para depuración de códigos

Para poder comunicar la cámara o nodo con el PC, se diseñó un circuito basado en la interfaz RS-232 compatible con la UART de los nodos CC2538EM y CC2538-CC2592EM y de la cámara. El mismo se conecta al PC utilizando un conector DB9 o por medio de un cable conversor RS-232/USB [21] y es utilizado para depuración de códigos y verificación de comandos de la cámara. El circuito implementado utiliza el integrado MAX232 y se muestra en la figura 2.12.



(a)



(b)

Figura 2.12: *(a)Esquemático, Interfaz PC-Cámara / PC-Nodo (b) PCB,Interfaz PC-Cámara / PC-Nodo (bottom view)*

Obs: Se puede prescindir de esta interfaz si se tiene un cable conversor de USB-TTL en 3Vdc.

2.6.3. Placa medidora de consumo

Se diseñó un PCB compuesto por dos circuitos independientes, uno para medir consumo de carga en el tiempo y otro para poder visualizar en detalle perfiles de corriente mediante un osciloscopio. Estos diseños serán utilizados por futuros Proyectos de Fin de Carrera (PFC) para caracterizar el consumo de carga del sistema en un escenario real.

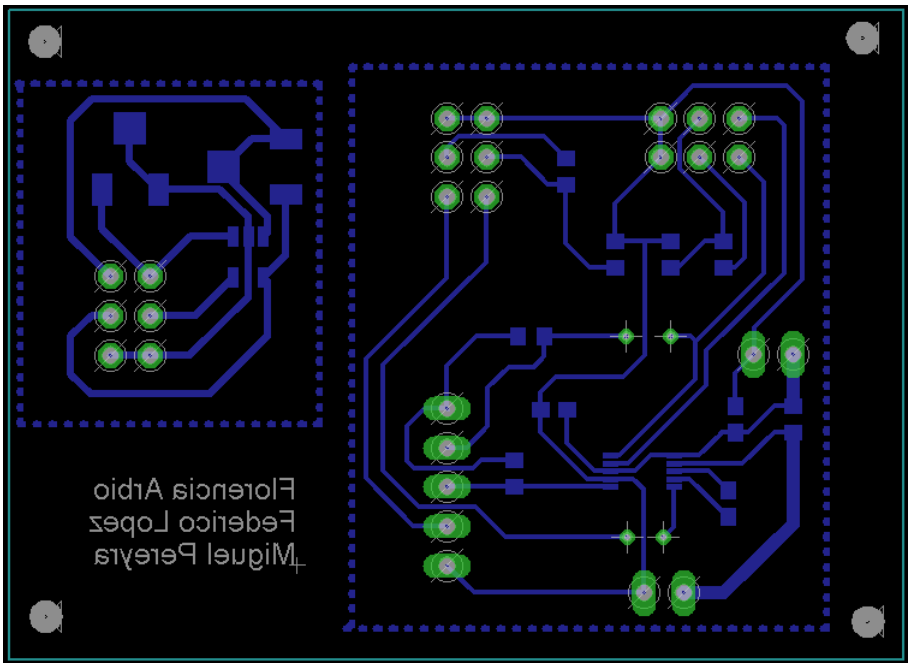
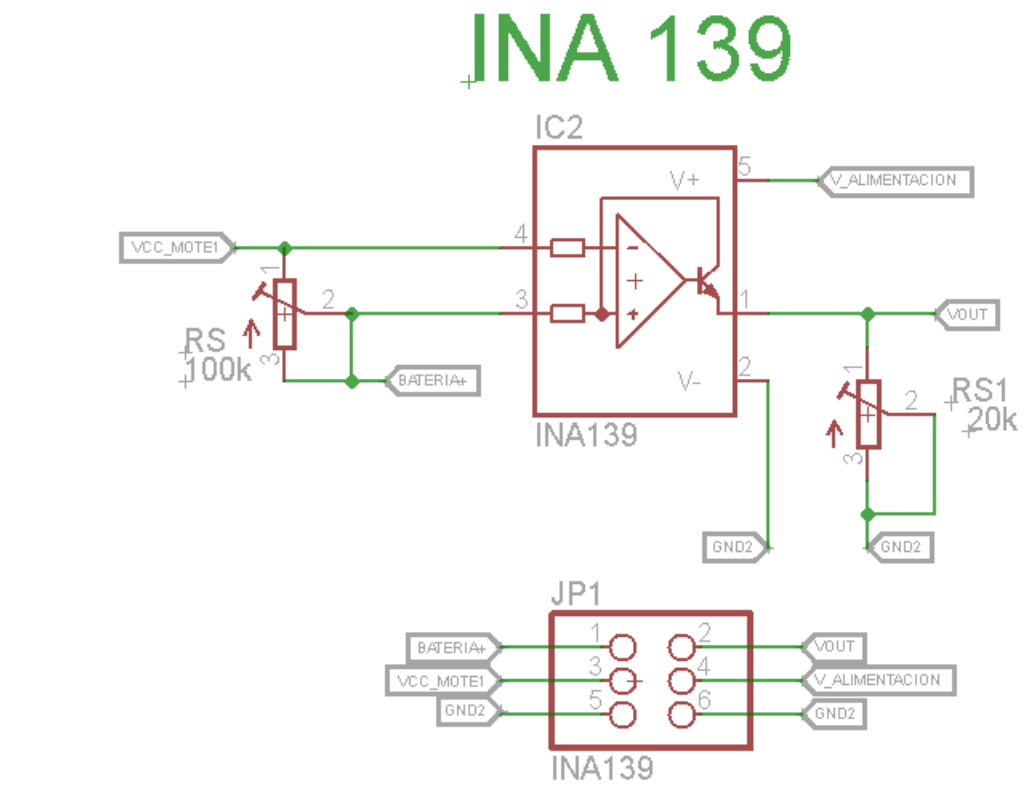
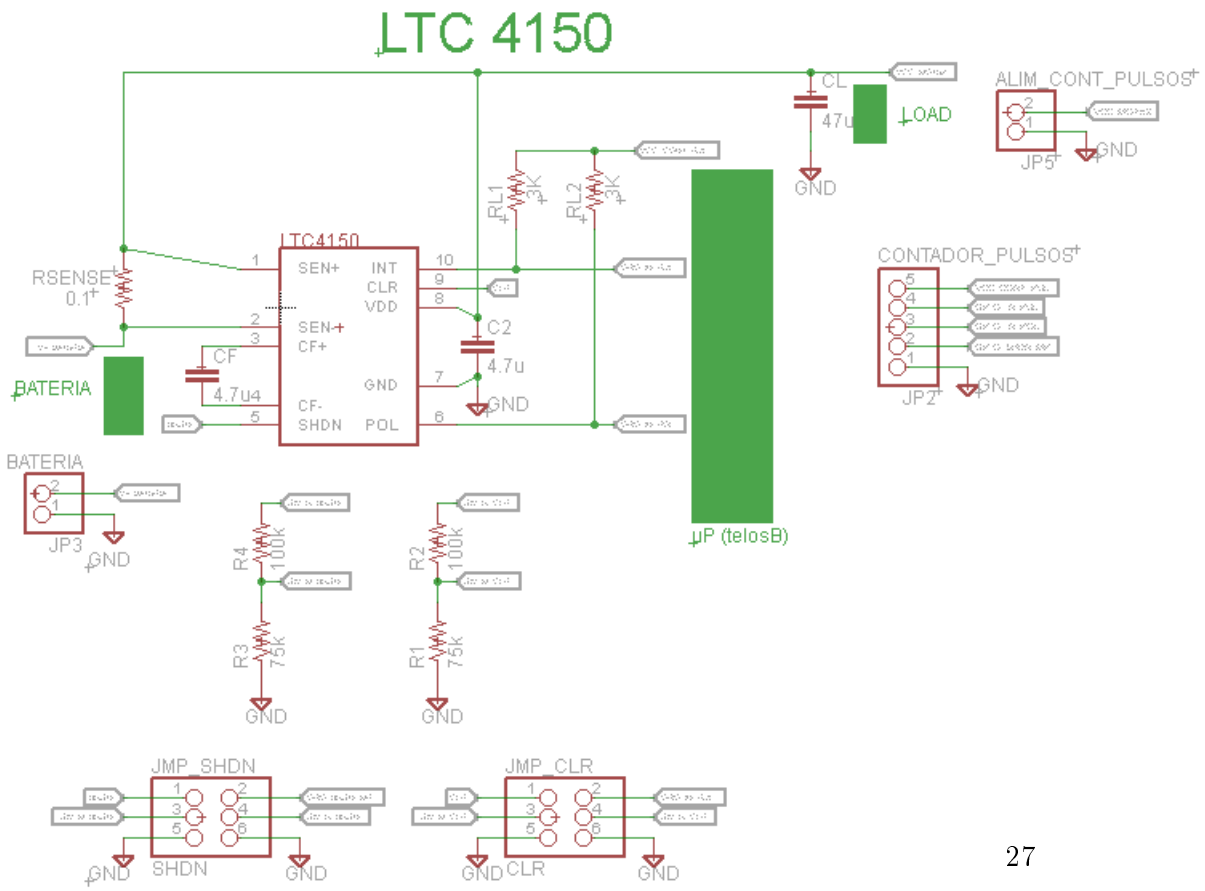


Figura 2.13: Layout placa medidora de consumo (bottom view)



(a)



(b)

Figura 2.14: *(a)Esquemático, placa medidora de perfiles de corriente (b) Esquemático, placa medidora de consumo de carga*

Medidor de perfiles de corriente

Este circuito está basado en el integrado INA139 [32] como se muestra en la figura 2.14a. El circuito amplifica la diferencia de potencial de un resistor shunt R_S por el cual circula la corriente que se quiere medir, utilizando el resistor R_{S1} para variar la ganancia de amplificación de 1 a 100.

Las principales características del amplificador INA139 se detallan a continuación:

- Intensidad de corriente $I_{bias} = 10\ \mu$A (valor típico).
- Intensidad de corriente de reposo de 60 μA (valor típico).
- Ancho de banda típico de 440 kHz para una resistencia $R_L = 10\ \text{k}\Omega$.
- Permite medir una diferencia de potencial de hasta 500 mV en bornes de la resistencia R_S y un voltaje máximo de 40 V en modo común.
- Permite alimentar el circuito de medida independiente de la carga.

Debido a que se pretende medir perfiles en un rango de unos pocos μA hasta 500 mA, se opta por colocar un preset en lugar de R_S y de R_{S1}, sacrificando precisión. De esta manera se puede tomar registro del consumo de corriente en el tiempo de los distintos ciclos del sistema (captura de imagen y envío de datos por RF, etc). En caso de requerir mayor precisión se debe sustituir R_S por una resistencia de valor fijo menor o igual a 1Ω de tolerancia 1 %.

Modo de uso Las conexiones se realizan en el conjunto de pines JP 1. El amplificador se puede alimentar desde las mismas baterías que alimentan al DUT, ó utilizar una alimentación independiente, a través de los pines P1 y P4 respectivamente. El perfil de corriente se tiene a través de un osciloscopio midiendo la tensión en el pin 2, la cual verifica la siguiente ecuación:

$$V_{out} = I_s\, R_s\, R_{s1}\, K \tag{2.2}$$

donde K = 1000μA/V, conociendo el valor de R_s y R_{s1} se puede despejar I_s.

Medidor de consumo de carga

Este circuito está basado en el integrado LTC4150 [28] como se muestra en la figura 2.14b. Es un circuito contador de carga en coulomb que convierte la diferencia de potencial en R_{sense} en frecuencia de forma que la cantidad de pulsos a la salida del conversor es proporcional a la cantidad de carga que entrega la batería.

Para poder interpretar los datos es necesario disponer de un PC o alguna plataforma de procesamiento que sea capaz de medir los pulsos a la salida y hacer la conversión correspondiente, dada por la ecuación 2.3 donde Q es la carga en Ah y N el número de interrupciones.

$$Q = \frac{N}{3600\, G_{fv}\, R_{sense}} [Ah] \tag{2.3}$$

En este circuito se encuentran disponibles los conjuntos de pines JP2, JP3, CLR, SHDN, que permiten además de conectar el circuito a la plataforma de procesamiento, variar las conexiones mediante jumpers según la aplicación. Las dos posibilidades son conectar el LTC4150 directo al dispositivo que cuenta pulsos o a través de un divisor resistivo de voltaje. En la figura 2.15 muestra cómo conectar los jumpers para cada caso.

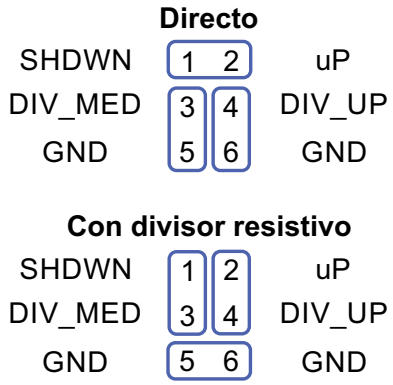


Figura 2.15: Diagrama de conexión de jumpers

- JP3 alimentación del dispositivo en estudio.
- JP2 conexiones hacia la plataforma de procesamiento.
- JP5 alimentación del circuito.

Capítulo 3

Estudio del sistema visual: geometría de la trampa, iluminación e imágenes.

3.1. Resumen

El conteo de polillas capturadas en la trampa implica que éstas puedan ser identificadas en la imagen por el personal que va a realizar el conteo. Para esto, es necesario encontrar la mejor configuración en cuanto a: i) tipo de iluminación (tipo, cantidad y posición de LEDs e intensidad de iluminaciones que se traduce en consumo de corriente), ii) altura de la cámara respecto al piso de la trampa, iii) modelo de lente y iv) geometría del piso de la trampa.

Para estudiar cómo estas cuatro variables afectan a la calidad de la imagen, se construyó un set-up experimental que permitió obtener un banco de imágenes modificando estas variables sistemáticamente. Este capítulo presenta las pruebas realizadas y cómo se logró obtener la configuración que asegura la identificación eficaz de las polillas. Ésta consiste en: lente (LS-20150), 4 LEDs (dispuestos en los extremos del techo de la trampa, dirigidos hacia arriba, blancos, de alta luminosidad), altura cámara - piso = 20,5 cm y piso de la trampa inclinado 20^o respecto a la horizontal . A través de este estudio, se desencadenó la aprobación del modelo de trampa por parte de la cooperativa JUMECAL. Sin embargo, no fue posible optimizar el sistema de iluminación ya que al momento de realizar los ensayos no se contaba con la trampa final del sistema. Por esto el capítulo concluye presentando propuestas a futuras mejoras.

3.2. Consideraciones previas

En este capítulo se utilizan conceptos básicos de óptica y fotografía, no siendo el objetivo profundizar en los mismos. Es por esto que se hará una breve introducción de los conceptos que se consideran de mayor relevancia, dejando referencias para que el lector pueda interiorizarse en los mismos si lo cree necesario.

- **Aberración:** Son los defectos de un sistema óptico. Las aberraciones producen distorsiones en las imágenes que empobrecen su calidad [25].

- **Distorsión:** Es una forma de aberración. Se manifiesta en la forma del objeto, afectando particularmente a los bordes (ver figura 3.1). La distorsión de barril es muy importante en objetivos gran angular y objetivos de ojo de pez que se utilizan en fotografía para registrar campos muy amplios ($60^o - 180^o$) en espacios reducidos. Los objetivos corregidos de esta distorsión se llaman ortoscópicos [25].

- **Brillos:** Cuando se ilumina una superficie, ésta refleja la luz al punto de ocasionar zonas donde la intensidad de luz interfiere con los puntos que se encuentran dentro de la misma (ver figura 3.2). Estos efectos interfieren visualmente y también pueden interferir con los algoritmos de procesamiento que puedan implementar a futuro.

- **Calidad de imagen:** No existe una definición para este término. En esta documentación se considera que una imagen de buena calidad es aquella en la que se optimizan el foco, iluminación y resolución minimizando efectos como la distorsión y así poder distinguir eficazmente las polillas en la trampa.

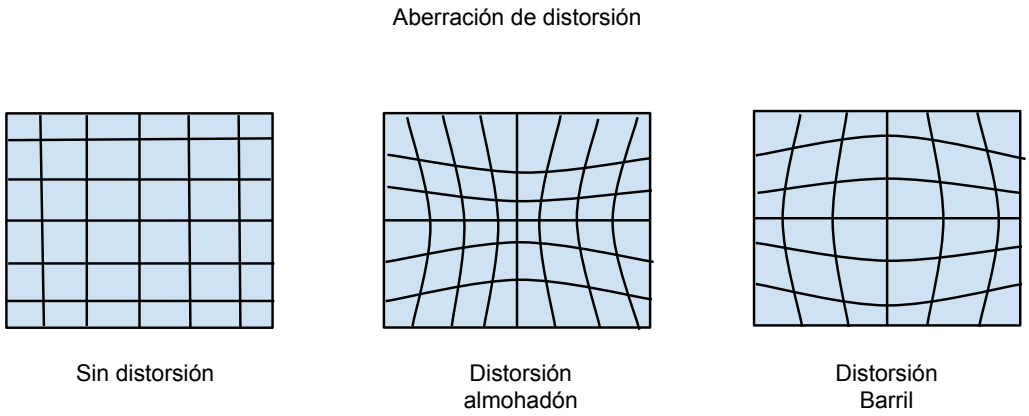


Figura 3.1: *Formas de aberración de distorsión.*

Figura 3.2: *Imagen 081 del banco, muestra señalado con rojo los brillos indeseados.*

3.3. Introducción

Con el fin de obtener una imagen donde un especialista logre reconocer los insectos de interés, se tomaron fotografías a pisos de trampas variando los siguientes factores:

- Lentes.
- Geometría del piso de la trampa.
- Resolución y compresión de imagen.
- Colores de LEDs de iluminación.
- Posicionamiento y cantidad de LEDs de iluminación.

En futuros PFC, se piensa implementar un algoritmo de conteo automático de polillas que podría disminuir el consumo de carga de los nodos, prescindiendo de enviar imágenes diariamente. Se debe considerar que para un algoritmo de procesamiento, algunos efectos en la imagen que no resultan molestos para el ojo humano, pueden afectar el resultado y viceversa. Aquí se presentan algunos ejemplos:

- Brillos como los mostrados en la figura 3.2, no interfieren en el reconocimiento de polillas por un humano, pero sí pueden afectar el resultado del algoritmo.

- La utilización de iluminación en distintos colores (longitudes de onda) si bien afecta la percepción de color de la imagen, puede llegar a ser beneficiosa para un algoritmo si logra resaltar las superficies de interés.

- Factores como la aberración pueden resultar molestos para el ojo humano pero no necesariamente para el algoritmo ya que el mismo podría utilizar procesamiento para invertir efectos no deseados.

Durante el desarrollo del proyecto se generó un banco de imágenes que puede ser utilizado a modo de lograr las condiciones óptimas para la aplicación de un algoritmo. Será de gran utilidad para el entrenamiento y validación del mismo. Además, permitirá ahondar en el estudio del sistema de iluminación. Este banco se adjunta al CD entregado junto con esta documentación.

Se debe destacar que gran parte de las imágenes presentes en tal banco no se encuentran centradas respecto al piso de la trampa y muchas otras se observan difusas. Esto se debe a que el soporte donde se enrosca el lente de la cámara utilizada tenía dos imperfecciones: *i)*- un defecto en la rosca que no permitía hacer foco por completo con alguno de los lentes, *ii)*- el soporte no se encontraba centrado respecto al foto-sensor. En el apéndice B se describe las soluciones para estos defectos.

3.4. Caracterización de imágenes

En esta sección, se caracterizan los distintos factores que influyen a la hora de la toma de imagen:

1. Lente a utilizar y geometría del piso de la trampa.

2. Sistema de iluminación considerando colores, cantidad y posicionamiento de LEDs en la trampa.

3. Compresión y resolución de imágenes JPEG.

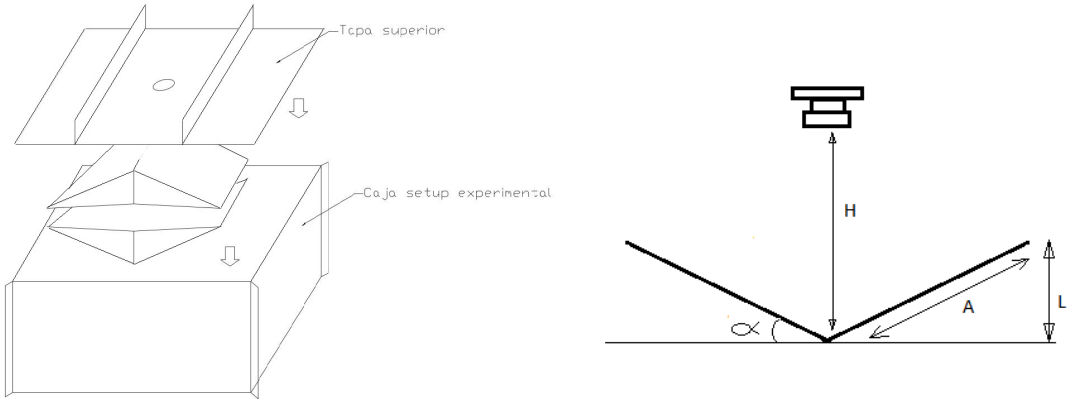
En base al lente elegido y determinación de la geometría del piso de la trampa, se captura una imagen que desencadena en el modelo final de la trampa, aprobada por JUMECAL. En la subsección 3.4.3 se presenta este modelo.

Para tomar las diferentes imágenes, se utilizó el programa ejecutable *LS-Y201-2MP.exe* [29] que provee el fabricante de la cámara. El mismo permite tomar y almacenar imágenes seleccionando su tamaño y compresión. Vale aclarar que este software tiene errores[1], lo que obliga en algunos casos a utilizar un programa de comunicación serial para enviar los comandos a la cámara, como son Realterm, Hyperterminal ó X-CTU. Para la comunicación Cámara-PC se puede utilizar el circuito interfaz Cámara-PC presentado en la sección 2.6.2 del capítulo de Hardware ó directamente un cable conversor TTL-USB [11].

A modo de evitar que las condiciones externas interfieran a la hora de tomar las imágenes, se utilizó un set-up experimental dentro del cual se deposita la trampa. Éste consiste en una caja que evita el ingreso de luz exterior. Su color es negro mate para evitar que se refleje

[1] Ver detalles en apéndiceB

la iluminación de los LEDs (ver figura 3.3). En las diferentes imágenes tomadas se varían el ángulo del piso de la trampa tipo Wing respecto a la horizontal, así como la altura de cámara en el setup (figura 3.3).



(a) (b)

Figura 3.3: *(a) Setup experimental. (b) Variables de configuración piso-cámara.*

3.4.1. Geometría del piso de la trampa

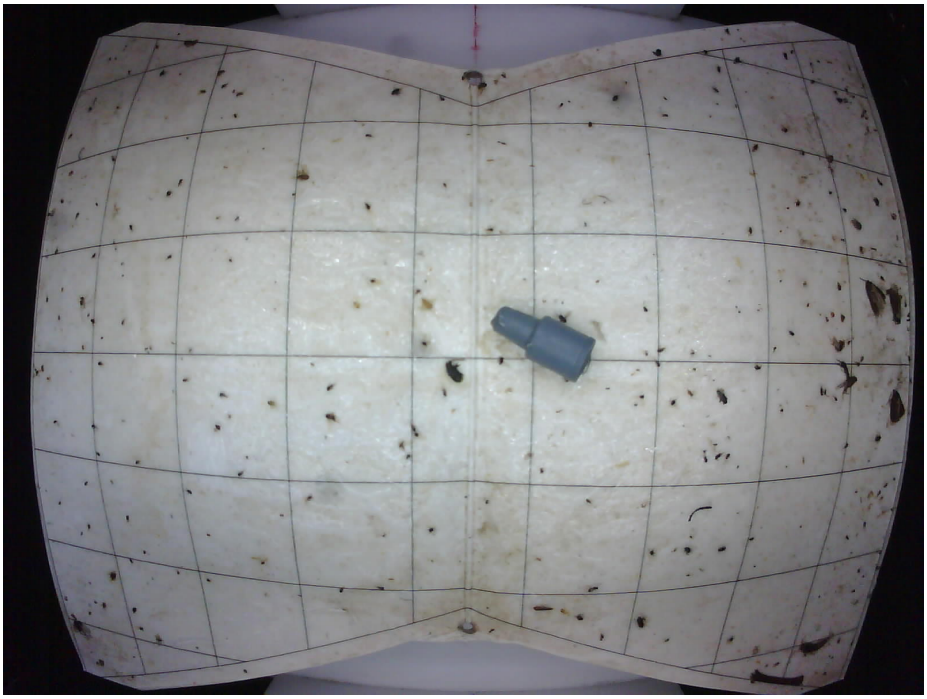
Tal como se detalla en el capítulo 2, los lentes a considerar en este proyecto son: LS-20150, LS-40166, HX-1820 y el que tiene por defecto la cámara (CL4022IR).

Debido a que los primeros tres lentes son de tipo gran angular, se debe definir si es conveniente utilizar una trampa con piso plano ó mantener la geometría original con piso inclinado. Esto tiene influencia en la iluminación, la aberración y foco de la imagen. Teóricamente el piso inclinado debe invertir las aberraciones que genera el lente gran angular y permitir tener foco en todos los puntos de la imagen. Efectivamente es lo que sucede y se puede comprobar en las figuras 3.4, 3.5 y 3.8:

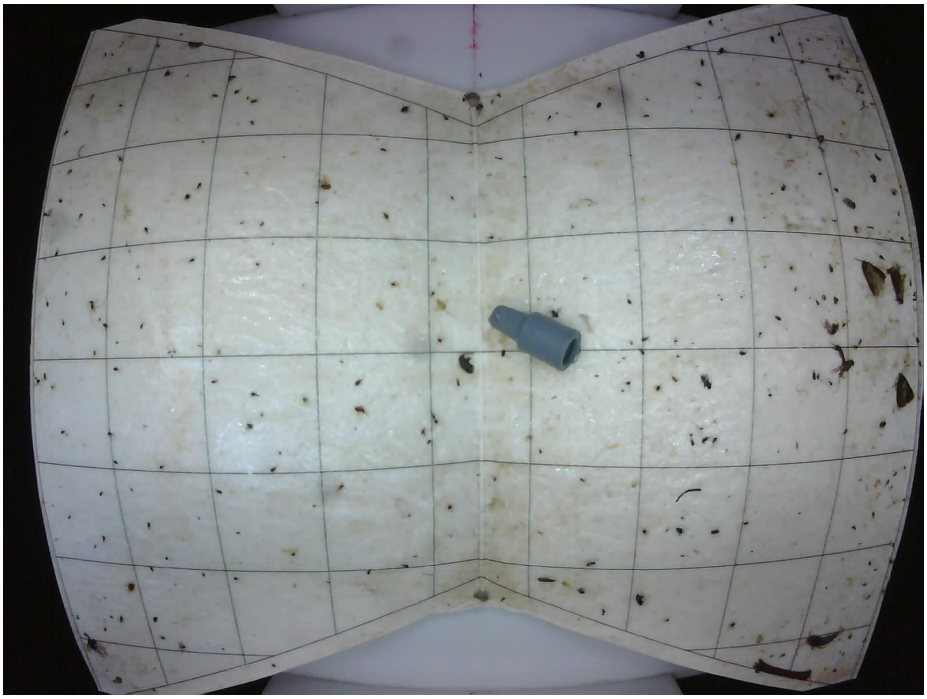
- En la figura 3.4a donde el piso es plano, se observa en las líneas horizontales una curvatura que aumenta sobre los bordes de la imagen. Esta curvatura disminuye notoriamente al tener el piso inclinado, como se muestra en la 3.4b.

- En las figura 3.5 se observa que no es posible tener foco en el centro y bordes de la imagen de forma simultánea cuando se tiene piso plano. Esto no sucede cuando el piso es inclinado y se puede apreciar en la figura 3.8.

De aquí se concluye que la trampa a utilizar tendrá piso inclinado.

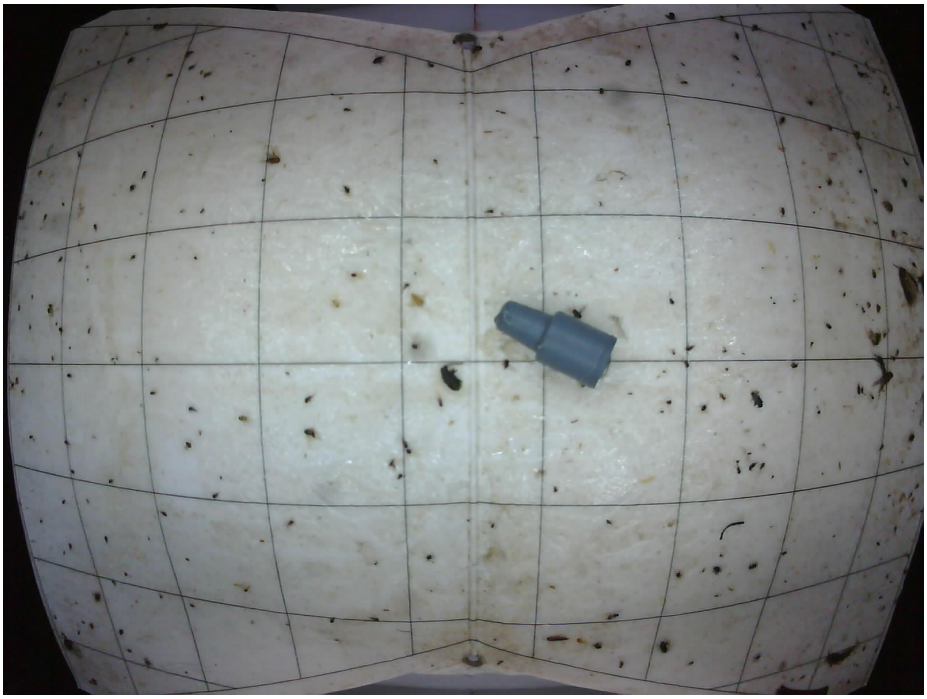


(a)

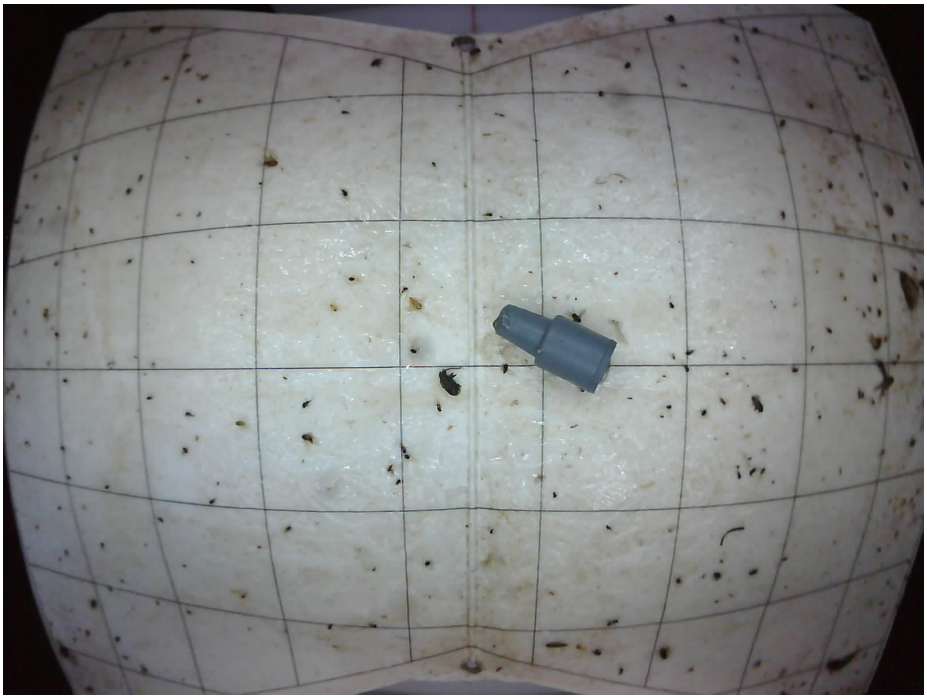


(b)

Figura 3.4: *Comparación piso inclinado vs piso plano y su efecto en la aberración. (a) Imagen 175 del banco de imágenes; Piso plano L=0mm , H=105mm , A=120mm; Lente HX-1820; Tamaño 130 kB. (b) Imagen 174 del banco de imágenes; Piso inclinado L=35mm , H=145mm , A=120mm; Lente HX-1820; Tamaño 127 kB.*



(a)



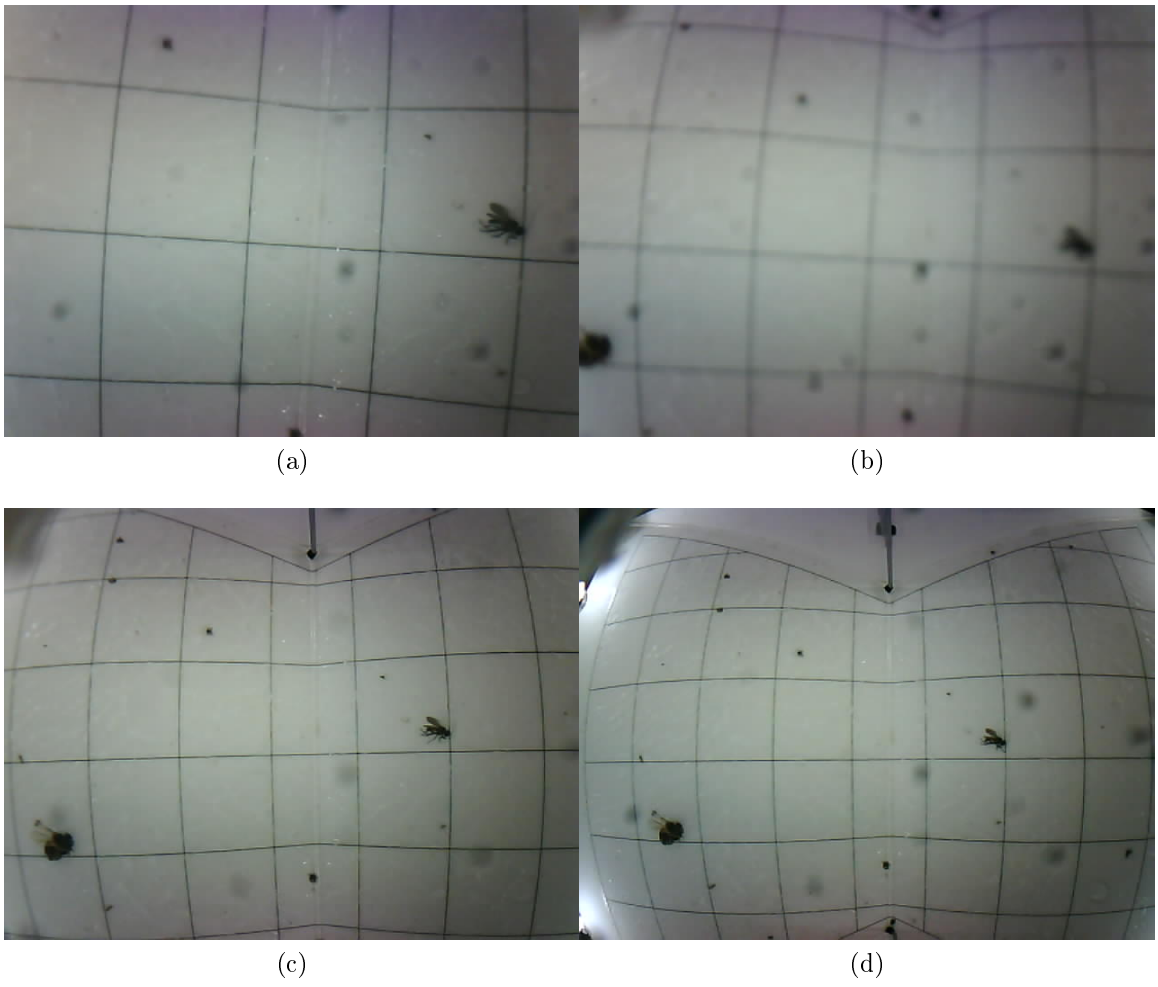
(b)

Figura 3.5: *Comparación en piso plano y su efecto en el foco. (a)Imagen 176 del banco; Piso plano L=35mm , H=145mm , A=120mm; Lente HX-1820; Foco en bordes; Tamaño 127 kB. (b)Imagen 177 del banco; Piso plano L=0mm , H=105mm , A=120mm; Lente HX-1820; Foco en centro; Tamaño 118 kB*

3.4.2. Caracterización de diferentes lentes y elección del lente a utilizar.

Elegida la geometría del piso de la trampa, se pasa a determinar el lente a utilizar.

Se dispone de los lentes CL4022IR, LS-20150, LS-40166 y HX-1820, ordenados de menor a mayor ángulo de cobertura. Manteniendo las dimensiones de la trampa original, se toman imágenes para determinar si con alguno de los lentes disponibles se logra capturar el piso de la trampa de forma completa. Se utiliza como sistema de iluminación cuatro LEDs blancos, colocados en las puntas del techo de la trampa, dirigidos hacia arriba. La cámara se coloca en el centro del techo de la trampa. Las imágenes se observan en las figuras 3.6. Se tiene que con el lente HX-1820 se logra capturar gran parte del piso, siendo necesario aumentar la separación entre el piso y techo 5 mm para lograr una captura completa. Se considera que este cambio no afecta las características de la trampa de forma significativa.



(a) (b)

(c) (d)

Figura 3.6: *Cobertura del piso de la trampa original con los diferentes lentes. (a)Lente CL4022IR. (b)Lente LS-20150. (c)Lente LS-40166. (d) Lente HX-1820.*

Se envió a JUMECAL la mejor imagen capturada con el lente HX-1820, presentada en

la figura 3.7, la cual no fue validada. Esto se debe a que las polillas de interés (señaladas con los círculos rojos en la figura 3.7) no se distinguen claramente por las aberraciones que genera el lente[2]. Se opta entonces por cambiar a un lente que disminuya estos efectos dentro de los gran angulares. Se toma el lente LS-20150, que es el de menor ángulo, obteniéndose la figura 3.8 bajo las siguientes condiciones:

1. Aumento de la altura de la trampa en 9 cm respecto a la original, a modo de tener una cobertura completa del piso.

2. Ángulo del piso de la trampa respecto a la horizontal de $\alpha = 20^o$.

3. Iluminación con 4 LEDs blancos apuntando hacia arriba.

Se observa que se obtienen mejoras en las aberraciones y foco. Sin embargo, se tiene que la iluminación puede ser optimizada. La figura 3.8 fue enviada a JUMECAL, desencadenando la validación del modelo de trampa que se presenta en este documento.

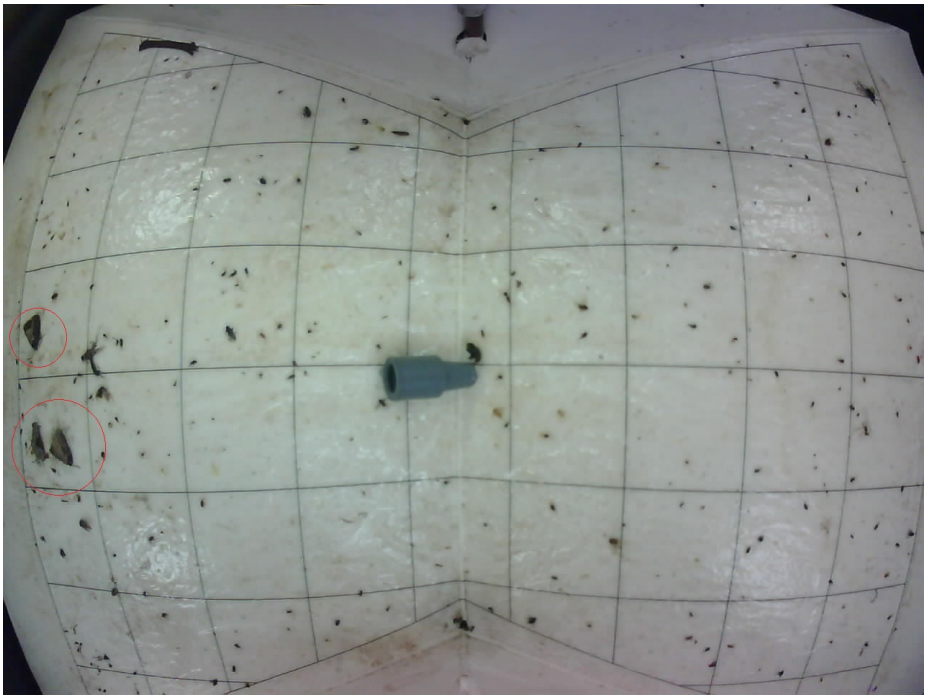


Figura 3.7: *Imagen 130 del banco de imágenes; Piso inclinado (inclinación por defecto); 4 LEDs blancos en esquinas del techo de la trampa, dirigidos hacia arriba; H=12,5 cm; lente HX-1820; Tamaño 80 kB*

[2]Las polillas se encuentran en los bordes donde la distorsión es mayor.

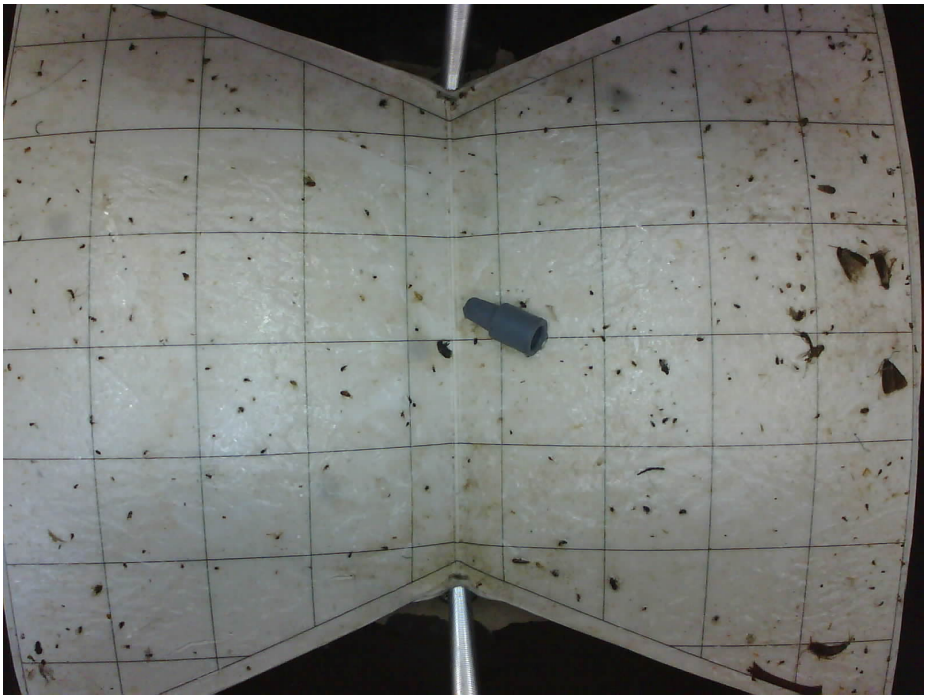


Figura 3.8: *Imagen 193 del banco de imágenes. L=41mm, H=215mm, A=120mm; lente LS-20150; peso 122 kB*

Las características del lente LS-20150 se pueden observar en el capítulo 2.

3.4.3. Geometría de trampa aprobada por JUMECAL

Luego del envío de la imagen 3.8, surge el modelo de trampa aprobada por JUMECAL. Una imagen de la misma se presenta en la figura 3.9.

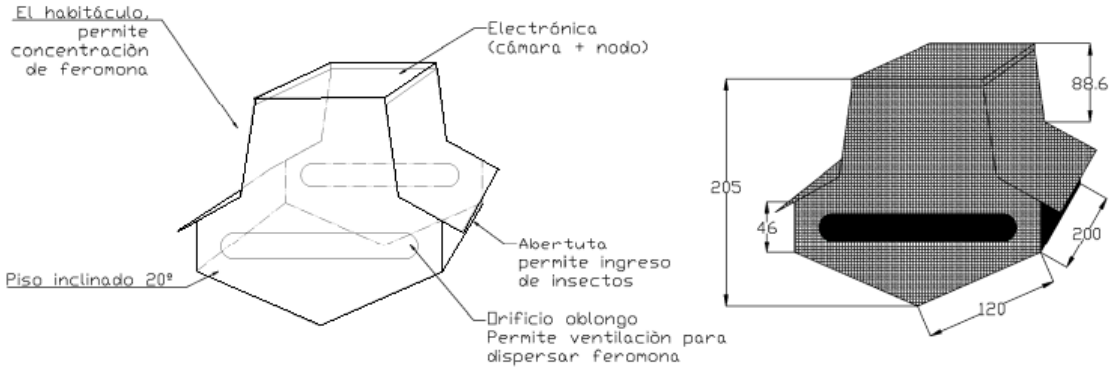


Figura 3.9: *Dimensiones y geometría de la trampa aprobada por JUMECAL*

En una reunión mantenida con integrantes de esta cooperativa, se llegó a esta geometría de trampa, considerando los siguientes puntos:

- Mantener aberturas laterales para favorecer la difusión de la feromona.
- Aberturas frontales para el ingreso de los insectos.
- Ángulo del piso de la trampa respecto a la horizontal de $\alpha = 20^o$, dado por el resultado de las pruebas.

Vale aclarar que esta validación se obtuvo semanas antes de presentar esta documentación. Es por esto que el estudio del sistema de iluminación no se realiza sobre la geometría de esta trampa.

En la próxima temporada de reproducción de polillas se evaluará si los cambios realizados en la geometría de la trampa no tiene efectos secundarios a la hora de atraer polillas.

3.4.4. Caracterización de iluminación con diferentes colores de LEDs

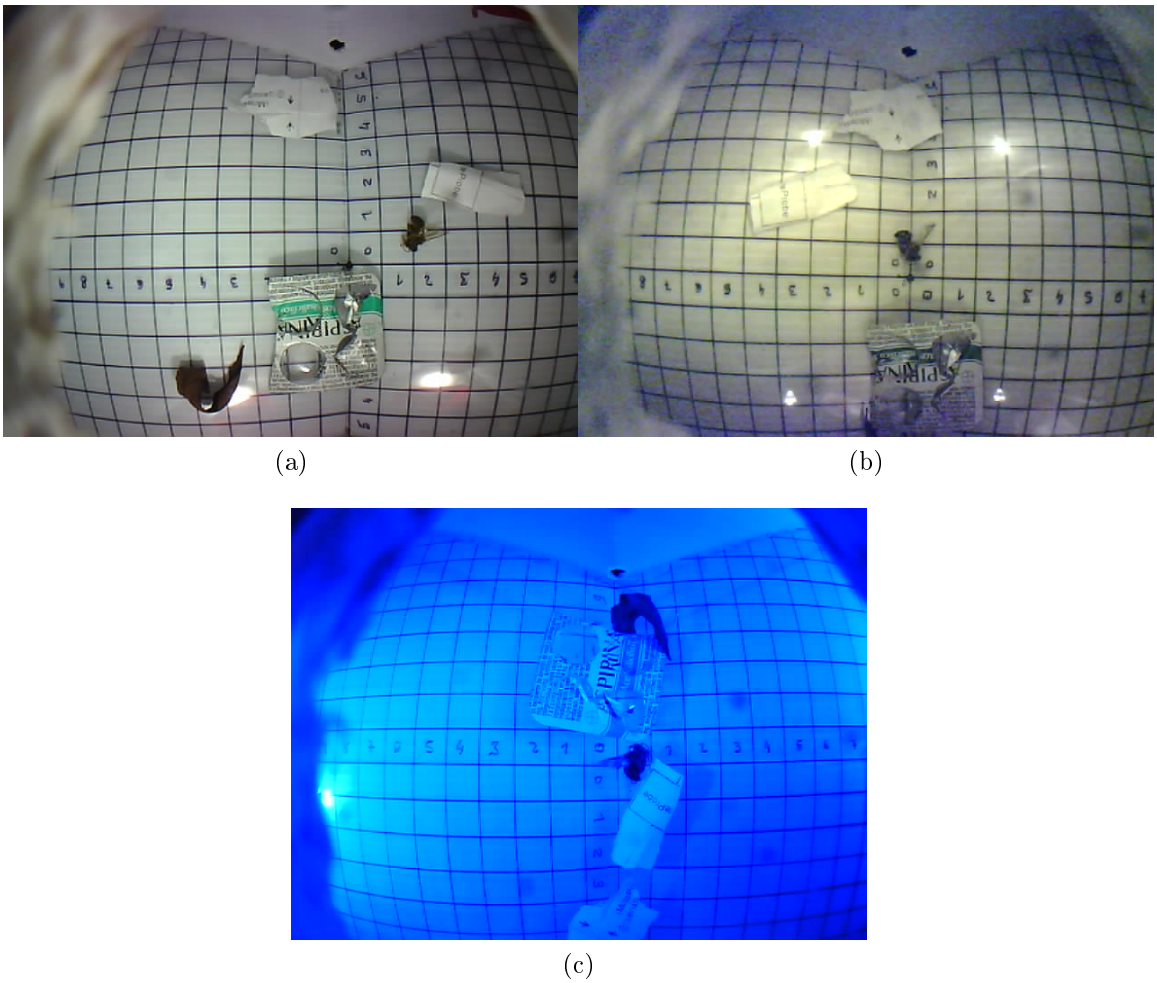
Con la intención de determinar el color de iluminación que facilite el reconocimiento de polillas en la trampa, se toman imágenes con LEDs de diferentes longitudes de onda, en las siguientes condiciones:

- Lente HX-1820.
- Compresión 05h y resolución 640x480 pixeles.
- Trampa auxiliar creada a partir de las medidas de la trampa original.

En las figuras 3.10a, 3.10b y 3.10c se muestran las imágenes obtenidas utilizando LEDs blancos, amarillos y azules respectivamente. Se tiene que la imagen tomada con LEDs blancos, resulta las más adecuada ya que no afecta significativamente la percepción de colores y permite distinguir con mayor claridad los objetos. Esto es de esperar ya que la cámara posee un sensor CMOS con sensibilidad RGB.

Otra opción considerada es utilizar la luz natural como iluminación. Si bien en este caso se disminuiría el consumo del sistema, la calidad de la imagen se vería afectada por diferentes factores como las condiciones lumínicas del día y la sombra de hojas y frutos que no permitirían una iluminación uniforme. Esta alternativa se descarta al realizar pruebas en campo, donde se observa que en ciertas ocasiones la iluminación solar provoca imágenes extremadamente blancas, sin poderse distinguir objetos con claridad en la trampa.

Es así que se concluye que la mejor opción es utilizar LEDs blancos y tomar las imágenes entre el atardecer y amanecer (noche).



(a) (b)

(c)

Figura 3.10: *Piso inclinado (inclinación por defecto); H=12,5cm. Imágenes obtenidas utilizando (a) LEDs blancos. (b) LEDs amarillos. (c)LEDs azules.*

3.4.5. Caracterización de iluminación variando cantidad y posicionamiento de LEDs

Con el presente ensayo se determina un posicionamiento óptimo de los LEDs, obteniendo una iluminación uniforme con el menor consumo de corriente posible. Se realizan pruebas con dos y cuatro LEDs de alta luminosidad colocándolos en diferentes lugares de la trampa. Se toman imágenes con y sin difusores dirigiendo los LEDs hacia arriba, los costados ó hacia el piso de la trampa.

Luego de tomar una gran cantidad de imágenes, se tiene que la mejor alternativa dentro de las opciones consideradas es:

- 4 LEDs blancos con su extremo limado para difundir la luz, según la figura 3.12, colocados en las esquinas de la trampa.

- Los LEDs se direccionan hacia arriba.
- No se utilizan difusores.

La imagen obtenida se muestra en la figura 3.11, donde se observa que no existen reflejos molestos al ojo humano ni que puedan afectar a la hora de utilizar un algoritmo de procesamiento.

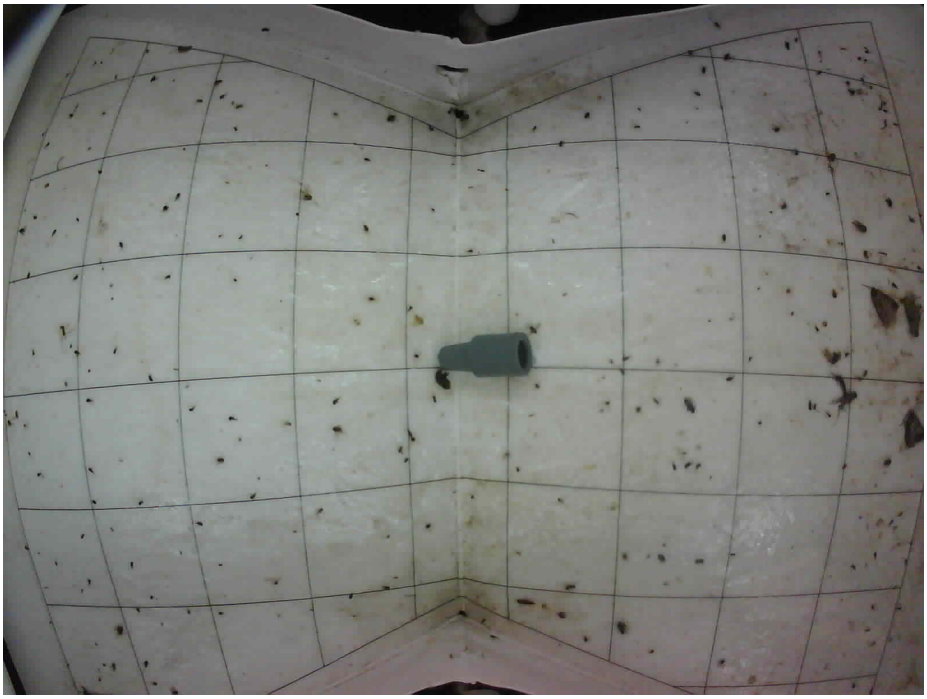


Figura 3.11: *Imagen obtenida utilizando cuatro LEDs blancos apuntando hacia el techo de la trampa; Piso inclinado (inclinación por defecto); H=13cm; lente HX-1820; peso 67 kB.*

Al momento de tomar estas imágenes, aún no se tenía la trampa validada por JUMECAL. La iluminación debe caracterizarse con esta trampa, pudiendo ser mejorada a través de un estudio exhaustivo, quedando como trabajo a futuro.

A continuación se dejan algunas recomendaciones que generaron buenos resultados al elaborar el banco de imágenes.

- Utilizar LEDs con el mayor ángulo de iluminación posible. Se pueden utilizar difusores con este propósito.
- Utilizar LEDs blancos con una leve componente de azul (luz fría).
- Utilizar LEDs de alta luminosidad a un porcentaje bajo de su capacidad máxima, así se evitan reflejos y halos de luz .
- No iluminar directo al piso de la trampa, hacer que la luz se refleje en el techo para evitar reflejos en el piso.

- La distancia entre cada LED y el techo debe ser de al menos 0.5 cm para obtener una mejor difusión de la luz.
- Para disminuir la intensidad luminosa de los LEDs, variar la corriente a través de una fuente, hasta obtener la intensidad deseada. Luego, colocar una resistencia en serie con los LEDs que asegure tener esta corriente.

En la figura 3.12 se muestra una forma de aumentar el ángulo de iluminación de un LED. Se debe limar el lente del LED dejándolo lo más recto posible y pulir la superficie limada.

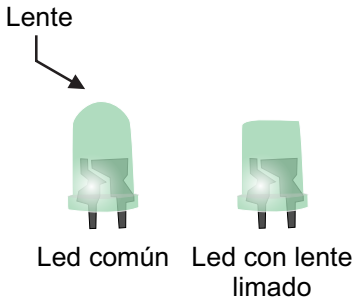


Figura 3.12: *Mejora del ángulo de iluminación de un led estandar.*

3.5. Conclusiones del capítulo

La configuración trampa-iluminación-lente con la que se obtienen imágenes de buena calidad, con poca aberración, buen foco, buena luminosidad y sin reflejos, es la siguiente:

- Tomar imágenes en la noche.
- Piso inclinado un ángulo $\alpha = 20^o$ de la horizontal.
- 4 LEDs de alta luminosidad blancos ubicados en los vértices del techo de la trampa, dirigidos hacia arriba de forma que se reflejen en el techo.
- Lente LS-20150.

El banco de imágenes generado aporta información sobre los parámetros estudiados en este capítulo y brinda las herramientas necesarias a la hora de aplicar mejoras al sistema de iluminación utilizando la trampa validada por JUMECAL. Adicionalmente, este banco brindará información para futuros PFC a la hora de aplicar un algoritmo de procesamiento para el conteo automático de polillas. Servirá para el entrenamiento del mismo y validación de resultados en función de las características del sistema.

Capítulo 4

Red y protocolo de comunicación

4.1. Resumen

En este capítulo se presenta el proceso de diseño del protocolo de comunicaciones propuesto. Se realiza una breve introducción a las redes de sensores inalámbricos destacando los requerimientos particulares que tiene que cumplir el sistema para la aplicación en estudio. Se analiza la implementación del *stack* de red en Contiki OS y se evalúa qué protocolos pueden ser reutilizados en *WNSvision*. Finalmente se presenta la implementación de un *stack* propio, y se detalla la topología de la red y el protocolo de nivel de aplicación implementado. [1]

4.2. Introducción

Las redes de sensores inalámbricos están constituidas por dispositivos autónomos distribuidos espacialmente, denominados comúnmente nodos. Los nodos son sistemas microelectrónicos embebidos de tamaño y hardware reducidos equipados con al menos un microcontrolador, uno o más sensores para monitoreo de magnitudes físicas o ambientales, un módulo de comunicación por radiofrecuencia y una fuente de alimentación, usualmente baterías.

En estas redes es sustancial minimizar el consumo de los nodos, de forma de aumentar la autonomía energética y maximizar el tiempo de vida de las baterías. Esto demanda especial consideración de los tiempos que duran las transmisiones y hace necesario que los módulos de radio, así como otros periféricos sean apagados cuando no se usan.

Dadas las limitaciones de hardware, estos dispositivos están diseñados para transmitir pequeñas cantidades de datos, en general algunos bytes representando valores numéricos de temperatura, humedad, presión o nivel de luz.

En la aplicación particular de RSI que es objeto de estudio en este proyecto se requiere la transmisión de grandes volúmenes de datos, debido a que se requiere transmitir imágenes de hasta 100 KB, como se vio en el capítulo 3. Esto constituyó uno de los principales retos enfrentados.

[1] La implementación del protocolo se detalla en capítulo 5

Debe garantizarse además, la correcta recepción de las imágenes en el destino, asegurando que los datos lleguen completos, en orden y sin duplicados, por lo que el protocolo de comunicación a diseñar debe ser confiable.

Estos requerimientos son los que se tomaron en cuenta para el diseño y posterior implementación de la topología de red y el protocolo de comunicaciones.

Para el diseño del protocolo se analizó el *stack* implementado en Contiki OS, y se definieron cuáles de ellos se adaptan de forma óptima a la aplicación en estudio. Esto se presenta en la sección 4.3.

En base a este análisis se diseñó un stack de protocolos propio, que consiste en la reutilización de las capas inferiores del stack de Contiki y la implementación de capas superiores que permitan cumplir los requerimientos mencionados. Esto es detallado en la sección 4.4.

4.3. *Stack* de comunicaciones en Contiki

Para la transmisión de datos en sistemas de comunicación, como lo son las RSI, se utilizan modelos basados en capas. Estas últimas refieren a distintos niveles de abstracción, donde cada capa realiza un conjunto de funciones bien definidas que ofrece como servicio a las capas superiores. Esta arquitectura de software simplifica el diseño y la implementación de las redes de comunicación.

El *stack* de comunicaciones implementado en Contiki se muestra en la figura 4.1.

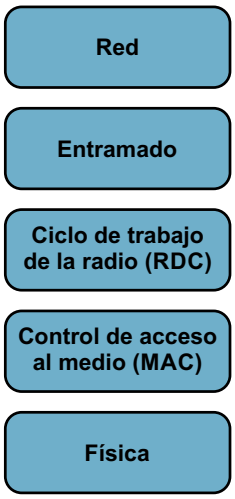


Figura 4.1: *Stack* de protocolos de Contiki

A continuación se describe cómo Contiki implementa las diferentes capas. Las capas física y de acceso al medio cumplen con el estándar IEEE 802.15.4 y se describen en la sección 4.3.1. En la sección 4.3.2 se describe la capa de ciclo de trabajo de la radio. El entramado también se implementa siguiendo el estándar IEEE 802.15.4 y se describe en la sección 4.3.3. Contiki ofrece dos opciones para el nivel de red, que se describen en la sección 4.3.4.

4.3.1. Capa física y capa MAC - Estándar IEEE 802.15.4 [4]

El *stack* de protocolos en Contiki comienza con el driver de la radio, el cual se encarga de enviar y recibir datos por RF así como de apagar y encender la radio según sea necesario. Como se describe en el capítulo 2 los nodos utilizados tienen integrado un transceiver de RF que opera en la banda de 2.4 GHz bajo el estándar IEEE 802.15.4.

Para el control de acceso al medio, el estándar IEEE 802.15.4 establece el uso del protocolo CSMA/CA no persistente.

El protocolo CSMA tiene como cometido arbitrar el acceso al medio en canales compartidos y funciona basado en la capacidad de los nodos de detectar cuándo se está usando el canal y actuar acorde a ello, de forma de disminuir la probabilidad de ocurrencia de colisiones.

En la modalidad no persistente, cada vez que un nodo desea transmitir una trama, se escucha el canal (detección de portadora). Si el canal está libre, comienza la transmisión, y si está ocupado espera un tiempo aleatorio y vuelve a sensar el canal, repitiendo el procedimiento.

La modalidad CA añade el anuncio por parte del nodo de su intención de transmitir antes de hacerlo. De esta forma, el resto de los nodos de la red conoce su intención, esperan un tiempo aleatorio adicional y si tras ese intervalo el medio continúa libre, realizan su transmisión. Esto reduce la probabilidad de colisiones en el canal.

4.3.2. Ciclo de trabajo de la radio

En las redes de baja potencia, el transceiver de radio debe estar apagado tanto como sea posible para ahorrar energía. En Contiki OS, esto se hace en la capa de Radio Duty Cycling (RDC) o Ciclo de Trabajo de la Radio. Contiki OS posee varios controladores de RDC [10], entre los que se encuentran ContikiMAC y NullRDC.

ContikiMAC enciende la radio de forma periódica en intervalos llamados *wake-ups* para sensar el canal y detectar si hay actividad. Si un paquete de transmisión es detectado durante un *wake-up*, el receptor es capaz de recibirlo. Cuando el paquete es recibido satisfactoriamente, el receptor envía un mensaje de reconocimiento (ACK). Para transmitir un paquete, el emisor envía el paquete de datos repetidamente hasta recibir un paquete de acuse del receptor. El emisor intenta sincronizarse con el receptor estimando cuándo ocurrirán los *wake-ups* a partir de los ACKs ya recibidos.

Por otro lado NullRDC, como su nombre lo indica, no implementa manejo del ciclo de trabajo de la radio. Es un protocolo que mantiene la radio encendida de forma permanente durante las transmisiones.

Experimentalmente se probaron ambos protocolos y se obtuvieron tiempos de envío más breves usando NullRDC lo que implica mantener la radio encendida por un menor tiempo, disminuyendo el consumo debido a la comunicación por RF. En base a esto y en simulaciones realizadas en Plagavision, es que se elige trabajar con NullRDC.

4.3.3. Entramado

El estándar IEEE 802.15.4 define cuatro tipos de tramas:

- Tramas de Beacon: son utilizadas como balizas para señalizar o establecer la configuración de red.
- Tramas de Datos: se utilizan para transmitir la información relevante o carga útil.
- Tramas de Reconocimiento de paquetes: se utilizan para confirmar las recepciones exitosas.
- Tramas de Comandos MAC: se utilizan para el control entre entidades pares de la capa de acceso al medio.

La estructura es la misma en todas las tramas, sólo se diferencian en el tamaño de la carga útil que llevan. Están compuestas por un encabezado de sincronización (SHR por sus siglas en inglés), un encabezado de capa física (PHR) y una unidad de datos de capa física (PSDU). La PSDU a su vez está compuesta por un encabezado y pie MAC (MHR y FHR respectivamente) y por la carga útil de la capa MAC (MSDU). Esta estructura se muestra en la figura 4.2.

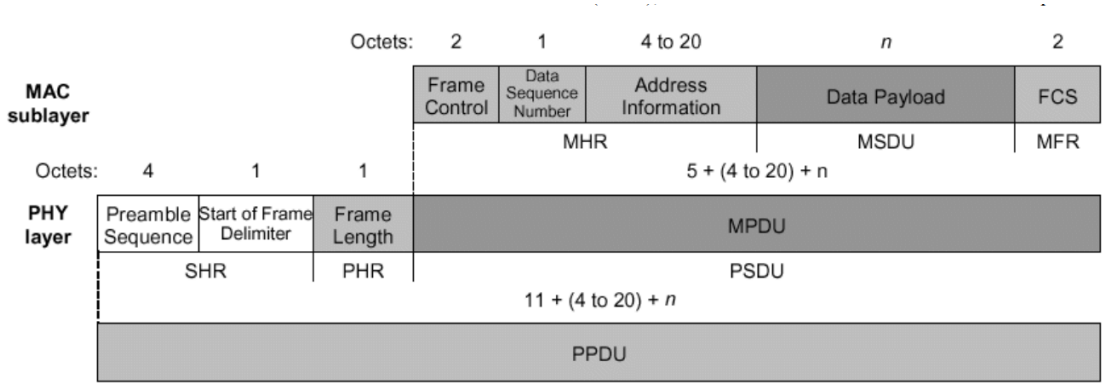


Figura 4.2: Estructura de trama

4.3.4. Capa de red

El sistema operativo Contiki OS ofrece dos opciones para comunicación a nivel de red. Por un lado el *stack* uIP (micro IP), una implementación de TCP/IP adaptada a redes inalámbricas de recursos limitados, y por otro el *stack* Rime, un conjunto de protocolos de red ligeros diseñados para redes inalámbricas de baja potencia.

El *stack* Rime está estructurado en capas muy simples de forma de minimizar la longitud de los encabezados. Rime ofrece una amplia gama de primitivas y protocolos sencillos que pueden ser combinadas por el desarrollador para lograr protocolos y mecanismos más complejos. En el anexo C se explica con más detalle el *stack* de protocolos Rime.

El *stack* uIP no fue estudiado en este proyecto.

4.4. *Stack* de comunicaciones *WSNvision*

Como fue mencionado en la sección 4.2 la aplicación para la que se desarrolla el protocolo de comunicaciones requiere la transmisión de grandes volúmenes de datos y que estos sean recibidos de forma íntegra en el nodo concentrador. Adicionalmente, se debe tener en cuenta el requerimiento de bajo consumo que tienen las RSI en general.

En las siguientes secciones se detalla cómo está implementado el *stack* de red utilizado en este proyecto.

4.4.1. Capas inferiores

Las capas física, MAC y de entramado del *stack WSNvision* están definidas por el transceiver que viene integrado en el SoC de los nodos utilizados, y son las definidas por el estándar 802.15.4.

Para la capa RDC se seleccionó el protocolo NullRDC por ser con el que se obtuvieron los menores tiempos de transmisión de imágenes.

Por encima de estas capas, se utilizan los servicios del *stack* Rime, en particular se utiliza la primitiva *unicast*, que añade a los paquetes enviados las direcciones de origen y destino. Así, cuando un nodo recibe un paquete, chequea que la dirección de destino coincida con la propia, descartando el paquete en caso contrario.

La primitiva *unicast* ofrece al desarrollador cinco funciones para construir sobre ellas su propio protocolo. Estas son: abrir y cerrar un canal de comunicación, enviar un mensaje y dos funciones de *callback*, una que se ejecuta cuando se recibe un mensaje desde la capa inferior, y otra que se ejecuta cuando se recibe el ACK de un mensaje enviado.

4.4.2. Topología de la red y direccionamiento

Para la topología de la red se definieron dos tipos de nodo: nodos sensores con capacidad visual *WimSN* y nodo concentrador *sink*. Se estableció además que la comunicación sea lineal y *multihop*.

Con la intención de cubrir el área más extensa posible dentro de los cultivos, minimizando la cantidad de nodos a utilizar, se propone una topología de red lineal, en la que cada nodo ve un solo vecino aguas arriba y un solo vecino aguas abajo. Esto permite realizar ruteo estático, donde cada nodo sabe que el destino o próximo salto para un paquete recibido es el vecino ubicado en la dirección opuesta a la que el paquete fue recibido.

Es necesario entonces que los datos sean transmitidos desde el nodo más lejano hasta el *sink*, por lo que el protocolo a implementar debe soportar envíos *multihop*.

Rime utiliza direcciones de 16 bits para lo que toma los dos bytes menos significativos de la dirección IEEE (EUI-64) [14] del nodo. Estos 2 bytes pueden ser seteados en tiempo de compilación utilizando la sentencia "make NODEID=1", que asigna al nodo la dirección 0x0001. Adicionalmente, la dirección del nodo puede consultarse en tiempo de ejecución utilizando la variable *node_id* del módulo *node-id.h* de Contiki OS.

Al crear la red se le asignó la dirección 1 al nodo *sink* y luego, yendo aguas arriba se asignan a los nodos *WimSN* las direcciones 2 a 5. De este modo cada nodo tiene asociado un identificador único en la red que se denomina de aquí en más como ID. Dada la topología lineal de la red, el nodo de ID x tendrá dentro de su rango de alcance al nodo de ID $x-1$

aguas abajo y al nodo de ID $x + 1$ aguas arriba. Esto es diferente en el caso del *sink*, que sólo tendrá dentro de su rango de alcance al nodo de ID 2.

4.4.3. Transmisión *multihop*

Para implementar el envío *multihop* de mensajes, cuando un nodo *WimSN* recibe un paquete de datos realiza la comparación entre su ID y el ID de destino del paquete recibido. Si estas no coinciden, determina que el paquete no es para él, por lo que procede a reenviarlo. Para determinar el sentido en que debe hacer el reenvío, verifica si el ID de destino es menor o mayor que su propia ID. Este procedimiento se realiza en la función de *recv_unicast*, la cual se detalla en la figura 5.7 en el capítulo 5.

4.4.4. Capa de Aplicación

La transmisión de datos se realiza de modo maestro-esclavo, siendo gobernada por el nodo *sink*. El *sink* se encarga de chequear conectividad, pedir el envío de imágenes y enviar órdenes a los nodos *WimSN* para que ingresen en bajo consumo, indicando el tiempo que deben permanecer en este estado.

Por su parte, los nodos *WimSN* tienen el rol de esclavos. Simplemente toman la foto y esperan los pedidos del *sink*, respondiendo según corresponda.

Para lograr un protocolo confiable, se parte de la primitiva *unicast* y sobre ella se agregan en nivel de aplicación: reenvíos, *buffers* de recepción y transmisión, y mensajes de control para administrar la red. También se agrega la capacidad de envíos *multihop*.

4.4.5. Unidades de datos en capa de Aplicación (APDU)

El estándar 802.15.4 define el tamaño máximo de paquete en 128 bytes. Considerando los encabezados de capa MAC y de Rime, la carga útil o *payload* que se puede transmitir es menor. Para la unidad de datos de capa de aplicación, ya sean mensajes de control o paquetes con fragmentos de imagen, se define una longitud de carga útil de 100 bytes y un encabezado de 12 bytes. La estructura de estos datos se muestra en la figura 4.3.

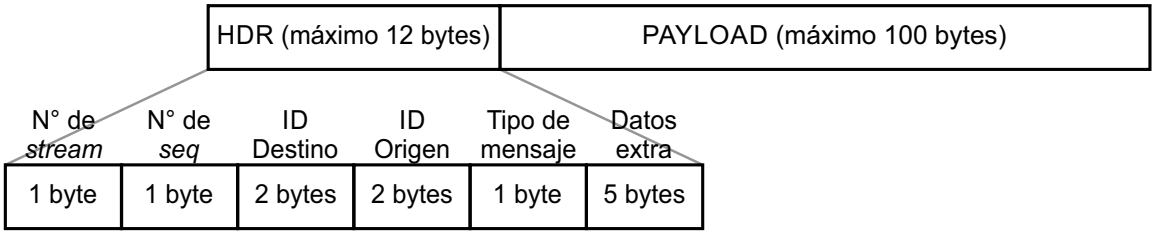


Figura 4.3: Unidad de datos de capa de aplicación

El encabezado de los paquetes de datos está compuesto por las siguientes variables:

- Número de *stream*: permite identificar a qué *stream* pertenece el paquete recibido. En los mensajes de control este campo es ignorado.

- Número de secuencia: identifica un paquete dentro de un *stream*. En los mensajes de control este campo es ignorado.
- ID del nodo destino: en los mensajes de control, es la dirección del nodo al que va dirigido el mensaje. Es el destino final, diferente al próximo salto en la red. En los paquetes con fragmentos de imagen, este valor siempre es 1, ya que las imágenes tiene a este nodo como destino final.
- ID del nodo origen
- Tipo de mensaje que se envía. Si se trata de un paquete que contiene parte de un *stream*, se identifica con el número 200.
- Datos extra. Dependen del tipo de mensaje.

Las imágenes se envían en ráfagas denominadas *streams*. Cada *stream* está compuesto por diez paquetes de datos pertenecientes a la imagen que se esté enviando. Los nodos WimSN envían un *stream* completo y esperan la respuesta del sink antes de seguir adelante. Este procedimiento se describe con mayor detalle en la sección 4.4.9.

4.4.6. Mensajes de control

Se implementan mensajes de control para gobernar el correcto funcionamiento de la red y la sincronización entre nodos. Cada uno de estos mensajes tiene una dirección determinada, unos van dirigidos desde el *sink* hacia los nodos *WimSN* (aguas arriba), y otros en sentido contrario. No existen mensajes de control entre nodos *WimSN*. Los mensajes y los datos extras que tienen asociados se describen a continuación:

- $PING$: Enviado por el *sink* a los diferentes nodos para consultar cuáles se encuentran activos en la red y listos para enviar la imagen de su trampa. No agrega datos extra. Espera como respuesta el mensaje $PONG$.
- $PONG$: Enviado por cada nodo *WimSN* como respuesta a un $PING$. No agrega datos extra.
- ASK_PIC: Enviado por el *sink* para solicitar a un nodo *WimSN* determinado que envíe su imagen. No agrega datos extra. Espera como respuesta el mensaje END_STREAM correspondiente al primer *stream* enviado por el nodo.
- END_STREAM: Este mensaje es enviado por un nodo *WimSN* par comunicar el fin de un *stream* o el reenvío de un paquete específico. No agrega datos extra. Espera como respuesta uno de los siguientes mensajes: $STREAM\_OK$, $STREAM\_WRONG$ ó ASK_PAQ.
- $STREAM\_OK$: Enviado por el *sink* para dar aviso de la recepción correcta de un *stream* y solicitar al *WimSN* el envío del siguiente. No agrega datos extra.
- $STREAM\_WRONG$: Enviado por el *sink* para dar aviso de que el número del *stream* recibido no es el esperado. No agrega datos extra.

- *ASK_PAQ*: Enviado por el *sink* para solicitar el reenvío de un paquete determinado de un *stream* que no fue recibido. No agrega datos extra.
- *END_PIC*: Aviso de fin de envío de imagen por parte del nodo *WimSN*. No agrega datos extra. Espera como respuesta el mensaje *OFF*.
- *OFF*: Orden enviada por el *sink* para que el nodo *WimSN* vaya a bajo consumo por un determinado periodo de tiempo. Añade dos bytes que representan el tiempo en minutos para que el nodo entre en bajo consumo. Espera como respuesta el mensaje *ACK_OFF*.
- *ACK_OFF*: Enviado por un nodo *WimSN* para avisar al *sink* que recibió el mensaje *OFF* y va a entrar en bajo consumo. No agrega datos extra.

En el listado anterior puede apreciarse que cada mensaje tiene asociado su correspondiente mensaje de respuesta, o espejo. Esto se tiene en cuenta en el momento de implementar las retransmisiones, sección 4.4.8.

4.4.7. *Buffers* de recepción y de transmisión

Ambos tipos de nodo implementan un *buffer* de recepción del tamaño de un paquete de capa de aplicación (112 bytes), en donde almacenan los mensajes recibidos por radio.

Adicionalmente, los nodos *WimSN* implementan en memoria RAM un *buffer* de transmisión del tamaño de un *stream* en el que guardan fragmentos de imágenes que van leyendo desde su memoria flash y luego transmiten de a paquetes.

Así mismo, el *sink* implementa un *buffer* de recepción del tamaño de un *stream* donde almacena en el orden correcto los paquetes recibidos hasta completar un *stream*, verificando la integridad del mismo, para luego guardarlo en su memoria flash.

4.4.8. Retransmisiones

El procedimiento de retransmisión de mensajes coincide en nodos *WimSN* y *sink*. Una vez que un nodo envía un mensaje de control, se inicializa un *timer*. Si vencido éste no se ha recibido respuesta, se procede a reenviar el mensaje. Además se añade un contador de reenvíos, de tal forma que si no se recibe la respuesta luego de un determinado número de reenvíos, se toma una acción predeterminada. Estas acciones dependen del tipo de mensaje, y se describen en la sección 4.4.9.

En el caso de envío de *streams* o paquetes de datos, una vez vencido el timer de reenvío se reenvían el *stream* o paquete seguidos además, del mensaje *END_STREAM*.

Para los mensajes *ACK_OFF*, *PONG*, *STREAM_OK*, *ASK_PAQ* y *STREAM_WRONG* no se realizan retransmisiones. Esto es posible porque los controles se llevan a cabo en sus mensajes espejo:

- Las pérdidas de los mensajes *STREAM_OK*, *ASK_PAQ* y *STREAM_WRONG* son controladas con retransmisiones del mensaje *END_STREAM*.
- Las pérdidas del mensaje *PONG* son controladas con las retransmisiones del mensaje *PING*.

- Las pérdidas del mensaje ACK_OFF son controladas con retransmisiones del mensaje OFF. Sin embargo estas pérdidas no interfieren en el funcionamiento de la red, dado que lo relevante en este caso es que el nodo *WimSN* recepcione el mensaje OFF e ingrese al modo de bajo consumo.

4.4.9. Funcionamiento de la red

El funcionamiento general de la red se compone de las siguientes etapas: i) Chequeo de conectividad, ii) Envío y recepción de imagen y iii) Envío al bajo consumo. Estas etapas se repiten tantas veces por día como haya sido configurado por el operador.

Para describir este funcionamento se describe en primera instancia el caso ideal: todos los mensajes y paquetes de datos llegan en tiempo y forma a su destino, y no se producen envíos duplicados ni desorden en la recepción. Más adelante se presentan los distintos escenarios de falla detectados y se describe cómo se comporta el protocolo en esos casos.

Para facilitar la comprensión para el caso ideal, en la figura 4.4 se presenta el funcionamiento en una red de tres nodos (el *sink* y dos *WimSN*). El funcionamiento para redes con mayor cantidad de nodos es análogo.

Funcionamiento ideal

i) Chequeo de conectividad

Luego de la inicialización, el nodo *sink* verifica hasta qué nodo *WimSN* tiene conectividad enviando un mensaje $PING$ y esperando un mensaje $PONG$ como respuesta. Dada la topología lineal de la red comienza por el nodo *WimSN2* y a medida que va recibiendo respuestas avanza aguas arriba. Los nodos *WimSN* cuando se incian, capturan su imagen y quedan a la espera del mensaje $PING$.

ii) Envío y recepción de imagen

Luego de enviar el mensaje $PONG$ los nodos *WimSN* quedan a la espera del pedido de su imagen por parte del *sink*.

El *sink* comienza a solicitar a cada nodo el envío de su imagen enviando el mensaje ASK_PIC. Comienza por el nodo más lejano que haya respondido en la etapa anterior, y avanza en dirección aguas abajo. De este modo se utiliza la topología lineal de la red para optimizar el consumo de carga, permitiendo que el nodo que ya envió su imagen puede entrar en bajo consumo ya que no tiene que servir de repetidor a otros nodos.

Cuando un nodo *WimSN* envía el último *stream* de su imagen, da aviso al *sink* con el mensaje END_PIC. Este proceso se describe con mayor detalle en la figura 4.5.

iii) Envío a bajo consumo

Cada vez que el *sink* recibe la imagen completa de un nodo *WimSN*, consulta al PC el tiempo en minutos que falta para el próximo ciclo y a continuación envía el mensaje OFF indicando al nodo que ingrese al bajo consumo por este tiempo.

El nodo *WimSN* responde con el mensaje ACK_OFF, ingresando inmediatamente al bajo consumo por el tiempo indicado, sin esperar respuesta.

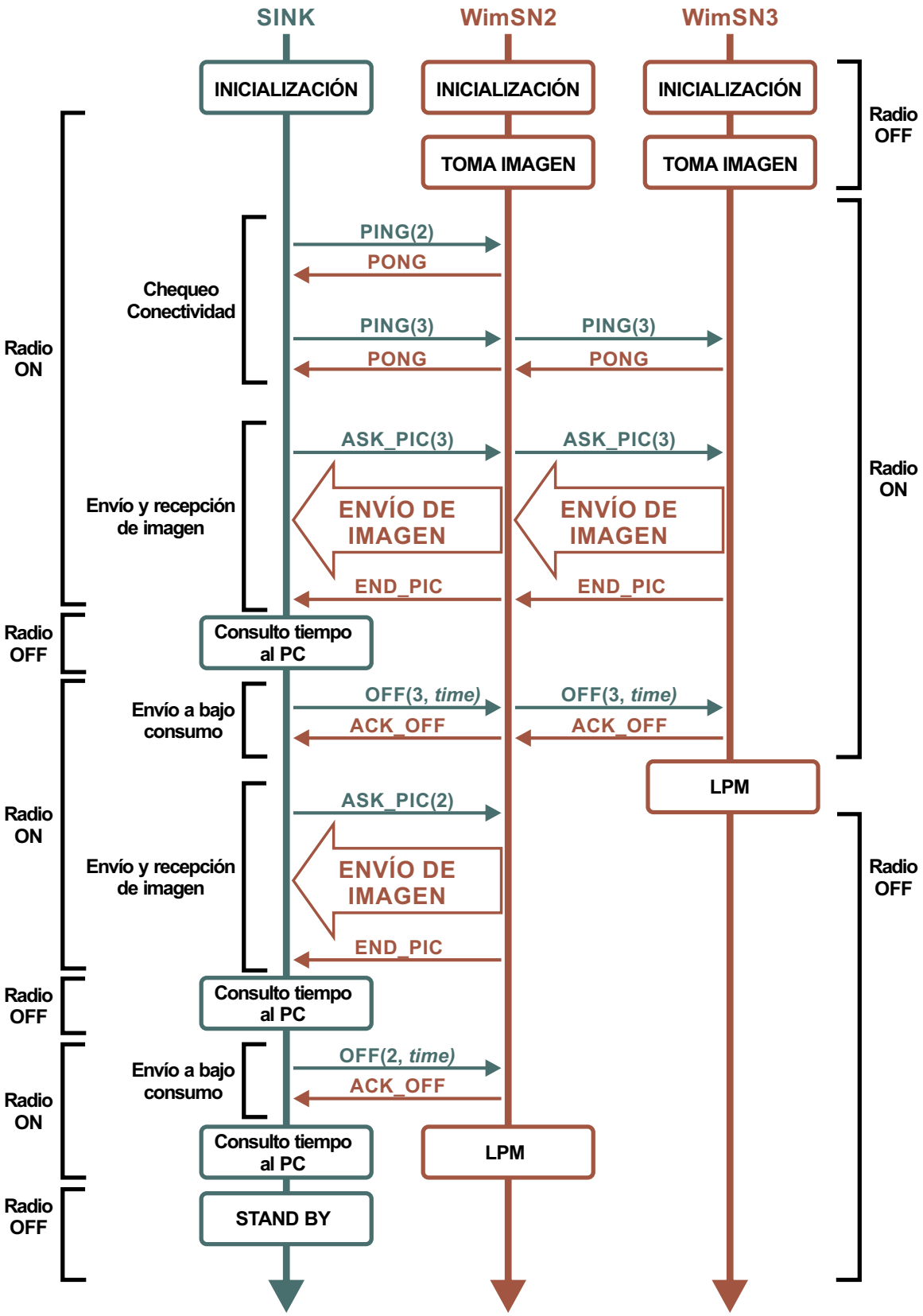


Figura 4.4: Funcionamiento ideal de la red, para una red de 3 nodos

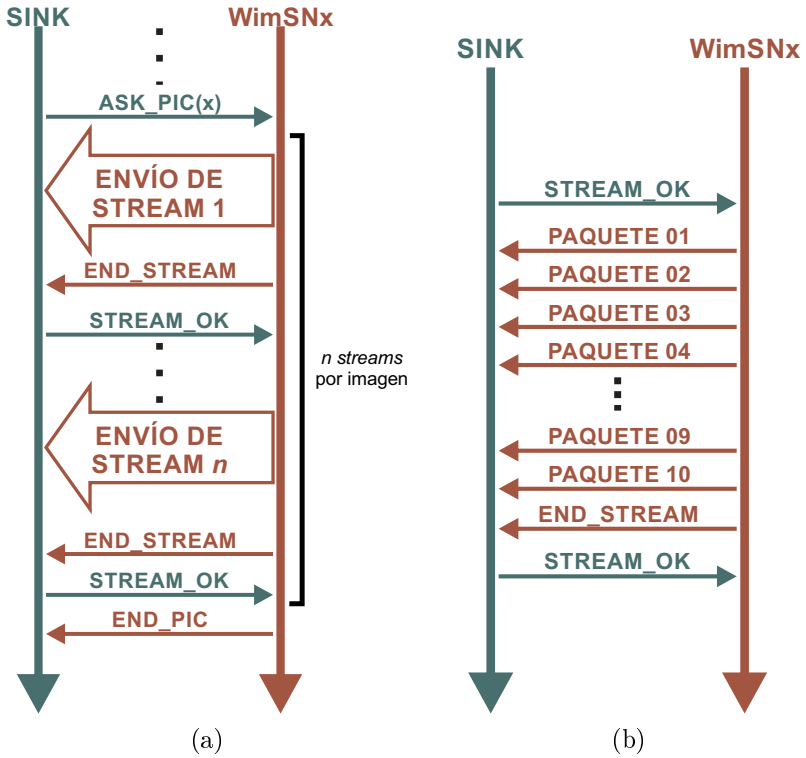


Figura 4.5: Envío y recepción de imágenes sin pérdidas. (a) Intercambio de mensajes en el envío de una imagen. (b) Envío de un *stream*

Detalle del proceso de envío y recepción de imágenes

Para solicitar el envío de imágenes el nodo *sink* envía a un nodo determinado el mensaje ASK_PIC y espera como respuesta el *stream* número 1 y su correspondiente END_STREAM.

Una vez que un nodo *WimSN* recibe un mensaje ASK_PIC, responde enviando el primer *stream* correspondiente a su imagen, enviando al final un mensaje END_STREAM para que el *sink* verifique que lo haya recibido correctamente. El nodo *WimSN* queda a la espera de la confirmación por parte del *sink*. En el caso ideal, la respuesta del sink es un $STREAM\_OK$ que indica que recibió todos los paquetes del *stream* esperado. Los casos de falla en la etapa de envío de imágenes se detallan a continuación junto con las otras fallas.

Escenarios de falla

A continuación se presentan los diferentes escenarios de falla relevados y se describe cómo se comporta el protocolo para asegurar la correcta recepción de las imágenes en el *sink*. Al final de esta sección se adjuntan imágenes para ayudar a entender los diferentes escenarios.

i) Fallas en Chequeo de conectividad

Durante la etapa de Chequeo de conectividad pueden ocurrir las siguientes fallas:

- No se reciben los mensajes $PING$ y $PONG$, figuras 4.6a y 4.6b.
- El nodo *WimSN* está capturando o almacenando su imagen en el momento en que el *sink* envía el mensaje $PING$, figura 4.6c. En este intervalo el *WimSN* tiene su radio apagada por lo que no recibirá el mensaje.

Para recuperarse de estas fallas el protocolo establece que el *sink* retransmita el mensaje $PING$. Si se alcanza el máximo número de retransmisiones sin obtener respuesta, el *sink* da por perdido el enlace con el nodo consultado y con los nodos ubicados aguas arriba, teniendo en cuenta la topología lineal, figura 4.6d. A continuación da aviso al PC de cuáles nodos *WimSN* respondieron, y comienza la etapa de envío de imágenes con los nodos alcanzados.

En caso de que no se tenga conectividad con el nodo más cercano, *WimSN2*, el *sink* consulta al PC el tiempo para el próximo ciclo y entra al estado de *stand-by* [2].

Los nodos que no lograron conexión con el *sink* continúan activos esperando nuevas órdenes que llegarán en el próximo ciclo.

ii) Fallas en Envío y recepción de imagen

Durante la etapa de Envío y recepción de imágenes pueden ocurrir las siguientes fallas:

- Pérdida del mensaje ASK_PIC. Luego del chequeo de conectividad los nodos *WimSN* quedan a la espera de este mensaje para comenzar a enviar su imagen, y hasta no recibirlo no efectúan ninguna acción. Para recuperarse de esta falla el protocolo establece que el *sink* retransmita el ASK_PIC. Una vez agotada la cantidad de reenvíos, se determina que no hay conectividad y se continúa con el siguiente nodo aguas abajo. Si esto ocurre con el nodo *WimSN2*, el *sink* consulta al PC el tiempo para el próximo ciclo y entra al estado de *stand-by*. Esta falla se muestra en la figura 4.7b.
- Pérdida del mensaje END_STREAM. El *sink* espera recibir el mensaje END_STREAM para chequear que el *stream* recibido sea el esperado y para verificar su completitud. Si no recibe este mensaje en una ventana de tiempo da por perdida la conexión con el nodo consultado y continúa con el nodo siguiente. Para recuperarse de esta falla el protocolo establece que los nodos *WimSN* retransmitan el *stream* entero y el mensaje END_STREAM. Alcanzado el máximo número de retransmisiones el

[2] El nodo sink no tiene requerimientos de bajo consumo por tener alimentación desde el PC al que está conectado.

nodo interpreta que no tiene conectividad y vuelve al inicio a esperar un nuevo ciclo. Esta falla se muestra en la figura 4.7a.

- El *sink* recibe mensaje END_STREAM pero el *stream* recibido no es el esperado. En este caso el *sink* envía el mensaje $STREAM\_WRONG$, agregando el número de *stream* esperado para que el nodo *WimSN* corrija y envíe el *stream* correcto. Este procedimiento se detalla en la figura 4.7c. Esto puede verse en la figura 4.4.6.

- El *sink* recibe mensaje END_STREAM pero el *stream* recibido está incompleto. Cuando el *sink* recibe un END_STREAM verifica que el *stream* esté completo. Para esto recorre el buffer de recepción chequeando que cada paquete haya sido recibido. Cuando encuentra un paquete faltante envía al nodo *WimSN* el mensaje ASK_PAQ indicando cuál es el paquete faltante. El nodo *WimSN* responde con el paquete correspondiente seguido de un END_STREAM, como se muestra en la figura 4.7d.

- El nodo *WimSN* no recibe respuesta luego de enviar END_STREAM. Luego de enviar un mensaje END_STREAM un nodo *WimSN* inicializa un *timer* de retransmisión. Si una vez expirado este *timer* no se ha recibido respuesta del *sink* el nodo vuelve a enviar el *stream* junto con el mensaje END_STREAM. Alcanzado el máximo número de retransmisiones el nodo interpreta que no tiene conectividad y vuelve al inicio a esperar un nuevo ciclo. Las posibles respuestas del sink son las descritas en la sección 4.4.6. Como ejemplo se muestra en la figura 4.7c el caso en que se pierde un mensaje $STREAM\_OK$.

iii) Fallas en Envío a bajo consumo

Durante la etapa de Envío a bajo consumo las fallas que pueden ocurrir son:

- Un nodo *WimSN* no recibe el mensaje OFF. Luego de que un nodo *WimSN* envía el mensaje END_PIC, queda a la espera del mensaje OFF, con la indicación del tiempo por el cual tiene que ir al bajo consumo. Si este mensaje no es recibido dentro de un tiempo fijado por un timer de reenvío, el nodo *WimSN* reenvía END_PIC. La cantidad de reenvíos es limitada por un contador, de forma tal que si no se recibió OFF el nodo *WimSN* vuelve al inicio para esperar los comandos del *sink* en una nueva ejecución del sistema.

- El nodo *sink* no recibe el mensaje ACK_OFF. Cuando el *sink* envía el mensaje OFF inicia un timer de reenvío y espera como respueta el mensaje ACK_OFF. Si vencido dicho *timer* el *sink* no ha recibido respuesta, retransmite su mensaje. Una vez alcanzado el máximo de retransmisiones sin obtener respuesta el *sink* continua con el siguiente nodo o, en caso de que haya llegado al final del ciclo, consulta al PC el tiempo para entrar en *stand-by*. Esta falla no interfiere en el funcionamiento de la red, dado que lo relevante en este caso es que el nodo *WimSN* recepcione el mensaje OFF e ingrese al modo de bajo consumo.

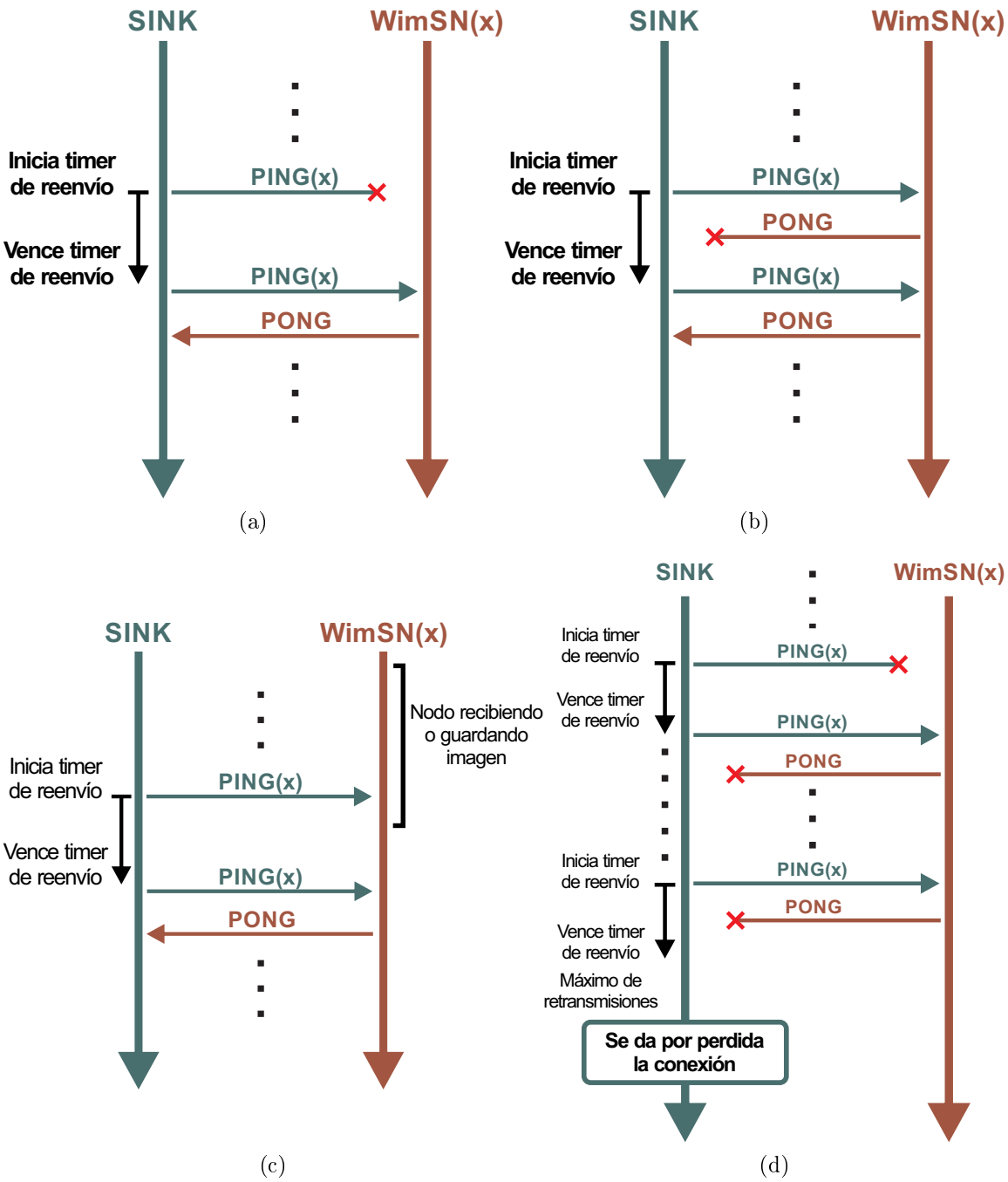


Figura 4.6: Fallas en etapa de Chequeo de conectividad. (a) Pérdida de mensaje $PING$. (b) Pérdida de mensaje $PONG$. (c) Envío de mensaje $PING$ durante toma de imagen. (d) Cantidad de reenvíos máxima del mensajes $PING$ sin recepción respuesta

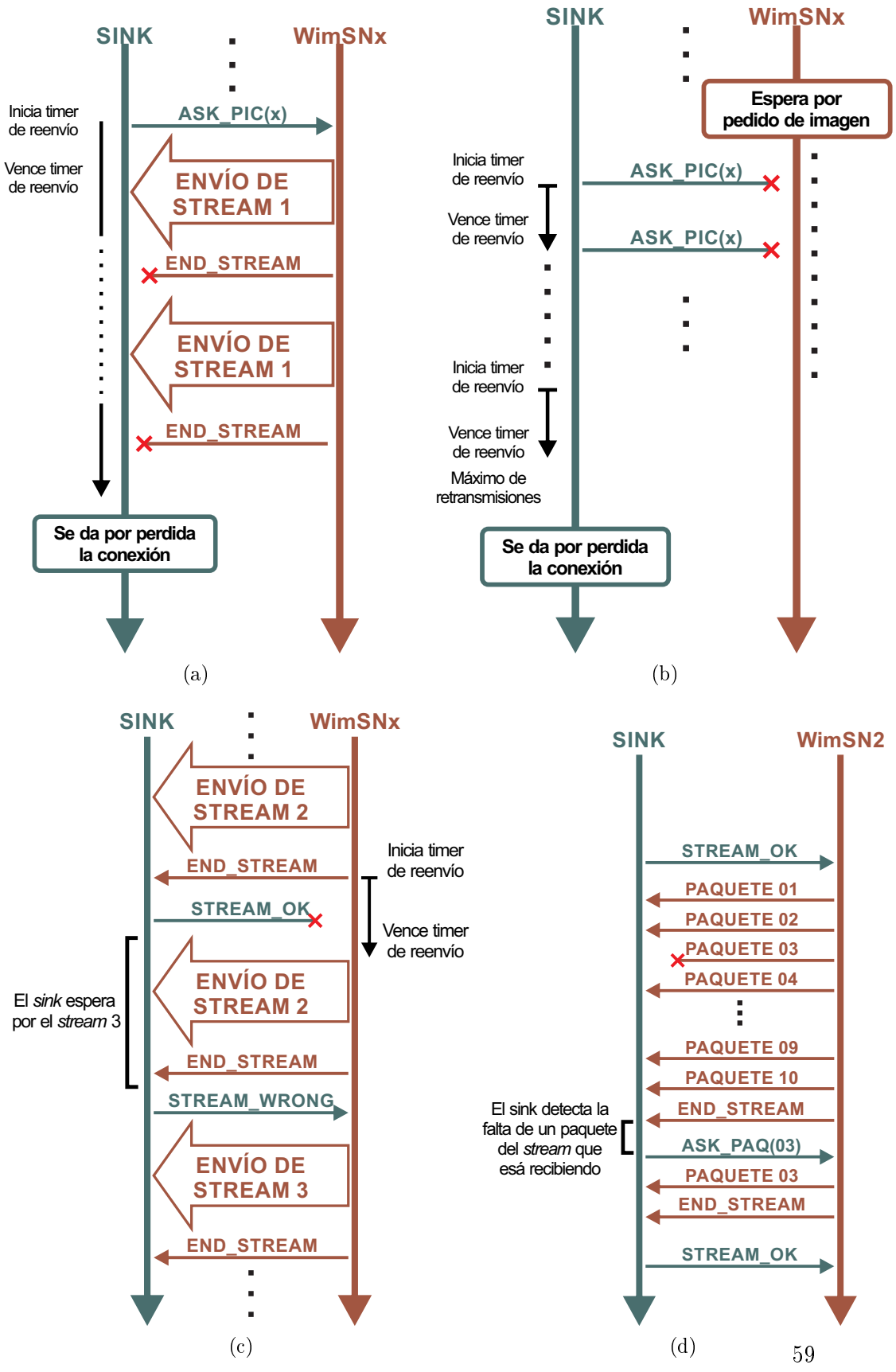


Figura 4.7: Fallas en etapa de Envío a bajo consumo. (a) Pérdida de mensaje END_STREAM. (b) Pérdida de mensaje ASK_PIC. (c) Pérdida de mensaje $STREAM\_OK$, y recepción de stream incorrecto. (d) Pérdido de un paquete

Capítulo 5

Software

5.1. Resumen

Este capítulo consta de tres secciones. En la primer sección se exponen conceptos básicos del sistema operativo Contiki OS. Luego se presenta el diseño de los módulos de software para el manejo de los nodos de la red y por último la implementación de estos módulos. En la tercer sección se especifica el procedimiento para la extensión del número de nodos de la red.

5.2. Contiki OS

Contiki OS [2, 15, 17], es el sistema operativo utilizado en la programación de los nodos. Se trata de un sistema operativo en tiempo real (RTOS) especialmente desarrollado para sistemas con pocos recursos de procesamiento y memoria. Es un RTOS de código abierto que utiliza lenguaje de programación C, y se encuentra portado para diferentes plataformas, entre las que se encuentra el SoC cc2538.

Contiki OS posee un *kernel* que implementa un modelo del tipo *event_driven* [13], con funcionalidades de *Multithreading*. Se basa en la idea de tener varios hilos de ejecución en paralelo, denominados procesos, que esperan ciertas condiciones para activarse. Una vez que se dan esas condiciones, llamadas eventos, los hilos de ejecución realizan las tareas pertinentes y finalizan o quedan a la espera de otro evento que los desencadene.

A continuación se enumeran conceptos básicos de Contiki OS que permitirán entender el código implementado.

- Procesos

  Los procesos son la estructura utilizada por Contiki OS para simular la multitarea a partir de la ejecución de varios hilos en paralelo. Son creados por el usuario para desarrollar tareas específicas. Cada proceso tiene un único *protothread* asociado, que se ejecuta cuando el proceso es llamado, y un bloque de control. Los estados de los procesos pueden ser desactivado, activado y llamado. Un proceso está en estado llamado cuando se está corriendo su *protothread*. Está activado cuando puede ser

llamado (mediante eventos) y está desactivado cuando ningún evento puede despertar al *protothread*.

Un *protothread* [13] es una función en C que permite estructurar código de forma de permitir al sistema realizar otras actividades mientras parte del código está esperando que algo suceda. Para los procesos, Contiki define diferentes macros de estados de *protothreads*:

- $PROCESS\_BEGIN()$ Declara el comienzo del código de un proceso
- $PROCESS\_END()$ Declara el fin del código de un proceso.
- $PROCESS\_EXIT()$ Finaliza el proceso.
- $PROCESS\_WAIT\_EVENT()$ *o* $PROCESS\_YIELD()$ Espera por algún evento
- $PROCESS\_WAIT\_EVENT\_UNTIL()$ Espera por un evento pero bajo determinada condición
- $PROCESS\_PAUSE()$ Pausa el evento temporalmente.

- Eventos

  Los eventos provocan la ejecución de un proceso pudiendo ser síncronos o asíncronos. Los primeros, al ser posteados son directamente dirigidos al proceso que se desea ejecutar, siendo su posteo similar a la llamada de una función a través de la sentencia *process_post_sync*(). Los eventos asíncronos son almacenados en una cola circular y procesados en forma cronológica un tiempo después, utilizándose la sentencia *process_post*() para desencadenarlos. Los eventos pueden ser ocasionados por hardware a través de interrupciones o software y siempre tendrán asociados un proceso a despertar.

- Timers

  Contiki OS ofrece una serie de librerías de timers que son usadas tanto por las aplicaciones programadas por el usuario como por el propio sistema operativo. Con estas librerías es posible chequear el paso de periodos de tiempo, sacar de modos de bajo consumo al sistema y realizar conteos de tiempo real.

  Las mismas son: timer, stimer, ctimer, etimer y rtimer.

  Las librerías timer y stimer son usadas para chequear el paso de un lapso de tiempo determinado. La diferencia entre ambos timers está en la resolución de tiempo. El primero utiliza los ticks del clock del sistema, mientras que el segundo tiene una resolución de segundos, pudiéndose setear periodos mas largos de tiempo.

  Las librería de etimer provee de timers por eventos, utilizados para esperar por un determinado tiempo desencadenando luego un post de un evento.

  La librería ctimer refiere a timers por callbacks: luego de que expira un determinado tiempo, se da paso a una función callback que ejecuta determinadas instrucciones.

  Por último, la librería rtimer es de suma utilidad para aplicaciones en las que se requiera de acciones en tiempo real. Permite la medida de largos intervalos de tiempo con precisión y la salida del modo de bajo consumo de los nodos sensores.

- Escritura, lectura y borrado de memoria flash.

  Con el fin de manejar escrituras y lecturas en memoria *flash*, Contiki provee de una API y una extensión de la misma denominadas Contiki Fyle System (CFS) y CFS-Coffee, respectivamente. Estas herramientas no se encuentran portadas para el nodo cc2538dk.

  En su defecto, existe el módulo *rom-util.h* que permite la lectura, escritura y borrado de flash. Cabe mencionar que las funciones presentes en este módulo tienen ciertas restricciones tales como la cantidad de bytes posibles de borrado por iteración (32768 bytes) y que el control de no escritura o borrado de lugares de interés para el sistema en flash las debe realizar el programador. Estas consideraciones serán detalladas en la página 64.

  Las funciones que fueron de utilidad en este proyecto son:

  - $uint32\_t\ rom\_util\_page\_erase(uint32\_t flash\_addr, uint32\_t size)$
    Función para el borrado de memoria *flash*. Recibe como parámetros la dirección de memoria flash ($flash\_addr$) a partir de la cual se desea borrar y la cantidad de bytes ($size$) a borrar.
  - $uint32\_t\ rom\_util\_program\_flash(uint32\_t\ *ram\_data, uint32\_t\ flash\_addr, uint32\_t\ byte\_count)$
    Función para la escritura en flash. Recibe un puntero a los datos que se quieren escribir ($*ram\_data$) y el tamaño de datos en bytes y la dirección de memoria flash donde se desea realizar la escritura ($flash\_addr$).
  - $uint32\_t\ rom\_util\_memcpy(void * dest, const void * src, size\_t n)$
    Función para lectura de memoria flash.

- Stack de protocolos de Comunicación

  Contiki OS provee de dos grupos de protocolos de red: Rime y uIP. Con uIP se puede utilizar protocolos de capa de transporte como TCP y UDP adapatados para sistemas de bajos recursos de hardware. Rime posee primitivas de red como unicast, multicast y broadcast. En este proyecto, se utilizan las primitivas del módulo unicast. Este provee funciones para envío de paquetes a un dispositivo cuya dirección rime es conocida. La comunicación es single-hop y a priori no se realizan controles de secuencia. Más detalles de este stack se encuentran en el apéndice C.

5.3. Diseño

A modo de cumplir con los requerimientos del sistema, el software se dividió en 9 módulos: *project-conf.h*, *nodo.c*, *rime-stream.c*, *radio.c*, *driverCamara.c*, *dormir.c*, *flash.c*, *sink.c* y *wsn-vision.py*.

- *project-conf.h* es el archivo de configuración del proyecto en Contiki OS. Aquí se configuran opciones referidas al bajo consumo y ciclo de la radio. Además, se configuran parámetros propios del software desarrollado tales como constantes y timers.

- *rime-stream.c* es el que maneja la comunicación por radio en los nodos *WimSN*, enviando la imagen e interpretando los mensajes enviados por el *sink*.
- *driverCamara.c* contiene las funciones y procesos necesarios para la comunicación con la cámara LS-Y201-2MP y adquisición y almacenamiento de la imagen en memoria flash del nodo *WimSN*.
- *nodo.c* es el bloque principal para los nodos *WimSN*, siendo el que gestiona la ejecución de los módulos *rime-stream.c* y *driverCamara.c*, pidiendo la toma y envío de la imagen cuando el *sink* lo ordena.
- *sink.c* es el módulo que utiliza el nodo *sink* para la comunicación por RF con los nodos *WimSN*, almacenamiento de imágenes y comunicación con el PC.

Los módulos *radio.c*, *dormir.c* y *flash.c* son de uso común para nodos *WimSN* y *sink.c*. El primero contiene las funciones necesarias para envío de datos por RF. El módulo *dormir.c* nuclea las funciones para enviar al nodo *WimSN* a bajo consumo y configurar tiempos de espera para comienzo del ciclo de toma y envío de imágenes. El bloque *flash.c*, se utiliza para el borrado, escritura y lectura de memoria flash.

Por último, se tiene el módulo *wsn-vision.py*, desarrollado en leguaje *Python* [9] que se ejecuta desde el PC conectado al nodo *sink*. Este programa implementa:

- La comunicación vía UART con el nodo *sink*.
- Interactúa con el usuario permitiendo configurar el número de capturas de imágenes diarias por nodo *WimSN* y el horario en que se realizan las mismas.
- En base al punto anterior, calcula los tiempos de envío a bajo consumo para los nodos *WimSN* luego de cada captura, y el tiempo de pedido de imágenes por parte del *sink*. De esta forma se tiene sincronismo en la red.
- Recibe las imágenes provenientes de los nodos *WimSN* agregando un *time-stamp* y dirección ID del nodo emisor, almacenándolas en formato JPEG.
- Envía las imágenes recibidas por correo electrónico a casillas configuradas por el programador.

5.3.1. Almacenamiento de la imagen

Tanto en el nodo *sink* como en los nodos *WimSN*, la imagen se almacena en la memoria flash interna de tamaño 512 kB del SoC CC2538 [35]. Esta última se encuentra dividida en páginas de 2048 bytes comenzando en la dirección de memoria 200000h y finalizando en 27FFFFh. La página 255 es la llamada *Lock Bit Page* y contiene los bits de bloqueo utilizados para impedir escribir o borrar determinada sección de memoria. Esta página no debería estar accesible, a menos que el registro $FLASH\_CTRL\_FCTL.UPPER\_PAGE\_ACCESS$ del microcontrolador se encuentre seteado.

Este procedimiento de almacenamiento parece ser innecesario en el caso de los nodos *WimSN* ya que a primera vista, podría enviarse la imagen por RF a medida que es pedida a

la cámara. Previendo la posibilidad de procesamiento de la imagen para conteo automático de polillas en futuros PFC, se decidió dejar resuelto el almacenamiento en flash para facilitar la aplicación de algoritmos. Por otro lado, dado que el conversor utilizado en la placa Nodo-Interfaz-Iluminación (ver capítulo 2) soporta una corriente máxima de 300 mA, tener la cámara y radio encendidas a la vez podría dañar a este integrado (observar tabla 2.1) además de ocasionar caídas de tensión en las baterías por picos de corriente y aumentos del consumo de carga.

El nodo TelosB utilizado en Plagavision, posee una memoria flash externa cuya gestión está resuelta por Contiki OS a través de la API CFS y su extensión CFS-Coffee. Estas herramientas proveen un set de funciones de uso sencillo para la escritura, lectura y borrado de memoria flash. Sin embargo, CFS y CFS-Coffee no se encuentran portados para el SoC CC2538 al momento presentar esta documentación. Luego de investigar y conectarse con uno de los desarrolladores del nodo OpenMote [1], se llegó al conocimiento de la existencia del módulo $rom - util.h$, especialmente desarrollado para estas funcionalidades.

Según el manual de usuario del SoC CC2538, es necesario borrar el sector de flash que se desea escribir posteriormente. El borrado es la única operación que setea en '1' los bits de la sección de trabajo. Al querer escribir una determinada palabra, el SoC realiza la operación AND entre el contenido de flash y la palabra que se desea escribir. Es así que los bits de memoria flash de valor '0', no se pueden setear en '1' sin un borrado previo.

A través de diferentes pruebas para conocer las posibilidades y restricciones de $rom - util.h$, se observó que no es posible borrar una cantidad mayor a 32 kB en una sola iteración. Si ésto no se respeta, el microcontrolador se resetea.

Vale considerar además, que la mínima unidad de borrado de memoria flash es 2 kB (una página), aunque las funciones utilizadas son capaces de realizar el cálculo del número de páginas en función de la cantidad de bytes que el usuario desee borrar.

Debe además tenerse en cuenta que la cantidad de bytes que se desea escribir o leer de flash, debe ser múltiplo de 4. Estas consideraciones fueron tomadas en la implementación del módulo *flash.c*.

5.3.2. Driver de la cámara

Es el módulo encargado de la toma de imágenes y fue reutilizado del proyecto Plagavision. Lleva a cabo la comunicación entre la cámara y el nodo *WimSN* vía UART. Se mantiene la estructura del código, realizándose cambios en la comunicación con la cámara y manejo de almacenamiento de la imagen en memoria flash. Esto se debe a que la cámara y nodo elegidos en Plagavision difieren de los utilizados en este proyecto. A continuación se describen estos cambios.

- *Almacenamiento de la imagen en memoria flash.*

  Tal como se expresa en 5.3.1, se utiliza el módulo *flash.c* implementado, por no disponer de CFS-Coffee.

- *Cambio de palabras de control.*

[1] Plataforma utilizado en primeras etapas del proyecto. Utiliza el SoC CC2538. Ver apéndice E

Se cambian las palabras de control de la cámara que difieren de las utilizadas en Plagavision. Es importante mencionar que la hoja de datos de la cámara LS-Y201-2MP presenta errores, siendo necesario corregir algunas de las palabras que allí se indican, tal como se detalla en el apéndice B.

- *Modificación de código para recibir imágenes de mayor tamaño.*

  La resolución máxima de la cámara utilizada en Plagavision es 640x480 pixeles (con imágenes de tamaño menor a 64 kB), mientras que en esta cámara es de 1600x1200 pixeles (con tamaño de imágenes que pueden superar fácilmente los 100 kB). Para enviar tamaños de imagen superiores a 64 kB, la cámara utiliza 3 bytes, en lugar de 2. Se modificó el código para el cálculo de tamaño de imagen y tipo de variable en que se almacena este tamaño [2].

- *Presentación de bytes para cambiar compresión y resolución en el archivo project-conf.h*

  Se dejaron accesibles las constantes *COMPRESION* y *RESOLUCION* en el archivo de configuración del sistema para poder variar las mismas sin ingresar en el código del driver.

- No se envía el comando para cambio de *baudrate* de la cámara, partiendo de la base que si ésta tiene un *baudrate* distinto al del nodo, no se puede tener comunicación. Tanto la cámara como el sistema operativo Contiki OS fijan por defecto un baudrate de 115200 bps, optándose por utilizar este valor en la aplicación.

La comunicación con la cámara se realiza vía UART. En la tabla 5.1 se detallan los comandos y ACKs intercambiados con esta cámara. El ACK del comando RESET se obtuvo experimentalmente ya que el que indica la hoja de datos es erróneo.

Todos los comandos comienzan con la secuencia "56h,XXh", donde XXh es por defecto 00h e indica el número ID de la cámara. Cualquier otra secuencia no es reconocida por la cámara. Ante un comando válido, exceptuando el pedido de reseteo, la cámara envía un ACK que comienza con la secuencia "76h, XXh". Si se recibe un comando inválido, la cámara responde reenviando lo recibido. En la sección 5.4.1 se detallan los diagramas de flujo que explican el funcionamiento del driver.

[2] La variable era del tipo $uint16\_t$, modificándose a $uint32\_t$

Comando	Descripción	ACK de comando
56h, 00h, 26h, 00h	Reset de cámara/Init End	0Dh, 00h, 49h, 6Eh, 69h, 74h, 20h, 65h, 6Eh, 64h, 0Dh, 0Ah
56h, 00h, 36h, 01h, 00h	Tomar foto	76h,00h, 36h, 00h, 00h
56h, 00h, 36h, 01h, 03h	Parar de tomar foto	76h, 00h, 36h, 00h, 00h
56h, 00h, 31h, 05h, 01h, 01h,12h, 04h, XXh	Factor de compresión	76h, 00h, 31h, 00h, 00h
56h, 00h, 54h, 01h, XXh	Resolución	76h, 00h, 54h, 00h, 00h
56h, 00h, 24h, 03h, 01h, XXh	Baudrate	76h, 00h, 24, 0x00, 0x00
56h, 00h, 34h, 01h, 00h	Tamaño de imagen	76h, 00h, 34h, 00h, 04h, 00h, HHh, MMh, LLh
56h, 00h, 32h, 0Ch, 00h, 0Ah, 00h, MHH, MH ,ML, 00h, KHH, KH, KL , XH, XL	Leer bloque de memoria	76h, 00h, 32h, 00h, 00h, ..*data*.., 76h, 00h, 32h, 00h, 00h

Tabla 5.1: Comandos y ACKs para comunicación con la cámara LS-Y201-2MP.

5.3.3. Comunicación por RF

Los respectivos módulos para *WimSN* y *sink* que involucran la transmisión por radio, tienen como objetivo manejar la comunicación entre éstos, transmitiendo las imágenes y realizando la administración de la red. La comunicación se realiza a través del stack Rime, y es multihop, especialmente diseñada para una topología de red lineal.

Se diseñó un protocolo de comunicaciones que permite la correcta administración de la red, previendo mensajes para envío de imágenes, reenvío de paquetes perdidos, envío de tiempos para ir a bajo consumo y chequeo de conectividad. El mismo se detalla en el capítulo 4.

La comunicación por RF debe ser tal que en caso de existir conectividad entre un nodo *WimSN* y el *sink*, la imagen se reciba de forma íntegra. Además se debe lograr realizar el intercambio de mensajes para enviar a bajo consumo al nodo *WimSN* hasta el momento de la próxima captura. Esto se logra a través de la implementación de controles de secuencia y confirmación de recepción de mensajes cruciales para la comunicación.

La imagen es enviada en ráfagas de paquetes de 100 bytes, delimitando el número por ráfaga para facilitar los controles. Las ráfagas se envían de a bloques denominados *streams*. [3] Cada paquete posee, entre otros datos, su número de secuencia y el número de *stream* al que pertenece. Con esta información, el nodo *sink* determina si el *stream* que recibió es el esperado y si además no hubo pérdida de paquetes. En caso de no ser el *stream* esperado, el *sink* le informa al nodo y pide reenvíos si existieron pérdidas en la transmisión.

Los nodos *WimSN* son capaces de distinguir si los paquetes que circulan en la red son para ellos o no. En caso negativo, actúan como router, reenviándolos al nodo que

[3] El número de paquetes por *stream* es configurable por el usuario en el archivo de configuración del sistema. Ver 5.4.8

corresponda.

Cabe destacar que si bien *WSNvision* implementó una red de cinco nodos, el aumento de la cantidad de nodos *WimSN* resulta transparente para el usuario, como se describe en la subsección 5.4.10.

5.3.4. Módulo de bajo consumo de carga y configuración de timers

Los nodos *WimSN* deben permanecer activos solo en el momento de la toma de la imagen y posterior envío. Dado que los mismos se alimentan de baterías, resulta imprescindible la reducción del consumo. La utilización de modos de bajo consumo contribuye a este hecho.

La arquitectura del SoC CC2538 permite implementar cuatro estados de bajo consumo, llamados PM0, PM1, PM2 y PM3, y ordenados de mayor a menor consumo [35, 36]. Los órdenes de consumo de corriente brindados por el fabricante son los siguientes:

1. PM1, 24 mA.
2. PM1, 0.6 mA.
3. PM2, 1.3 uA.
4. PM3, 0.4 uA

Del estado de menor consumo (PM3) solo se puede salir por interrupciones por hardware (por flanco en pines) ya que todos los timers permanecen inactivos [35]. La forma de salir de los modos PM0 a PM2 es por interrupciones por hardware (por flanco en pines) ó eventos de rtimers ya que el reloj de 32 kHz permanece en funcionamiento en estos estados.

Contiki OS implementa un algoritmo que al habilitar el ingreso al bajo consumo, gestiona el modo al que se ingresará, optando entre PM0, PM1 y PM2. La gestión se realiza en base a criterios como:

1. Si la radio permanece encendida.
2. Si los periféricos permiten el ingreso a PM1 o PM2.
3. Tiempo estimado en que el sistema permanece en bajo consumo.

Contiki OS permite al programador configurar el máximo modo de bajo consumo al que se desea llegar. En esta aplicación, se desea el mínimo consumo durante el tiempo en que el nodo *WimSN* permanece inactivo.

El módulo dormir, implementa funciones para el ingreso a bajo consumo del nodo *WimSN* y seteo del rtimer utilizado para salir de este estado. A su vez, este módulo se utiliza para configurar un rtimer en el nodo *sink* que permita dar aviso del momento en que el *sink* deba comenzar a pedir las imágenes. Los rtimers son configurados con tiempos calculados por el PC que permiten que la red esté sincronizada.

5.3.5. Comunicación PC-*sink*

Con el fin de que el usuario del sistema pueda configurar la frecuencia de captura de imágenes y visualización y almacenamiento de las mismas, se tiene el módulo *wsn-vision.py* desarrollado en lenguaje Python.

wsn-vision.py posee cinco funcionalidades principales:

- Interpretación de órdenes del usuario para configurar cantidad de imágenes, horario y frecuencia de toma de las mismas.
- Cálculo de tiempos de toma de imágenes según la hora actual para que el *sink* pueda enviar a los nodos y la red permanezca sincronizada.
- Cálculo de tiempo que debe esperar el nodo *sink* para comenzar a pedir las imágenes según hora del PC.
- Recepción de los archivos de imágenes y etiquetado de las mismas.
- Almacenamiento de las imágenes en el PC y posterior envío vía mail a casillas de correo configurables por el programador.

El etiquetado de las imágenes se realiza colocando el horario y fecha de recepción de la imagen y el nodo del cual proviene.

5.4. Implementación

Como se mencionó en la sección 5.3, el software se dividió en 9 módulos:

- *project-conf.h*: Archivo de configuración del sistema.
- *nodo.c*: Módulo principal para los nodos *WimSN*.
- *rime-stream.c*: Módulo para envío de imagen y comunicación por RF. Utilizado por los nodos *WimSN*.
- *sink.c*: Módulo del nodo *sink*. Encargado de comunicación por RF con nodos *WimSN*, almacenamiento de imágenes y comunicación con PC.
- *radio.c*: Módulo auxiliar utilizado por *sink* y nodos *WimSN* con funciones para envío de paquetes y mensajes.
- *flash.c*: Módulo auxiliar utilizado por *sink* y nodos *WimSN* para borrado, lectura y escritura en la memoria flash del nodo cc2538dk.
- *dormir.c*: Módulo utilizado por *sink* y nodos *WimSN* para configuración de rtimers y manejo de modos de bajo consumo.

El sistema posee dos tipos de nodos:

- Nodos *WimSN*: Nodos que tienen capacidad visual, encargados de tomar imágenes y enviarlas al *sink* cuando éste lo ordene.
- Nodo *sink*: Es el nodo conectado al PC vía UART que pedirá las imágenes y las enviará al PC. Enviará a los nodos el tiempo en que deben ingresar en bajo consumo hasta la toma de una nueva imagen.

En la figura 5.1 se observa un diagrama de interacción de los diferentes bloques de software con los módulos de Contiki OS y el hardware involucrado para el nodo *WimSN*. A continuación se detallan los diferentes módulos implementados.

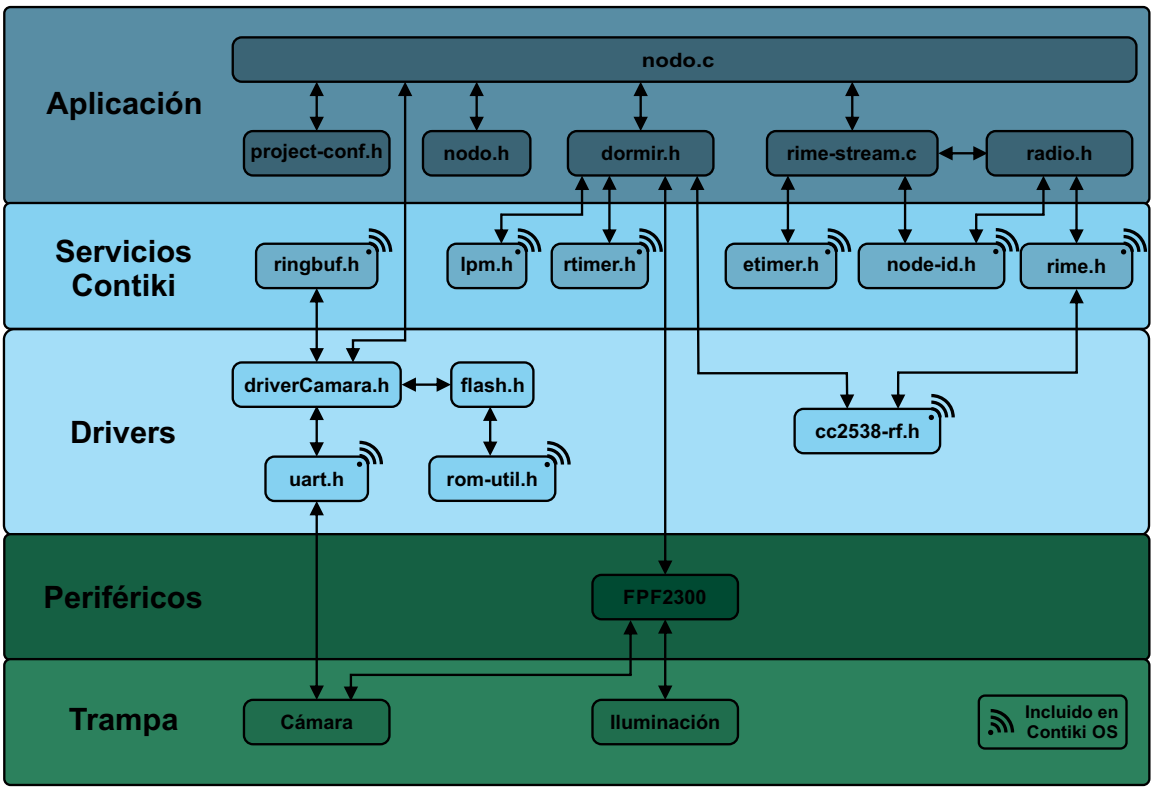


Figura 5.1: Diagrama de capas para el nodo *WimSN*

5.4.1. Módulo *driverCamara.c*

El módulo *driverCamara.c* implementa el driver de comunicación con la cámara LS-Y201-2MP. Consta de tres procesos: *comando_cam*, *camara* y *foto* encargados de envío de comandos, inicialización y toma y almacenamiento de imagen en memoria flash, respectivamente. Los comandos y ACKs intercambiados con la cámara se pueden observar en la tabla 5.1.

El proceso *camara* es el que realiza la inicialización, según se observa en la figura 5.2. En primer lugar, se inicializa la UART para comunicación con la cámara y se envía el comando RESET. Luego, se envían los comandos para cambiar compresión y resolución, según las

constantes *COMPRESION* y *RESOLUCION* asignadas en el archivo de configuración del sistema (ver 5.4.8). Por último, se encienden los LEDs de iluminación y se espera un tiempo de dos segundos recomendado por el fabricante de la cámara, previo a la toma de imagen. Este tiempo permite además la estabilización de iluminación.

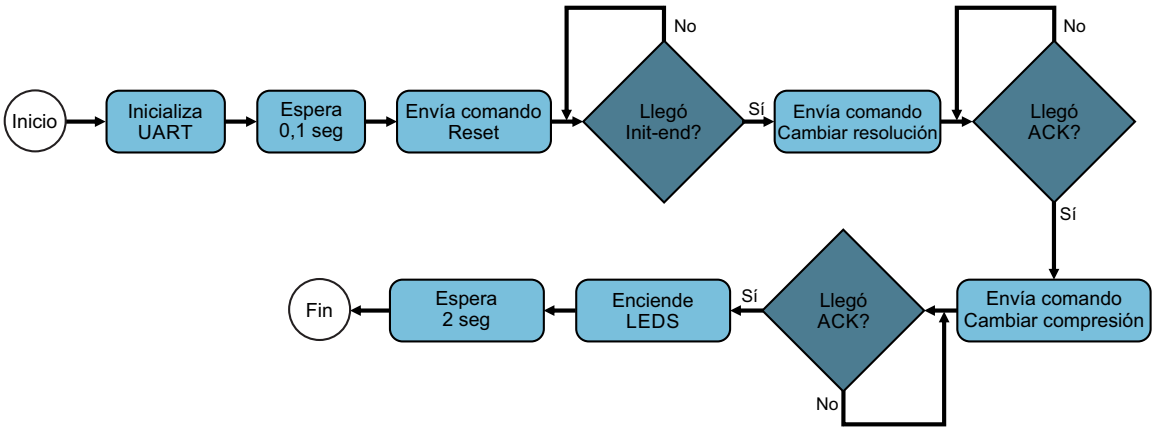


Figura 5.2: Diagrama de flujo del proceso *camara*

El proceso *foto* se encarga del pedido y recepción de la imagen, almacenándola en memoria flash. Un diagrama de flujo del proceso se puede observar en la figura 5.3.

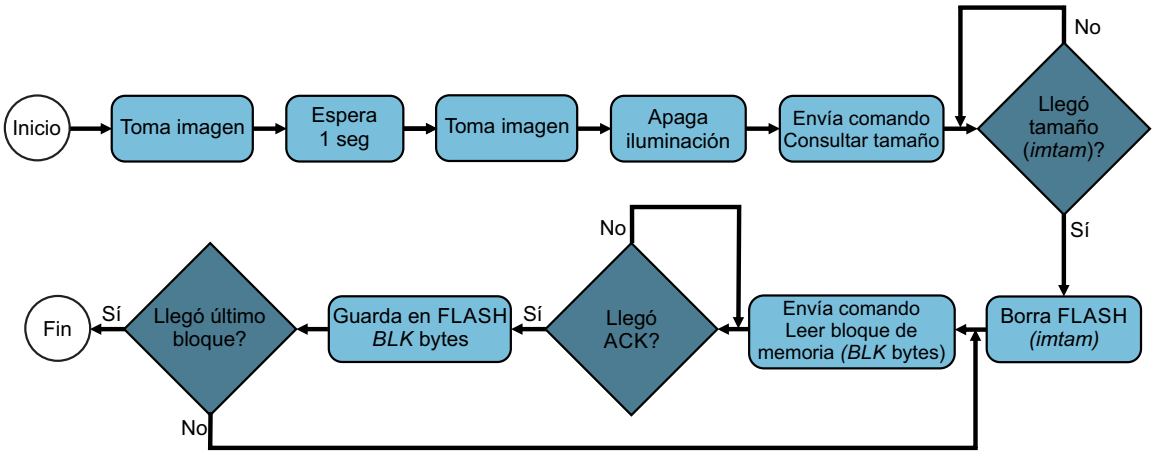


Figura 5.3: Diagrama de flujo del proceso *foto*

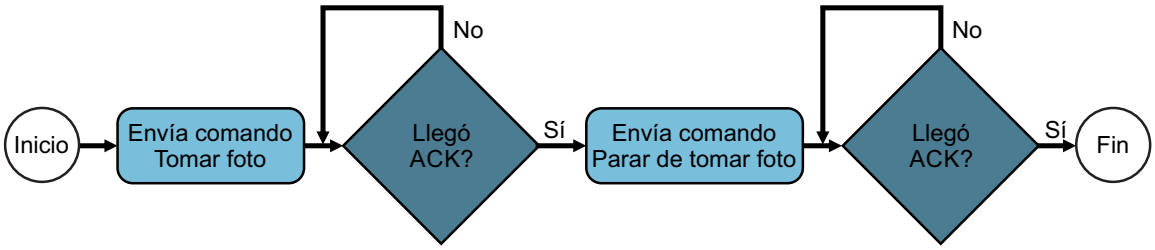


Figura 5.4: Diagrama de flujo de la toma de imagen en el proceso *foto*.

En laboratorio se comprobó que tomando la imagen dos veces consecutivas ésta mejora. No se tiene certeza sobre el motivo de este efecto, pero se estima que la cámara realiza un balance de blancos entre dos fotos consecutivas. Por ésto, se toma dos veces la imagen, desactivando luego el sistema de iluminación. Se envía el comando para consulta del tamaño de la imagen (*imtam*) capturada. La cámara responde con el ACK presentado en la figura 5.5, enviando los bytes HH, MM y LL que conforman el tamaño de la misma.

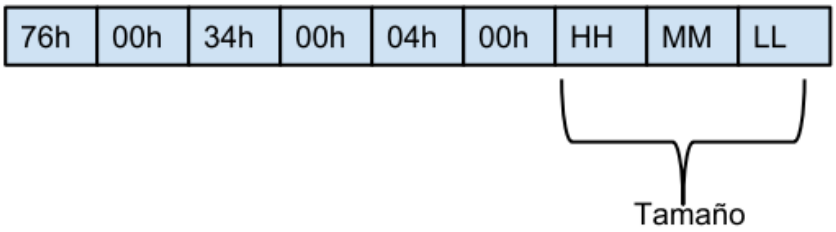


Figura 5.5: ACK al comando de pedido de tamaño de imagen.

Se borra la sección de memoria necesaria para almacenar la imagen, según el valor de *imtam*, utilizando la función $erase\_flash()$.

Luego, se envía el comando de lectura de bloques de memoria que se detalla en la figura 5.6.

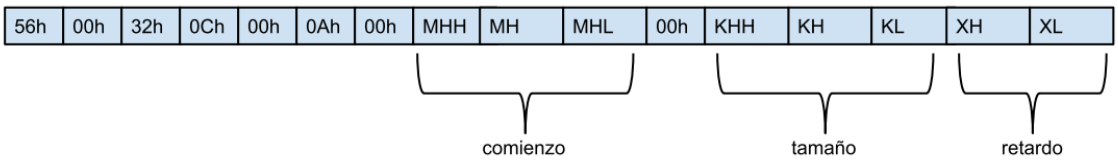


Figura 5.6: Comando de lectura de bloque de memoria.

Los bytes MHH, MH y ML conforman la dirección en memoria de la cámara desde donde se quieren leer los datos, comenzando en 00h 00h 00h. Los pedidos de imagen se realizan de

a bloques de 112 bytes [4], por lo que la dirección se incrementa en este valor entre un pedido y otro. KHH, KH y KL especifican el tamaño de bloque que se desea leer. En este caso, dado que se lee de a 112 bytes, sus valores son 00h, 00h y 70h respectivamente. Cada bloque es enviado entre un ACK inicial y final. XH y XL especifican el tiempo de envío entre los ACKs iniciales y finales y los datos. Estos ACKs coinciden y se corresponden con 76h 00h 32h 00h 00h.

Cada vez que se recibe un bloque, se almacena en memoria flash a través la función $write\_flash()$, actualizando el puntero de escritura en cada iteración.

El ciclo de lectura se realiza tantas veces como el resultado del cociente de la división entera entre *imtam* y el tamaño del bloque más una vez, ésto es

$$\frac{imtam}{112} + 1 \tag{5.1}$$

Por último, se tiene el proceso *comando_cam*, encargado de realizar el envío de comandos a la cámara. Este proceso no toma medidas de contingencia en caso de no recibir ACKs ó tener ACKs corruptos. Estas medidas deben tomarse como trabajo a futuro, ya que al tener fallas de comunicación con la cámara, el nodo *WimSN* permanece en un *loop infinito*.

5.4.2. Módulo *radio.c*

El módulo *radio.c* reúne las funciones necesarias para el envío de la imagen y de mensajes de control entre *sink* y nodos *WimSN* por RF. La imagen se envía en ráfagas de paquetes denominadas *streams*, mientras que la comunicación de órdenes y acciones entre nodo y *sink* se realiza a través de mensajes de control detallados en el capítulo 4. Estos mensajes son: $PING$, $PONG$, END_STREAM, $STREAM\_OK$, $STREAM\_WRONG$, ASK_PAQ, END_PIC, OFF y ACK_OFF.

Es en el archivo *header* de *radio.c* donde se enumeran estos mensajes, además de declararse la estructura *msg* para su envío y el de *streams*. *msg* incluye los siguientes campos:

- Número de stream y secuencia del paquete a enviar (si se trata de un mensaje estos campos no se toman en cuenta).
- ID del nodo destinario.
- ID del nodo emisor del mensaje o *stream*.
- Tipo de mensaje que se envía. Si se trata de un *stream*, se identifica con el número 200. En caso contrario, será uno de los mensajes antes enumerados.
- Datos a enviar.

Las funciones implementadas en *radio.c* son $send\_msj()$, $send\_stream()$ y $send\_pqtes()$:

- $send\_msj()$ envía los mensajes de control, recibiendo como parámetros el mensaje a enviar, el destino y datos. Esta función es utilizada por nodos *WimSN* y *sink*.

[4] constante BLK en código del driver

- *send_stream*() envía los paquetes de un *stream*. Recibe como parámetros el número de *stream* a enviar y un puntero a un arreglo en memoria RAM que contiene al *stream*.

- *send_pqtes*() se utiliza para reenviar un paquete de un determinado *stream*. Recibe como parámetros el número de *stream* al que pertenece el paquete, el número de secuencia del paquete y un puntero al lugar de memoria RAM donde se encuentran los datos a enviar.

Los *streams* y paquetes son enviados siempre desde un nodo *WimSN* hacia el *sink*, por lo que las funciones *send_stream*() y *send_pqtes*() enviarán los datos al nodo con ID inmediatamente menor (aguas abajo) e ID de destino 1 (hacia el *sink*).[5]

La función *send_msj*() es utilizada por nodos *WimSN* y *sink*. Por ésto, si la ID de destino es mayor que la de origen, significa que el *sink* está emitiendo un mensaje, y la función deberá enviarlo al nodo de ID 2 con la ID de destino correspondiente. En caso contrario, el emisor es el nodo *WimSN*, por lo que *send_msj*() enviará el mensaje al nodo de ID inmediatamente inferior con ID de destino 1 (aguas abajo). En el capítulo 4 se encuentran los detalles sobre la implementación de mensajes e interacciones entre nodo *WimSN* y *sink*.

5.4.3. Módulo *rime-stream.c*

rime-stream.c es el encargado del envío de la imagen y comunicación por RF en los nodos *WimSN*. Para la implementación de la comunicación por radio, se utilizó el módulo *unicast* del stack de protocolos Rime. Se implementó multihop especialmente para una topología lineal y un protocolo de transmisión de información el cual es detallado en el capítulo 4.

rime-stream.c consta de dos procesos:

1. *main_rime_process*, quien se encarga de abrir el canal de comunicaciones *unicast*.

2. *mandar_stream_process* que envía la imagen al nodo *sink*.

Se tiene además un *callback* propio del módulo *unicast* que se ejecuta al recibir datos por RF y la función *interpretar_msj*() quien realiza acciones específicas según el mensaje recibido. Vale aclarar que los mensajes tienen en su cuerpo la ID del nodo al que se desea enviar el mensaje (ID de destino) y la ID de origen.

Un diagrama del funcionamiento del *callback* de recepción se presenta en la figura 5.7. Este *callback*, al ejecutarse verifica qué tipo de datos se recibió. Si se trata de un paquete de un *stream*, significa que un nodo *WimSN* está enviando su imagen al *sink*, por lo que reenvía el paquete aguas abajo. En caso contrario, chequea si la ID de destino coincide con la del nodo en cuestión. Si existe coincidencia, será la función *interpretar_msj*() quien determine las acciones a tomar según el mensaje recibido. En otro caso, el *callback* reenviará al nodo de ID inmediatamente inferior o superior para destinos de ID menores o mayores a la propia respectivamente.

[5] Ver capítulo 4 por detalles en el funcionamiento y topología de la red.

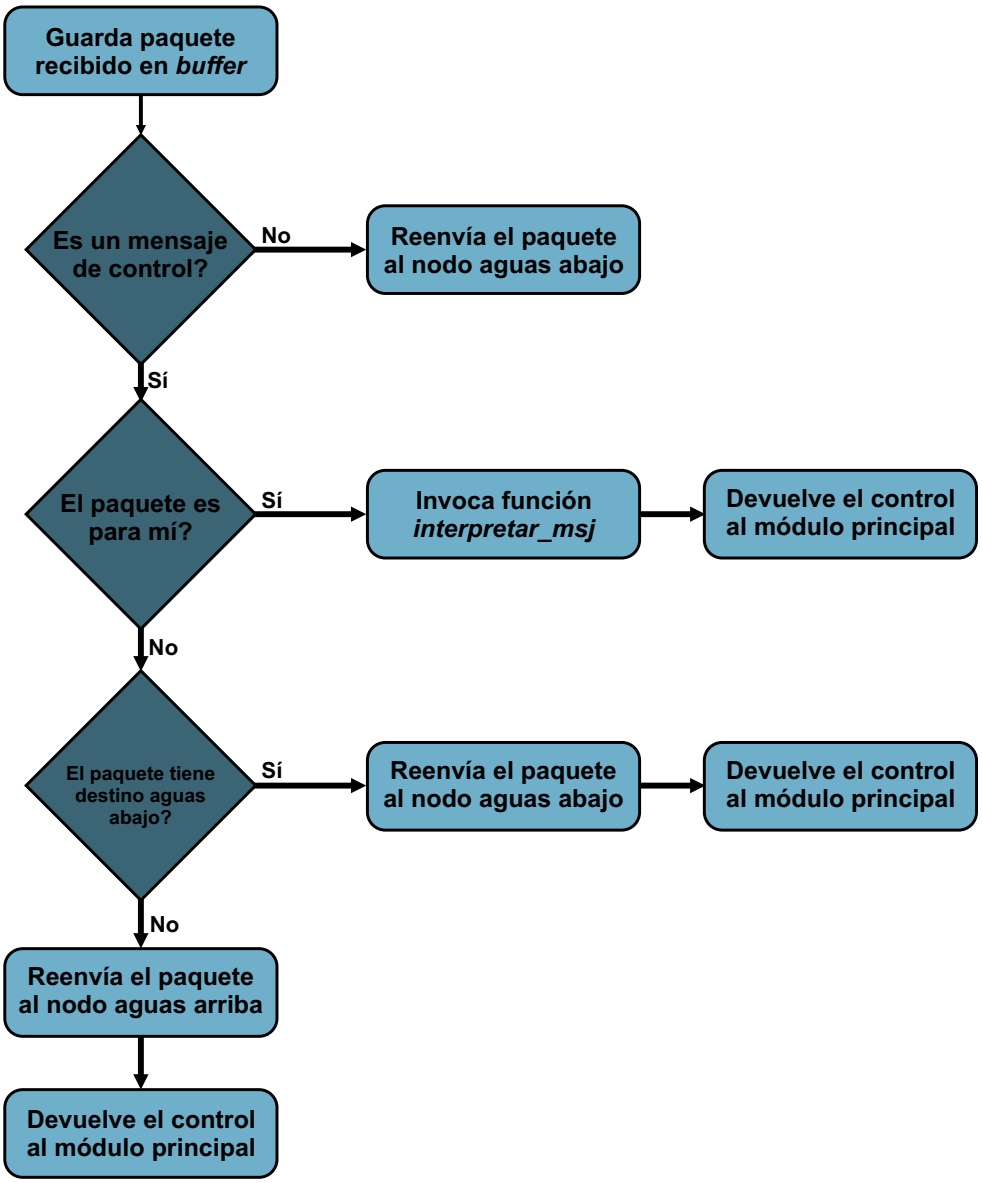


Figura 5.7: Función *callback* de recepción de los nodos *WimSN*

El proceso $mandar\_stream\_process$ en primer lugar realizará el cálculo de la cantidad de *streams* que debe enviar en base al tamaño de la imagen. Luego, comenzará con el envío. Cada vez que el nodo completa el envío de un *stream*, avisa al *sink* con el mensaje END_STREAM. Luego de ésto existen cuatro escenarios posibles:

- El nodo *sink* afirma que el *stream* fue recibido de forma correcta y por lo tanto el nodo *WimSN* enviará el siguiente *stream*.
- El nodo *sink* da aviso de la pérdida de un paquete, por lo que el nodo reenviará el mismo seguido del mensaje END_STREAM.

- El nodo *sink* indica que el *stream* recibido no se corresponde con el esperado, por lo que el nodo *WimSN* envía el *stream* pedido seguido del mensaje END_STREAM.

- No se reciben respuestas del *sink* por un determinado tiempo. El nodo reenvía el *stream*, quedando a la espera de alguno de los tres casos anteriores. Si se repite este escenario con un mismo *stream* por un determinado periodo de tiempo, se da por fallida la comunicación, reseteándose el nodo.

A menos que se dé por perdida la comunicación, el ciclo se repite hasta completar el envío de la foto, momento en el cual se da aviso al nodo a través del mensaje END_PIC.

Aquí se espera por el mensaje que indica el tiempo durante el cual el nodo debe ir a bajo consumo, terminando el proceso al recibirlo y dando aviso al módulo *nodo.c* a través de un *post*.

Un diagrama de flujo del envío de imágenes con el módulo *rime-stream.c* se muestra en la figura 5.8.

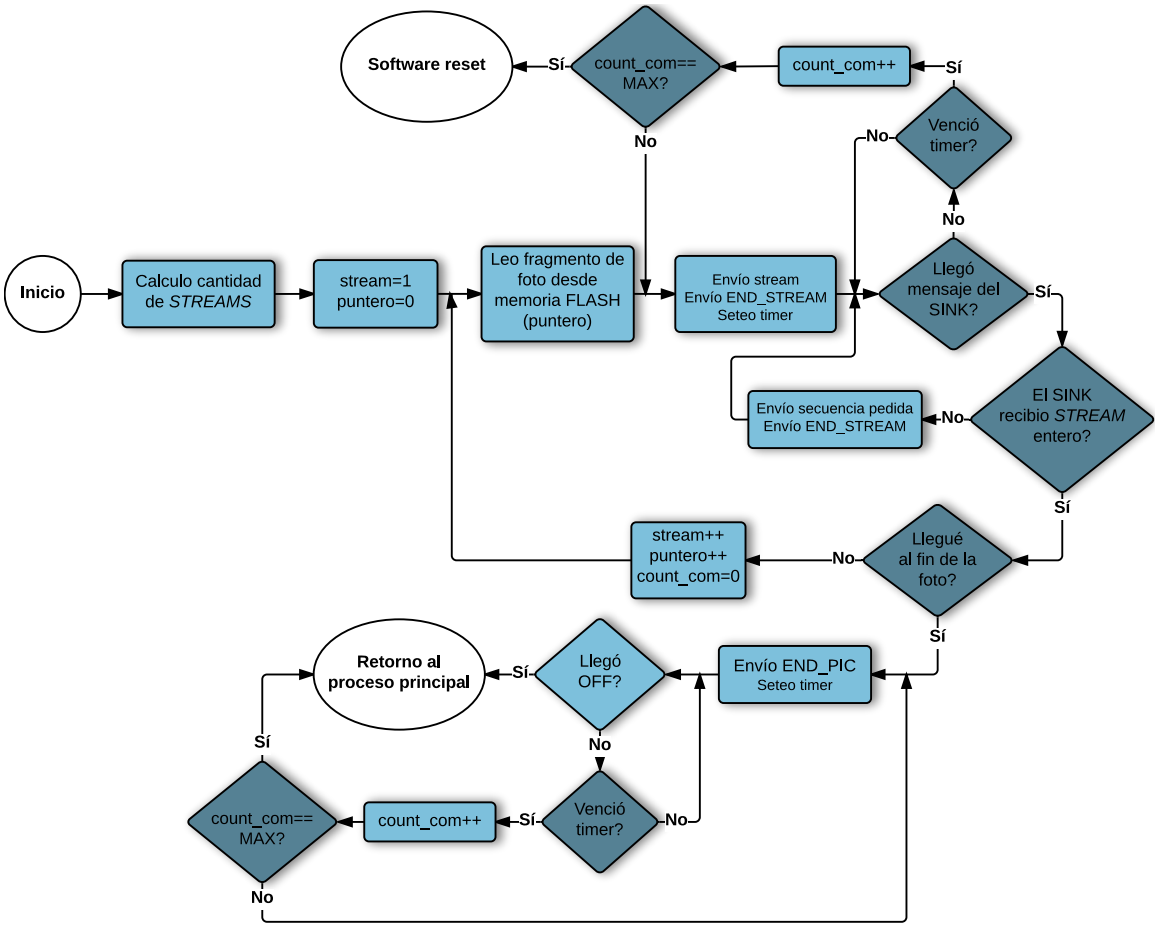


Figura 5.8: Diagrama de flujo de envío de una imagen utilizando el módulo *rime-stream.c*.

5.4.4. Módulo *dormir.c*

El módulo *dormir.c* implementa funciones para enviar a bajo consumo al nodo *WimSN*, configurar rtimers para sincronizar los ciclos de nodos *WimSN* y *sink* y apagar y encender cámara, LEDS de iluminación y radio. Las funciones son:

- $apagar\_radio()$: evoca a la función $off()$ del módulo de Contiki OS *cc2538-rf.c*. Tiene el fin de deshabilitar la radio en los momentos en que no se transmite a modo de bajar consumo de carga y no interferir con la UART. Si no se deshabilita al tomar la imagen en caso del nodo *WimSN* o interactuar con el PC para el *sink*, surgen problemas de comunicación.
- $encender\_radio()$: evoca a la función $on()$ del módulo *cc2538-rf.c*. Habilita la radio.
- $apagar\_camara$ y $encender\_camara$: apagan y encienden la cámara de fotos respectivamente.
- $apagar\_LEDS$ y $encender\_LEDS$: apagan y encienden los LEDS de iluminación presentes en la trampa del nodo *WimSN*.
- $dormir\_nodo()$: Recibe la hora en minutos para setear el rtimer que se utiliza tanto en los nodos *WimSN* como en el *sink* para sincronizar la red. En caso de nodo *WimSN*, esta función también envía a bajo consumo al mismo y apaga su radio.
- $rt\_callback$: Es el callback que se activa al expirar el rtimer. En caso de que ya se haya llegado al periodo de tiempo para iniciar un nuevo ciclo, este *callback* retira del bajo consumo al nodo *WimSN* y en ambos tipos de nodos realiza un *post* al proceso principal.

Dado que los rtimers admiten como constante de tiempo un $uint16\_t$, teniendo como unidad de medida el segundo, el máximo intervalo de tiempo que se puede configurar en una instancia es de aproximadamente 18 horas. Como criterio, si el tiempo a setear excede las 12 horas, luego de que expire el primer timer, se vuelve a configurar con el tiempo que reste para completar el intervalo.

5.4.5. Módulo *flash.c*

El módulo *flash.c* contiene las funciones $erase\_flash()$, $write\_flash()$ y $read\_flash()$ implementadas para borrar, escribir y leer desde flash respectivamente. Además, se encuentra la función auxiliar $ctrl\_lock\_bits()$ que controla en cada operación no modificar la página de bloqueo $LockBitPage$.

Las funciones deshabilitan interrupciones antes de realizar cualquier acción. Esto que resulta trivial, trajo problemas en el proyecto cuando aún no se tenía un buen manejo de las herramientas que provee $rom - util.h$. Si no se deshabilitan las interrupciones, se tienen fallas aleatorias en la lectura y escritura desde flash, lo que puede hacer que el archivo imagen no se envíe de forma correcta.

Como se mencionó en la sección 5.3.1, la mínima unidad de borrado de memoria flash es una página, lo que equivale a 2048 bytes. Empíricamente se tuvo que en una iteración se

puede realizar un borrado máximo de 16 páginas (32768 bytes). Vale aclarar además que la cantidad de bytes que se pueden escribir en flash o leer desde esta memoria debe ser múltiplo de 4.

5.4.6. Módulo *nodo.c*

El módulo *nodo.c* es el encargado de manejar el ciclo en los nodos *WimSN*, con el proceso *main_process*. Esto es:

1. Tomar la imagen.
2. Gestionar el comienzo de envío de la imagen por RF.
3. Dar la orden de envío a bajo consumo del nodo por el tiempo dado por el *sink*.

La captura de imagen se realiza a través de los procesos del módulo *driverCamara.c* en el siguiente orden cronológico:

1. Se enciende la cámara y se da comienzo al proceso *camara*. Este proceso realiza un *post* a *main_process* al terminar de realizar sus tareas y finaliza.
2. Se toma y almacena la imagen en memoria flash del nodo *WimSN* mediante el proceso *foto*. *foto* postea un evento a *main_process* al terminar las tareas y finaliza. Se apaga la cámara.

Luego de la toma de una imagen, se enciende la radio, siendo *rime-stream.c* quien envía la imagen por RF cuando el *sink* lo ordene. Al terminar el envío y recibir mensaje el mensaje OFF, se realiza un *post main_process*.

nodo.c apagará la radio y enviará a bajo consumo al nodo mediante la función *dormir_nodo*() con el tiempo enviado por el *sink*.

El proceso comienza nuevamente al expirar el tiempo de envío a bajo consumo.

El diagrama de flujo de la figura 5.9 brinda una abstracción del ciclo realizado por un nodo *WimSN*.

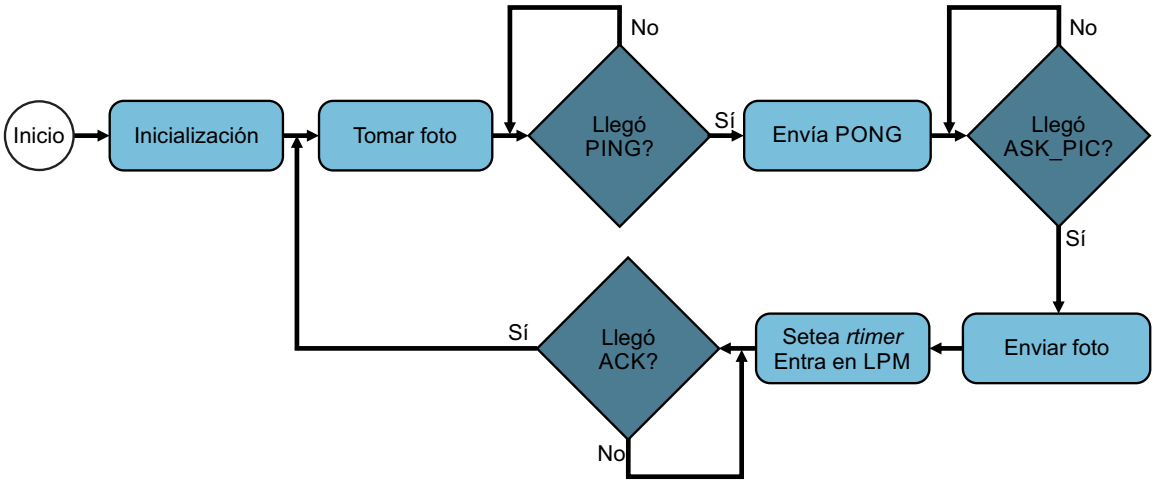


Figura 5.9: Diagrama de flujo del funcionamiento del nodo *WimSN*

5.4.7. Módulo *sink.c*

El módulo *sink.c* es el utilizado en el nodo *sink* para la comunicación por RF con los nodos *WimSN* y comunicación con el PC vía UART. Es este módulo el que gobierna el funcionamiento del sistema. En la figura 5.10 se observa un diagrama de interacción entre los diferentes módulos de Contiki OS, los módulos implementados y el hardware para el nodo *sink*.

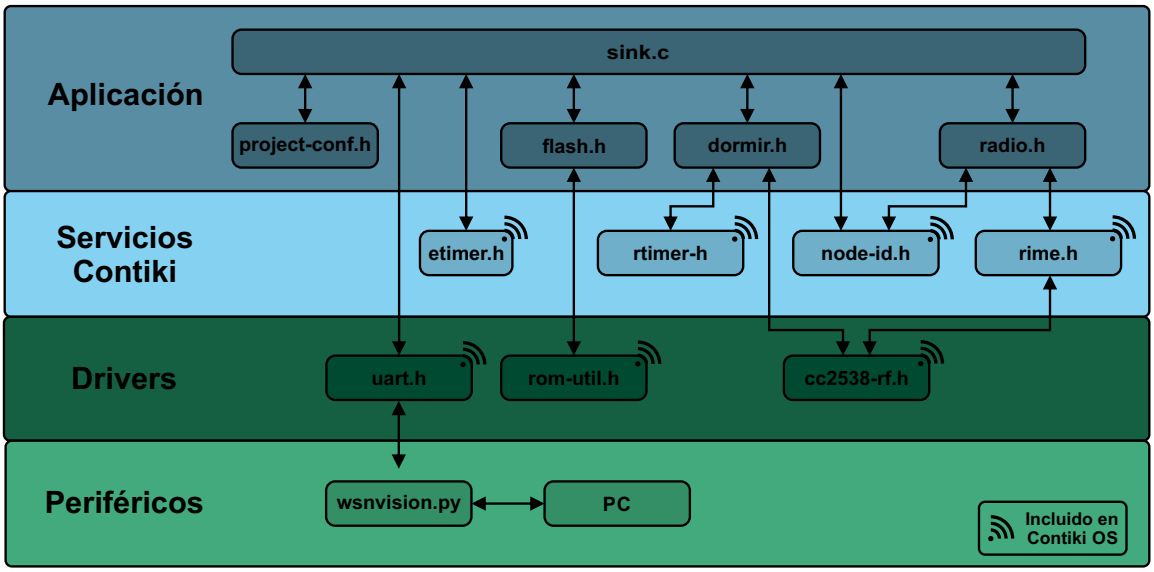


Figura 5.10: Diagrama de capas para el nodo *sink*

sink.c posee tres procesos:

1. *main_rime_process*, encargado de abrir el canal de comunicaciones *unicast*.
2. *rec_stream_process*, que realiza recepción de la imagen controlando número *stream* y secuencia de paquetes.
3. *main_process*, proceso principal que se encarga de desencadenar los ciclos de operación.

Además se tienen las siguientes funciones:

- *inicializar*(), que inicializa variables y la UART para comunicación con el PC, enciendo además la radio.
- *interpretar_msj*(), que realiza acciones al recibirse un mensaje de control por RF.
- *interpretar_msj_PC*(), análoga a *interpretar_msj*(), tomando acciones al recibir mensajes vía UART.
- *uart_callback*(),es el *callback* de recepción vía UART. Cada vez que se recibe un mensaje del PC, este callback evoca a *interpretar_msj_PC*().

- *recv_unicast*(), es el *callback* de recepción por RF para el nodo *sink*.

El diagrama de flujo de la función *recv_unicast*() se muestra en la figura 5.11. Cada vez que se reciben datos por RF, esta función verifica si se trata de un mensaje de control ó un paquete. Si es un mensaje, se evoca a la función *interpretar_msj*(). Si se trata de un paquete, se verifica si éste pertenece al *stream* esperado; en caso afirmativo se almacena el paquete en un *buffer* de memoria RAM y se registra en una arreglo el número de secuencia recibido. Si se trata del primer *stream* se da aviso al proceso *main_process* con un *post*.

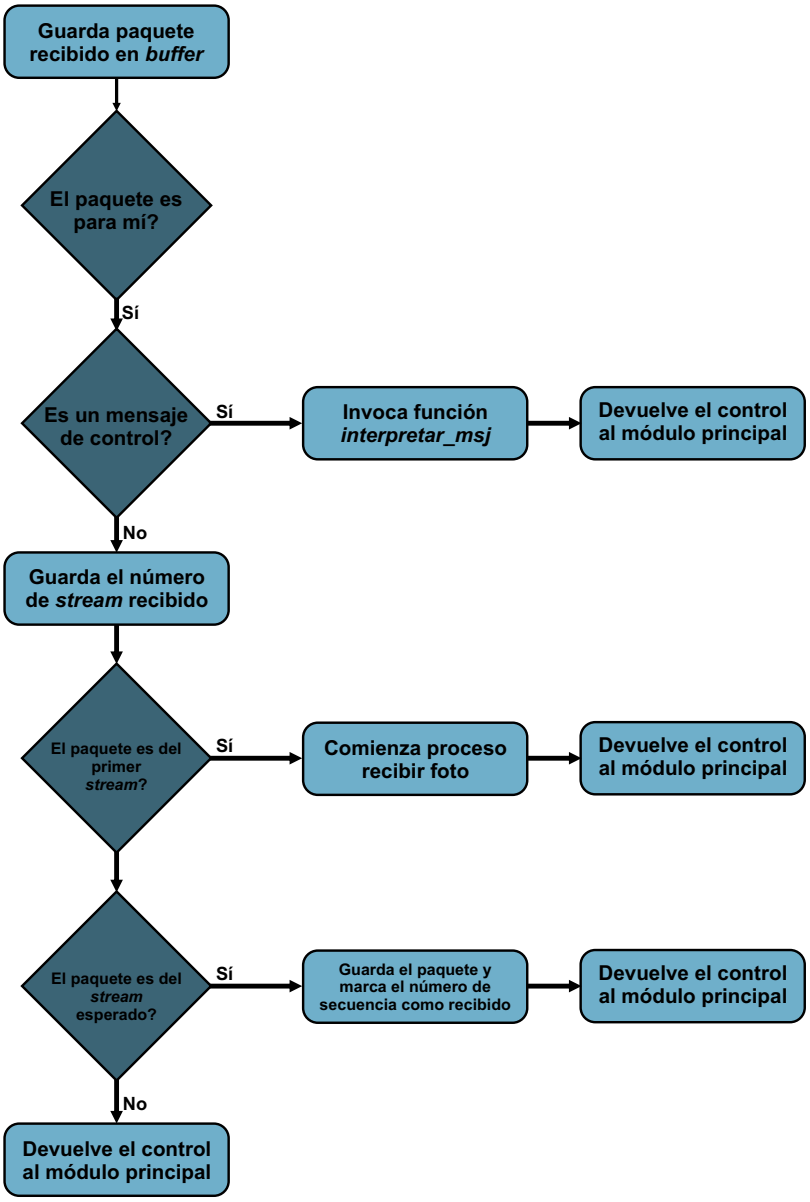


Figura 5.11: Funicón *callback* de recepción del *sink*

El funcionamiento del módulo *sinc.c* se muestra en la figura 5.12. En primer lugar se realiza un chequeo de conectividad para determinar los *nodos WimSN activos* en la red. Si luego de este chequeo, se tiene que no existen *nodos activos* en la red, el módulo consultará al PC el tiempo en que se debe volver a realizar el chequeo. En caso de existir *nodos activos*, se realiza el pedido de la imágenes comenzando por el nodo de ID superior que haya respondido al chequeo de conectividad.

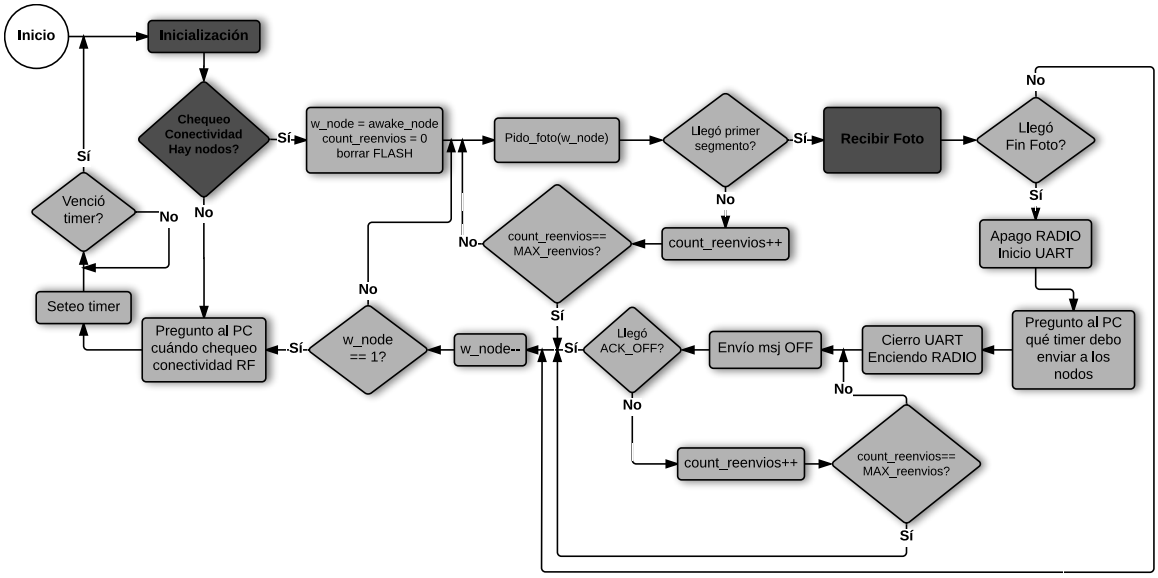


Figura 5.12: Diagrama de funcionamiento del módulo *sink*

Cada vez que se recibe una imagen, se consulta al PC el tiempo en que el nodo *WimSN* emisor debe permanecer en bajo consumo, para tener sincronismo en la red. Se envía dicho tiempo al nodo a través del mensaje de control OFF.

Se implementa el sistema de retransmisiones presentado en el capítulo 4 para los mensajes de control teniendo especial énfasis en que el *sink* no permanezca en un *loop* infinito.

El chequeo de conectividad se realiza en el proceso $main\_process$. Un diagrama de flujo de este mecanismo se presenta en la figura 5.13. Esta verificación consiste en el envío del mensaje $PING$ con el sistema de retransmisiones, comenzando desde el nodo de ID 2[6]. Si no se recibe el mensaje $PONG$ de un nodo en un determinado tiempo, se da por perdida la conectividad con el mismo y los de ID superior, finalizando el chequeo.

[6] El nodo de ID 2 es llamado $FIRST$ en la figura 5.13

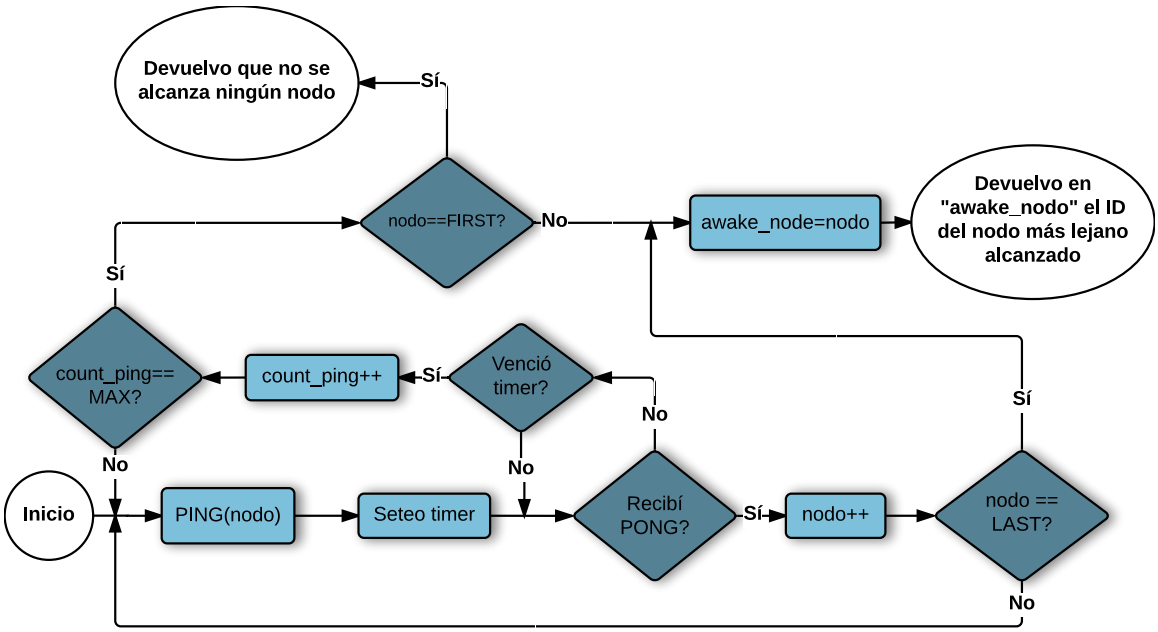


Figura 5.13: Chequeo de conectividad de la red

La recepción de la imagen se realiza con el proceso $rec\_stream\_process$, según el diagrama de flujo de la figura 5.14. Este proceso comienza a ejecutarse una vez que se reciben paquetes del primer *stream*. Cada vez que se recibe el mensaje END_STREAM, la función $interpretar\_msj()$ realiza un *post* a $rec\_stream\_process$ el cual actúa del siguiente modo:

- Verifica si el *stream* recibido es el esperado. En caso negativo se envía el mensaje $STREAM\_WRONG$, quedando a la espera de un nuevo END_STREAM.

- Si se recibieron todos los paquetes del *stream*, se almacena este último en memoria flash y se envía el mensaje $STREAM\_OK$.

- Si existieron pérdidas de paquetes, se pide reenvío mediante el mensaje ASK_PAQ. Este mensaje de control tiene implementado el sistema de retransmisiones descrito en el capítulo 4.

- Cada vez que se recibe el mensaje END_STREAM, se configura un timer. Si este timer expira antes de haber recibido el próximo END_STREAM, se considera que la comunicación se perdió. Siendo así, se da aviso al proceso $main\_process$ a través de un *post*, y se termina el proceso.

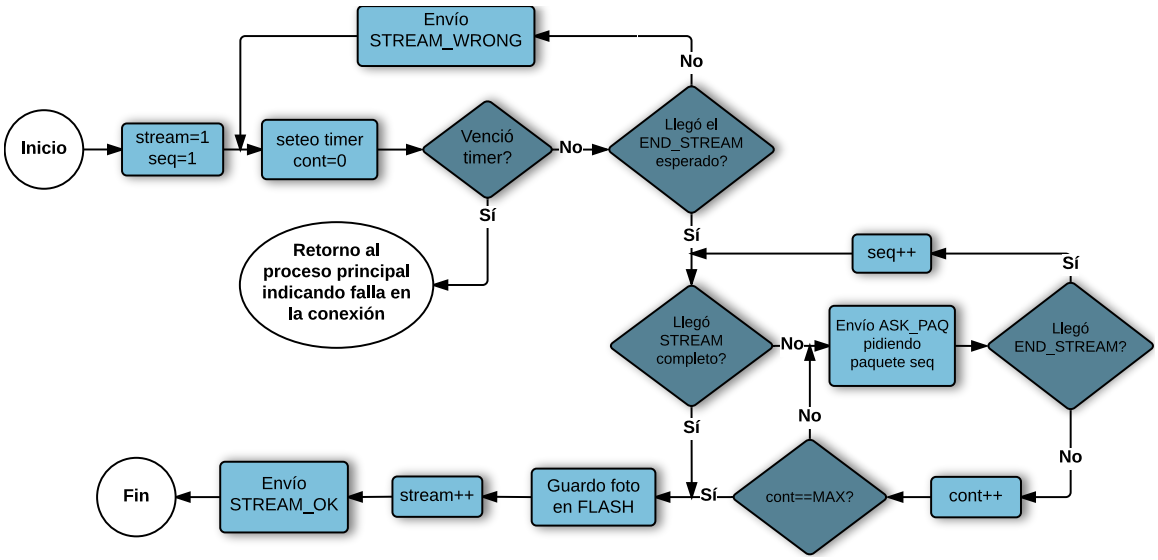


Figura 5.14: Recepción de imágenes en el proceso $rec\_stream\_process$

5.4.8. Archivo de configuración del sistema

project-conf.h es el archivo de configuración del sistema, en lo que respecta a los módulos desarrollados en Contiki OS.

Desde aquí se modifican constantes que serán de importancia a la hora de optimizar el sistema en proyectos futuros y se seleccionan aspectos referentes al ciclo de radio y modos de bajo consumo. A continuación se detallan esas constantes, para facilitar la continuación del proyecto.

- $RESOLUCION$: byte con el que se varía la resolución de la imagen a tomar. Las diferentes opciones son:
  - 1Dh - 800x600 pixeles
  - 1Ch- 1024x768 pixeles
  - 1Bh -1280x960 pixeles
  - 21h- 1600x1200 pixeles
- $COMPRESION$: byte con el que se selecciona la compresión deseada de la imagen. La máxima compresión se corresponde con el valor FFh.
- $LAST\_NODE\_ID$: valor que determina la ID del último nodo de la red. A modo de ejemplo, si la red está compuesta por cinco nodos, esta constante deberá valer 5.
- $PONG\_ET$: Tiempo (en segundos) de configuración del timer de reenvío de mensajes para nodos *WimSN* y *sink*.
- $COUNT\_PINGS$: Contador de intentos de envío de mensajes por parte de los nodos *WimSN* y *sink*.

- *STREAM_ET*: Tiempo en segundos de reenvío de *streams* en nodos *WimSN*. Es el timer que expira al no recibir respuestas luego de un *END_STREAM*.
- *COUNT_COM*: El producto de esta constante por *STREAM_ET* determina el tiempo luego del cual se da por perdida la comunicación en nodos *WimSN* y *sink* durante el envío de una imagen.
- *COUNT_ALIVE*: El producto de esta constante por *PONG_ET* determina el tiempo en que el *sink* determina que no existe conectividad con un nodo *WimSN* determinado.
- *DATASIZE* : Tamaño en bytes de un paquete.
- *STREAMLEN*: Cantidad de paquetes que conforman un *stream*.

5.4.9. Módulo *wsn-vision.py*

El módulo *wsn-vision.py* fue implementado en el lenguaje de programación Python y es la interfaz entre el usuario y el sistema. Al ejecutarse el programa, *wsn-vision.py* consulta al usuario la cantidad de imágenes por nodo que desea adquirir diariamente y en qué horarios, almacenando estos datos. El usuario debe ingresar estos datos por teclado. *wsn-vision.py* abre el puerto de comunicación serial en el que estará conectado el nodo *sink* con su placa interfaz y se queda escuchando ese puerto.

Existen tres mensajes que el nodo *sink* envía a *wsn-vision.py*:

- *PING_TIME*: El nodo *sink* consulta el tiempo en que debe comenzar a chequear conectividad nuevamente.
- *FOTO_RDY*: El nodo *sink* avisa que tiene una foto para enviar, informando del nodo *WimSN* que proviene.
- *SLEEP_UNTIL*: El nodo *sink* consulta el tiempo que debe enviar al nodo *WimSN* para que ingrese a bajo consumo.

Al recibir el mensaje *SLEEP_UNTIL*, *wsn-vision.py* verifica cual es la próxima hora de envío. En base a ésto, realiza el cálculo de tiempo en minutos y se lo envía al *sink*.

Al recibir el mensaje *FOTO_RDY*, envía un ACK y comienza la recepción de la imagen vía UART. Al detectarse el final de la imagen [7], *wsn-vision.py* avisa al *sink* que se recibió una imagen completa. Luego de ésto, crea un mail y adjunta la imagen, enviándola a las direcciones configuradas por el programador.

Las acciones tomadas por *wsn-vision.py* al llegar el mensaje *PING_TIME* son análogas a las de la recepción de *SLEEP_UNTIL*. Luego de responder al *sink* con el tiempo en minutos para configurar su rtimer, *wsn-vision.py* setea su propio timer con el intervalo enviado menos treinta segundos. Por ese periodo se deja de escuchar el puerto serial.

Este módulo no contempla medidas de contingencia en caso de falla de comunicación con el *sink*. Una posible solución es la configuración de timers y contadores para reenvío de mensajes y comandos, quedando como posible trabajo a futuro.

[7] Los bytes FFD9h indican el final de una imagen en un archivo JPEG

5.4.10. Extensión del número de nodos *WimSN* de la red

Si bien este proyecto implementa una red de 5 nodos, el software desarrollado permite la extensión de la misma de modo transparente, hasta un máximo de 255 nodos. Para ésto se debe modificar en el archivo de configuración del sistema la constante $LAST\_NODE\_ID$ con el número de nodos de la red, tal como se especifica en la subsección 5.4.8.

El módulo *nodo.c* debe compilarse especificando la dirección ID del nodo *WimSN* al que se desea cargar el código. Esto se realiza con la sentencia MAKE $TARGET = cc2538dk$ $nodo$ $NODEID = ID$ [3].

Capítulo 6

Caracterización de alcance de radio y protocolo de transmisión de información

6.1. Resumen

En este capítulo se presentan los ensayos realizados a fin de caracterizar el alcance de la radio de los nodos *WimSN* y *sink*, estableciendo si se logra un área de cobertura de un nodo *WimSN* por hectárea. Adicionalmente, se presentan las pruebas realizadas al protocolo de transmisión de información en campo de cultivos frutales.

Se tuvo que con visibilidad entre antenas se puede llegar a distancias superiores a los 250 m entre nodos. La situación cambia si las antenas no tienen visibilidad, llegando a pérdidas mayores al 10 % a 60 m de distancia. Se logró recibir imágenes íntegras a través de la red distribuida en campo.

6.2. Introducción

Con el objetivo de caracterizar el alcance de la radio de los nodos *WimSN* y *sink* y probar el protocolo de transmisión en un ambiente real, se realizaron ensayos en plantaciones de manzanos.

El predio fue cedido por JUMECAL y se encuentra próximo a la sede de esta cooperativa. En la figura 6.1a se tiene imagen satelital del mismo, donde se observan las plantaciones agrupadas en canteros. Los árboles frutales se plantan en surcos, de forma alineada, como se muestra en la figura 6.1b. Entre dos surcos existe una separación de 5 m, medida con un GPS. Las trampas tipo Wing se ubican sobre una caña en el tercio más alto del árbol, tal como se muestra en la figura 6.2.

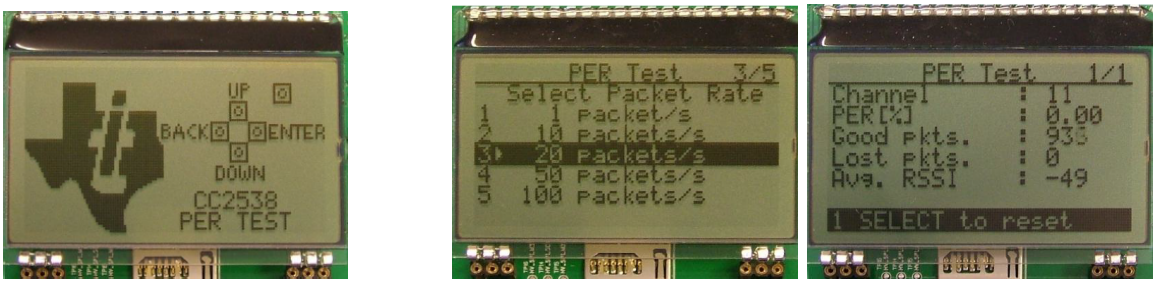
(a) (b)

Figura 6.1: (a) Predio cedido por cooperativa JUMECAL para pruebas en campo. (b) Disposición de árboles frutales en campo.

Figura 6.2: Trampa tipo Delta colocada para control de plagas.

Los insumos utilizados en las pruebas fueron:

- Cuatro nodos CC2538-CC2592EM y un nodo CC2538EM.
- Dos placas SmartRF06EBK.
- Cuatro placas interfaz Nodo-Cámara-Iluminación.
- Una cámara modelo LS-Y201-2MP.
- Doce pilas AA para alimentación de nodos.



(a) (b) (c)

Figura 6.3: (a) Pantalla de inicio del software *per-test*, tomada de [33]. (b) Selección de cantidad de paquetes por prueba con software*per-test*, tomada de [33].(c) Presentación de resultados luego de una prueba con software *per-test*, tomada de [33]

.

- Un PC para visualizar las imágenes.
- GPS para determinar posiciones y distancias entre nodos.

6.3. Caracterización del alcance de radio

Para poder caracterizar el alcance de radio, se envían ráfagas de mensajes de un nodo emisor a un receptor. Se registran luego el procentaje de mensajes perdidos y RSSI (Received Signal Strength Indicator).

El RSSI es un indicador del nivel de potencia de las señales recibidas por un dispositivo en redes inalámbricas. Depende de una curva de calibración interna de la radio del dispositivo que se desea caracterizar.

Texas Instruments provee de forma libre el software *per-test* que permite caracterizar el alcance de la radio de los nodos CC2538-CC2592EM y CC2538EM. Este software está implementado en lenguaje C puro, y puede ser adquirido en la página del desarrollador [26]. Permite el envío de mensajes de un nodo a otro a una determinada cadencia, registrando RSSI y porcentaje de paquetes perdidos. Para la interacción con el usuario, este software utiliza el Display y botones de la placa SmartRF06EBK. Imágenes de la interfaz que presenta *per-test* se observan en las figura 6.3.

Utilizando los botones de la placa SmartRF06EBK el usuario navega por diferentes pantallas, seleccionando las siguientes opciones:

- Nodo CC2538EM ó CC2538-CC2592EM.
- Modo emisor o receptor de mensajes.
- Canal de comunicaciones. Nodo emisor y receptor deben configurarse en el mismo canal.
- Potencia de transmisión Tx (seleccionable en modo emisor): Tx puede configurarse dentro del rango -15 dBm a 22 dBm en caso de nodo CC2538-CC2592EM. Si se trata de un nodo CC2538EM, el máximo se reduce a 7 dBm.

- Número total de paquetes a enviar en la ejecución (seleccionable en modo emisor). Las opciones son 1000, 10000, 100000 ó 1000000 paquetes por instancia.
- Cantidad de paquetes por segundo que se desean enviar (modo emisor). Se opta entre 1, 10, 20, 50 ó 100 paquetes por segundo.

Con este software se realizaron los ensayos de caracterización entre dos nodos *WimSN* y entre el nodo *sink* y un nodo *WimSN* bajo las siguientes condiciones:

1. En los nodos *WimSN* se utilizó su antena externa.
2. Los nodos se posicionaron a la altura del tercio más alto de los árboles, por ser ésta la distancia a la que se colocan las trampas tipo Wing.
3. La potencia de transmisión seleccionada fue la máxima (7 dBm y 22 dBm para nodos *sink* y *WimSN* respectivamente).
4. Se enviaron 10000 paquetes por instancia.
5. Las pruebas se realizaron en el mes de mayo, momento en el cual los árboles habían perdido parte de su follaje.
6. Se registraron instancias a diferentes distancias entre nodos, con y sin visibilidad entre antenas.

En la tabla 6.1 se presentan los resultados obtenidos realizando los ensayos entre un nodo *WimSN* y el nodo *sink*, con visibilidad entre antenas. La ubicación de los nodos se observa en la figura 6.4.

Emisor	Receptor	Distancia (m)	Paquetes /segundo	Paquetes perdidos (%)	RSSI (dBm)
Nodo *WimSN*	Nodo *sink*	197.3	100	0,06	-82
Nodo *WimSN*	Nodo *sink*	258.2	100	1,39	-74
Nodo *sink*	Nodo *WimSN*	258.2	100	2,66	-96

Tabla 6.1: Ensayos con software *per-test* entre nodos *sink* y *WimSN* con visibilidad entre antenas.

Se desprende de esta tabla que es posible una separación entre nodo *WimSN* y *sink* de al menos 258 m, con pérdidas que no superan el 3 %. Se puede observar además que cuando el nodo *sink* es el emisor de mensajes, las pérdidas de paquetes son mayores y el RSSI es menor. Esto resulta coherente ya que este nodo emite con una potencia máxima significativamente menor a la de los nodos *WimSN*. Se concluye que a esta distancia las pérdidas son muy bajas, pudiéndose superar la cobertura de un nodo por hectárea.

Figura 6.4: Disposición de nodos con visibilidad entre antenas.

Se realizó el mismo ensayo entre dos nodos *WimSN* a diferentes distancias, con visibilidad entre antenas, registrándose los datos en la tabla 6.2. En este caso se tienen pérdidas menores al 1 % a una distancia de 327 m, lo que indica nuevamente que es posible cumplir con el requerimiento de área de cobertura.

La situación cambia si no existe visibilidad entre antenas. Se colocaron los nodos dentro del follaje de los árboles, en diferentes surcos, como se muestra en la figura 6.5, obteniéndose los resultados registrados en la tabla 6.3. Se observa que a una distancia entre nodos de 60 m equivalente a 12 surcos, las pérdidas son del 11 %, ascendiendo al 70 % cuando la separación es de 80 m (16 surcos).

Emisor	Receptor	Distancia (m)	Paquetes /segundo	Paquetes perdidos (%)	RSSI (dBm)
Nodo *WimSN*	Nodo *WimSN*	68	100	0	-57
Nodo *WimSN*	Nodo *WimSN*	197.3	100	0	-71
Nodo *WimSN*	Nodo *WimSN*	327.8	100	0,37	-76

Tabla 6.2: Ensayos con software *per-test* entre dos nodos*WimSN* con visibilidad entre antenas.

Con ésto, se tiene que si se colocan los nodos sin visibilidad entre antenas, la máxima distancia admisible son 60 m, sin llegarse a tener la cobertura de un nodo por hectárea.

Figura 6.5: Disposición de nodos *WimSN* entre surcos. Los nodos se ubican en los extremos de la línea roja de la imagen.

Emisor	Receptor	Distancia (m)	Paquetes /segundo	Paquetes perdidos (%)	RSSI (dBm)
Nodo *WimSN*	Nodo *WimSN*	35 (7 surcos)	100	0,63	-89
Nodo *WimSN*	Nodo *WimSN*	45 (9 surcos)	100	0,89	-95
Nodo *WimSN*	Nodo *WimSN*	60 (12 surcos)	100	10,56	-99
Nodo *WimSN*	Nodo *WimSN*	70 (14 surcos)	100	12,2	-102
Nodo *WimSN*	Nodo *WimSN*	80 (16 surcos)	100	67,9	-105

Tabla 6.3: Ensayos con software *per-test* entre dos nodos*WimSN* colocados entre el follaje de los árboles.

Para cumplir con el requerimiento, se proponen dos alternativas:

- *Colocar nodos que actúen únicamente como routers, reenviando los paquetes.*

  Se podría reutilizar el *callback* de recepción por RF para nodos *WimSN*, reenviando mensajes según la ID del nodo emisor. Debería realizarse un estudio de diferentes modelos de nodos priorizando el bajo precio y bajo consumo de carga.

- *Colocar las antenas de los nodos de modo tal que tengan línea vista entre sí.*

  Sería posible extraer un cable extensor desde la trampa, colocando la antena por encima de la altura de los árboles.

Estos dos puntos pretenden ser meramente opciones de trabajo a futuro.

6.4. Ensayo a protocolo de transmisión de información

Previo a la prueba del protocolo de transmisión en campo, se realizaron ensayos en laboratorio con el fin de verificar si el protocolo se comportaba de acuerdo a lo esperado. A continuación se presenta un breve resumen de esos ensayos:

- Se provocaron fallas en número de secuencias de paquetes y número de *stream* en los nodos *WimSN*. El nodo *sink* identificó estas fallas y pidió los paquetes faltantes y el *stream* correcto.
- Se quitó la alimentación de un nodo intermedio cuando cuando un nodo *WimSN* de ID superior estaba enviando una imagen. El nodo *WimSN* reenvió el *stream* la cantidad de veces fijada por el programador y se reseteó al superar el número de máximo de reenvíos.
- Se provocaron pérdidas de mensajes de control entre nodos *WimSN* y *sink* obteniéndose los resultados esperados.
- Se probó la red sin la presencia de los nodos de ID 4 y 5. El nodo *sink* luego del chequeo de conectividad descartó a estos dos últimos, pidiendo la imagen al resto de los nodos.
- Se ensayó el sistema en laboratorio con los nodos *WimSN* utilizando su antena impresa. Se logró tener pérdidas reales y el sistema se recuperó al reducir la distancia entre nodos.
- Se dejó el sistema en funcionamiento durante una semana. Nuevamente los resultados fueron satisfactorios.

Dado que los ensayos anteriores se realizaron en laboratorio, se decidió probar la red en campo. La disposición de los nodos se muestra en la figura 6.6.

Los números 1 a 5 presentes en la figura, se corresponden con las direcciones ID de los nodos. Las distancias entre nodos se detallan a continuación:

- 26,7 m entre nodo 2 y *sink*.
- 49 m entre nodo 2 y nodo 3.
- 37 m entre nodo 3 y nodo 4.
- 19 m entre nodo 4 y nodo 5

Se ejecutó el sistema enviando una imagen del nodo 5 al *sink*, recibiéndose de forma satisfactoria. Vale aclarar que no se realizaron una mayor cantidad de pruebas estresando el protocolo por falta de tiempo el día de la visita de campo. Se coordinó otra visita que se realizará luego de la entrega de documentación, donde se pretenden realizar más ensayos al sistema en un escenario real. Los resultados se mostrarán en la presentación del proyecto.

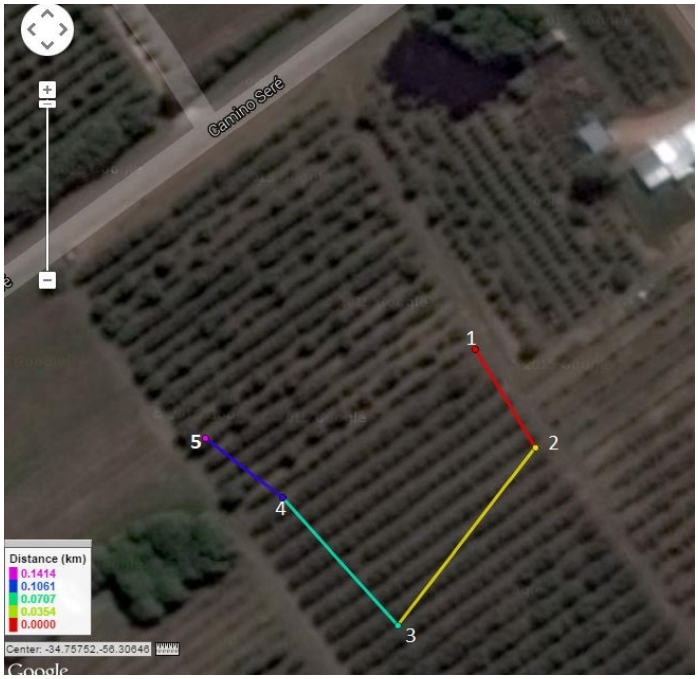


Figura 6.6: Disposición de nodos de la red en campo.

6.5. Conclusiones del capítulo

Dada la caracterización del alcance de radio, es posible tener 100 m de separación entre nodos solo si se tiene visibilidad entre antenas. En caso contrario, se obtiene una distancia máxima de 60 m. Para cumplir con el requerimiento se propone la utilización de nodos de bajo costo y consumo que actúen como *routers*. Otra alternativa sería colocar la antenas fuera de las trampas, buscando visibilidad entre éstas.

El sistema se comportó de acuerdo a lo esperado en un escenario real. Sin embargo, es necesario realizar una mayor cantidad de pruebas en campo, estresando el protocolo de transmisión de información y verificando que el mismo logra recuperarse ante pérdida de mensajes.

Capítulo 7

Conclusiones

Se implementó una red de sensores inalámbricos con capacidad visual que permite realizar el monitoreo de polillas de las especies *Cydia Pomonella* y *Cydia Molesta* presentes en cultivos de manzanos, perales, membrilleros y nogales.

Esta red está compuesta por nodos con capacidad visual, *WimSN*, y por un nodo concentrador, *sink*. Los nodos *WimSN* se disponen embebidos en trampas y están compuestos por un módulo CC2538-CC2952EM, una cámara Linksprite LS-Y201-2MP, un sistema de iluminación y un circuito interfaz. El nodo *sink* está compuesto por un módulo CC2538-CC2952EM y un circuito interfaz. Ambos tipos de nodo presentan mejoras respecto al prototipo implementado en Plagavision. Tienen mayor capacidad de procesamiento y almacenamiento, mayor potencia de transmisión RF y permiten imágenes de mayor resolución.

Se implementó la red con topología lineal, siendo probado el sistema en escenario real. La red estuvo compuesta por cinco nodos, siendo transparente su extensión al número deseado por el usuario. Se logró transmitir imágenes de forma íntegra a través de la red de 5 nodos tanto en campo como en laboratorio.

Se diseñó e implementó un protocolo confiable de comunicación por RF que asegura la transmisión de imágenes de forma íntegra. El mismo implementa mensajes de control que permiten el manejo de la red y el funcionamiento cíclico del sistema. Este protocolo presenta un sistema de retransmisión de los mensajes considerados críticos para el funcionamiento de la red. Se tuvo en cuenta la necesidad de que los nodos *WimSN* estén en modo de bajo consumo durante la mayor cantidad de tiempo posible, minimizando el tiempo en el que el nodo está transmitiendo.

Se desarrolló en Python un programa interfaz PC-*sink* que permite la configuración de cantidad y horario de toma de imágenes, además del almacenamiento de las mismas en el PC y su posterior envío por correo electrónico. La interfaz etiqueta a estas imágenes con la fecha y hora de recepción y número de nodo emisor, enviándolas vía correo electrónico a casillas configuradas por el usuario. Este programa recibe las cosultas del *sink* y responde con el tiempo que falta para el próximo envío de imágenes, asegurando la sincronización de la red.

Se elaboró un banco de imágenes que permitió definir la configuración óptima del sistema: lente e iluminación empleados, características de la trampa, y resolución y compresión a ser seteadas en la cámara.

En base a las imágenes tomadas se determinó el uso de 4 LEDs blancos colocados en los vértices del techo de la trampa, apuntando hacia arriba. Se dedujo además que la mejor opción es tomar las imágenes en la noche ya que la exposición directa a la luz solar provoca imágenes con alto contenido de blancos, en las cuales las polillas no pueden ser distinguidas.

Se corroboró que con piso inclinado en la trampa se obtienen mejores imágenes que con piso plano, dado que las mismas presentan menor aberración y permiten tener foco uniforme en todo el piso de la trampa. El mejor resultado se obtuvo con una inclinación de 20° respecto al plano horizontal.

Con el objetivo de obtener imágenes que cubran el piso completo de la trampa, se evaluaron lentes gran angular, entre los cuales se seleccionó el lente LS-20150, por ser el que permitió cubrir todo el piso con menor aberración en la imagen. Si bien la elección de este lente implicó cambios en el diseño de la trampa, este diseño fue validado por productores de la cooperativa JUMECAL.

El banco de imágenes permitirá en etapas futuras profundizar sobre otras mejoras y fundamentalmente será de utilidad en el entrenamiento y validación de posibles algoritmos para conteo automático.

Se realizaron ensayos en campo del alcance de la radio de los nodos, logrando transmisiones a distancias superiores a 200 m cuando existe línea vista entre antenas. En las pruebas en que no se tiene esta visibilidad, por ejemplo con los nodos separados en forma transversal a los surcos, la distancia obtenida se redujo a 60 m. En caso de ser necesario aumentar la separación entre nodos, la red podría complementarse con nodos económicos que simplemente actúen como routers o repetidores, o se podrían usar extensores para colocar las antenas de los nodos por encima de la copa de los árboles asegurando visibilidad. Los ensayos fueron realizados posicionando los nodos en el tercio superior de la copa de los árboles, por ser la altura en que se colocan las trampas habitualmente.

Dado que no se contó con participación del Grupo de Tratamiento de Imágenes del IIE, no se tuvo un algoritmo de procesamiento de imágenes para implementar el conteo automático de polillas. El conteo automático se considera fundamental para encarar en trabajos futuros, ya que el mayor consumo de los nodos se produce durante la transmisión de datos. La implementación del conteo automático permitiría disminuir el consumo de los nodos si se decide por ejemplo enviar imágenes solamente cuando se supera un umbral de polillas atrapadas. Adicionalmente, este conteo permitiría automatizar la centralización de información, liberando el sistema de errores humanos y facilitando el proceso.

Al momento de entregar esta documentación no se han completado medidas de caracterización del consumo de carga de los nodos. Se podrían realizar estimaciones con los valores teóricos brindados por los fabricantes, pero éstas no serían representativas del sistema en un escenario real.

En la presentación formal de este proyecto se presentarán estimaciones obtenidas a nivel laboratorio, adjuntando un anexo con estos datos.

7.1. Análisis en función de los criterios de éxito

- **Implementación de la red de 5 nodos en laboratorio, tomando imágenes al menos dos veces al día y enviándolas al *sink* por RF sin pérdidas.**

Se logró implementar la red de cinco nodos a nivel laboratorio tomando dos imágenes por día. El ensayo se realizó por un periodo de una semana en la que cada nodo envió su respectiva imagen, almacenándose en el PC y siendo luego enviada vía e-mail.

- **Implementación de la red en campo pudiendo transmitir una imagen sin pérdidas a través de los cinco nodos.**

  Se logró transmitir imágenes a través de la red de forma íntegra en un escenario real. Se probó la red en cultivos de manzanos y perales obteniéndose resultados satisfactorios. De todos modos, no se estresó el protocolo en cuanto a distancias entre nodos por no disponer del tiempo suficiente en la visita a los cultivos.

- **Lograr una cobertura de un nodo *WimSN* por hectárea. Para esto, se impone una distancia mínima entre dos nodos de al menos 100 m en campo.**

  Dados los ensayos presentados en el capítulo 6, se tiene que si existe visibilidad entre antenas, colocando los nodos a la altura de un tercio de la copa de los árboles, se logran distancias superiores a los 200 m.

  Sin embargo, si no existe visibilidad, la distancia se reduce a 60 m.

  Para cumplir con este requerimiento, se propone la utilización de nodos que actúen como routers. Otra opción radica en colocar la antena de los nodos fuera de las trampas, buscando visibilidad.

- **Obtener imágenes del piso de la trampa que sean validadas por los entomólogos de JUMECAL.**

  Al momento de entregar esta documentación recién se tiene la aprobación de la trampa presentada por parte de la cooperativa agraria JUMECAL, por lo que no fue posible realizar un intercambio de imágenes para su aprobación.

7.2. Dificultades enfrentadas en el transcurso del proyecto

En el transcurso del proyecto, se debieron afrontar distintas dificultades. Aquí se presentan las que resultaron de mayor implicancia.

1. **Cambio de plataforma de hardware.**

   En las primeras etapas de proyecto se trabajó con la palataforma Openmote. Si bien la misma posee el SoC cc2538, no se encuentra portada de forma correcta en Contiki OS debido a temas constructivos.

   Aquí se detectaron errores en la asignación de puertos GPIO asociados a LEDs y botones. Por otro lado, al intentar manejar su memoria flash, se detectó que el nodo OpenMote almacenaba porciones de software en diferentes lugares de memoria flash, de forma aleatoria. En este momento no se tenía fluidez en el manejo de la plataforma ni en el sistema operativo Contiki OS, por lo que estas dos complicaciones tardaron en detectarse y resolverse.

   A mediados de diciembre de 2014, ya transcurridos ocho meses del comienzo del proyecto, se pasó a trabajar en la plataforma $CC2538EM$ que se encuentra portada de forma correcta en Contiki OS.

2. **Falta de herramientas de simulación en Contiki OS.**

   Contiki OS posee un programa de simulación (Contiki Cooja) que permite la prueba de software de forma sencilla. Aquí se pueden crear redes de nodos, simular ambientes con pérdidas, observar el flujo de paquetes en las comunicaciones, etc.

   El SoC cc2538 no se encuentra portado en Cooja, por lo que la depuración de códigos y detección de errores implicó una gran inversión de tiempo.

   A modo de ejemplo, en cada momento que se deseaba probar un cambio en el funcionamiento de multihop se debía armar una red real, llevando un tiempo considerable entre compilación y armado.

3. **Falta de herramientas de harware y retraso en compras en el exterior.**

   Al inicio del proyecto, al trabajar con la plataforma OpenMote se contaba con dos nodos y una placa programadora, lo que no permitía paralelizar tareas de programación. Adicionalmente, se dañó uno de los nodos lo que complicó aún más esta situación.

   Al comenzar a trabajar con la plataforma CC2538EM, solo se contaba con una placa programadora y un nodo, lo que no permitía paralelizar tareas de programación ni realizar la depuración de códigos de forma rápida.

   Por otro lado, la realización de las placas interfaz nodo-iluminación-cámara, se dilató en el tiempo dado que la compra de componentes para estas placas estuvo en Aduana por varios meses.

4. **Funcionamiento de multihop**

   En el marco del curso Redes de Sensores inalámbricos, dictado por el IIE, se realizó el estudio de la aplicación udp-stream implementado por los desarrolladores de Contiki. El mismo utiliza el stack uip de Contiki OS, y realiza ruteo dinámico. Se determinó que podría ser de utilidad para el envío de grandes cantidades de datos en redes de sensores. Por ésto, en marco de *WSNvision* se modificó *udp-stream* para garantizar el envío de la imagen, agregando controles de secuencia.

   Una vez que se logró el envío de la imagen nodo a nodo, se pasó a probar el multihop. Luego de un gran tiempo invertido, no se logró el multihop, decidiendo implementar un protocolo propio basado en el stack Rime de Contiki OS. Este último es el protocolo que se presenta en este proyecto.

7.3. Trabajo a futuro

En esta sección se presentan sugerencias para aportar mejoras al sistema. Estas sugerencias serán tomadas por otro grupo de PFC que continuará con el desarrollo en una etapa 3. Esta etapa consiste en: aumentar a diez la cantidad de nodos de la red, optimizar el funcionamiento del sistema para disminuir consumo de carga de los nodos e implementar el conteo automático de polillas en nodos *WimSN*.

1. *Mejoras en el Driver de la cámara*

En el software desarrollado para la comunicación con la cámara LS-Y01-2MP, no se tomaron medidas de contingencia en caso de falla en la comunicación UART entre nodo y cámara o en caso de la recepción de una imagen corrupta.

Se sugiere:

a) Agregar algún tipo de control que permita que ante una pérdida de comunicación con la cámara, el nodo *WimSN* retome su ciclo de operación.

b) Verificar los *markers* de JPEG recibidos de forma de identificar cuando se reciben imágenes corruptas para tomar medidas de contingencia.

2. *Software de interfaz en Python PC-sink*

   El código *wsn-vision* no debe ser tomado como un software final de interfaz PC-*sink*. El mismo fue desarrollado con el fin de realizar pruebas de la red.

   Para mejorarlo, se deben tomar medidas de contingencia en caso de falla de comunicación UART o falta de conectividad a Internet para el envío de imágenes.

3. *Mejoras en sistema de iluminación*

   Si bien en este proyecto se estudiaron diferentes configuraciones de iluminación en cuanto a posicionamiento, color y cantidad de LEDs, resta realizar un estudio utilizando la trampa aprobada finalmente por JUMECAL. Al momento de realizar el banco de imágenes no se tenía el modelo de trampa definitiva del proyecto.

4. *Mejoras en tiempos de envío de imágenes*

   Si bien el sistema garantiza el envío de imágenes en la red, en los diferentes códigos implementados se utilizan timers para envío de mensajes. El sistema no se encuentra optimizado en cuanto a tiempos, por lo que deben realizarse pruebas en campo transmitiendo imágenes y registrando pérdidas de mensajes al disminuir los tiempos.

   Es deseable disminuir el tamaño de imágenes sin comprometer el reconocimiento de insectos en la misma, mejorando la iluminación y aumentando la compresión de JPEG.

   Además, puede variarse la cantidad de paquetes por *stream* buscando optimizar el envío en cuanto a pérdidas y tiempo.

   A modo de ejemplo, luego del envío de un *stream* se tiene un timer de 1 segundo antes de enviar el mensaje END_STREAM. Si no se coloca un timer, el mensaje de control se pierde ocasionalmente. Esto implica un retardo de 1 segundo entre el envío de un *stream* y el siguiente. Este tiempo no está optimizado y puede disminuirse.

5. *Caracterización del consumo de carga del sistema*

   Para relevar el consumo real del sistema, queda pendiente realizar medidas de largo aliento en campo con la placa de medidora consumo diseñada en este proyecto. Se debe dejar el sistema en funcionamiento en campo con esta placa conectada al nodo más comprometido (nodo de ID 2 por ser el que permanece encendido por mayor tiempo) para poder sacar conclusiones certeras del consumo de los nodos y la duración de las baterías.

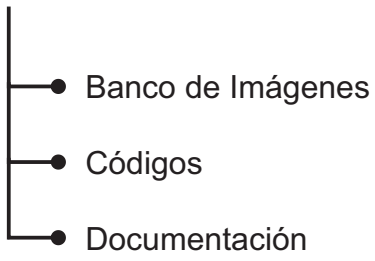
6. *Robustez en comunicación UART PC-sink* El módulo wsn-vision.py no contempla medidas de contingencia en caso de falla de comunicación con el *sink*. Tampoco se toman medidas paliativas en el módulo *sink* en lo que refiere a fallas de comunicación UART. Una posible solución es la configuración de timers y contadores para reenvío de mensajes y comandos.

Apéndice A

Contenido CD

A.1. Estructura del CD

Junto con esta documentación, se hace entrega de un CD organizado según el siguiente árbol de directorios:



A.2. Descripción de archivos

El directorio **Codigos** contiene las carpetas **Implementación Contiki** e **Implementación Python** con los archivos fuente implementados para el proyecto.

- **Documentacion doxigen:** Contiene documentación doxigen del código implementado en Contiki.
- **Implementación Contiki:** Contiene los archivos: *project-conf.h, dormir.c, dormir.h, driverCamara.c, driverCamara.h, flash.c, flash.h, nodo.c, nodo.h, radio.c, radio.h, rime-stream.c, sink.c.*
- **Implementación Python:** Contiene los archivos fuente implementados en lenguaje *Python*: *consola_ io.py, scan.py, wsnvision.py.*

La carpeta **Documentacion** contiene la documentación del proyecto en el archivo **tesis.pdf**.

La carpeta **Banco de Imagenes** contiene el archivo **imagenes.xls** desde donde se puede acceder a todas las imágenes que componen el banco generado durante el transcurso del proyecto, especificando los detalles de configuración de cada una de ellas.

A.3. Requisitos del sistema

Para compilar los códigos o realizar cambios en los mismos, es requisito tener instalado el *toolchain* de Contiki OS 2.7[1] y actualizar el archivo cc2538-rf.c que se encuentra en la carpeta Codigos/cambios. Para poder visualizar los archivos relativos a la documentación es necesario disponer de:

- Visor de PDF.
- Ubuntu 12.04 LTS o superior.[2]
- Visor de imágenes.
- Visor de planillas de cálculo, Microsoft Excel o equivalente.

[1] Disponible en agregar pag

[2] Durante el transcurso del proyecto se utilizo una máquina virtual con InstantContiki 2.7 y se puede descargar de la página

Apéndice B

Dificultades en el uso de la cámara

B.1. Defectos de la cámara

Durante la operación de la cámara surgieron algunos inconvenientes debido a defectos de fabrica y errores en la hoja de datos. En este apéndice detallamos estos errores y se deja un registro de las correcciones necesarias.

Defectos en el soporte del lente: Se encontraron 3 defectos en el soporte del lente los cuales se solucionan construyendo soportes nuevos.

- Defecto en la rosca.
- El lente no se encuentra centrado respecto al sensor CMOS.
- El largo del soporte no permite lograr foco utilizando el lente LS-20150.

En primer lugar se comprobó en uno de los soportes que no era cilíndrico sino que tenía una leve forma cónica que no permitía enroscar por completo los lentes para poder tener foco. Para solucionar primero se limó levemente la rosca y se pasó un macho con el pase y diámetro que especifica la hoja de datos del los distintos lentes.

Otro defecto, se muestra en la figura B.2b donde se puede observar que el sensor CMOS no se encuentra centrado al eje óptico. Para corregir ésto es necesario modificar el sistema de soporte para el lente, de forma que el centro del lente quede alineado al centro del sensor. Según las medidas tomadas, el soporte debe moverse -0.51 mm en el sentido del del eje x y 0.123 mm en el sentido del eje y, para que quede centrado. Hasta el momento solo se corrigió provisoriamente para una de las cámaras, destornillando el soporte y moviéndolo hasta tener de forma experimental una imagen centrada. Una vez que se logró esto, se pego el soporte con silicona en esa posición.

Como se puede observar en la figura B.2a existe una gran desviación entre el eje óptico y el sensor. En la figura B.2b se corrigió de forma experimental, como ya se explicó, este defecto. En ambas fotografías el eje óptico pasa por la cruz que se ve en el piso.

Por último, se tuvo que las dimensiones del soporte que trae por defecto la cámara, no permite lograr foco con el lente LS-20150. Para solucionar esto se construyó un nuevo soporte donde se rebajó la altura del mismo. En la figura B.3, la opción 1 muestra las dimensiones que debe cumplir el nuevo soporte.

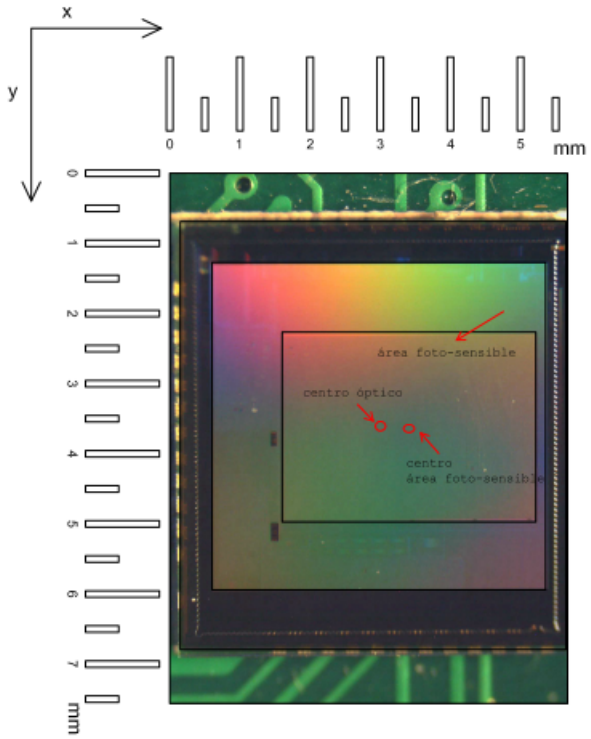


Figura B.1: Posición del sensor CMOS respecto al eje óptico

Hoja de datos: Existen varias hojas de datos para esta cámara que difieren en algunas especificaciones como valores de tensión de alimentación, consumo de corriente. En la hoja de datos tomada como referencia en este proyecto, existen comandos y ACKs erróneos [30]. A continuación se detallan los comandos ya corregidos, que no figuran en la hoja de datos o que tienen alguna diferencia con lo que declara el fabricante en su hoja de datos. Experimentalmente se probaron los comandos Power saving, Quit saving, Control exposición, Modo día y Modo Noche y no se obtuvieron resultados alentadores ni que aporten algo para este proyecto.

(a) (b)

Figura B.2: *(a) Fotografía tomada con la cámara centrada, y soporte por defecto. (b) Fotografía tomada con la cámara centrada, y soporte centrado.*

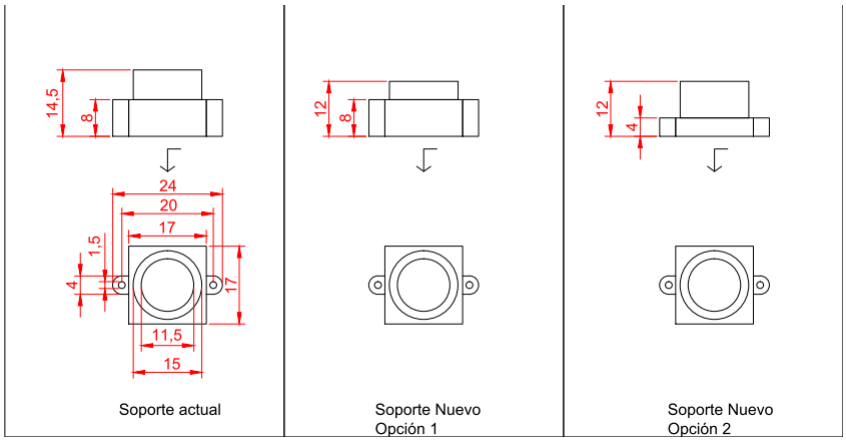


Figura B.3: Dimensiones del soporte original y dimensiones para nuevo soporte.

Comando	Descripción	ACK de comando
56h, 00h, 26h, 00h	Reset de cámara/Init End	0Dh, 00h, 49h, 6Eh, 69h, 74h, 20h, 65h, 6Eh, 64h, 0Dh, 0Ah
56h, 00h, 3Eh, 03h, 00h, 01h, 01h	Power saving	76h, 00h, 3Eh, 00h, 00h
56h, 00h, 3Eh, 03h, 00h, 01h, 00h	Quit saving	76h, 00h, 3Eh, 00h, 00h
56h, 00h, 46h, 00h, XXh	Control exposición	76h, 00h, 46h, 00h, 00h
56h, 00h, AAh, 00h, 00h	Modo día (Filtro IR)	76h, 00h, AAh, 00h, 00h
56h, 00h, AAh, 01h, 00h	Modo Noche (Filtro IR)	76h, 00h, AAh, 00h, 00h

Tabla B.1: Comandos/ACKs para comunicación con la cámara LS-Y201-2MP.

Apéndice C

Stack de comunicaciones Rime

A continuación se detallan las diferentes primitivas que ofrece Contiki OS en su *stack* Rime, y cómo se organizan.

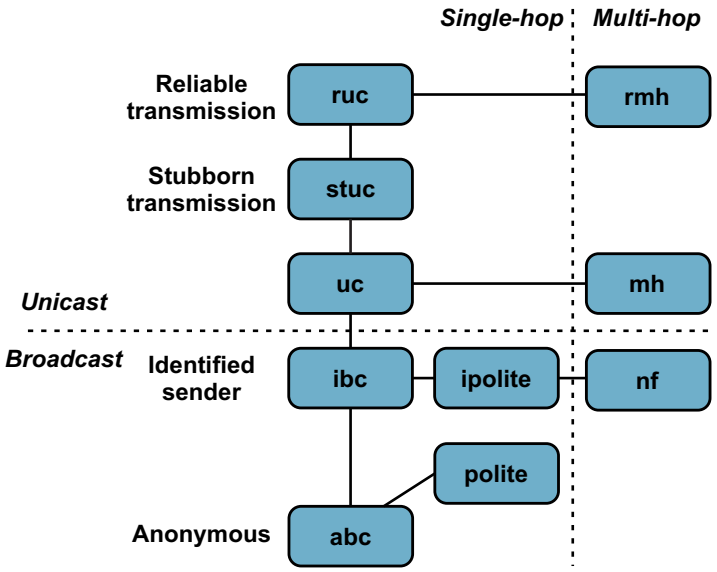


Figura C.1: *Stack* de primitivas Rime

abc: *"Anonymous broadcast"* es la primitiva más básica. Envía paquetes a todos los vecinos que escuchan el canal en el que se envió el paquete. No se envía información de quién envió el paquete. El resto de las primitivas están basadas en esta.

ibc: *"Identified sender"* utiliza la primitiva *abc* agregando la dirección del nodo remitente.

uc: *"Unicast"* envía un paquete a un nodo vecino determinado. Usa la primitiva *ibc* y agrega la dirección del destinatario. Cuando un nodo recibe un paquete chequea la dirección del destinatario, si esta no coincide con la propia, descarta el paquete.

suc: *"Stubborn unicast"* envía repetidamente un paquete usando *uc* hasta que la capa superior cancele la transmisión.

ruc: *"Reliable unicast"* utiliza ACKs y retransmisiones para asegurar que el nodo vecino haya recibido exitosamente un paquete. Cuando el destinatario responde el ACK, el módulo *ruc* notifica a la capa superior. Utiliza la primitiva *suc* para realizar retransmisiones y agrega números de secuencia para confirmar los ACKs de los paquetes enviado.

sibc: *"Stubborn Identified broadcast"* utiliza la primitiva *ibc* enviando un mensaje repetidamente hasta que es cancelado por la capa superior o hasta que se envía un nuevo mensaje.

ribc: *"Reliable Identified broadcast"* agrega a la primitiva *sibc* ACKs y contrrol de secuencia.

polite: *"Polite anonymous best effort local broadcast"* para reducir la cantidad de paquetes enviados verifica si un mensaje ya fue enviado por otro nodo. La capa superior que usa este protocolo setea un intervalo de tiempo en el cual se debe evitar el paquete. Si cuando el intervalo expira no se recibió el paquete que se está por enviar, entonces sí hace el envío.

ipolite: *"Identified polite anonymous best effort local broadcast"* agregar a la primitiva *polite* el identificador del nodo remitente.

nf: *"Best-effort network flooding"* envía un paquete a todos los nodos de la red. Usa la primitiva *ipolite* para cada hop para reducir el número de transmisiones redundantes. Usa el atributo *time to live* para eliminar paquetes.

Apéndice D

Antena externa y extensor de rango CC2592

D.1. Cambio de antena impresa del nodo CC2538-CC2592EM por antena externa a través de conector SMA

El nodo CC2538-CC2592EM brinda la posibilidad de utilizar dos tipos de antenas: una antena impresa tipo F invertida [1, 12] o una externa a través de un conector SMA. Por defecto, el nodo tiene activa la antena impresa. La conexión con el extensor de rango CC2592 se realiza a través del capacitor C3110, como se muestra en la figura D.1.

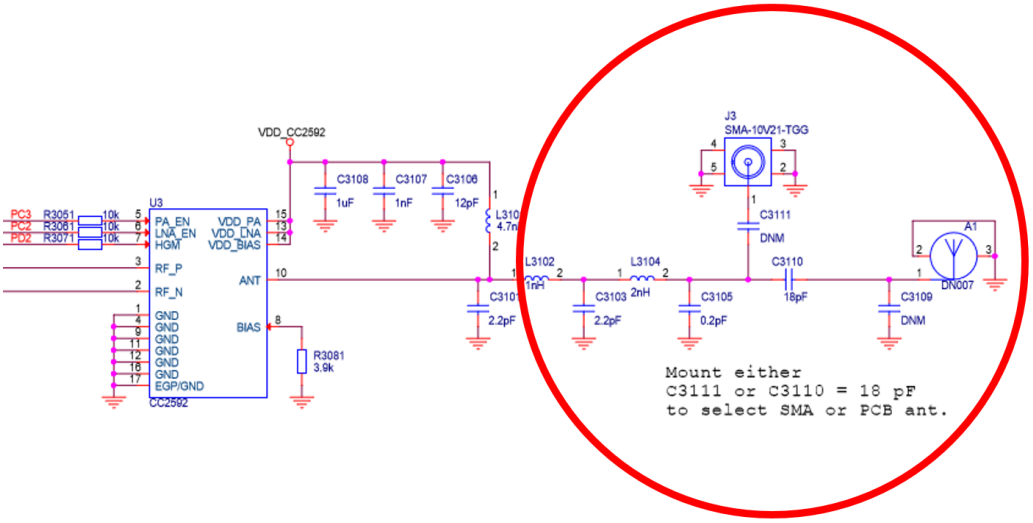


Figura D.1: Selección de antenas de nodo CC2538-CC2592EM a través de capacitor C3110. Imagen tomada de [40]

Para utilizar la antena antena externa a través del conector SMA, el fabricante especifica que se debe rotar este capacitor 90° en sentido antihorario.

En la figura D.2 se muestra un PCB del nodo CC2538-CC2592EM, señalizando la posición de este capacitor. Se tiene que el mismo se encuentra debajo de un blindaje que

debe ser retirado momentáneamente para realizar el cambio.

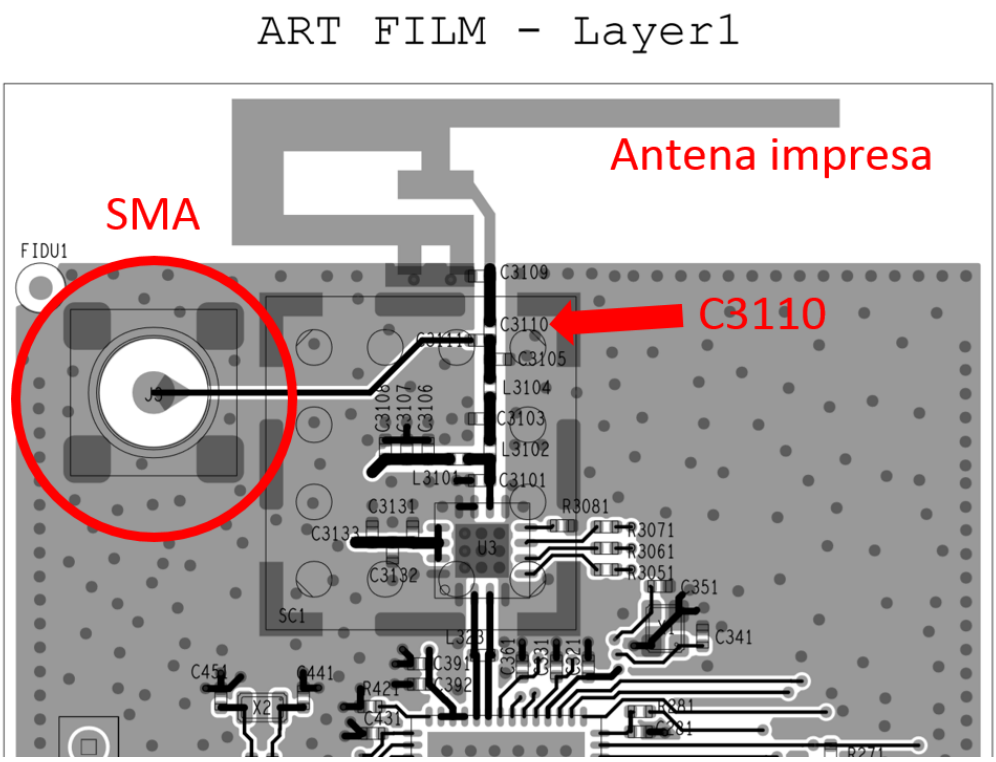


Figura D.2: Ubicación de capacitor C3110 en PCB de nodo CC2538-CC2592EM. Imagen tomada de [31]

D.2. Control del extensor de radio CC2592 [12]

El extensor de rango CC2592 posee tres pines para su control: PA_EN, LA_EN y HGM. Estos pines se conectan al SoC CC2538 como se muestra en la figura D.3.

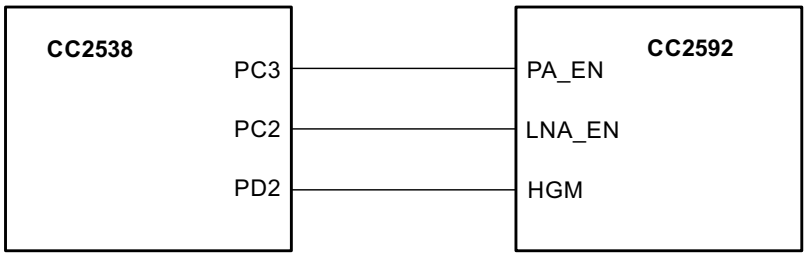


Figura D.3: Interconexión CC2538-CC2592, tomada de [12]

La lógica de control se presenta en la tabla D.1.

PA_EN	LNA_EN	HGM	Modo de operación
0	0	X	Power Down
X	1	0	RX Low Gain Mode
X	1	1	RX High Gain Mode
1	0	X	TX

Tabla D.1: Lógica de control en el SoC CC2538 para manejo del amplificador CC2592

Registro CC2538	Valor
$CCTEST\_OBSSEL2$	81h
$CCTEST\_OBSSEL3$	80h
RFC_OBS_CTRL0	6Ah
RFC_OBS_CTRL1	68h

Tabla D.2: Registros a modificar en el SoC CC2538 para manejo del amplificador CC2592

Para que el CC2538 pueda controlar el extensor de rango a través de los pines PC2 y PC3, es necesario configurar los registros presentados en la tabla D.2.

Para manejar este chip, se modificó el archivo $cc2538 - rf.c$ del sistema operativo, configurando los registros mencionados en la tabla D.2 y modificando el valor de salida de los pines PD2, PC2 y PC3 según se quiera transmitir o desactivar el extensor de radio.

Vale aclarar que los pines PC3 y PC4 son utilizados por la placa programadora SmartRF06EBK para el manejo de los LEDS 3 y 4.

Apéndice E

Plataforma OpenMote

OpenMote [8] es un nodo sensor desarrollado por la universidad de Berkely en 2012 que utiliza el SoC CC2538, utilizado en las primeras etapas del proyecto. A continuación se detallan sus principales componentes.

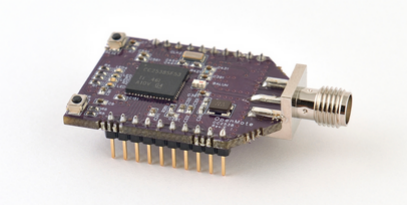


Figura E.1: Plataforma OpenMote, tomada de [8].

- CC2538: SoC de Texas Instruments.
- TPS62730: Es un conversor DC/DC de Texas Instruments con dos modos de funcionamiento, salida regulada en 2.1 V o ByPass 3 V, poniendo a la salida la alimentación de entrada.
- ABM8G: Es un cristal de 32 MHz que se utiliza como reloj para el MCU y radio transceiver.
- ABS07: Es un cristal de 32,768 kHz que se utiliza para el reloj del MCU, RTC (Real Time Clock).
- LEDs: Cuenta con 4 LEDs de color, rojo, verde, amarillo y naranja.
- Botones: Dispone de dos botones, uno de reset y otro de usuario.
- Conector para antena. Permite la conexión de una antena externa.
- Formato XBee: OpenMote es totalmente compatible con el formato XBee.

Existen dos dispositivos que permiten interactuar con el nodo:

- OpenBase: permite programar el nodo mediante un J-TAG o por conexión Micro-USB, conexión por UART mediante conector Micro-USB, Conector Ethernet y acceder algunos Puertos del MCU.
- OpenBattery: Permite alimentar al nodo mediante 2 pilas AAA e incluye en su última versión sensor de humedad y acelerómetro.

En las primeras etapas de proyecto se trabajó con esta plataforma, siendo descartada luego por su alto costo económico en comparación con la plataforma CC2538-CC2592EM [1]. Si bien la misma posee el SoC CC2538, no se encuentra portada de forma correcta en Contiki OS debido a temas constructivos. Aquí se detectaron errores en la asignación de puertos GPIO asociados a LEDs y botones. Por otro lado, al intentar manejar su memoria flash, se detectó que el nodo OpenMote almacenaba porciones de software en diferentes lugares de memoria flash, de forma aleatoria.

[1] 90 dólares es el precio del OpenMote, mientras que es posible adquirir dos plataformas CC2538-CC2592EM por un valor de 100 dólares

Apéndice F

Perfiles de corriente y consumo de carga

Es de gran importancia poder estimar la autonomía del sistema y disponer de la información necesaria para poder estudiar que procesos se deben optimizar si se quiere reducir aún más el consumo del sistema. Para obtener los perfiles de corriente se utilizó el circuito mostrado en la figura F.1, midiendo mediante un osciloscopio la tensión en bornes de una resistencia conectada en serie con la alimentación del sistema.

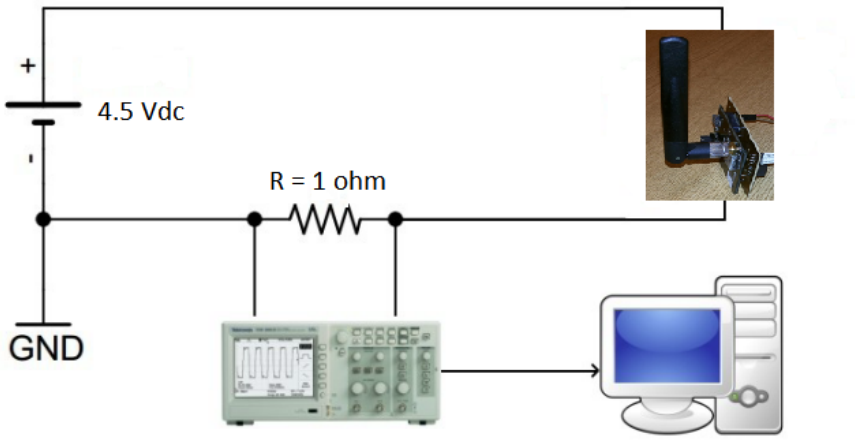


Figura F.1: Circuito de medición.

De esta manera y utilizando la Ley de Ohm se despeja el valor de la corriente que consume el sistema, como se muestra en la ecuación F.2. En este caso se utilizó una resistencia de 1Ω por lo que la escala vertical del osciloscopio refleja directamente el valor de la corriente consumida.

$$I = \frac{V}{R} \tag{F.1}$$

Se tomaron perfiles de corriente correspondientes a los diferentes estados del sistema, procesando los datos obtenidos en Matlab. De aquí se tiene la figura F.2, donde se muestra el perfil de corriente correspondiente a un ciclo en un nodo *WimSN*, enviando una imagen de 50 kB en condiciones de laboratorio.

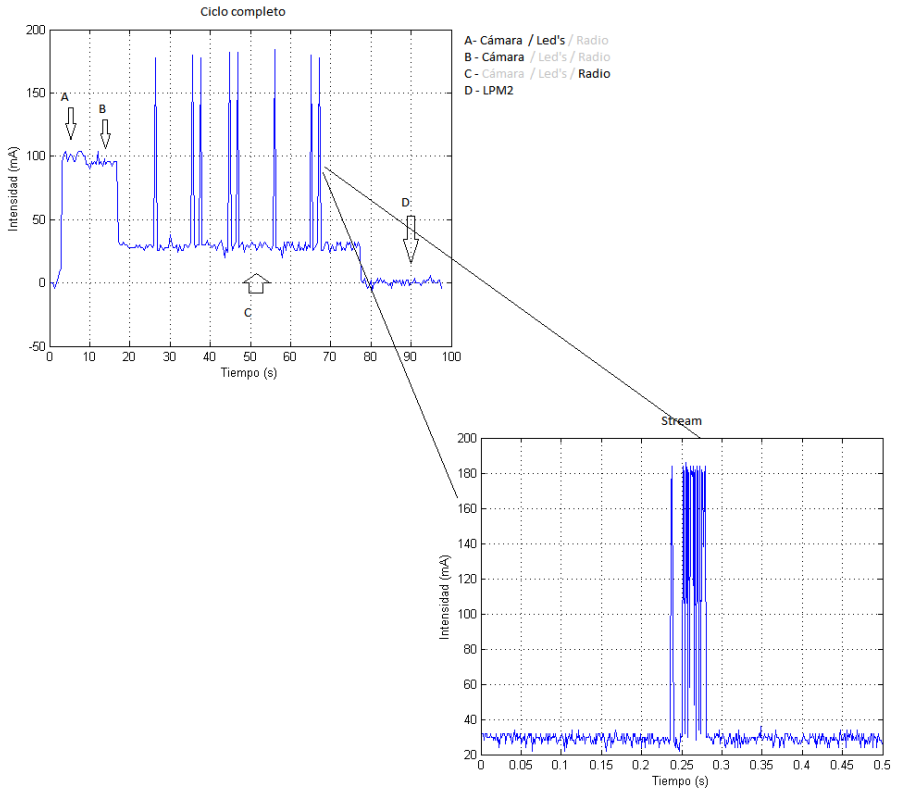


Figura F.2: Perfil de corriente de un ciclo (tomar imagen, transmitir imagen por RF y LPM).

Se puede disgregar el estado del sistema en cuatro zonas:

- Zona A: Un instante luego de despertarse, el nodo enciende cámara, LEDs y toma la foto (5 seg.)
- Zona B: Apaga LEDs y transmite la foto por UART desde la cámara hacia el nodo y almacenándola en flash(8 seg.)
- Zona C: Apaga cámara y LEDs y comienza la transmisión por RF[1] (60 seg.)
- Zona D: Apaga radio y entra en modo de bajo consumo (1 día - 73 seg.)

Utilizando el software de procesamiento Matlab se integra la medida de la diferencia de potencial en el tiempo y se divide entre la resistencia de 1Ω. Una vez que se tiene la carga

[1]Es importante destacar que debido a la resolución del osciloscopio no es posible observar todos los *streams* utilizando esa escala. Por esto se toma posteriormente el perfil correspondiente a un *stream*.

consumida, se puede expresar el resultado en mili-Ampere hora (mAh) para comparar de manera directa con la carga total de las pilas. Para hacer esta conversión basta con dividir entre un factor de 3600 ya que:

$$1mAh = 1mAx3600s \tag{F.2}$$

Se realizan los cálculos de consumo asumiendo que se toma una imagen de 50 kB por día en una red de cinco nodos, con transmisión por RF sin pérdidas y que en modo LPM el sistema consume una corriente de 0.5 mA. Se tiene que el nodo más comprometido (nodo *WimSN* de ID 2) tendría un consumo por día de 14.7110 mAh. Si se utilizan pilas AA de 2800 mAh, la duración de las mismas será de 6.34 meses. En particular los consumos de carga calculados son:

- 1 stream $\simeq$ 0.0092 mAh
- Tomar foto $\simeq$ 0.188 mAh
- LPM $\simeq$ 11.94 mAh

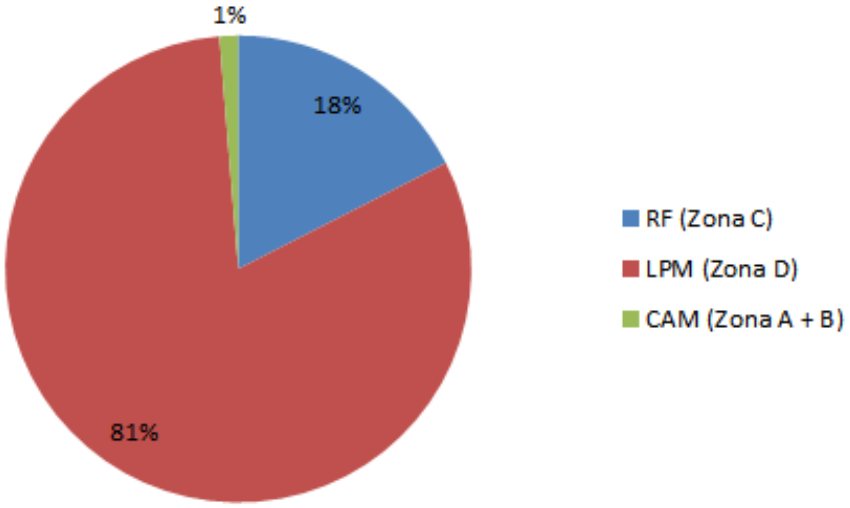


Figura F.3: Peso de los diferentes estados respecto al consumo de carga total.

En la figura F.3 se muestra el peso que tiene cada estado en el consumo de carga, donde claramente se observa que el mayor porcentaje de consumo se da mientras el sistema se encuentra en stand-by, siendo necesario bajar el consumo en corriente del sistema en modo LPM si se quiere alargar la vida útil de las baterías.

Referencias

[1] 2.4 ghz inverted f antenna-design note. http://www.ti.com/lit/an/swru120b/swru120b.pdf. Accedido 20-06-2015.

[2] Contiki community wiki. https://github.com/contiki-os/contiki/wiki. Accedido 17-06-2015.

[3] Getting started with contiki for ti cc2538dk. https://github.com/contiki-os/contiki/blob/master/platform/cc2538dk/README.md. Accedido 15-05-2015.

[4] Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans). https://standards.ieee.org/about/get/802/802.15.html. Accedido 25-06-2015.

[5] Instituto nacional de investigación agropecuaria. http://www.inia.uy/. Accedido 17-06-2015.

[6] Juventud melilla cooperativa agraria. http://www.jumecal.com.uy/. Accedido 17-06-2015.

[7] Ls-20150 2.8 mm de longitud focal m12xp0.5 lente de cámara. http://www.ebay.es/itm/LS-20150-2-8mm-Focal-Length-M12xP0-5-Camera-Lens-/271557924253. Accedido 17-06-2015.

[8] Openmote, open hardware for the internet of things. http://www.openmote.com/hardware/openmote-cc2538-en.html. Accedido 20-06-2015.

[9] Python, programming language. https://www.python.org/. Accedido 20-06-2015.

[10] Radio duty cycling. https://github.com/contiki-os/contiki/wiki/Radio-duty-cycling. Accedido 25-06-2015.

[11] Usb to ttl serial cable - debug. http://www.adafruit.com/product/954. Accedido 17-06-2015.

[12] An130 - using cc2592 front end with cc2538. http://www.ti.com/lit/an/swra447/swra447.pdf, Febrero 2014.

[13] T. Voigt A. Dunkels, O. Schmidt and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *ACM SenSys 2006*.

[14] IEEE Standards Association. Guidelines for 64-bit global identifier (eui-64). https://standards.ieee.org/develop/regauth/tut/eui64.pdf. Accedido 25-06-2015.

[15] University of Southern California Autonomous Networks Research Group. Contiki tutorials. http://anrg.usc.edu/contiki/index.php/Contiki_tutorials. Accedido 23-06-2015.

[16] L. Mitchell B. Pennebaker. *JPEG:Still Image Data Compression Standard*. Springer US, 1993.

[17] T. C. Community. Contiki - the open source os for the internet of things. http://www.contiki-os.org/. Accedido 15-05-2015.

[18] PACIFIC BIOCONTROL CORPORATION. Isomate c plus. http://www.pacificbiocontrol.com/Pacific_Biocontrol_Corporation/Brochures_&_Tech_Sheets_files/CPLUSSpanish.pdf. Accedido 17-06-2015.

[19] Instituto Nacional de Investigación Agropecuaria. Caracterización espacial de los lepidípteros plaga de los frutales de pepita en la zona sur de uruguay. http://www.inia.uy/Publicaciones/Documentos%20compartidos/18429160112092452.pdf. Accedido 21-06-2015.

[20] Universidad de la República Departamento de Protección Vegetal, Facultad de Agronomía. Cydia pomonella. http://www.pv.fagro.edu.uy/fitopato/SSD/Insectos/Cydia/Carpocapsa.html. Accedido 21-06-2015.

[21] BB Electronics. Usb to serial adapter model uc232a. http://www.bb-elec.com/Products/Datasheets/uc232a_4513ds-pdf.pdf. Accedido 20-06-2015.

[22] Fairchild Semiconductor. *FPF2300/02/03, Dual-Output Current Limit Switch Datasheet*, June 2009.

[23] INC GREAT LAKES IPM. Scentry tm wing trap, assembly and use instructions. http://www.greatlakesipm.com/instrscwing.pdf. Accedido 21-06-2015.

[24] S. Nunez I. Scatoni. Estrategias apropiadas para el control de grafolita en montes de durazneros y membrilleros. http://www.fagro.edu.uy/~crs/extension/Gusano%20del%20durazno%20y%20membrillo%20-%20CRS%202008.pdf. Accedido 21-06-2015.

[25] UdelaR Instituto de Física, Facultad de Ingenierína. Aberraciones, seminario de ingeniería óptica. http://www.fing.edu.uy/if/cursos/intr_optica/Material/aberraciones.pdf. Accedido 17-06-2015.

[26] Texas Instrument. Software cc2538 per test v1.1.0. http://www.ti.com/tool/cc2538-cc2592emk. Accedido 20-06-2015.

[27] N. Wainstein J.Schandy, M.González. Plagavision-Desarrollo de un prototipo de detección temprana de plagas en WSN con capacidad visual., 2013.

[28] Linear Technology. *LTC4150 Coulomb Counter*, 2003.

[29] Linksprite. Evaluation utility for linksprite jpeg color camera serial uart interface ls-y201. `http://www.linksprite.com/download/showdownload.php?id=36&lang=en`. Accedido 17-06-2015.

[30] LinkSprite. *LinkSprite JPEG Color Camera Serial UART Interface User Manual*, Julio 2013.

[31] Texas Instrument. *PCB plataforma CC2538-CC2592EM,versión 2.0.*

[32] Texas Instrument. *High-Side Measurement CURRENT SHUNT MONITOR Datasheet*, Diciembre 2000.

[33] Texas Instrument. *CC2538 Development Kit Quick Start Guide*, Abril 2013.

[34] Texas Instrument. *CC2538 Evaluation Module Kit Quick Start Guide*, Abril 2013.

[35] Texas Instrument. *CC2538 System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee/ZigBee IP Applications, User's Guide*, Mayo 2013.

[36] Texas Instrument. *A Powerful System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN and ZigBee Applications, Datasheet*, Abril 2013.

[37] Texas Instrument. *SmartRF06 Evaluation Board User?s Guide*, Mayo 2013.

[38] Texas Instrument. *CC2538-CC2592 Evaluation Module Kit Quick Start Guide*, Febrero 2014.

[39] Texas Instrument. *CC2592 2.4-GHz Range Extender Datasheet*, Febrero 2014.

[40] Texas Instrument. *Circuito Esquemático plataforma CC2538-CC2592EM*, Enero 2014.

Índice de tablas

Índice de figuras

Esta es la última página.
Compilado el jueves 27 agosto, 2015.
http://iie.fing.edu.uy/