MONCEL

Transmisión de una variable fisiológica desde un dispositivo pasivo subcutáneo al celular

MEMORIA DE PROYECTO PRESENTADA A LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE LA REPÚBLICA POR

Ramiro Barrón, Andrea Cukerman, Juan Martín Ortega

EN CUMPLIMIENTO PARCIAL DE LOS REQUERIMIENTOS PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELECTRICISTA

TUTOR

Prof. Ing. Franco Simini ..... Universidad de la República

CO-TUTOR

Prof. Agr. Ing. Juan Pablo Oliver..... Universidad de la República

Montevideo, Uruguay
31 de Julio de 2014

Resumen

MONCEL transmite una variable fisiológica desde un dispositivo pasivo subcutáneo al teléfono celular.

Durante el desarrollo de la enfermedad de la diabetes, la medición y monitoreo continuo de los niveles de glucosa en la sangre es una importante fuente de información para asegurar un control adecuado de la enfermedad. Conociendo esta medida, el paciente es capaz de tomar acciones correctivas para retornar los valores a un nivel aceptado.

Dentro de este marco se diseña MONCEL, un sistema sensor electrónico sin batería capaz de trasmitir continuamente al celular del paciente los valores de la variable fisiológica medida.

En una primera etapa, al no disponer del sensor subcutáneo de glicemia en sangre, se usa un sensor de temperatura. La elección de la tecnología de radio frecuencia como fuente de alimentación permite desarrollar un sistema sensor completamente pasivo capaz de medir y trasmitir el valor de la variable sensada cada vez que es consultado, recibiendo su energía de un sistema externo.

Este segundo dispositivo que durante el proyecto se nombra Pasarela, cumple la función de ser por un lado el lector del sistema sensor, y además, de trasmitir los datos obtenidos via Bluetooth al teléfono celular.

Por su parte, la aplicación de celular desarrollada en Android, recibe y almacena los datos enviados por la Pasarela, y alerta a pantalla y con un mensaje de texto si el valor supera el rango aceptable predefinido por el usuario y/o médico tratante.

MONCEL alcanza valores de ±0.1ºC de incertidumbre en la medida y de distancias de hasta 16m entre el sensor del paciente y su teléfono celular.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Agradecimientos

Dedicamos esta carilla para agradecer a quienes colaboraron con el desarrollo del proyecto. Con temor a olvidarnos de nombrar a alguno, agradecemos a:
Franco Simini, por la continua disponibilidad, soporte conceptual y emocional durante momentos críticos del proyecto.
Juan Pablo Oliver, por la ayuda académica y respondernos consultas en el desarrollo
Andrea Camelliti, por la información y referencias sobre la diabetes
Leonardo Steinfeld, por permitirnos utilizar el FETDebugger más tiempo de lo debido
Jorge Villardino de Biogénesis, por el sensor de temperatura para piel
Nuestras familias, parejas y amigos, por el apoyo moral incondicional y por habernos trasmitido el valor del esfuerzo y estudio con el ejemplo

Y a todos aquellos que de forma directa o indirecta contribuyeron con el proyecto.

MUCHAS GRACIAS

Capítulo 1
Introducción

1.1. Motivación

La diabetes es una de las principales causas de mortalidad en el mundo, produciendo más muertes en el año que la suma del cáncer de seno y SIDA según las estadísticas de la Federación Internacional de Diabetes. El estudio muestra que la enfermedad fue padecida por 382 millones de personas en el año 2013 y se estima que este número se incremente 55% para el 2035 [2][3]

La enfermedad es causada por varios trastornos, siendo el principal la baja producción de insulina. Esta hormona producida por el páncreas se combina con la glucosa transportada en la sangre para que pueda absorberse a nivel celular y ser transformada en energía. Sin la insulina, las células son incapaces de absorber los azúcares, resultando en un incremento de los niveles de glucosa en la sangre. [1]

Existen 3 tipos de diabetes: Tipo 1, Tipo 2 y la diabetes gestacional que ocurre únicamente en el embarazo. En el caso de los diabéticos Tipo 1, el páncreas es incapaz de producir esta hormona por lo que necesitan inyecciones diarias y un control muy riguroso de la glicemia. Generalmente se diagnostica en niños o adultos jóvenes que representan entre el 5 y 10% de todos los casos. El restante de los enfermos es Tipo 2 donde usualmente se produce la llamada Resistencia a la Insulina. El páncreas produce la hormona pero no es aceptada por el cuerpo y se necesitan fármacos adicionales, inyecciones o un plan de ejercicios y alimentación adecuados para evitar complicaciones.

En cualquiera de los dos casos, una baja producción de la hormona resulta en un incremento de los niveles de glucosa que ocasiona entre otros efectos daño en los vasos sanguíneos, afectando principalmente la vista y las extremidades. Si no es controlado puede producir úlceras en los pies, daños en la piel, debiéndose recurrir a amputaciones.

La enfermedad también produce incrementos en el colesterol, que obstruye las arterias y puede resultar en derrames cerebrales o infartos. Las estadísticas proporcionadas por la ADA (American Diabetes Association) muestran que 2/3 de los diabéticos en Estados Unidas mueren a causa de un infarto o un derrame cerebral [1].
Las células a su vez queman los lípidos de reserva del cuerpo para utilizarlos como fuente de energía. Esto produce como efecto secundario la liberación de ácidos en la sangre llamados cetonas y si este proceso continúa se alcanza una cetoacidosis que deriva en el llamado coma diabético.

Estas sintomatologías pueden ser evitadas mediante un riguroso control y un plan de alimentación sugerido por el médico en base al registro de estos valores. La necesidad de un monitoreo continuo de los niveles de glucosa en los diabéticos pasa a ser imprescindible para el control adecuado de la enfermedad. Solamente conociendo esta medida el paciente es capaz de tomar acciones correctivas para que los valores vuelvan a un nivel aceptado.
Desde 1962 los pacientes utilizan glucómetros para el monitoreo de estos niveles. Consisten en instrumentos de medida que utilizan un método electroquímico para obtener el valor de glicemia. Los pacientes deben extraer una pequeña muestra de sangre en una tirilla de papel a través de un dispositivo de punción estéril e insertar la tirilla dentro del instrumento. La glucosa en la sangre reacciona con un electrodo que contiene una enzima que se oxida por medio de un agente. Esta reacción en el electrodo genera una corriente eléctrica y la carga total es proporcional a la cantidad de glucosa en la sangre que ha reaccionado con la enzima.
Estos instrumentos cumplen con los estándares de exactitud, y el costo de los monitores varía entre los USD 50 y 100, y casi USD 1 cada tirilla de papel. [1]

El mayor inconveniente de los glucómetros radica en la falta de continuidad en el monitoreo de los valores. El paciente debe interrumpir sus actividades para controlar las medidas y estar alerta a los síntomas de una posible hyper o hipoglucemia.

Desde los últimos 20 años existen sistemas de monitoreo continuo que consisten en sensores implantables que toman una muestra de sangre periódicamente y transmiten el valor de glucosa inalámbricamente a un receptor o "beeper". La vida útil de la batería del trasmisor es de aproximadamente 5 días, debiendo retirarse cada vez que ésta se agota lo que ocasiona incomodidad y daño. Asimismo los pacientes se vuelven vulnerables al estigma social que cargar con dicho aparato conlleva.
Este sistema de monitoreo continuo es comercializado por sólo 2 empresas en el mundo y su costo, a julio 2014, es superior a los USD 2.000

MONCEL es un proyecto que responde a la necesidad de una medición controlada de la glucosa tomando como prioridad la comodidad de los pacientes, automatizando el procedimiento de análisis sin interrumpir sus actividades.

Se desarrolla un sistema sensor implantable y sin batería, que utilizando únicamente como fuente de energía a un dispositivo externo, logra trasmitir continuamente los valores sensados. Este lector, luego de recibir los datos, los retransmite al celular del paciente el cual lo alertará si el valor está fuera del rango aceptado y además guardará un registro de lo acontecido para una posterior consulta.

Considerando la dificultad del propio sensor de glicemia en cuanto a su obtención o diseño este proyecto utiliza la temperatura como variable corporal para materializar su desarrollo.

1.2. Antecedentes

MONCEL se desarrolla en el marco del Núcleo de Ingeniería Biomédica de las Facultades de Medicina e Ingeniería de la Universidad de la República en Uruguay.

La definición del proyecto surge de la expresión de deseos de varias personas con diabetes y de docentes de las especialidades involucradas.
Se investiga la enfermedad para obtener la información primaria para el planteo del proyecto.

Se utiliza el código del obligatorio C del curso Sistemas Embebidos 2012 del IIE para la configuración del microprocesador elegido.

La propuesta del proyecto fue aprobada como Proyecto Final de Carrera de Ingeniería Electrónica en Abril 2013 bajo la tutoría del Ing. Franco Simini, director del Núcleo de Ingeniería Biomédica y el Ing. Juan Pablo Oliver.

1.3. Objetivo

Registrar y trasmitir una variable fisiológica medida desde un sensor sin batería y subcutáneo a una aplicación de celular.

Estas variables son obtenidas a través de un biosensor que tendrá características de ser implantable y pasivo. Se deberá desarrollar un sistema electrónico embebido que actuará como fuente de energía al sensor predefinido y a su vez deberá trasmitir los datos al celular.

Mediante el manejo de un sistema de alertas en la aplicación, el paciente logrará un monitoreo continuo de la variable medida.

Se elige la temperatura por su facilidad de implementación. Esta unidad podrá ser sustituida o ampliada a los sensores tanto de glicemia u otros como ser sensores que midan la humedad de la piel o la radiación ultravioleta, entre otros.

1.4. Alcance

El proyecto incluye el registro y trasmisión en el tiempo de la temperatura corporal como variable fisiológica.
Dentro del alcance están también el estudio, selección y manejo de la fuente de energía del biosensor, y el desarrollo de la aplicación del celular donde se hará la recepción y almacenamiento de los datos.

Si bien el biosensor será subcutáneo y se diseñará el sistema embebido para cumplir con las especificaciones que sea implantable, el desarrollo del sistema físico a presentar se limitará a una versión cutánea para registrar la temperatura corporal.

1.5. Descripción del Documento

El presente documento incluye los siguientes capítulos y apéndices:

Capítulo 1: Introducción al proyecto. Se presenta una prevé descripción de la diabetes como principal motivación, los antecedentes en los cuales nos basamos durante el trabajo, y el objetivo y alcance definidos en el Curso de Gestión de Proyecto en Abril 2013.

Capítulo 2: Especificación de MONCEL. Definición de principales bloques y flujos de MONCEL y requerimientos de funcionamiento del sistema.

Capítulo 3: Diseño de MONCEL. Criterios de elección de tecnologías de comunicación entre Pasarela - Sistema Sensor y Pasarela – Celular, y diseño de cada uno de los componentes.

Capítulo 4: Implementación. Puesta en funcionamiento de MONCEL. Se destaca: Procesos de calibración en el Sistema Sensor, Interacción de bloques y análisis energético en Pasarela, Diagrama de clases en la Aplicación de celular.

Capítulo 5: Pruebas y Resultados. Se describe la realización de 8 ensayos evaluando la distancia y ángulo entre Pasarela – Sistema Sensor y Pasarela – Celular. Se analiza en particular la dependencia del dato obtenido con la distancia y su corrección de funcionamiento.

Capítulo 6: Gestión del Proyecto. Evaluación del cronograma y análisis de riesgo planificados en Abril 2013. Énfasis en las dificultades encontradas durante el desarrollo del proyecto.

Capítulo 7: Proyecto de Producción. Descripción del diseño de la antena del Sensor para disminuir su tamaño y esquemático de la Pasarela. Estimación de costos para una producción de lote de 10, 100 y 1000 unidades.

Capítulo 8: Conclusiones. Conclusiones de MONCEL y sugerencias de mejoras.

Apéndice A: Cálculo de incertidumbres en la serie de medidas en cada ensayo del Capítulo 5.

Apéndice B: Fotos de pantalla de los ensayos del Capítulo 5.

Apéndice C: Documentación en inglés (Paper) del proyecto.

Apéndice D: Documentación del software de la Pasarela elaborado utilizando la herramienta de generación de código de Doxygen.

Apéndice E: Documentación del software de la Aplicación del celular elaborado utilizando la herramienta de generación de código de Doxygen.

Capítulo 2
Especificación de MONCEL

2.1. Introducción

En el presente capítulo se brinda una idea global del funcionamiento del sistema y sus diferentes partes.

Se detallan en primer lugar sus bloques constitutivos, indicando que función cumple cada uno junto con sus principales características. Posteriormente, se muestra como es el intercambio de datos entre los mismos, y una vez detallado esto se describen todas las restricciones y supuestos tomados en cuenta a la hora de comenzar con el proyecto.
Por último, se listan los criterios de éxito elegidos para poder evaluar el correcto funcionamiento de todo el sistema.

2.2. Diagrama de Bloques

En personas que padecen de diabetes, el monitoreo de los niveles de glucosa resulta imprescindible para un control adecuado de la enfermedad. El paciente necesita un sistema que sea capaz de medir la variable en forma periódica sin interrumpir sus actividades, y contar con un manejo de alertas en el caso que la variable se encuentre fuera del rango para poder actuar en consecuencia.

Bajo este marco se desarrolla MONCEL, un sistema electrónico que toma las medidas de un sensor pasivo y subcutáneo y las retrasmite al celular que alertará al paciente cuando se encuentren fuera del rango prefijado.

El proyecto está dividido en 3 grandes bloques:

1. Sistema Sensor: circuito integrado que toma los datos del sensor.
   El sistema tendrá las características de ser implantable logrando sensar continuamente la variable de interés sin necesidad de interrumpir las actividades del paciente, y sin fuente de energía propia evitando así la extracción del sensor para remover su batería.
   Si bien el desarrollo subcutáneo está por fuera del alcance del proyecto, la necesidad de contar con esta característica es determinante en las elecciones de las tecnologías de comunicación.

Se trabajará con un sensor de temperatura simulando el mecanismo de funcionamiento del sistema como si se contara con sensores de glicemia.

2. Pasarela: sistema embebido externo al paciente que alimenta el Sistema Sensor, recibe los datos obtenidos y los retrasmite a la Aplicación de Celular.
   Tiene como función principal intercomunicar el Sistema Sensor con la Aplicación de Celular. Deberá suministrar la energía suficiente para que el Sistema Sensor logre obtener la medida, y ser capaz de leer e interpretar el dato y retrasmitirlo hacia el celular.
   Contará con una fuente de alimentación propia para alimentar al Sensor y a su propio circuito.

3. Aplicación de Celular: único bloque de interacción con el usuario.
   El usuario deberá descargar la aplicación en su celular y será su único mecanismo de interacción y configuración del sistema.
   La aplicación deberá recibir los datos recibidos de la Pasarela y almacenarlos en una base de datos. Si el valor excede el rango definido por el usuario, manejará un sistema de alertas en el propio celular y vía mensajes de texto avisa a terceros para prevenir una posible complicación.

En resumen, el paciente cuenta con 3 dispositivos físicos: el Sistema Sensor sin batería implantado en el cuerpo que sensa continuamente la variable fisiológica, el dispositivo externo Pasarela que está cerca del celular o cerca del sensor de acuerdo a las tecnologías de comunicación definidas, y el celular donde la aplicación MONCEL lo alertará si el valor recibido se encuentra fuera de rango.

Los bloques definidos son ilustrados en la Figura 1.

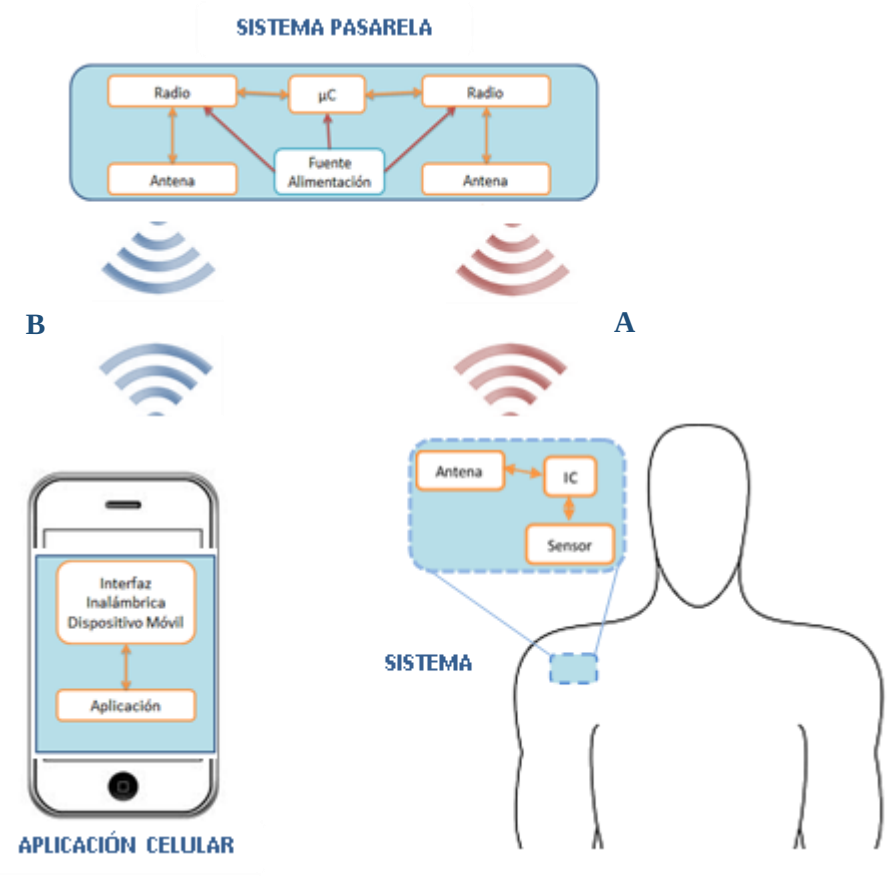


Figura 1 - Diagrama de Bloques funcionales del Sistema MONCEL. Notar que hay dos comunicaciones inalámbricas, A de cercanía con el electrodo/sensor y B entre la Pasarela y el celular.

2.3. Diagramas de flujo

La figura 2 muestra el diagrama de flujo utilizado para la configuración inicial que deberá utilizarse por única vez al momento que el sistema sensor sea implantado. La configuración del sistema sensor (calibración, modos de operación) deberá efectuarse previamente a la implantación por un técnico correspondiente. Esto es detallado en Configuración Inicial del sistema sensor en el capítulo 4.

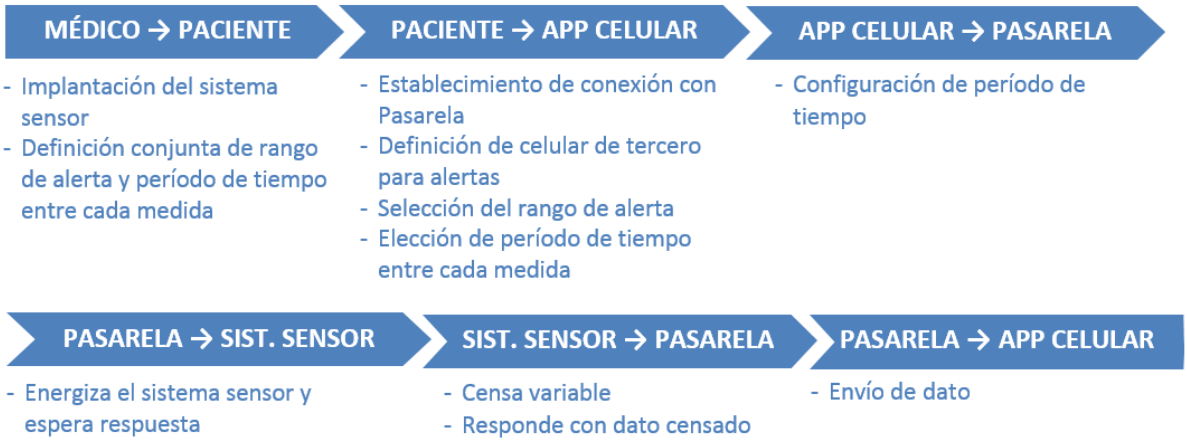


Figura 2 - Diagrama de flujo de comunicación en la configuración inicial de MONCEL

La figura 3 representa la trasmisión sistemática de los datos sensados. El flujo es repetido en cada intervalo definido por el usuario en el punto anterior.

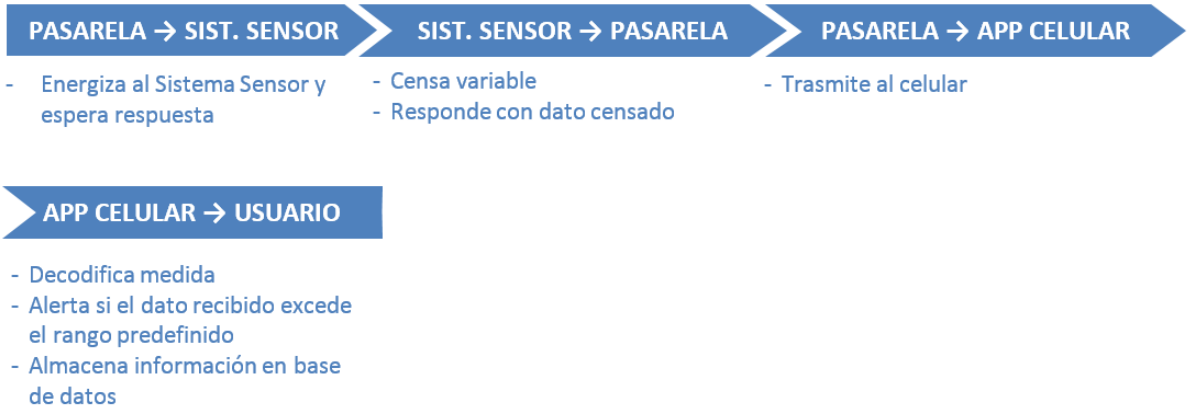


Figura 3 - Diagrama de flujo de comunicación recurrente de MONCEL

En los casos en que la comunicación o el hardware fallaran, se presentan caminos alternativos de respuesta definidos en el capítulo 4 Implementación.

2.4. Supuestos

Los datos asumidos como ciertos a efectos de la planificación del proyecto fueron los definidos a continuación:

- Dispositivo a implementar cumple con normas de seguridad del paciente
- El presupuesto previsto es suficiente
- Dato brindado por el biosensor es correcto con incertidumbres acotadas.
- Los usuarios son capaces de interactuar con una aplicación del celular
- La tecnología de enlace para la trasmisión funcione en el celular
- El celular esté encendido en la medición de los valores
- Existe una plataforma del microcontrolador y de radio enlace que satisfaga los requerimientos del proyecto

2.5. Restricciones

- No se cuenta con tecnología para desarrollar un sensor implantable de glicemia
- MONCEL deberá operar no más allá de 4m de distancia entre el celular y el biosensor.
- El sensor deberá tener características de ser subcutáneo
- El sensor no deberá tener fuente de alimentación propia

Por otra parte, se considera como restricción en el desarrollo del proyecto que el equipo de trabajo tiene una disponibilidad de 15 hrs. Semanales cada uno.

2.6. Actores involucrados

Los actores involucrados directamente en el proyecto son los siguientes:

- Núcleo de Ingeniería Biomédica (NIB)
- Usuarios Finales: pacientes con condiciones crónicas
- Médicos interesados en evaluar la propuesta
- Directores de Proyecto
- Integrantes del grupo de trabajo

2.7. Criterios De Éxito

Los criterios que determinan si el proyecto es exitoso se definieron en el Curso de Gestión de Proyectos en Abril del 2013 y son los siguientes:

- Implementar un protocolo auditable, que transmita el 100% de los valores adquiridos por el biosensor y que el 100% de las medidas trasmitidas sean recibas por el celular
- Alcance de la conectividad con ropa de uso diario en Uruguay
- El error en la medición de temperatura admisible es +/- 0.1 ºC
- La aplicación alerta el 100% de las veces que la medida esté fuera del rango preestablecido (el cual es configurable), e indica fallas en el sistema
- El promedio de la encuesta realizada a 10 personas seleccionadas por los tutores al final del proyecto sea mayor a 4/5 en términos de la amigabilidad y facilidad de uso de la aplicación.
- Autonomía del dispositivo acompañe a la del celular asociado

Capítulo 3
Diseño de MONCEL

3.1. Introducción

En esta sección se analizarán las diferentes tecnologías de comunicación a usar entre el dispositivo pasarela y el sistema sensor considerando los requerimientos y limitaciones propias del medio en la que se encuentran.
En una segunda parte, se describe el diseño general para cada uno de los bloques definidos en el capítulo anterior.

3.2. Tecnología de Radio Frecuencia

El objetivo del proyecto es de registrar y trasmitir una variable fisiológica medida desde un sensor a una aplicación de celular.

Considerando que el sistema sensor deberá permanecer dentro del organismo surge la necesidad que sea completamente pasivo. Esto quiere decir que no contará con fuente de alimentación propia sino que dependerá necesariamente de un dispositivo secundario (*dispositivo pasarela*) encargado de brindarle al sistema sensor la energía necesaria para realizar la medida y re trasmitirla posteriormente al celular.

Teniendo en cuenta la dependencia que existe entre el sistema sensor pasivo y el dispositivo secundario que trasmite la energía, se analiza la posibilidad de utilizar la tecnología RF (Radio Frecuencia) como medio de comunicación entre ambos elementos.

La tecnología consiste en la comunicación de una etiqueta o transponder que contiene la información a trasmitir, junto con su lector o transceiver correspondiente que envía la señal RF para que la etiqueta trasmita la data almacenada.

En particular la etiqueta o tag RF contiene dos elementos fundamentales: la antena de radiofrecuencia que permite la comunicación vía radio, y un pequeño chip o circuito integrado. Dependiendo de la aplicación final del sistema, el substrato donde se encapsula el chip y la antena RF será diferente permitiendo la adaptación de sus características a los requisitos de la adaptación.

Clasificando a las etiquetas según sus características básicas de funcionamiento, procedimos en la elección de su tipología que se adecúe a nuestros requerimientos.

Activas, Semiactivas o Pasivas

Las etiquetas pasivas son aquellas que no poseen fuente de alimentación alguna. La señal recibida de los lectores induce una corriente eléctrica pequeña y suficiente para operar un circuito integrado CMOS, de forma que puede generar y trasmitir una respuesta. Esta respuesta puede ser cualquier tipo de información, no sólo un código de identificación.
Las etiquetas pasivas suelen tener distancias de uso práctico comprendidas entre los 10 cm y llegando hasta los 3 metros, dependiendo de la frecuencia utilizada, el protocolo de comunicación y el tamaño de la antena.

La falta de una fuente de alimentación propia hace que el dispositivo pueda tan pequeño como su tamaño de chip y antena utilizada, resultando también en ser las etiquetas de mayor bajo costo del mercado partiendo desde U$S 0.25 valor a la fecha.[4]

Tanto las etiquetas activas como las semiactivas conllevan una fuente de alimentación propia, descartadas de inmediato en la elección del proyecto.

Por lo tanto la tipología de la etiqueta elegida fue la **PASIVA**.

Write Once/Write Many

Los tags que contienen la característica Write Once, permiten al usuario configurar o escribir su valor una sola vez, después de modificar el valor inicial es imposible alterarlo.
En cambio las etiquetas Write Many, tienen la capacidad de escribir y reescribir tantas veces como se desee.

Debido a nuestra necesidad de reescribir continuamente el valor del dato sensado, la etiqueta deberá tener la característica de **WRITE MANY**.

Anticolisión, Seguridad y Encriptación

Si existen varias etiquetas próximas al lector, éste puede tener dificultad de recibir la información de la etiqueta de interés. La característica de anticolisión permite al tag conocer cuándo debe trasmitir para no entorpecer otras lecturas. Por otra parte algunos tags permiten encriptar la información en la comunicación para evitar la decodificación del mensaje por parte de otros lectores. Este es el caso por ejemplo del pago de crédito a través de la tecnología NFC en los celulares.

En nuestro caso esto último no sería imprescindible, sin embargo sí se deberá contar en la comunicación con un **PROTOCOLO DE ANTICOLISIÓN**.

Rango de frecuencia: LF, HF o UHF

Dentro de las dificultades en los criterios de selección que nos encontramos , la elección de los rangos de frecuencia a utilizar fue el principal motivo de discusión por las limitaciones asociadas a cada opción.

Los elementos considerados al momento de seleccionar el rango adecuado se pueden clasificar en los siguientes:

- Modalidad del Transponder (Activo o Pasivo)
- Distorsión y atenuación del cuerpo humano
- Tamaño de antena
- Alcance de lectura

Previo al estudio de cada uno de los elementos, se identifican los siguientes rangos de frecuencia de transmisión:

- LF (low frecuency) : 3-300 kHz
- HF (high frecuency) / RF (radio frecuency) : 3-30 MHz
- UHF (ulta-high frecuency): 300-3 GHz / Microwave : >3 GHz

LF: Es el sistema menos susceptible a los liquidos y metales. Velocidad de comunicación baja. Rango máximo de lectura no supera los 50cm.

HF: Respuesta en presencia de líquidos es buena aunque no la misma que en bajas frecuencias. Velocidad de comunicación es aceptable para sistemas estaticos o de baja velocidad. Rango máximo de lectura es alrededor de 1m.

UHF: Presenta inconvenientes en la interferencia provocada por metales y liquidos. Velocidad de comunicación alta. Rango de lectura alcanza hasta los 9m.

Las etiquetas que trabajan en UHF necesariamente deben ser etiquetas activas por lo cual se descarta este rango de frecuencias. [5]

Considerando que el sistema sensor deberá ser subcutáneo, se toma como segundo criterio de selección la atenuación y distorsión que presenta las ondas en el cuerpo humano. La tasa de absorción del cuerpo humano es considerablemente más baja en bajas frecuencias que en el rango más alto, lo que implica que la penetración tiende a disminuir a medida que la frecuencia aumenta. De esta manera el rango de frecuencias idóneo en el proyecto sería LF o HF. [6]

No se debe seleccionar la banda de frecuencias sin considerar respectivamente sus tamaños de antenas ya que afectan de manera primordial el tamaño de los componentes. Tomando en cuenta la relación inversa que existe entre la frecuencia y la antena, cuanto menor sea el rango en el que trabajemos, mayor será el tamaño de la antena de trasmisión, resultando impráctico en sistemas implantables. [5]

Esto significó para el proyecto que el rango LF sea descartado. Si bien el tamaño de la antena depende además de la distancia que exista entre el transciever y el transponder, se debe de tomar en consideración la comodidad para el usuario de llevar el dispositivo pasarela consigo.
De esta manera se decide trabajar en el **rango de frecuencias de HF**, sabiendo que aunque no cuenta con la tasa de absorción más baja, las frecuencias permiten un tamaño de antena que puede llegar a ser subcutáneo sin incomodar al paciente.

Por otra parte hay que destacar el sistema de acoplamiento que existe en los diferentes rangos de frecuencias y su sensibilidad a la interferencia electromagnética. Tanto LF como HF funcionan a través de acoplamiento inductivo, mientras que UHF y Microwaves es a través de campos electromagéticos y su sensibilidad varía de acuerdo a estas características. [5]

En nuestro caso esto no debería de ocasionar errores, tomando en cuenta además la experiencia en el área de etiquetas implantables en el rango de bajas frecuencias.

Resumen de criterios de selección de Tecnología de RF

La Tabla 1 muestra un resumen de los criterios de elección para la tecnología de comunicación entre Pasarela y Sensor definidos en la sección anterior.

Tecnología	Frecuencia de Trabajo	Alcance	Permeabilidad en la piel	Tamaño de Antena	Tipo de Transponder
LF	3 - 300kHz	< 50cm	SI	Grande	Pasivo/Activo
HF	3 - 30MHz	~ 1m	SI	Mediana	Pasivo/Activo
UHF	300MHz - 3GHz	< 9m	NO	Pequeña	Activo

Tabla 1 - Resumen de criterios de elección de banda de frecuencias

Como primera aproximación, la tecnología UHF es descartada inmediatamente porque únicamente logra comunicarse con un transponder Activo lo cual se contradice con el objetivo del proyecto. Si bien el tamaño de la antena es el que mejor se adecua al propósito de ser subcutáneo, la permeabilidad de la piel distorsiona la comunicación lo que resulta en errores en los datos recibidos afectando el sistema de monitoreo.

La Baja Frecuencia o LF cumple con los criterios de pasividad del transponder y la atenuación que presentan las ondas en el cuerpo humano son mínimas comparado con el resto de las tecnologías estudiadas. Si bien el alcance es el menor, pero el criterio de éxito vinculado a la distancia de 4m entre el sensor y el celular se podría llegar a alcanzar si seleccionamos una tecnología de comunicación de larga distancia entre la Pasarela y el celular.
El mayor de los inconvenientes de LF radica en el tamaño de la antena. Tomando en cuenta que la comodidad del usuario es un elemento fundamental en el éxito de la implementación, fue descartada esta opción del proyecto.

Finalmente se concluye que la frecuencia de trasmisión utilizada en la comunicación del lector a sistema sensor será de **HF – 13.56MHz**, que cumple con los criterios de permeabilidad, tamaño de antena y tipo de transponer. El alcance deberá ser tomado en cuenta en la selección de la tecnología de comunicación del celular compensando de esta manera la distancia mínima predefinida.

Una vez obtenida la información relevada, el siguiente paso consistió en seleccionar en el mercado los elementos que cumplieran con los requisitos del sistema propuesto.

Selección y compra

Selección del Transponder

En la siguiente tabla se resumen las especificaciones técnicas relevadas para la selección del transponder:

Tipología	Pasiva
WriteOnce/WriteMany	WriteMany
Protocolo de Anticolisión	SI
Rango de frecuencia	HF – 13.56MHz
Distancia max entre Reader y Tag	1m

Tabla 2 - Resumen de especificaciones de etiquetas

Tras investigar las diferentes opciones del mercado de etiquetas pasivas RF que tengan la posibilidad de modificar el valor de identificación continuamente de acuerdo a la lectura de un dispositivo externo (en nuestro caso de un biosensor) seleccionamos las siguientes opciones:

RF430FRL15xH NFC ISO15693 Sensor Transponder / Texas Instruments

Descripciones Generales

El integrado se compone de un chip con un microcontrolador programable MSP430 de bajo consumo. Es capaz de funcionar 100% sin batería, y tiene la opción de conectar un sensor externo resistivo y analógico. Trabaja en el rango de frecuencia de 13.56MHz, y utiliza el protocolo NFC de comunicación. [7]

Si bien cumple con todas las especificaciones mencionadas, a la fecha no se encuentra disponible en ninguno de los distribuidores del fabricante.

Tras contactarnos con técnicos de la empresa de Texas Instruments en Julio 2013, nos mencionaron que podría existir la posibilidad de que se comercialice a partir de Diciembre 2013, fecha a la cual se planificó ya contar con el sistema funcionando. Por este motivo fue descartada la opción del RF430.

MLX90129 13.56MHz Sensor Transponder IC / Melexis Microelectronic Integrated Systems
Descripciones Generales

Este chip es un circuito integrado que es capaz de operar en modo completamente pasivo, que cuenta internamente con un sensor de temperatura, y que además, tiene la capacidad de ser conectado adicionalmente con hasta dos sensores resistivos externos cualesquiera.

Este último punto es de suma importancia en el proyecto si se recuerda que el objetivo de MONCEL es la de medir la glucosa, ya que este chip ofrece la opción de conectar en esta etapa el sensor de temperatura, y pasar luego muy rápidamente a cualquier otro sensor resistivo que se requiera en un futuro.
Además, como puede ser conectado con hasta dos sensores resistivos simultáneamente, el chip brinda también la posibilidad extra de que en un futuro casi que inmediato, se puedan monitorear dos variables fisiológicas en lugar de una. Más aún, si se admitiera la exactitud de su sensor de temperatura interno, las variables podrían llegar a ser hasta 3 diferentes, o mismo la temperatura en 3 puntos distintos del cuerpo.

El rango de la frecuencia de funcionamiento es de 13.56MHz y su protocolo de comunicación se adapta a la norma ISO 15693 (protocolo con características de Anti colisión, ISO 15693-Part 3: Anticollision and transmission protocol) [8]. A su vez, cuenta con la opción de comunicación con un microcontrolador externo a través de un puerto SPI. [9]

En resumen, el MLX90129 de Melexis cumple con todas las características necesarias de acuerdo a la especificación de requerimientos relevada, aunque también se deberá analizar qué tipo de sensor de temperatura externo adicional se deberá comprar que disponga de un mayor de nivel de exactitud que el sensor interno del integrado de ±2.5ºC.

El costo de cada integrado a la fecha fue de U$S 4.56 c/u hasta 10 unidades, y de U$S 132.41 el Evaluation Board del MLX90129 (EVB090129, contiene todos los elementos necesarios para efectuar las pruebas de funcionamiento del MLX90129).

La Tabla 3 muestra una comparación entre las distintas opciones en el mercado que se evaluaron para el sistema sensor.

Contando con únicamente 2 opciones, el RF430FRL15xH NFC ISO15693 Sensor Transponder y el MLX90129 que trabajan en el rango de frecuencias HF que especificamos (cabe mencionar que existieron otras opciones de Microchips y TI pero trabajan a UHF y LF respectivamente por lo cual no satisfacían nuestros requerimientos), y considerando que el sistema de Texas Instruments no estará disponible hasta Diciembre siendo la fecha límite de entrega del proyecto en Abril, se opta por el integrado de Melexis, distribuídos estos por DigiKey y Future Electronics.

	MLX90129	**RF430FRL15xH**	**RFPIC12F675F-I/SS**	**TMS37157**
Rango Frecuencia RF	13,56MHz	13,56MHz	434MHz	134,2kHz
Disponible en Stock	Si	No	Si	Si
Fabricante	Melexis	Texas Instruments	Microchip	Texas Instruments
Precio	U$S 4,56	U$S4,02	U$S2,32	U$S3,47

Tabla 3 - Comparación entre las diferentes opciones de compra para el transponder utilizado en el Sistema sensor

De esta manera, se decide adquirir el **Evaluation Board EVB90129** y **5 chips de MLX90129** a través de la empresa distribuidora DigiKey.

Compra	**Cantidad**	**Proveedor**	**Costo Total**
Evaluation Board EVB90129	1	MELEXIS	U$S 132.41
MLX90129	5	MELEXIS	U$S 22.80

Tabla 4 - Compra de sistema sensor

Selección del Transciever

Ahora que contamos con el sistema sensor de nuestro proyecto, debemos buscar su correspondiente lector para utilizar en el dispositivo pasarela.

Asimismo deberá contar con la opción de conexión de un microcontrolador externo para poder retrasmitir la señal recibida de HF a una señal legible por el celular. Esta elección de tecnología de comunicación entre el dispositivo pasarela y el sistema sensor se analizará en el capítulo 3.

Se analizaron 3 opciones en el mercado:

NFC – Utilización del propio celular como Transciever

NFC es un interfaz inalámbrica entre dispositivos muy similar a bluetooth, sin embargo tiene características similares a los sistemas RF. NFC utiliza campos magnéticos alternantes para establecer la comunicación en el rango de los 13.56MHz. Para establecer la comunicación se necesita que la contraparte este dentro del campo magnético del lector, lo que resulta en rangos máximos de lectura de alrededor de 10cm.

En caso de utilizarse esta solución, se eliminaría el dispositivo pasarela, ya que los celulares de última generación cuentan con lector NFC, algo no menor y que merece una consideración especial en nuestro proyecto. Sin embargo, la utilización del propio dispositivo celular como pasarela a través de NFC, implicaría que el mismo estuviese muy próximo al sistema sensor para cada lectura. Esto resultaría en que el usuario cada vez que desee conocer el valor de la variable a medir, se acerque el celular a la zona del cuerpo donde esté implantado el sensor para obtener la lectura. La continuidad en este caso se vería seriamente afectada, algo que no estamos dispuestos a comprometer en nuestro proyecto.

MLX90121 13.56MHz Transciever / Melexis Microelectronic Integrated Systems

<u>Descripciones Generales</u>

El MLX90121 es un circuito integrado lector de RFID a frecuencias 13.56MHz. Provee hasta 250mW de potencia de radio frecuencia con un voltaje de alimentación de 5V. El chip es configurado a través de una interfaz serial.

A la fecha es desarrollado por Melexis y distribuido por DigiKey y Future Electronics, y la empresa desarrolladora recomienda la paridad entre el MLX90129 y el MLX90121.

Si bien protocolo de comunicación se adapta a la norma ISO15693 por lo que es compatible con el transponder seleccionado, no cuenta con el entramado del dato que se adapte a esta norma, por lo que en caso de adquirirlo sería necesario simular el algoritmo característico del protocolo ISO15693 para poder enviar o recibir un paquete intercambiado con el sensor. [10]

Esto implica en el proyecto una demora en la programación de la codificación y decodificación de la norma que podría llegar a ser evitable por lo que se consideraron otras opciones.

TRF7960EVM 13.56MHz Transciever / Texas Instruments

Descripciones Generales

El TRF7960A es un front-end analógico para un sistema de lectura / escritura RFID 13,56 MHz. Opciones de programación incorporadas hacen que sea adecuado para una amplia gama de aplicaciones para proximidad y sistemas de identificación alrededores, siendo posible configurar el lector se seleccionando el protocolo deseado en los registros de control.

Las características que influyen directamente en el proyecto del TRF7960 EVM incluyen:

- Apoyo a la norma ISO 15693
- Cuenta con una antena de cuadro de a bordo 13.56 MHz
- Se comunica con el software de host en un PC basado en Windows a través de un cable USB estándar
- LED de indicación de Protocolo necesarios para indicar la detección de una etiqueta.
- Soporta tanto paralelo como interfaces de comunicación SPI entre la TRF7960 y el MSP430 integrado (configurable mediante un ajuste de jumpers)
- Lleva integrado un MSP430 de bajo consumo. El TRF7960EVM utiliza el estado de la MSP430F2370 con velocidades máximas de hasta 16 MHz

[11]

La Tabla 5 muestra una comparación entre los diferentes transcievers mencionados.

	TRF7960A	**MLX90121**	**Celular**
Rango Frecuencia RF	13,56MHz	13,56MHz	13,56MHz
Disponible en stock	Si	Si	Si
Manejo del protocolo de comunicación	Si	No	Si
Alcance	10cm	10cm	2cm
Fabricante	Texas Instruments	Melexis	Depende del modelo de Celular
Precio	U$S 100	U$S 138	Costo del celular

Tabla 5 - Comparación entre las diferentes opciones de compra para el transciever utilizado en la Pasarela

Considerando la adaptación de las características mencionadas al proyecto, se decidió adquirir el **TRF7960EVM** de la empresa Texas Instruments a un costo de U$S 100 la compra.

Por otra parte, como la empresa fabricante el sensor elegida recomienda el MLX90121 como el lector asociado al sensor, se decide adquirir además este transciever para contar con una alternativa en el caso que no lograramos la comunicación entre el chip de Texas Instruments y el Melexis por alguna incompatibilidad entre marcas.

Compra	**Cantidad**	**Proveedor**	**Costo final**
Evaluation Board EVB90121	1	MELEXIS	U$S 138.26
MLX90121	5	MELEXIS	U$S 35.43
TRF7960EVM - EVALUATION MODULE FOR TRF7960	1	TEXAS INSTRUMENTS	U$S 100

Tabla 6 - Compra de transciever para pasarela

3.3. Comunicación entre Pasarela y Celular

En esta sección se analizan las diferentes tecnologías de comunicación y/o soluciones para la transferencia de datos desde el dispositivo pasarela hacia la aplicación de celular considerando los siguientes elementos:

- Distancia máxima para la comunicación
- Hardware del propio celular
- Consumo de energía
- Facilidad de implementación.

En el alcance de este proyecto no se tiene en cuenta la modificación del hardware del dispositivo móvil, por lo que para la elección de la solución se tomaron en cuenta las tecnologías con las que cuentan en el 2013 la mayoría de los dispositivos móviles de última generación.

Bluetooth

La tecnología inalámbrica Bluetooth es un sistema de comunicaciones de corto alcance destinado a sustituir el cable entre distintos dispositivos electrónicos. Sus principales características son robustez, bajo consumo de energía y bajo costo.

Existen dos formas de comunicación bluetooth: Basic Rate (BR) y Low Energy (LE). Ambos sistemas incluyen la detección de dispositivos, establecimiento de la conexión y mecanismos de conexión.
La principal diferencia entre ambas queda determinada por el consumo de energía. Bluetooth LE es optimizada para los dispositivos que requieren una máxima duración de batería en lugar de una alta tasa de transferencia de datos. Se estima que consume entre ½ y 1/100 de la potencia de la tecnología de Bluetooth clásico. [12]
Los dispositivos que implementan ambos sistemas pueden comunicarse entre sí. De esta manera un celular con Bluetooth LE es capaz de transferir datos a otro dispositivo con Bluetooth BR y viceversa.

Esta tecnología se encuentra presente en la mayoría de los dispositivos móviles de última generación desde el año 2000. Su funcionamiento no depende de la red de telefonía, solo se limita por la distancia entre el dispositivo móvil y el dispositivo pasarela que puede llegar hasta los 10m dependiendo de la capacidad modelo del celular del usuario.

Una vez configurada la conexión, habilitar bluetooth en el dispositivo móvil permite establecer la conexión automáticamente sin necesidad de una nueva activación por parte del usuario.

Wi-fi

Wi-fi es un mecanismo de conexión entre dispositivos electrónicos que funciona en el rango UHF. Es adecuado para implementar redes inalámbricas, ya que posee alta velocidad de transmisión y seguridad.

Se puede utilizar Wi-fi para establecer la conexión de dos modos: utilizando un punto de acceso inalámbrico o utilizando Wi-fi Direct.

La opción como punto de acceso inalámbrico no se adapta al proyecto, ya que sería necesario un tercer equipo que sería el encargado de gestionar la conexión. El mismo puede no estar disponible en todas las ocasiones y a su vez la interconexión requiere de una configuración previa.

Wi-fi Direct es una norma que permite conectar dos dispositivos Wi-fi sin necesidad de un punto de acceso intermedio, logrando grandes velocidades de transmisión entre dispositivos. Una desventaja de Wi-fi direct es que no permite conectarse a otra red Wi-fi cuando esta activado, perdiendo el usuario la posibilidad de conectarse a internet.

GPRS/Servidor Web

Esta solución consiste en subir los datos obtenidos a un servidor en la web, utilizando una conexión GPRS. Luego, desde el dispositivo móvil utilizando una conexión de datos se descargan desde ese servidor a la aplicación.

La principal ventaja de esta solución es que no hay requerimientos de distancia entre el dispositivo pasarela y el dispositivo móvil. Otra ventaja que presenta esta solución es que los datos pueden ser accedidos por cualquier dispositivo que cuente con una conexión a internet.

La primera desventaja de esta solución es que para que el sistema funcione correctamente el dispositivo pasarela tiene que estar dentro del área de cobertura del proveedor de servicios telefónicos, de otra forma no se subirán los datos al servidor. A su vez ese servicio de datos tiene un costo fijo. Pero no solo se necesita la conexión de datos en el dispositivo pasarela sino que también el dispositivo móvil requiere de una, lo cual le da aún más complejidad y mayor costo a esta solución.

Resumen de criterios

La Tabla 7 muestra un resumen de las principales ventajas y desventajas en la elección de la tecnología de comunicación entre Pasarela y la aplicación de celular definidos en la sección anterior.

Tecnología	Ventajas	Desventajas
BLUETOOTH BR	Popularidad en el mercado de celulares Bajo costo de los módulos BT Seguridad en la trasmisión Bajo consumo de energía del módulo	Alcance de hasta 10m Alto consumo de batería del celular
BLUETOOTH LE	Bajo costo de los módulos BT Seguridad en la trasmisión Bajo consumo de energía del módulo Permite comunicación entre LE y BR Bajo consumo de batería en el celular	Alcance de hasta 10m Presente solo en celulares de última generación
WIFI	Alta velocidad de trasmisión Seguridad en la trasmisión Popularidad en el mercado Alcance de hasta 100m en aire	Necesidad de un punto de acceso para gestionar conexión (Tradicional) Imposibilidad de conectarse a otra red para navegar (Direct)
GPRS	No hay requerimientos de distancia Los datos pueden accederse desde internet Seguridad en la trasmisión	Necesidad de conexión a internet por plan de datos del celular Necesidad de conexión a internet por plan de datos de Pasarela Dificultad en el desarrollo de la aplicación

Tabla 7 - Comparación entre tecnologías de enlace con celular

Si bien las tecnologías de enlace de WiFi y GPRS permiten el mayor alcance entre el dispositivo pasarela y el celular, en el caso de la primera es necesario disponer de un punto de acceso para gestionar la comunicación, lo cual implica que si el paciente se encuentra fuera de su hogar (suponiendo que es allí donde se efectúa la mayor cantidad de trasmisión de datos) deja de existir la conexión entre el celular y la pasarela y por lo tanto el monitoreo continuo de la medida.
En el caso de GPRS, tanto el módulo en la pasarela como el celular del usuario deben de contar con un plan de datos de navegación para permitir que los valores se almacenen en un servidor web y posteriormente sean consultados. Esto implica un costo importante mensual en la implementación innecesario a nuestro criterio teniendo presentes las demás soluciones.

La solución Bluetooth permite un alcance de hasta 10m entre la pasarela y el celular, por lo que se estaría cumpliendo el criterio de la distancia mínima de 4m entre el sensor y el celular. Si consideramos el alcance máximo de 1m de la tecnología de RF seleccionada, se tendrá una distancia máxima de hasta 11 metros entre el sistema sensor subcutáneo en el cuerpo del paciente y su propio móvil. Si se supone que existen paredes y demás obstáculos en la conexión, aunque esta distancia máxima se redujera un 50% se seguiría cumpliendo con criterio de distancia mínima propuesto.

Tanto el Bluetooth standard como el Low Energy son consideradas las mejores opciones dentro del proyecto. Se priorizará la compra de un módulo BT que permita la conexión LE teniendo presente la comunicación bilateral que existe entre ambas tecnologías y el ahorro de energía que posibilita BT LE en el celular del paciente.

Selección y compra

JYMCUBLUETOOTH / Core Electronics

Descripciones Generales

Se trata de un módulo Bluetooth inalámbrico que proporciona una interfaz simple para conectarse a Arduino ®, y otras aplicaciones de microcontroladores.

El módulo proporciona un método para conectarse de forma inalámbrica con un PC o un teléfono Bluetooth para transmitir / recibir datos incrustados como datos GPS, lectura de voltaje ADC y otros parámetros. [13]

BR XB LE4.0 – S2/ Blueradios

Descripciones Generales

El módulo cuenta con un integrado de C2540 de Texas Instruments con lo necesario para poder conectarlo a otros microcontroladores.

Cualquiera de estos integrados deberá ser capaz de conectarse a la UART del microprocesador MSP430F2370 para poder trasmitir el dato del obtenido del sistema sensor al celular. Un esquemático de estos bloques se encuentra descritos en el capítulo 4.

Debido a que el módulo BR XB LE4.0 – S2 no se encontraba disponible en stock de la empresa, se adquirió en EBay, y al momento de la implementación el integrado no funcionaba. Se decidió entonces utilizar el **JYMCUBLUETOOTH** de la empresa Core Electronics a un costo de U$S 8 la compra.

Compra	**Cantidad**	**Proveedor**	**Costo final**
JYMCUBLUETOOTH	1	Core Electronics	U$S 8
BR XB LE4.0-S2	1	BlueRadio	U$S 38

Tabla 8 - Compra de módulo Bluetooth

3.4. Sistema Operativo de celular

En el año 2014, existen las siguientes plataformas en cuanto a dispositivos móviles se refiere: Android OS, iOS, Windows Phone y BlackBerry OS.
A continuación se presenta una breve descripción de las mismas junto con sus principales características.

iOS:

Es una plataforma propietaria, desarrollada por Apple. Es intuitiva, atractiva visualmente, proporcionando a los usuarios una excelente interfaz gráfica. Sin embargo, es una plataforma de código cerrado la cual además, presenta ciertos problemas de compatibilidad frente a dispositivos que no sean desarrollados por Apple.

Windows Phone:

Plataforma propietaria desarrollada por Microsoft. La PC deberá contar necesariamente con el sistema operativo de Windows para desarrollar. Adicionalmente se deberá abonar una tarifa de USD 49 anualmente si se desea publicar la aplicación en Windows Store y USD 99 si se publica en Windows Phone Store, valores a la fecha abril 2014.

BlackBerry OS:

Plataforma propietaria, desarrollada por RIM (Research In Motion). Al comienzo del proyecto fue analizada como una opción viable dentro del estudio, aunque se descartó debido a la falta de popularidad en el mercado. Un año después, la compañía se ha declarado en crisis financiera y no debe ser considerada si el proyecto sufre alguna modificación en el futuro.

Android OS:

Plataforma de código abierto desarrollada por Google. Es una de las plataformas más innovadoras del mercado y pionera en el desarrollo de aplicaciones móviles junto con iOS. La principal ventaja que presenta es de ser libre de distribución y código abierto, lo que implica que el desarrollo no lleve ningún costo asociado. Cuenta además con la mayor tienda de aplicaciones gratuitas.

Decisión del proyecto: adoptar **Android OS** como plataforma para el desarrollo de MONCEL. Esta decisión descarta iOS por incompatibilidad y costos asociados, y Blackberry por escasas persepctivas de difusión en el mercado. No obstante cabe aclarar que de querer desarrollarse para otra plataforma a futuro, no existirá impedimento alguno en cuanto a la compatibilidad con el resto del hardware del sistema. La

aplicación solamente deberá interpretar a través de la antena BT los datos recibidos y ser capaz de manejar un sistema de alertas nuevo.

3.5. Diseño de Sistema Sensor

El sistema sensor es el bloque encargado de medir la variable fisiológica que será monitoreada por el sistema a intervalos predefinidos. Estos intervalos son configurables por el paciente o médico tratante a través de la aplicación de celular.

Este bloque al ser totalmente pasivo, estará la mayor parte del tiempo "desconectado", y cada vez que se quiera realizar una medición de la variable fisiológica, recibirá de forma inalámbrica, un comando proveniente del bloque pasarela el cual hará que se dé comienzo al proceso de medición. Una vez culminado el mismo el resultado medido es consultado por la pasarela.

El Sistema Sensor que se utiliza en MONCEL está compuesto principalmente por el Sensor Tag MLX90129 de Melexis, el cual a su vez tiene conectado, de forma que se detallará en breve, el sensor de temperatura TITANIUM de la empresa Biogénesis[14].

Vale la pena aclarar que el MLX90129 ya cuenta internamente con un sensor de temperatura por lo cual, a priori, resultaría innecesario el tener que conectarle un sensor de temperatura adicional que resulta en un diseño físico más complejo. Sin embargo, éste último se agrega dado que por los requisitos establecidos, el error en la medición de la temperatura no debe ser superior a +/- 0,1ºC. Este requisito no es cumplido por el sensor de temperatura interno que tiene un error de medida de ± 2.5ºC [9]

Como sensor de temperatura externo, dadas las características ya mencionadas del MLX90129, se puede utilizar cualquier sensor resistivo.

En MONCEL se usa el sensor TITANIUM debido a que el mismo es diseñado específicamente como sensor de temperatura para piel, está además ampliamente probado en la industria médica, y tiene un error de ±0.1ºC

Interconexión de los componentes

La configuración empleada para la medición de la temperatura es la de un puente de Wheatstone donde se destaca el hecho de que el voltaje común ya es provisto internamente por el MLX90129. Es decir, la "rama fija" del puente es interna al propio chip lo cual hace que el diseño físico final sea más sencillo puesto que sólo debe agregársele externamente al MLX90129 la rama que tiene al sensor. Esto puede observarse en la Figura 5.

De hecho, también sería posible incluso hasta eliminar la otra resistencia de la rama variable teniendo que conectar únicamente el sensor.

Esto se puede hacer debido a que dentro de las bondades del MLX90129, está el tener una red de resistencias interna totalmente configurable donde, entre otras cosas, es posible conectar el sensor externo a una resistencia en serie variable cuyo valor se puede programar como se especificará en el capítulo 4.

Sin embargo, se opta por conectar la resistencia en serie con el sensor de forma externa de modo de poder controlar más fácil y precisamente el punto de equilibrio del puente, y llevarlo así a una temperatura predefinida.

Se opta por utilizar el puente de Wheatstone dado que la tensión a la salida del mismo resulta proporcional a la variación en la resistencia del sensor, la cual a su vez, es proporcional a la temperatura, y además, es lo suficientemente grande para el rango de temperatura de funcionamiento elegido, como para poder relevar con precisión la misma.
De hecho, para el rango de uso del sistema la resistencia del sensor varía en el orden de los 20-26 Ω por cada décima de grado que varíe la temperatura según lo indica la planilla otorgada por Biogénesis 10k.xls anexada en el CD.

El diagrama con el conexionado descripto del sistema sensor se puede ver en la figura 4.

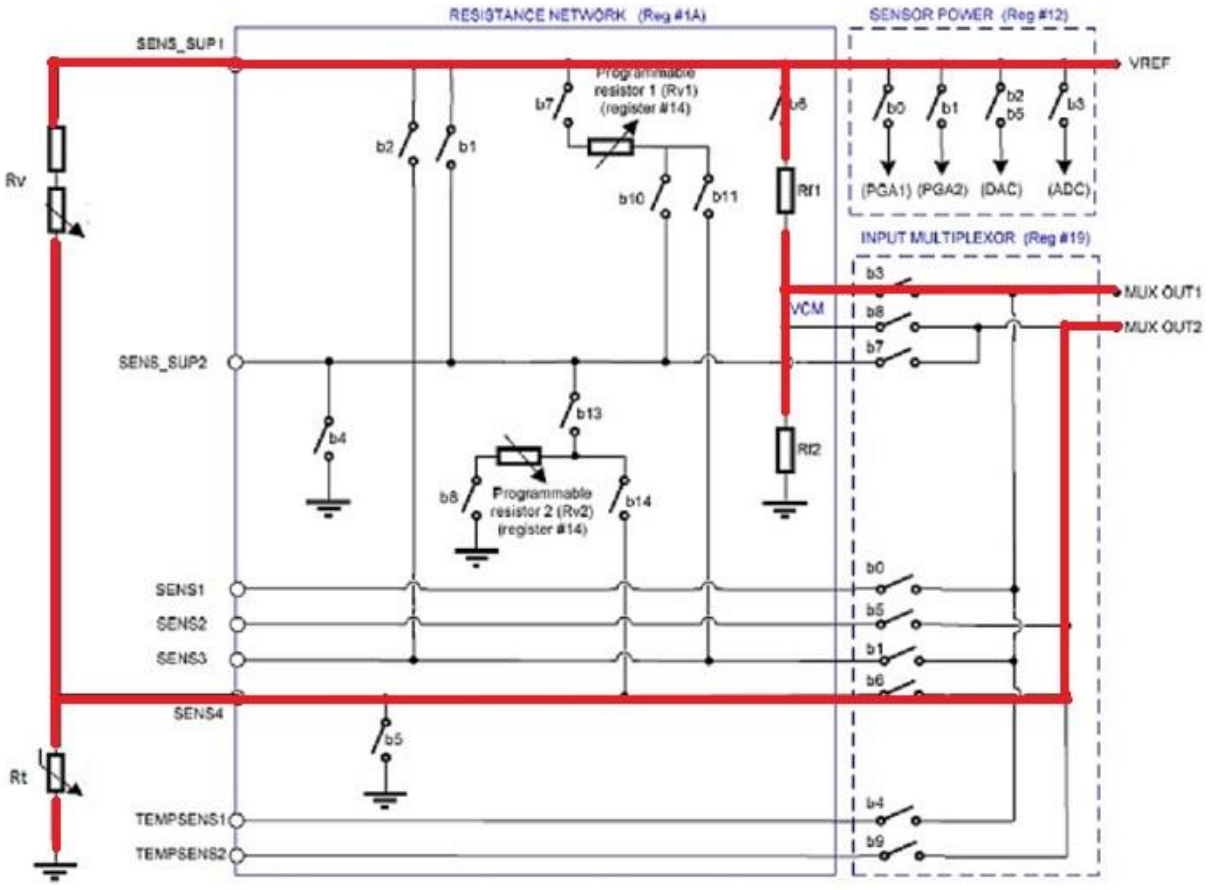


Figura 4 - Diagrama de conexiones del Sistema Sensor tomada de "MLX90129 Datasheet" pag.44 [9]

Esta temperatura predefinida es la de 38,5ºC la cual es el punto medio del rango de interés.

De la tabla otorgada por el fabricante, se tiene que la resistencia del sensor para esta temperatura es de 5,659 kΩ por lo que, para poder obtener el voltaje común de Vref/2, se requiere conectarle en serie una resistencia de igual valor.

Como claramente este no es un valor usual de resistencia, se opta por implementar la misma con la conexión en serie de dos resistencias, una de valor fijo y otra de valor variable el cual se irá ajustando hasta lograr el equilibrio del puente a la temperatura mencionada de 38.5ºC.

Los valores de resistencia que se usan son: 4,7 kΩ + un preset de 1 kΩ.

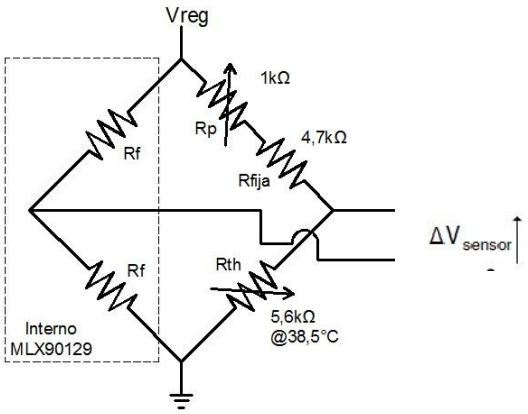


Figura 5 - Puente de Wheathstone para sensar la temperatura de MONCEL

Sensor de Temperatura para piel TITANIUM

El sensor de temperatura empleado, aparte de contar con la precisión requerida de +/- 0,1ºC, tiene también otras ventajas que se detallan a continuación.

Permite detectar muy rápidamente (en el orden de los 20 segundos según los datos proporcionados por el fabricante) las variaciones de temperatura gracias a su ultra rápida lectura térmica.

El estar fabricado en Titanio le otorga una alta relación resistencia/peso la cual es superior al resto de los metales y aleaciones de ingeniería para el rango de uso.

Además, el titanio es un metal muy liviano (pesa un 45% menos que el acero y tan sólo el 60% más que el aluminio), pero a su vez, posee una excelente resistencia tanto a la fatiga como a la corrosión.
Tiene coeficientes de expansión lineal y de conductividad térmica inferiores a los del aluminio y a los del acero aleado, y por si eso fuera poco, el titanio no es magnético. Esto último dato no es menor si se recuerda el requisito que debe ser un sensor completamente pasivo el cual por consiguiente recibirá la alimentación a través de ondas electromagnéticas.

MELEXIS MLX90129

Se decidió adoptar como parte central del Sistema Sensor el circuito integrado de Melexis MLX90129 porque incluye el manejo de un sensor (externo o interno) además del procesamiento asociado a radio frecuencia.

Diagrama de bloques

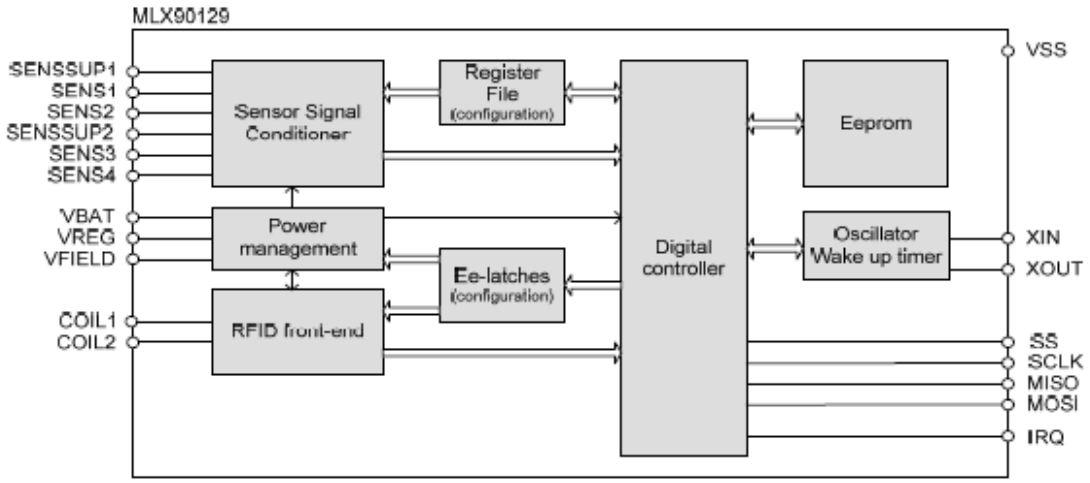


Figura 6 - Diagrama de bloques del MLX90129 tomada de "MLX90129 Datasheet" pag.10 [9]

El bloque ***Sensor Signal Conditioner*** es utilizado para amplificar, filtrar y convertir la diferencia de potencial de los sensores resistivos en una salida digital provista por su ADC de 16bits. Su funcionamiento será explicado con más detalle cuando se explique el procedimiento de configuración y calibración en el capítulo 4.

El bloque de ***Power Management*** es la encargada de administrar los distintos modos de energía del chip: monitorea el nivel de batería en caso que se utilice una batería externa, controla los osciladores y los modos de funcionamiento, y administra también la energía recibida a través del campo magnético por medio del módulo ***RFID front-end,*** almacenándola en un capacitor externo conectado al pin VFIELD. El diagrama de bloques se muestra en la Figura 7. En el caso de MONCEL, como es requisito que el sensor sea totalmente pasivo, el switch entre VFIELD y VBAT deberá permanecer cerrado con el fin de que la alimentación provenga de afuera y no de la batería la cual no estará.

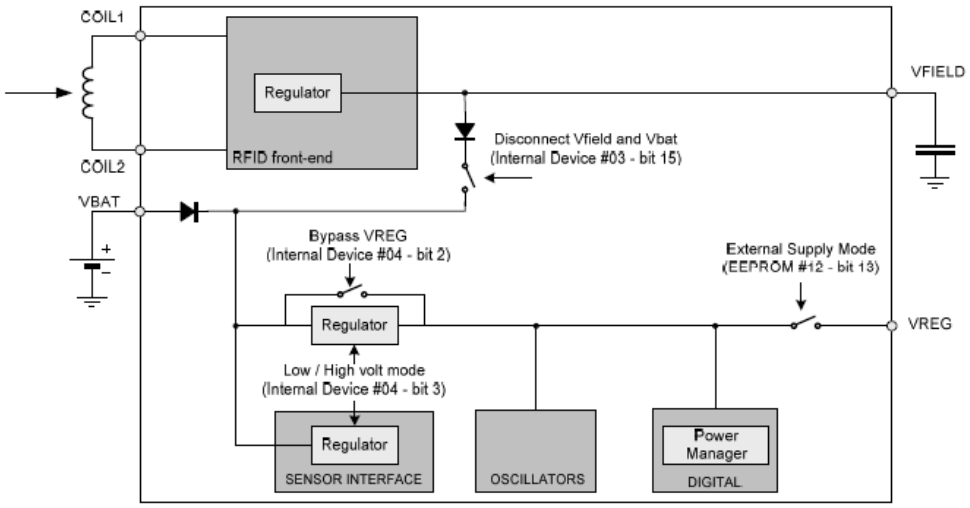


Figura 7 - Diagrama de Bloques de Power Management tomada de "MLX90129 Datasheet" pag.53[9]

El bloque ***Oscillator*** contiene diferentes tipos de osciladores: un oscilador RC interno de muy baja frecuencia (1kHz) utilizado como despertador, un oscilador de alta frecuencia de 5MHz el cual es usado como reloj para toda la parte digital, y un oscilador externo de cuarzo de 32.768kHz, utilizado como base precisa de tiempo.

El bloque ***RFID front-end*** es quien recibe a través de una antena externa un campo magnético a una frecuencia de 13.56MHz proveniente del lector, y a partir del mismo genera su propia fuente de alimentación, recupera el reloj, y controla la modulación de la portadora así también como recupera el dato de la señal entrante.

La interfaz RFID de este chip cumple con los requisitos de la norma ISO15693 lo cual implica que los datos sean recuperados de la señal usando ASK ya sea 10 o 100%, a una velocidad de 26kBit/s con codificación de ¼ de pulso. Asimismo, los datos hacia el lector usan codificación Manchester, y se envían al mismo usando modulación de la carga, también a una velocidad de 26KBit/s, y empleando ya sea una o dos subportadoras (a 423KHz y 484KHz). En la figura 8 se muestra el detalle de la interfaz RFID.

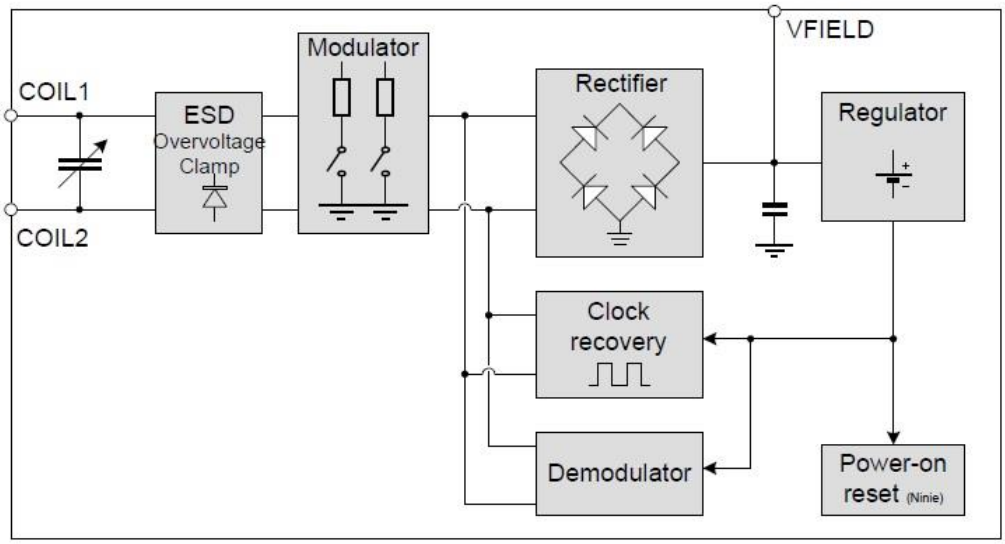


Figura 8 - Esquema de la interfaz RFID tomada de "MLX90129 Datasheet" pag.18 [9]

Finalmente, el bloque ***Digital Controller*** es quien administra el acceso de las diferentes interfaces (SPI, RFID) a las memorias disponibles (***EEPROM***, ***Register File***) y el sensor. Es quien maneja el protocolo ISO 15693 para la comunicación RF y los protocolos de SPI, controla el bloque ***Sensor Signal Conditioner,*** y quien guarda o envía la salida del ADC. Además, tiene también la capacidad de ejecutar otras aplicaciones (como funcionar como Datalogger por ejemplo) gracias a su bloque interno de Direct Memory Access (DMA). La función Datalogger si bien sería la idónea para el sistema MONCEL, no puede configurarse dado que para que funcione debe contarse con una batería externa lo cual no es compatible con el requisito de que el Sistema Sensor sea pasivo. Por ende, no será utilizada en MONCEL.

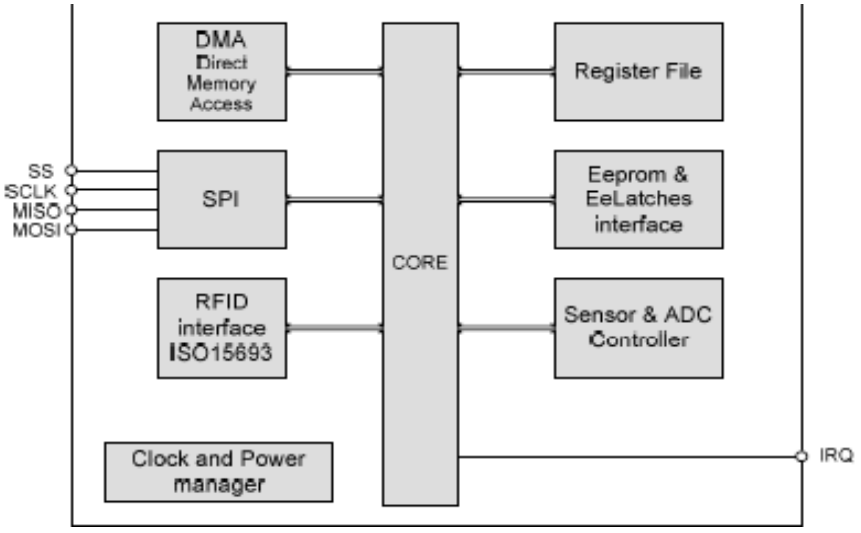


Figura 9 - Diagrama de Bloques del Digital Controller tomada de "MLX90129 Datasheet" pag.11 [9]

Dominios de memoria

El MLX90129 cuenta con 3 dominios de memoria bien diferenciados: la EEPROM, el Register File, y los Internal Devices. Cuenta con la opción de agregar una memoria externa en caso que el usuario requiera más espacio de almacenamiento. Estos dominios se pueden ver en la Figura 10.

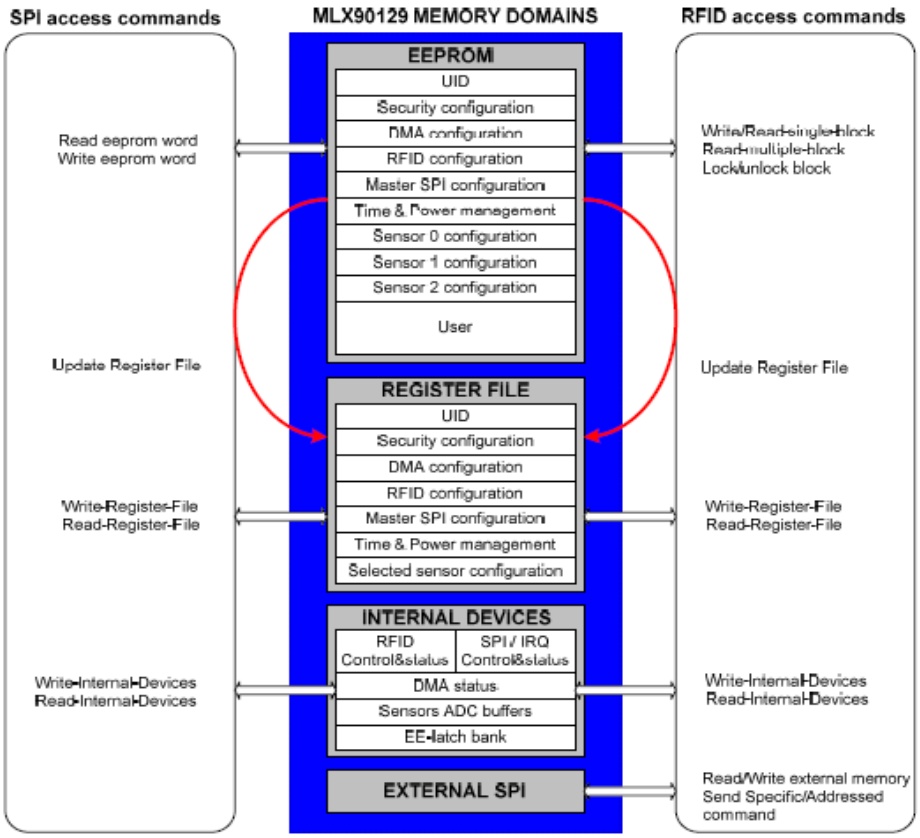


Figura 10 - Dominios de memoria del MLX90129 tomada de "MLX90129 Datasheet" pag. 12 [9]

La ***EEPROM*** es una memoria no volátil de 4Kbits, organizada como 256 palabras de 16 bits, de las cuales 39 se reservan para la configuración, 2 para los valores por defecto del trimming, mientras que las 215 restantes quedan disponibles para la aplicación del usuario (aproximadamente 3,4kbits).

Dado que es una memoria que no depende de la energía y que preserva su contenido aún ante la falta de la misma, se usa principalmente para almacenar los datos definidos por el usuario, así también como una imagen del Register File para que la misma pueda ser cargada inmediatamente después de un Power-On.

El ***Register File*** se usa para guardar la información de todos los parámetros de configuración del chip en todo momento que esté en uso. La misma o bien es cargada desde la imagen guadada en la ***EEPROM*** cada vez que es encendido, o también, a través de comandos del usuario ya sea vía SPI o RFID. Viene a ser la RAM del integrado que es dependiente de la energía por lo que es necesario actualizarlo cada vez que se enciende el chip.

El bloque ***Internal Devices*** no es otra cosa que los registros que se usan para configurar todas las unidades digitales, como ser: la interfaz de los sensores, la interfaz RFID y SPI, el DMA, los buffers del ADC, etc. Todos estos registros pueden ser accedidos también mediante comandos, ya sea vía RFID o SPI, al igual que los del **Register File**, y la diferencia principal con los mismos es que éstos registros no son cargados automáticamente desde la EEPROM.

Los ***EE-Latches*** son otro tipo de memoria no volátil aparte de le EEPROM, que forma parte del bloque Internal Devices. Estos son utilizados cuando los parámetros de configuración deben quedar disponibles inmediatamente. Se usan principalmente para la seguridad del chip, el control de energía, y para guardar el ajuste fino tanto de los osciladores como también de la capacitancia de la antena. Además, por seguridad, los mismos cuentan también con una copia de sus valores en un lugar fijo de la EEPROM.

3.6. Diseño de Pasarela

La Pasarela consta de una placa que contiene el lector de radio frecuencia a 13.56MHz que interactúa con el sensor biológico y se comunica internamente a través de una interfaz serie con un microprocesador de bajo consumo. Se utiliza un módulo BT externo a la placa que establece la conexión BT con el celular y luego transmite por un puerto serie los datos recibidos.

Arquitectura del sistema

Se desarrolló un sistema que cuenta con dos procesos principales independientes, adquisición de datos desde sensor y transmisión de datos al dispositivo móvil. La tasa a la cual adquiere los datos siempre es mayor o igual a la que los transmite, por lo que se implementa un buffer donde se almacenan los datos a ser enviados posteriormente.

El proceso de adquisición consiste en interrogar el sensor para obtener el dato válido y luego almacenarlo en el buffer. La implementación del buffer se realizó utilizando una cola redundante, donde se van almacenando los datos obtenidos.

El proceso de transmisión se encarga de establecer la comunicación con el dispositivo móvil a través del modulo BT. Una vez establecida la comunicación, envía los datos disponibles en el buffer de comunicaciones.

La placa utilizada fue EVMTRF7960A de Texas Instruments, desarrollada para facilitar el desarrollo de aplicaciones que utilizan el integrado TRF960A, lector de radio frecuencia a 13.56MHz, exhimiédonos de todo desarrollo de hardware para esta etapa. Además de este integrado, la placa cuenta con un microprocesador MSP430F2370 y posee una interfaz, serie o paralelo, entre estos dos componentes para establecer la comunicación. Se utiliza un módulo BT externo y se comunica con el MSP430F2370 de la placa de desarrollo a través de la interfaz serie (UART). De esta manera la aplicación en el celular va a recibir los datos biológicos mediante el protocolo Bluetooth.

Plataforma de desarrollo para la pasarela

Dado que la funcionalidad de la pasarela para fundamentalmente por su programación es necesario incorporar una plataforma de desarrollo.

IAR Embedded Workbench dispone un conjunto de herramientas de desarrollo para la construcción y depuración de aplicaciones embebidas usando ensamblador, C y C++.
Ofrece un ambiente de desarrollo integrado, que incluye un administrador de proyectos, editor, herramientas de compilación y depurador. Permite crear archivos de origen y proyectos, construir aplicaciones y depurarlas en un simulador o en el hardware.

Su interfaz de usuario es independiente del microcontrolador elegido, lo que facilita su uso al trabajar

con diferentes dispositivos. Permite utilizar objetivos generales y específicos de apoyo para cada dispositivo.
El compilador IAR C/C++ contiene optimizaciones que garantizan un código de tamaño pequeño, aprovechando todas las características específicas del dispositivo seleccionado.

Caso de uso

Recurrencia Pasarela			
Versión	1.0 (24/06/2014)		
Dependiencias	Archivo Moncel Android		
	Caso de uso Recurrencia Aplicación		
	Caso de uso Configuracion Incial		
Precondición	El caso de uso de Configuración Inicial se ha efectuado exitosamente		
	El sistema sensor se encuentre a menos de 10cm del sistema Pasarela		
	El celular se encuentre a menos de 10m del sistema Pasarela		
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso una		
	vez que se haya efectuado la Configuración Inicial		
Secuencia normal	**Paso**	**Acción**	
	1	El Timer B1 interrumpe al microprocesador	
	2	El microprocesador envia comando de lectura al TRF7960	
	3	El TRF7960 envía comando de lectura al sensor MLX90129	
	4	El sensor MLX90129 comienza proceso de sensado:	
		4.1	El sensor responde a TRF7960 que está ocupado
		4.2	El sensor adquiere el dato y lo almacena en un registro interno
	5	El TRF7960 envía comando de lectura al sensor MLX90129	
	6	El sensor responde con el dato al TRF7960	
	7	El TRF7960 envía dato al microprocesador	
	8	El microprocesador agrega el dato a la cola	
	9	El Timer B2 interrumpe al microprocesador	
	10	El microprocesador envia la cola de datos a través de la UART al módulo Bluetooth	
	11	El módulo Bluetooth envia los datos al celular	
	12	Se realiza el caso de uso Recurrencia Aplicación	
Postcondición	El microprocesador ha recibido correctamente el dato sensado		
	El módulo Bluetooth ha enviado la serie de datos al celular		
Excepciones	Paso	Acción	
	3, 5, 6,	Si el TRF7960 pierde conexión con el sensor:	
	7	E.1.	El TRF7960 comunica al microprocesador el error
		E.2	El microprocesador envía error al celular via el módulo de BT
Comentarios	La excepción mencionada se ha desarrollado hasta el punto E.1.		

Tabla 9 - Caso de uso de Recurrencia Pasarela

3.7. Diseño de Aplicación de Celular

Se realiza habitualmente el desarrollo de programas para Android mediante el lenguaje de programación Java junto con el conjunto de herramientas de desarrollo SDK (Software Development Kit). Las herramientas del SDK compilan el código en un APK: un paquete de Android, archivo con un sufijo apk que contiene la aplicación y es el archivo que los dispositivos con Android utilizan para instalar la aplicación.

Una vez instalado en un dispositivo, cada aplicación Android vive en su propio entorno limitado de seguridad implementando el principio de privilegios mínimos. Esto significa que cada aplicación, por defecto, sólo tiene acceso a los componentes que necesita para ejecutarse.
Sin embargo, existen maneras para que una aplicación logre compartir datos con otras aplicaciones o acceder a los servicios del sistema. Una app puede solicitar permiso para acceder a los datos del dispositivo, como los contactos del usuario, mensajes SMS, el almacenamiento capaz de (tarjeta SD), cámara, Bluetooth pero debe ser concedida por el usuario en el momento de la instalación.
Esto crea un ambiente seguro en el que la aplicación no puede tener acceso a partes del sistema para el cual no se le permite.

Definido el concepto de garantía en la seguridad de acceso de la información, se presentan a continuación tres conceptos básicos y fundamentales para el desarrollo:

Componentes de App

Los componentes definen el marco de la aplicación.
Se constituyen en cuatro bloques esenciales, cada uno con un propósito distinto y ciclo de vida que define cómo se crea y destruye el componente:

1. Actividades:
   Una actividad representa una única pantalla con una interfaz de usuario. Aunque las actividades trabajan en conjunto para formar una experiencia de usuario coherente, cada una es independiente de las demás y como tal, una aplicación diferente puede iniciar cualquiera de estas actividades (si la misma lo permite).

2. Servicios:
   Un servicio es un componente que se ejecuta en segundo plano para realizar operaciones de larga ejecución o para realizar un trabajo para procesos remotos. Un servicio no proporciona una interfaz de usuario. Por ejemplo, un servicio puede reproducir música en segundo plano mientras el usuario está en una aplicación diferente, o puede obtener los datos sobre la red sin bloquear la interacción del usuario con una actividad. Otro componente, como una actividad, se puede iniciar el servicio y dejar que se ejecute o se unen a ella con el fin de interactuar con él.

3. Proveedores de contenido:
   Un proveedor de contenido gestiona un conjunto compartido de datos de aplicaciones. Puede almacenar los datos en el sistema de archivos, una base de datos SQLite, en la web, o cualquier otro lugar de almacenamiento permanente donde su aplicación pueda tener acceso. A través del

proveedor de contenido, otras aplicaciones pueden consultar o incluso modificar los datos (si el proveedor lo permite).

4. Receptores de radio difusión:
   Un receptor de radiodifusión es un componente que responde a transmitir anuncios de todo el sistema. Por ejemplo, una emisión de anunciar que la pantalla se ha apagado, la batería está baja, o una imagen fue capturada. Aunque los receptores de radiodifusión no muestran una interfaz de usuario, es posible crear una notificación de la barra de estado para alertar al usuario cuando se produce un evento de difusión.

Archivo Manifest

Antes de que el sistema Android puede iniciar un componente de aplicación, el sistema debe saber que existe el componente mediante la lectura de archivo AndroidManifest.xml. Su aplicación debe declarar todos sus componentes en este archivo, que debe estar en la raíz del directorio del proyecto de aplicaciones.

Además de declarar los componentes de la aplicación, el archivo logra:

- Identificar los permisos de usuario de la aplicación requiere, como el acceso a Internet o acceso de lectura a los contactos del usuario.
- Declarar el Nivel API mínima requerida por la aplicación, a partir del cual las API de los usos de aplicaciones.
- Declarar características de hardware y software utilizados o requeridos por la aplicación, tales como una cámara, los servicios Bluetooth o una pantalla multitouch.

Recursos

Una aplicación para Android se compone por recursos separados del código fuente, como imágenes, archivos de audio, y todo lo relacionado con la presentación visual de la aplicación.
Las animaciones, menús, estilos, colores y el diseño de interfaces de usuario de una actividad se deben definir utilizando archivos XML. El uso de los recursos de la aplicación hace que sea más sencillo de actualizar varias características de su aplicación sin necesidad de modificar el código y al proporcionar conjuntos de alternativas recursos, Android permite optimizar la aplicación para una variedad de configuraciones de dispositivos (como diferentes idiomas y tamaños de pantalla). [15]

Plataforma de desarrollo

El SDK de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen Linux, Mac OS X 10.4.9 o posterior, y Windows XP o posterior.

La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Eclipse junto con el complemento ADT (Android Development Tools plugin) y es la utilizada en el proyecto aunque cabe destacar que también puede utilizarse un editor de texto para escribir ficheros Java y Xml y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones.

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

Casos de uso

A continuación se describen los dos casos de uso principales definidos en el sistema Aplicación de Celular.
Las restricciones del sistema, supuestos y actores involucrados son definidos en el capítulo 2 del documento.

Configuración Inicial		
Versión	1.0 (24/06/2014)	
Dependencias	Archivo moncel c	
Precondición	El usuario ha incorporado el sistema sensor en el cuerpo de acuerdo a las indicaciones del médico El sistema sensor se encuentre a menos de 10cm del sistema Pasarela El celular se encuentre a menos de 10m del sistema Pasarela El usuario ha descargado la aplicación de celular	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario ejecuta por primera vez la aplicación de celular	
Secuencia normal	**Paso**	**Acción**
	1	El usuario selecciona la aplicación MONCEL en el celular
	2	La aplicación se ejecuta
	3	El usuario selecciona el botón Encender
	4	La aplicación ejecuta proceso de encendido de Bluetooth
		4.1 El usuario otorga permiso de encendido de Bluetooth
		4.2 La aplicación enciende Bluetooth
	5	El usuario selecciona el botón de Configuración
	6	La aplicación despliega pantalla de Configuración
		6.1 El usuario ingresa en la aplicación el Numero de Celular de Alerta con el prefijo del país
		6.2 La aplicación almacena los datos
		6.3 El usuario ingresa en la aplicación el Rango de Alerta
		6.4 La aplicación almacena los datos
		6.5 El usuario ingresa en la aplicación el Tiempo de Muestreo y Tiempo de Trasmisión
		6.6 La aplicación almacena los datos
		6.7 El usuario ingresa en la aplicación el dato de MAC Bluetooth proporcionado por fabricante
		6.8 La aplicación almacena los datos
	7	El usuario selecciona botón de Volver
	8	La aplicación despliega pantalla principal
	9	El usuario selecciona botón Conectar
	10	La aplicación establece conexión con Bluetooth del sistema Pasarela
	11	La aplicación envía al sistema Pasarela los datos de Tiempo de Muestreo y Tiempo de Trasmisión
Postcondición	La aplicación ha almacenado los datos correctamente	
	La aplicación ha establecido conexión con Pasarela	

	Los datos de Tiempo de Muestreo y Tiempo de Trasmisión se han transferido de la aplicación a Pasarela		
Excepciones	Paso	Acción	
	4	Si el bluetooth se encuentra encendido en el celular del usuario:	
		E.1	El botón Encender de la aplicación queda deshabilitado
	6	Si el usuario ingresa en formato incorrecto alguno de los datos mencionados:	
		E.1	La aplicación alerta a pantalla el error
	10	Si no logra establecer conexión con Pasarela:	
		E.1	La aplicación alerta a pantalla el error
Comentarios	El caso de uso Recurrencia continua a Configuración Inicial Las excepciones del Paso 6 y 10 no han sido desarrolladas pero deberán ser consideradas en una producción futura.		

Tabla 10 - Caso de uso Configuración Inicial

Recurrencia Aplicación			
Versión	1.0 (24/06/2014)		
Dependiencias	Archivo moncel c Caso de uso Recurrencia Pasarela Caso de uso Configuracion Incial		
Precondición	El caso de uso de Configuración Inicial se ha efectuado exitosamente		
	El sistema sensor se encuentre a menos de 10cm del sistema Pasarela		
	El celular se encuentre a menos de 10m del sistema Pasarela		
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso una vez que se haya efectuado la Configuración Inicial		
Secuencia normal	**Paso**	**Acción**	
	1	Se realiza el Caso de uso Configuración Inicial	
	2	Se realiza el Caso de uso Recurrencia Pasarela	
	3	La aplicación de celular recibe los datos por Bluetooth	
	4	La aplicación de celular decodifica el dato recibido obteniendo una medida	
	5	La aplicación almacena medida en base de datos	
	6	La aplicación despliega la medida en pantalla junto con la fecha y hora de recepción	
	7	La aplicación compara la medida con el Rango de Alerta definido	
	8	Si la medida se encuentra dentro del Rango de Alerta:	
		8.1	La aplicación alerta a pantalla el mensaje: "Alerta! La temperatura es de *[valor en ºc]*"
		8.2	La aplicación envia mensaje SMS al celular definido con el texto: "Alerta! La temperatura es de *[valor en ºc]*"
Postcondición	La aplicación ha recibido los datos enviados de la Pasarela		
	La aplicación alertó si el dato recibido excedió el rango definido		
	La aplicación ha almacenado en base de datos los valores recibidos		
Excepciones	Paso	Acción	
	3	Si la aplicación pierde la conexión de Bluetooth con Pasarela:	
		E.1	La aplicación notifica a pantalla que se ha perdido la comunicación
Comentarios	La excepción del Paso 3 no ha sido desarrollada pero deberá ser considerada en una producción futura.		

Tabla 11 - Casos de uso Recurrencia Aplicación

Capítulo 4
Implementación

4.1. Introducción

A continuación se presentan los principales puntos realizados durante el desarrollo del proyecto.

En la sección del Sistema Sensor, se podrá encontrar la configuración inicial que describe los registros utilizados y la elección de los parámetros asociados a los mismos, y la calibración del sensor encontrando el mayor rango posible de la conversión para disminuir lo máximo posible la incertidumbre en las medidas.

Se describe la implementación de la Pasarela, adjuntando diagramas de bloques y las descripciones correspondientes del funcionamiento. Se encontrará en esta sección un análisis energético del bloque, la elección de la batería y su máxima duración.

Por último se describe la aplicación del celular, su diagrama de clases y los casos de usos asociados a las diferentes actividades que se presentan. Además se incluyen impresiones de pantallas de la aplicación.

4.2. Sistema Sensor

Configuración Inicial

De la descripción de bloques en la sección de Diseño, resulta claro que antes de poder utilizar el MLX90129 es necesario configurarlo correctamente, a efectos de tener conectado de la forma deseada el sensor de temperatura, y además, para habilitar o no los bloques pertinentes.

Para lograr este cometido hay que, en una primera instancia tener definida la forma de conexión del sensor al chip la cual en este caso ya fue descripta anteriormente, y una vez se tiene eso claro, hay que habilitar los módulos necesarios e implementar dicho conexionado en el MLX90129. Todo esto se realiza en el bloque **Sensor Signal Conditioner** cuyo diagrama de bloques se muestra en la figura 11.

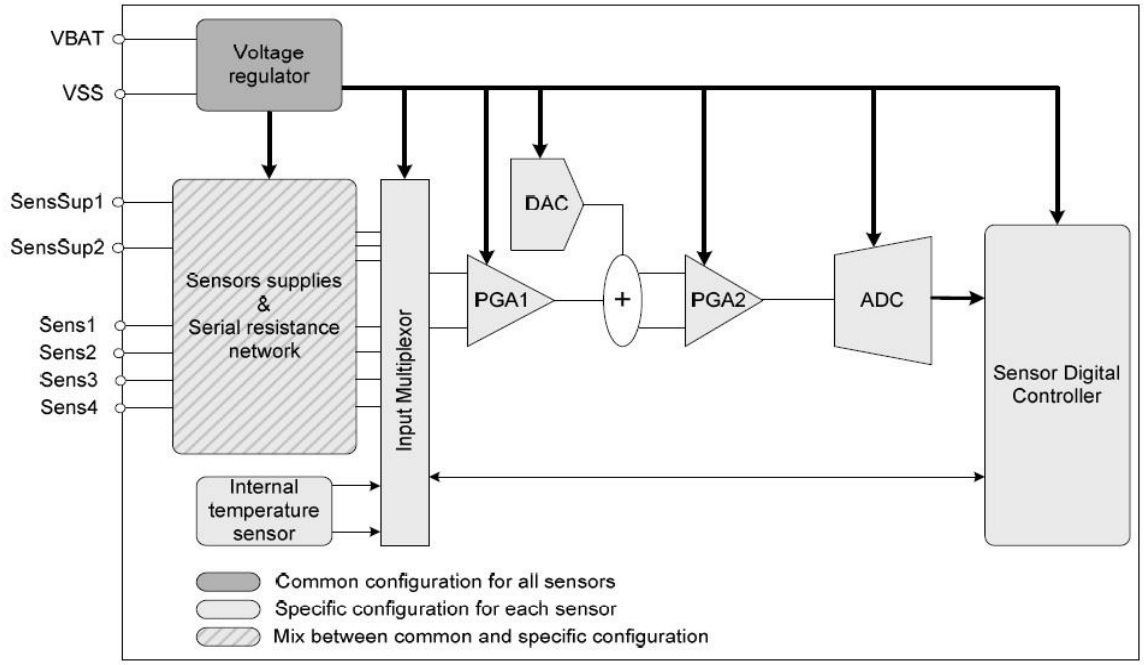


Figura 11 - Diagrama de bloques del Sensor Signal Conditioner tomada de "MLX90129 Datasheet" pag. 43 [9]

Las configuraciones requeridas se hacen mediante los siguientes registros:

1. Sensor Power Configuration Word
2. Programmable resistance
3. Sensor Control Word
4. Sensor low threshold
5. Sensor high threshold
6. Signal Conditioner Configuration Word
7. Connections Configuration Word
8. Serial Resistance Configuration Word

En lo que sigue se describe la configuración a usar para todos estos registros, haciendo hincapié en los parámetros necesarios para configurar MONCEL.

Sensor Power Configuration Word (#12)

Este registro se encuentra en la dirección #12 y es común a todos los sensores del chip.
El mismo es el encargado, principalmente, de habilitar o deshabilitar los módulos presentes dentro del bloque. Esto es, habilita o deshabilita el ADC, los amplificacores PGA1 y PGA2, el DAC, el regulador

de voltaje, y el sensor de temperatura interno. Además, también es quien indica si el sensor externo es también alimentado mediante el regulador interno o no.

En el caso de MONCEL, dado que se usarán todos estos bloques y que el sensor de temperatura externo será alimentado por el propio regulador interno (recordar que el Sistema Sensor se requiere sea pasivo), este registro se debe configurar con el siguiente dato:

Sensor's Power Configuration Word (#12) 007F

Programmable Resistance (#14)

Ubicándose en la dirección #14 de la EEPROM, acá es donde se almacena el valor de las resistencias programables que forman parte de la red de resistencias. Al igual que en el caso anterior, este registro es común a todos los sensores. Las resistencias programables, si bien son una opción válida a la hora de minimizar el diseño, no serán usadas por MONCEL dado que como se explicara anteriormente, se opta por calibrar más finamente el equilibrio del puente, lo cual se hace mediante una resistencia variable *conectada de forma externa*.

Se hace notar sin embargo, que por requerimientos del fabricante, los bits 6 a 9 deben permanecer como están (0100 en dicho orden) por lo que el valor a cargar en el registro es:

Programmable Resistance (#14) 0080

Sensor Control Word (#15 - #1B - #21)

Se encuentran en las direcciones #15, #1B o #21 según se trate de los sensores 0, 1 o 2 respectivamente, y es donde se configura el ADC y las opciones para data logging, las cuales no serán utilizadas por MONCEL.

Se configura el modo de mayor precisión para el ADC (modo 11) por más que es el más lento, dado que las demoras manejadas no son significativas para los tiempos generales del sistema (demora 21 o 42mseg según funcione en modo normal o de bajo consumo respectivamente). Por otra parte, se configura el ADC para que devuelva el promedio de 32 muestras con la mayor precisión.

El valor que se configura entonces es el siguiente:

Sensor Control Word (#15, #1B o #21) C003

Sensor Thresholds (#16-#17, #1C-#1D, #22-#23)

Estos registros definen los umbrales, por arriba y por debajo, que se usan en conjunto con el registro anterior, para el caso del data logging.
En el anterior se configura cuando guardar el dato (si cuando el valor estuviese por encima, por debajo o entre los límites), mientras que en estos es donde se definen dichos límites.

Al no ser usados por MONCEL todos ellos se configuran en 0:

Low Threshold (#16, #1C o #22)	0000
High Threshold (#17, #1D o #23)	0000

Signal Conditioner Configuration Word (#18, #1E, #24)

Esta palabra de configuración es una de las más importantes dado que es donde se guardan tanto las ganancias para los amplificadores PGA1 y PGA2, así también como el valor del DAC a usar por cada sensor.

El procedimiento para la elección de estos valores será explicado en la Procedimiento de Calibración en el Capítulo 4 donde se explica el procedimiento de calibración.

Los valores que usa MONCEL para el sensor de temperatura que tiene conectado actualmente son los mostrados en la Tabla 12:

PGA1	12,6
PGA2	1
DAC	-12
Sensor Conditioner Word (#18, #1E, #24)	028C

Tabla 12 - Parámetros de calibración de la cadena de medida

Sensor Connections Config. Word (#19, #1F, #25) y Serial Resistence Config. Word (#1A, #20, #26)

Son estos registros los que definen la forma de conexión del sensor de temperatura externo (o cualquier otro) con las resistencias internas y el resto del integrado, así también como la forma de conectar la salida de la red de resistencias a la etapa de amplificación que le sigue.

En MONCEL esta conexión ya fue descripta previamente y se puede ver en la figura 4.

Para lograr esto, se deben configurar los siguientes valores:

Serial Resistance Config. Word (#1A, #20 o #26)	0040
Sensor Connections Config. Word (#19, #1F o #25)	0048

Proceso de medición

En la sección que sigue se analiza el proceso de obtención del dato digital tanto desde el punto de vista del Hardware, como también desde el punto de vista lógico a implementar a través del Software.

Hardware

Para poder medir es necesario conocer en primera instancia, cuales son las etapas que tiene el chip para adquirir señales de los sensores.
Las mismas, ya fueron mencionadas en secciones anteriores y son las que se muestran en la siguiente figura 12:

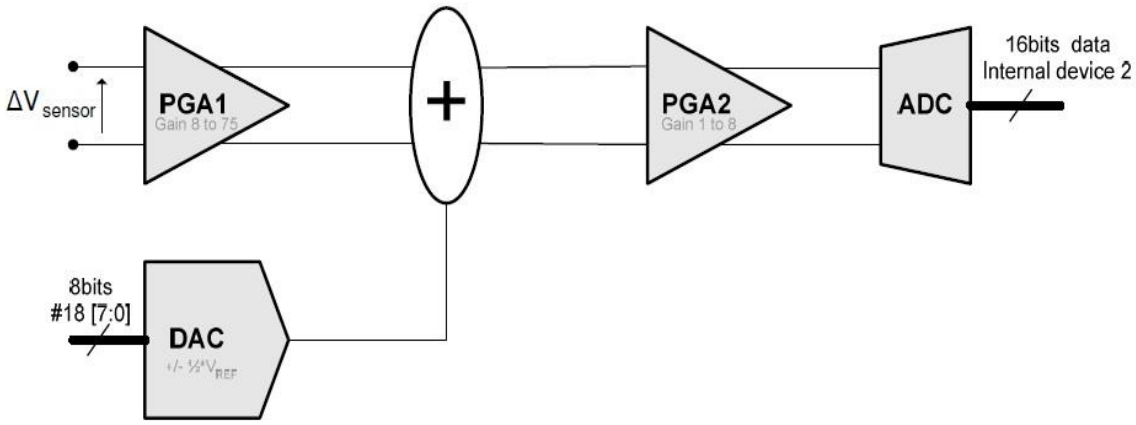


Figura 12 - Cadena de adquisición del MLX90129 tomada de "MLX90129 Datasheet" pag.2 [16]

Como puede verse, la cadena de adquisición del dato consta esencialmente de 2 etapas amplificadoras, a saber PGA1 y PGA2, una compensación de offset dada por el DAC, y finalmente el pasaje a digital realizado por el ADC.

En la primer etapa de amplificación se aumenta el voltaje diferencial presente a la salida del puente de Wheatstone (es el punto de ingreso a la cadena de adquisición del chip) el cual en el caso de MONCEL está en el rango de +/- 75mV para los mayores desequilibrios esperados para el puente acorde al rango de temperatura de interés (que se recuerda es de 35 a 42ºC).

En esta etapa de amplificación la ganancia puede ser ajustada libremente por el usuario en valores discretos que van desde 8 hasta 75 V/V.

Una vez amplificado, al voltaje resultante se le suma otro voltaje proveniente del DAC, el cual es configurado por el usuario teniendo un máximo de compensación de hasta ±Vref/2. Esta tensión se suma con el fin de obtener el máximo rango dinámico posible para la adquisición, lo cual se traduce en intentar llevar el offset generado por el sensor lo más cerca que se pueda de Vref/2.

En el caso de MONCEL, se va a calibrar el Sistema Sensor tratando que el equilibrio del puente se dé para el punto medio del rango de temperatura de interés que resulta en 38,5ºC.

Es decir, se ajustará todo de la mejor manera posible para que a 38,5ºC, en el punto medio de la rama donde se encuentra el sensor se midan Vref/2 volts lo que ocasionará que la salida del puente sea nula, o casi nula.

Una vez amplificado y con el offset ya compensado, el voltaje resultante vuelve a ser amplificado, en esta ocasión por el bloque PGA2. La ganancia de dicho bloque también es configurable por el usuario y puede ir desde 1V/V hasta 8V/V.

Finalmente, la salida del bloque anterior se inyecta directamente a la entrada del conversor analógico-digital, el cual es un ADC de 16 bits de salida que tiene un máximo de hasta 11 bits efectivos.

Además, debe tenerse en cuenta la limitante de voltaje a la entrada del mismo dado que cada entrada debe estar comprendida entre Vref/4 y 3Vref/4. Esto hace que el ADC admita una entrada diferencial de hasta ±Vref/2 , traduciéndose el – Vref/2 en el código de salida 0x0000, y el Vref/2 en el código 0xFFFF.

Todas las etapas antedichas tienen su contraparte matemática en las ecuaciones descritas en el Application Note MLX90129 Acquisition Chain anexada en el CD [16]

Software

Si bien todo lo antedicho es lo que acontece cuando se adquiere un valor desde el punto de vista del Hardware, desde el punto de vista lógico la cosa afortunadamente se simplifica considerablemente.

Esto se debe a que el MLX90129 cuenta con varios comandos propietarios los cuales hacen la tarea más sencilla. Vale la pena aclarar sin embargo, que si bien estos comandos son exclusivos para este integrado, igualmente cumplen también la misma norma ISO 15693. De todos modos, el chip admite también la opción de adquirir los valores de los sensores usando únicamente los comandos nativos provistos en la ISO 15693, a saber, Read Single Block y Write Single Block.

Si bien esta opción sería la ideal dado que es la que permite el más rápido intercambio el día de mañana de un transponder por otro debido a que éste último sólo tiene como requisito el que cumpla también con la citada norma, esta opción está basada en las características de data logging que posee el integrado las cuales en reiteradas veces se ha aclarado ya que no son compatibles con los requisitos impuestos.

Por este motivo, MONCEL utilizará para relevar el valor proveniente del sensor de temperatura, los comandos propios implementados por el fabricante Melexis.

En particular se usará principalmente el comando Read Internal Device XX, donde XX indica cual de los 3 sensores se quiere medir pudiendo ser este valor #06, #07, #08 que hacen referencia a los sensores 0, 1, y 2 respectivamente. Asimismo, también podría ser #02 para leer el buffer del RFID que es donde quedará almacenado el valor digital ya convertido.

La cronología de la medición es la siguiente:

El lector envía al MLX90129 el comando **A21F** (Read Internal Device) seguido de alguno de los valores antedichos para indicar el sensor a medir.

En cuanto el integrado recibe este comando, automáticamente carga la configuración del sensor especificado desde la EEPROM al Register File y comienza inmediatamente la conversión siguiendo los parámetros establecidos en dicha configuración. Como el proceso de conversión, aún en el modo más rápido de funcionamiento del ADC, está en el orden de algunos milisegundos (3,2 según la hoja de datos de MLX90129 página 9), mientras que la respuesta del transponder al lector es enviada luego de aproximadamente 300 micro segundos, la respuesta a este comando será siempre el código de error 0xA1 el cual indica que el dispositivo está ocupado según indica la hoja de datos MLX90129 página 29 [9]

Luego de iniciada la conversión se tiene que leer el dato medido. Para esto se usa el mismo comando **A21F**, pero se lee el Internal Device #02 el cual es el buffer del RFID que es donde se almacena la salida del ADC.

Cabe destacar que el lector debe conocer de antemano en que momento tendrá los datos válidos y disponibles en la salida.Para esto, se pueden implementar dos técnicas distintas.

La primera técnica consiste en estimar aproximadamente la duración de la conversión y leer entonces el buffer de salida una vez se tenga certeza de que el dato ya será válido y estará disponible.

La segunda técnica consiste en leer ciertas flags disponibles en el MLX90129 (nuevamente usando el comando A21F y leyendo la dirección #01 – RFID Interrupt y Status Word), y esperar hasta que las mismas indiquen que la conversión ya se terminó.

Como se tiene interés por ahorrar toda la energía que sea posible, MONCEL intenta minimizar todo lo que se pueda la cantidad de comunicaciones entre el Sistema Sensor y el dispositivo Pasarela. A esto se le suma además, que en el caso de este sistema la duración de la conversión es siempre la misma y conocida pues siempre se mide el mismo sensor y de la misma manera. También hay que mencionar el hecho de que los tiempos usuales entre conversiones son varios órdenes mayores a los tiempos necesarios para realizar la conversión.

Por todo lo anterior, MONCEL implementa la primera de éstas técnicas para obtener el valor.

De todos modos, para no correr riesgos debido a cualquier variación que pueda existir en los tiempos dados por los fabricantes, y para darle así mayor robustez al sistema de medición, entre el comienzo de la medición cuando se le envía al transponder el primer comando, y la consulta al buffer para conocer el dato resultante MONCEL hace una pausa de 2 segundos. Este tiempo es más que suficiente ya que cubre aún el peor caso en cuanto a tiempos de medición. Dicho caso se da cuando se configura el modo de mayor precisión en el ADC (modo = 11), el cual se configura además para devolver el promedio de 32 muestras. Si a eso se le suma el tiempo de inicialización da una demora máxima estimada de 1472mseg.

Con los 2 segundos se tiene un margen de más de un 35% del máximo esperado que es bastante mayor que el 10-15% recomendado por el fabricante incluso.

Procedimiento de calibración

Dado que ahora se cuenta con una idea más o menos clara de cual es el funcionamiento interno del MLX90129 y como es que el mismo logra llegar al valor digital, en lo que sigue se explica el método seguido para realizar la calibración del sensor de temperatura.

Claramente, si bien en esta etapa ya se tiene todo configurado y ya se pueden obtener valores digitales provenientes del ADC, si no se realiza esta calibración del sensor usado para optimizar así su uso en el rango de temperatura deseado, es más que probable que los valores obtenidos o sean incorrectos o sencillamente no se tenga la mejor resolución posible.

Para calibrar el Sistema Sensor hay que tener en cuenta los siguientes puntos:

- obtener el máximo rango dinámico posible
- verificar y compensar el offset introducido en la primer etapa

Basados en un Application Note dado por el fabricante [15] se implementó una planilla de cálculo (adjunta en el CD/ PlanillaCalculoCalibracionSensor) en la cual se generaron, una a una, todas las salidas de las distintas etapas de la cadena de adquisición.

Partiendo del valor nominal de la resistencia del termistor para cada temperatura perteneciente al rango de interés, se generó en primera instancia la salida del puente de Wheatstone (ver figura 5) para cada caso. Una vez obtenida la salida del puente (en mV) se procedió a calcular la salida del primer amplificador (PGA1), a dicha salida luego se le sumó el valor del DAC, se obtuvo después la salida del segundo amplificador (PGA2), y finalmente se calculó la salida que debería dar el ADC.

Como los valores a configurar tanto en el PGA1, PGA2, como en el DAC no son conocidos de antemano, todos estos valores se dejaron como parámetros a definir durante el proceso de calibración.

De todos modos, si bien los valores tanto para el PGA1 como para el PGA2 son discretos y pocos lo que permitiría que los valores óptimos sean hallados mediante el método de prueba y error, no se partió de cualquier valor, sino que se partió de la ganancia total estimada. Es decir, tomando como punto de partida el valor estimado teórico para el producto de PGA1 y PGA2, se comenzaron a variar los mismos hasta que se logró obtener el mayor rango para la salida del ADC sin que el mismo saturara en ningún momento.

Los valores óptimos obtenidos fueron de 12.6V/V para el PGA1 y de 1V/V para el PGA2.

Se puedo ver que si se aumenta el valor del PGA1, para las temperaturas más bajas del rango seleccionado el ADC ya satura, mientras que si se elige un valor menor al dado se comprueba que el rango del ADC disminuye.

Asimismo, si se elige otro valor que no sea el mencionado para el PGA2, se nota que el rango del ADC se incrementa considerablemente, pero a la vez, también hace que el ADC sature.

Una vez establecidos estos valores, se comenzó a variar el valor del DAC, el cual no afecta el rango del ADC, hasta que se obtuvo la menor diferencia posible para el valor de equilibrio del puente (38,5ºC), el cual corresponde al punto medio del rango (correspondiente al valor 8000h del ADC). En el caso del chip y sensor utilizado el mejor valor encontrado para el DAC fue de: 8C.

Sistema Sensor implementado

A continuación se muestra en la Figura 13 el Sistema Sensor a un lado de una tarjeta de crédito a modo de comparación del tamaño físico. Las dimensiones para el Sistema Sensor son de 8cmx6cmx1cm.

Está compuesto por la placa de evaluación del MLX90129 proveniente de la empresa Melexis, y cuenta con un sensor de temperatura externo (cable de color blanco en la figura 13) de 1m de largo el cual tiene adosado en su punta un termistor de 10kΩ que se apoya debajo del brazo como se haría con un termómetro de mercurio.

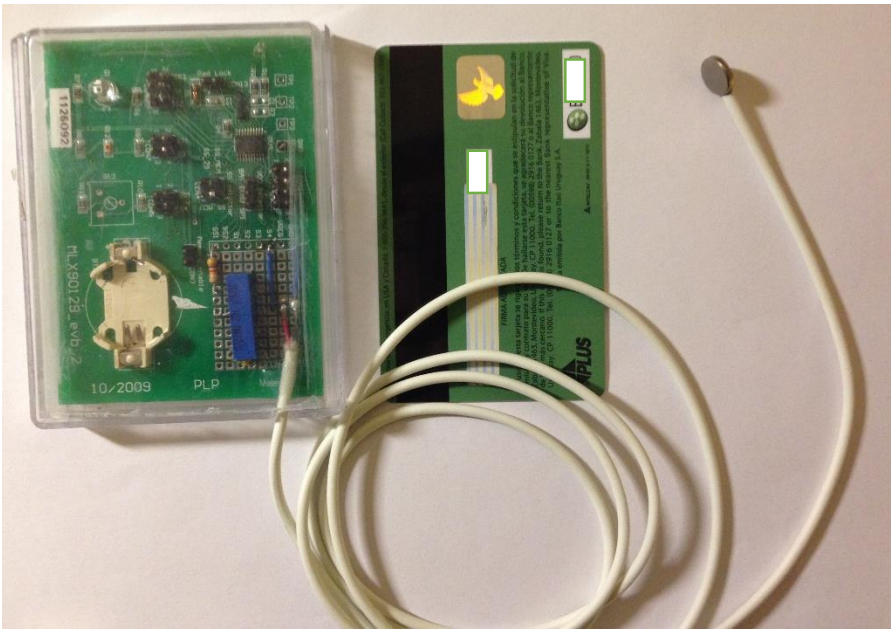


Figura 13 - Fotografía del Sistema Sensor de MONCEL en comparación con una tarjeta de crédito

4.3. Sistema Pasarela

De las especificaciones del proyecto surge la necesidad de implementar un intermediario entre el sensor y el dispositivo móvil. Este dispositivo debe ser capaz de comunicarse con el sensor mediante un enlace de RF y con el dispositivo móvil mediante BT.

En MONCEL se utilizó la placa de evaluación EVMTRF7960A de Texas Instruments, desarrollada para facilitar el desarrollo de aplicaciones que utilizan el integrado TRF960A de dicha empresa.
Sus principales componentes son el integrado TRF7960A y un microprocesador MSP430F2370. Posee una interfaz, serie o paralelo, entre estos dos componentes para establecer la comunicación. Se utiliza la interfaz en paralelo por ser más robusta.

Además de la placa mencionada anteriormente se utilizó un módulo BT serie. Este pequeño dispositivo establece la conexión BT y luego transmite por un puerto serie los datos recibidos.
Se utiliza la interfaz serie (UART) del MSP430F2370 de la placa de desarrollo para comunicar la placa de desarrollo con este dispositivo.
La interfaz serie del microprocesador, no es accesible en la placa de desarrollo, ya que se utiliza para comunicar la misma a través de una interfaz USB que no utilizamos en nuestro desarrollo. Se cortaron las pistas que unen el integrado de la interfaz USB con la UART del microprocesador, y se soldó la interfaz serie del módulo BT en su lugar.

El TRF7960A es un integrado para desarrollar sistemas de radio frecuencia que funcionan a 13.56MHz. Su configuración mediante registros internos lo hace adecuado para una gran variedad de aplicaciones, ya sea como adaptador de las señales analógicas o como codificador/decodificador de los estándares de RFID tales como ISO15693, ISO14443A/B entre otros. El TRF7960A se comunica con un microprocesador utilizando una interfaz que puede ser serial o paralela, utilizando registros internos para la correcta comunicación.

La familia de microprocesadores MSP430 de Texas Instruments, construida con CPU de 16bits, está diseñada para aplicaciones de bajo consumo energético. Incluyen periféricos analógicos y digitales, y algunas series ofrecen funcionalidades más avanzadas como cifrado AES, RF embebido entre otras. Su programación se realiza a través de JTAG o a través de bootstrap loader (BSL) mediante RS-232. Texas Instruments pone a disposición del desarrollador una gran variedad de documentos que facilitan el desarrollo con estos microprocesadores.

Diagrama de Bloques

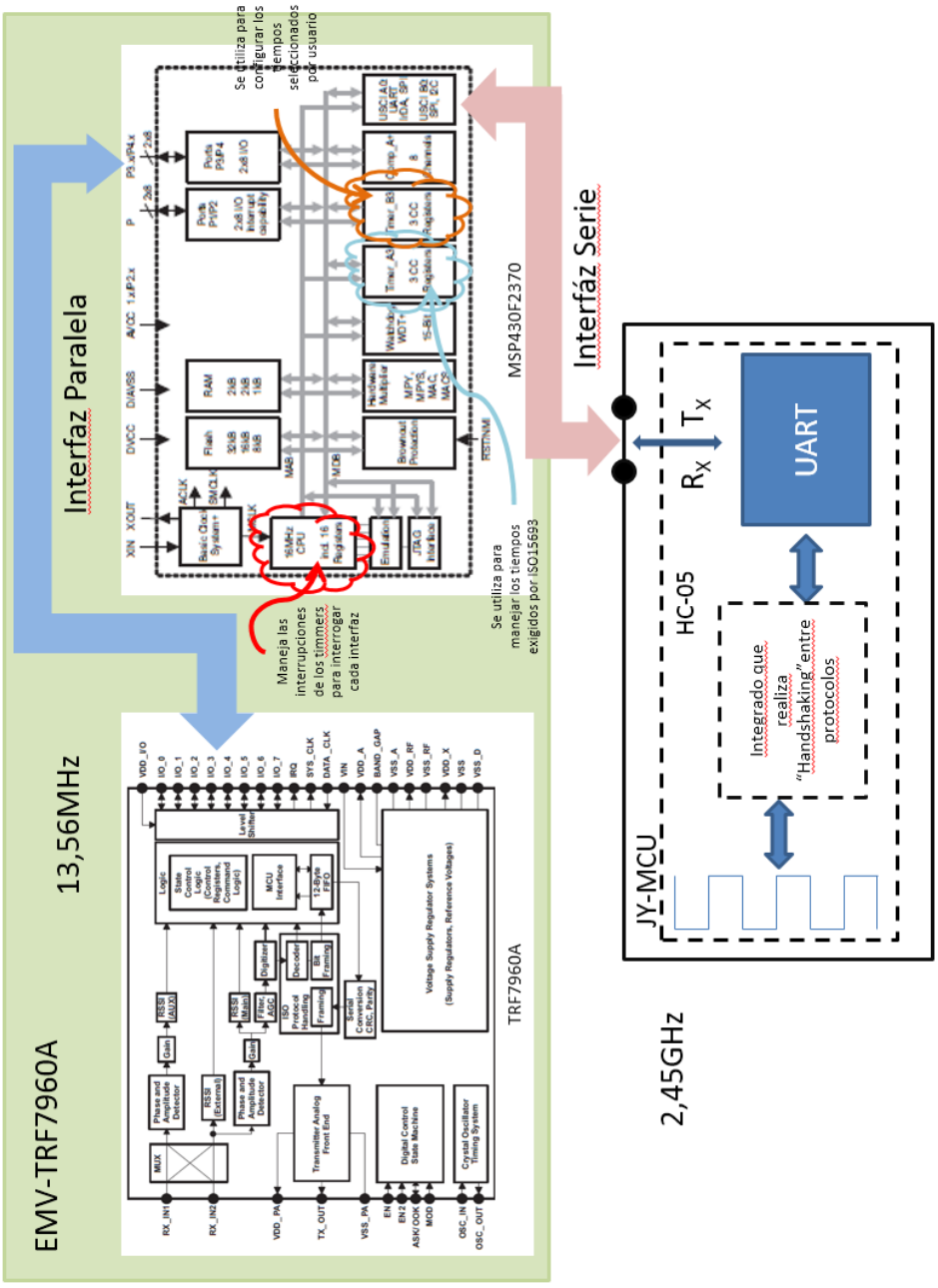


Figura 14 - Diagrama de bloques Sistema Pasarela

Sistema Pasarela implementado

En las Figuras 15 y 16 se muestra el resultado de la implementación física del sistema pasarela. Se compara su tamaño de 10cmx6cmx1cm con un celular Samsung Galaxy S5.

La pasarela es montada sobre la placa de evaluación del lector de radio frecuencia que interactúa con el sistema sensor, y cuenta además con un microprocesador que maneja el flujo de información. Se conecta un módulo BT externo (módulo que cuenta con un led de color rojo en la figura 15) y una batería Samsung de 1300mAh como se muestra en la figura.

Figura 16 - Fotografía de frente del sistema Pasarela de MONCEL en comparación con un celular Samsung S5

Figura 15 – Fotografía de perfil del sistema Pasarela de MONCEL en comparación con un celular Samsung S5

4.4. Aplicación de Celular

Diagrama de clases

El desarrollo de la aplicación se dividió en 5 clases principales como se muestra en la Figura 17.
Las clases RegistroOperations, DataBaseWrapper y Registros tienen como función crear y administrar la Base de Datos donde se guardan los datos recibidos. Sus funciones principales se encuentran descritas debajo de cada una de ellas representadas en la figura. Por su parte, SettingsFragment y UserSettingActivity son las responsables de gestionar las preferencias del usuario. Dentro del MainActivity, core del desarrollo de la aplicación, se destacan las funciones de cada definición, enfatizando y diferenciando de esta manera el manejo de los datos recibidos, la conexión al Bluetooth y las alertas efectuadas en el caso que la medida exceda el rango predefinido.

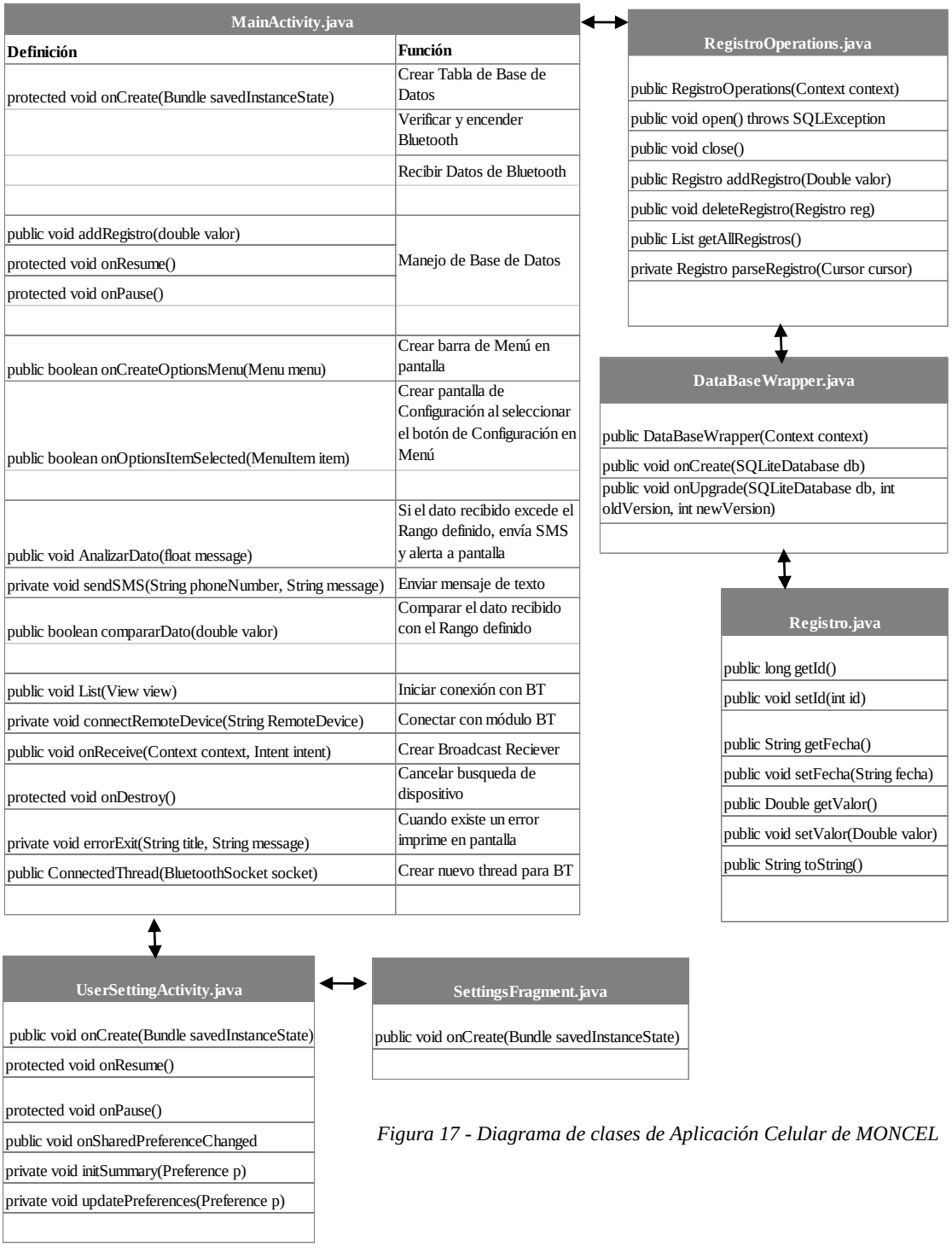


Figura 17 - Diagrama de clases de Aplicación Celular de MONCEL

La implementación de las clases fue compilada automáticamente por el sistema de desarrollo. El archivo resultante se llama MONCEL_APP.apk, que es un paquete de Android que contiene la aplicación y es el archivo que los dispositivos utilizan para instalar lo programado. Un detalle de este paquete se puede ver en el Apéndice E. Documentación Celular.

Aplicación de celular implementada

La aplicación cumple con los casos de uso descritos, y se muestra en las siguientes figuras impresiones a pantalla de la aplicación ejecutada en un celular Samsung Galaxy GT-I9082.

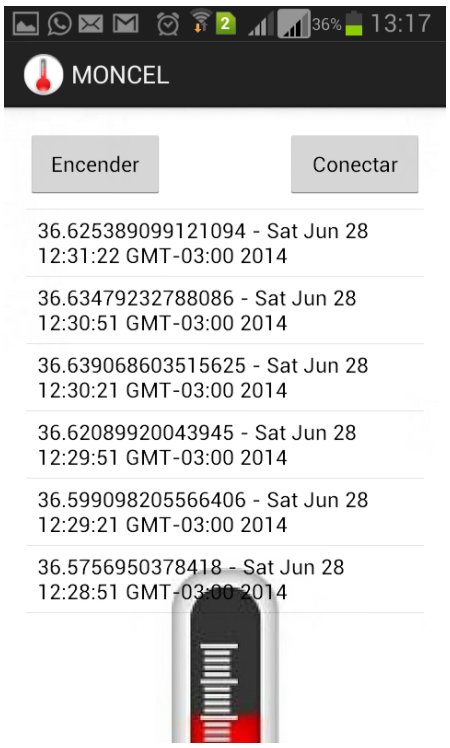


Figura 19 – Pantalla de inicio de MONCEL

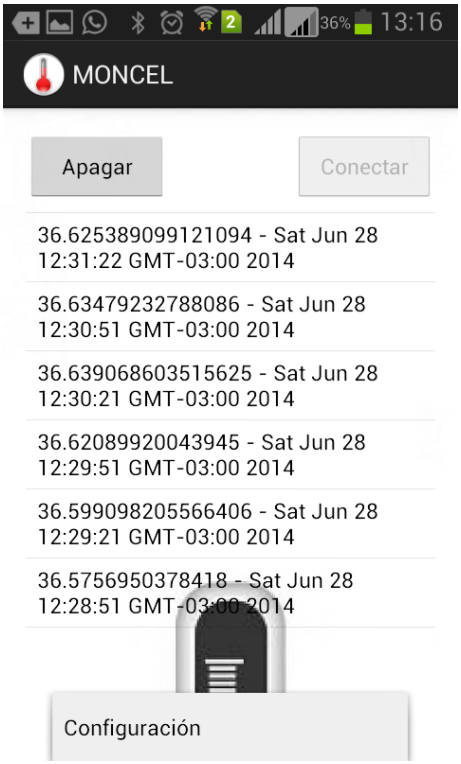


Figura 18 – Pantalla de inicio de MONCEL conectado al BT

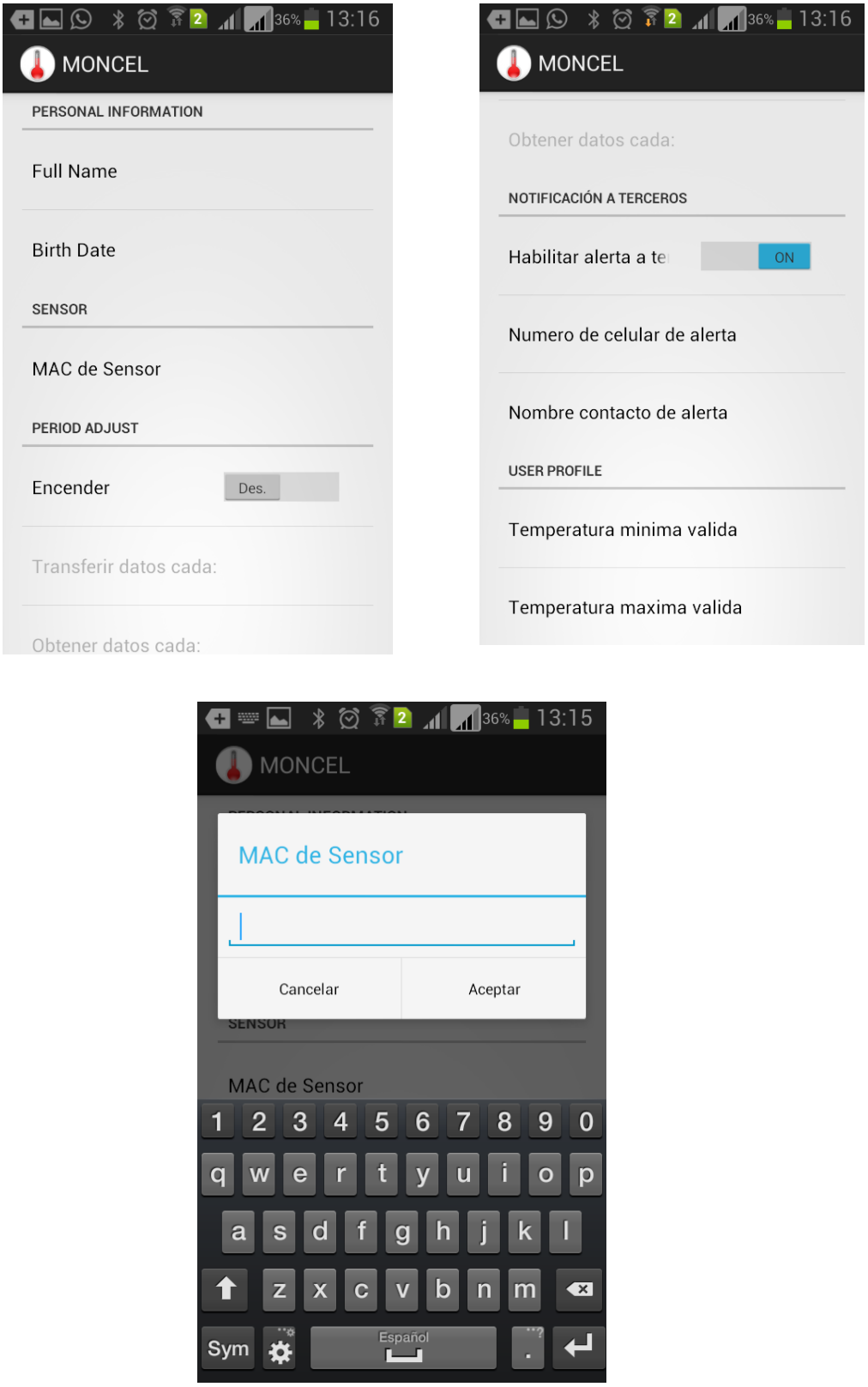


Figura 20 – Pantalla de Configuración de MONCEL

4.5. Análisis Energético

Modo energético de integrados y módulos

Microprocesador (MSP430F2370)

Los modos energéticos que puede estar el microprocesador son:

- Active Mode: Microcontrolador en modo activo.
- LPM1: CPU deshabilitada, mientras que los demás componentes permanecen activos.

RF Front End (TRF7960A)

- Modo 1: Transmisor y receptor se encuentran apagados. Reloj interno activo.
- Modo 2: Transmisor apagado. Resto del integrado activo.
- Modo 4: Todos los componentes se encuentran activos.

Módulo BT (HC-05)

- "Pairing": El módulo se encuentra visible esperando que se establezca el enlace.
- "Connected": Enlace establecido. El consumo en este modo es constante sin importar se esté procesando información o no.

Estados de funcionamiento

Se realiza el análisis energético considerando los estados de funcionamiento que se describen a continuación:

I. Encendido de la pasarela: Se realiza la configuración inicial de la pasarela.
II. Establecimiento enlace bluetooth: Se establece el enlace bluetooth entre la pasarela y el dispositivo móvil.
III. Pasarela en "Stand By".
IV. Envío comando de lectura: Se envía comando de lectura al sensor utilizando radio frecuencia.
V. Esperando respuesta sensor: Se espera respuesta del sensor.
VI. Envío valores obtenidos a dispositivo móvil: Se transmiten los valores disponibles en la pasarela al dispositivo móvil mediante Bluetooth.

Los estados de cada módulo están representados a en la Tabla 13.

Estado	MSP430F2370		TRF7960A			HC-05	
	Active	LPM1	Mode 1	Mode 2	Mode 4	"Pairing"	"Connected"
I	X		X			X	
II		X	X			X	
III		X	X				X
IV	X				X		X
V	X			X			X
VI	X		X				X

Tabla 13 - Modos de los módulos de la Pasarela según el estado

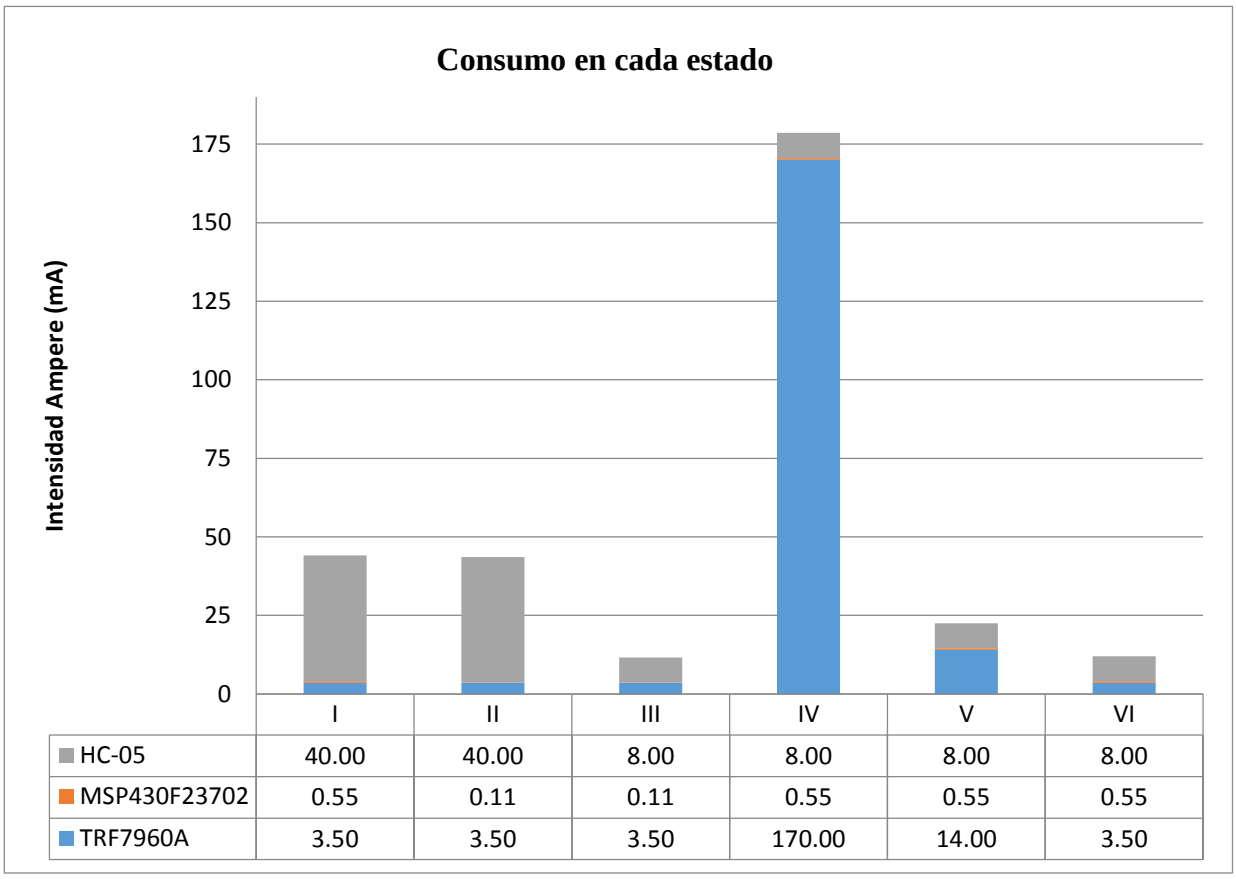


Figura 22 - Análisis de consumo energético de cada módulo por estado

Flujo de los estados en la aplicación

La pasarela puede atravesar los estados definidos en más de una ocasión y en distinto orden.
Para poder analizar el consumo energético, se definieron los procesos de trabajo Inicializar, Lectura y Transmitir. Cada uno de estos procesos está compuesto por algunos de los estados mencionados.

Considerando que el flujo normal de la aplicación es:
1. Inicializar
2. Lectura
a. Se repite cíclicamente con la frecuencia configurada por el usuario hasta que sea momento de transmitir
3. Transmitir
4. Se vuelve a 2.

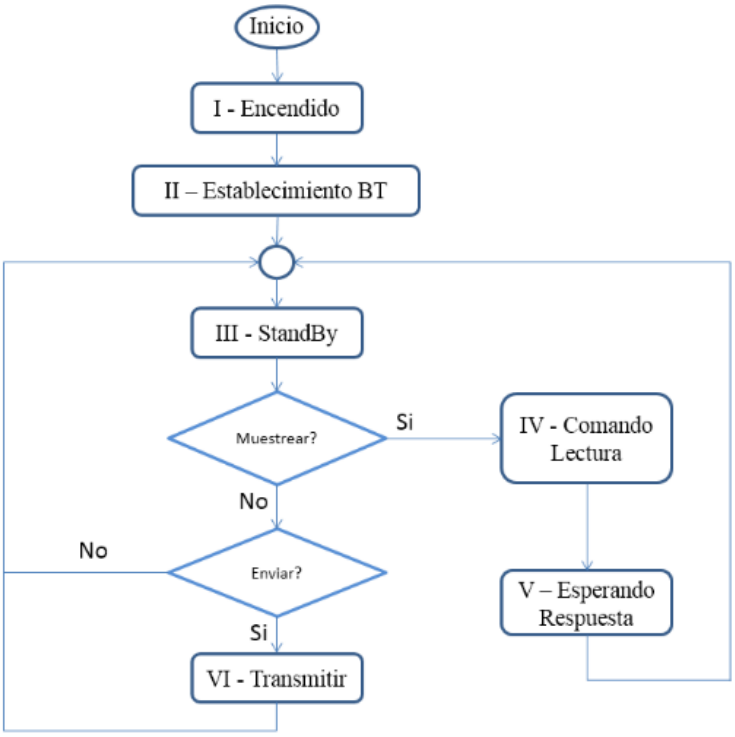


Figura 23 - Flujo de funcionamiento de la pasarela a través de los estados I, II, III, IV y V

La figura 24 muestra la distribución del % de la carga en función del intervalo de tiempo en cada etapa del flujo. Se puede observar que el estado de Standby es el que mayor consumo tiene en proporción al resto. Entre 10s y 900s de muestras, el estado de stand by ocupa casi el 99% del consumo de la carga.

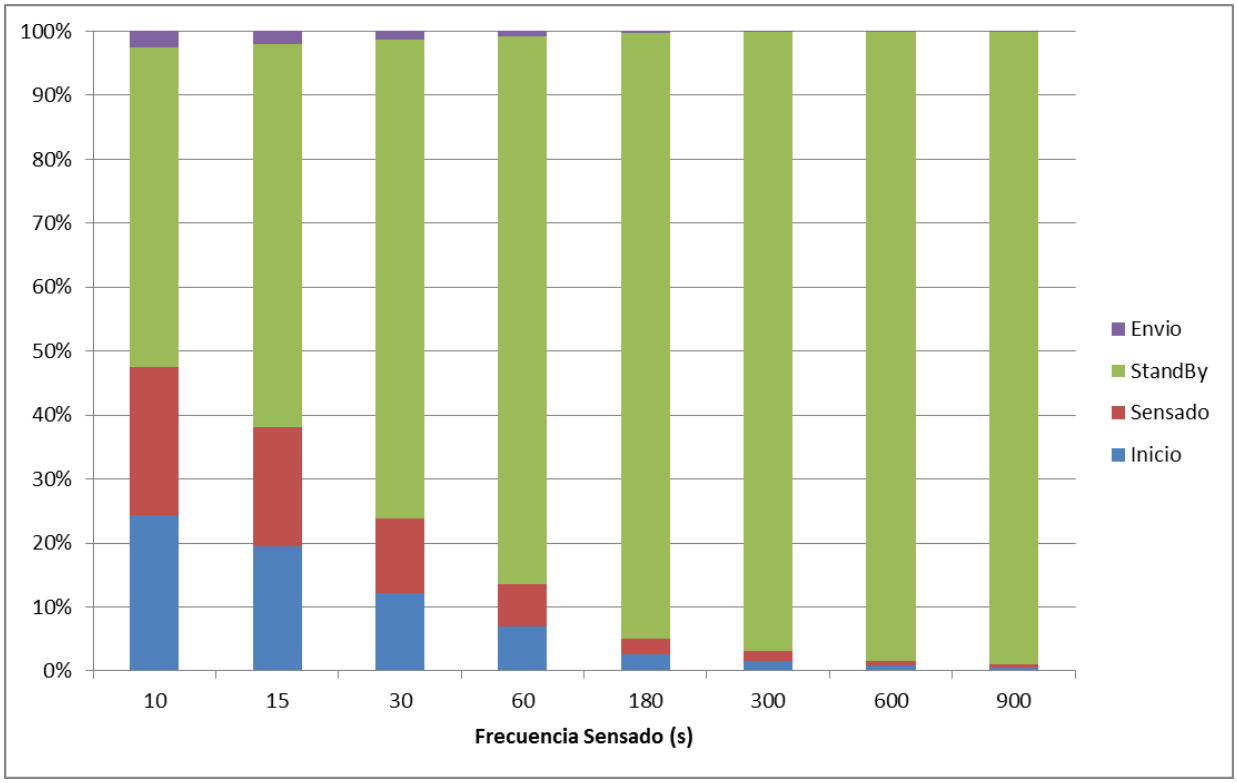


Figura 24 – Porcentaje de consumo de cada etapa en una batería de 1300mAh en función del intervalo de tiempo entre sensado

Cálculo de consumo energético

Se realiza el cálculo de autonomía suponiendo la lectura y transmisión es cada 5 minutos.

Como se puede ver en la Figura 22, el estado más determinante para el cálculo del consumo energético es el número IV, etapa en la cual la pasarela se encuentra en comunicación con el sensor.

Debido a que el proceso de configuración inicial se realiza una única vez y consume 87.66 mA, no se toma en cuenta para el cálculo del tiempo de duración de la batería.

A su vez se realizan los cálculos con una batería de 1300mAh.

Tomando en cuenta la duración de cada etapa y la suma de consumo de cada uno de los componentes involucrados se calcula el consumo total de la pasarela por etapa como:

Consumo (mAs) = Duración * Intensidad

Estado	**Duración (ms)**	**Consumo (mAs)**
IV	150	26,78

V	200	4,51
III	2000	23,22
IV	150	26,78
V	200	4,51
III	360	4,18
VI	500	6,02
TOTAL		**96,01**

Tabla 14 - Estimación de consumo según la duración de cada estado

En total el proceso dura: 3,56s.

Si el período entre trasmisiones es de 5 minutos, la pasarela se encuentra en Stand By 296,44s, consumiendo 3441,67mAs.
El consumo total por ciclo es entonces de: 3537,7mAs.

Sabiendo que la capacidad de la batería es de 1300mAh, la cantidad de ciclos será de:
Ciclos = (1300mAh/3537,7mAs)*3600s/h ≈ 1300 ciclos

Por lo tanto, la duración de la batería para un periodo de trasmisiones cada 5minutos se estima en:
Duración Batería (hrs) = (1300ciclos*5min/ciclo)/60min/hr =**108hrs**

Considerando un coeficiente de seguridad de un 70%, la duración de la batería será de aproximadamente de **3 días**.

Si el período entre trasmisiones fuera de 30 minutos, la pasarela se encuentra en Stand By 1796.4s, consumiendo 20838mAs.
El consumo total por ciclo es entonces de: 20934mAs.

Sabiendo que la capacidad de la batería es de 1300mAh, la cantidad de ciclos será de:
Ciclos = (1300mAh/20934mAs)*3600s/h ≈ 224 ciclos

Por lo tanto, la duración de la batería para un periodo de trasmisiones cada 5minutos se estima en:
Duracion Bateria (hrs) = (224ciclos*30min/ciclo)/60min/hr =**112hrs**

Nota:
Como se observa, la duración de la batería es muy poco sensible al período entre muestras para frecuencias del orden de los minutos.

Para mejorar la duración de la batería debemos disminuir el consumo entre tomas, para lo cual se debe apagar el Bluetooth y volver a encenderlo previo a la trasmisión.

Si hacemos esto, el consumo total para toma cada 5minutos será de: 96,01mAs + 43,61mAs(consumo de estado II) = 139,62mAs.
El estado de stand by entre muestras será de 295s consumiendo ahora 3,61mAs.

La duración de la batería pasa a ser de **325hrs**, o de **9 días** con un factor de seguridad del 70%.

A continuación se muestra en la Figura 25 la duración de la batería en hrs si apagamos el Bluetooth entre las medidas y lo volvemos a encender cada vez que se realiza una muestra en el sistema sensor, en comparación con la duración de la batería si dejamos permanentemente el BT encendido.

Como se puede observar, incluso tomando medidas cada 10s la duración es mayor encendiendo y apagando el BT, y esta diferencia aumenta considerablemente a medida que el intervalo de muestreo aumenta, por lo que conviene apagar la antena de Bluetooth siempre que se encuentra en stand by la Pasarela.

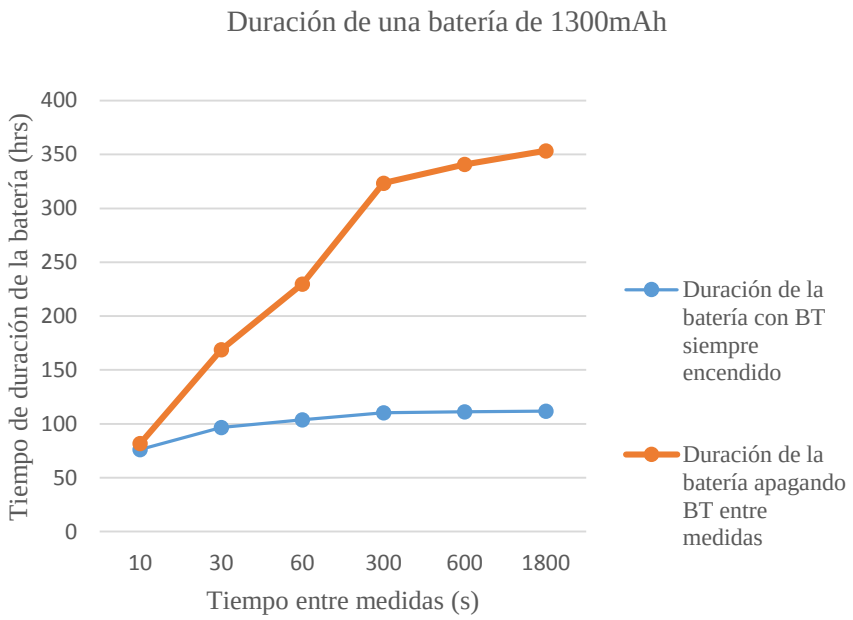


Figura 25 - Duración de una batería de 1300mAh en Pasarela en función del intervalo de tiempo entre medidas si se dejara el Bluetooth permanentemente encendido y si se apagara entre muestras.

Capítulo 5
Pruebas y Resultados

5.1. Introducción

En el presente capítulo se describen los experimentos realizados con el fin de testear el funcionamiento del sistema MONCEL.

MONCEL está compuesto por tres bloques distintos intercomunicados por enlaces de RF distintos. Estos enlaces, tienen alcances distintos que resultan determinantes para el normal funcionamiento del sistema. En particular se releva en primera instancia la comunicación entre la Pasarela y el Sistema Sensor, evaluando la influencia de la distancia y posición entre ellos.

El acoplamiento inductivo que se crea durante la comunicación Pasarela-Sensor depende fuertemente también del ángulo existente entre las antenas correspondientes de cada bloque.
Esto hace que se ensaye la respuesta del sistema frente a la variación de éste ángulo.

Por último, se evalúa la influencia del sistema según la distancia entre la Pasarela y el Celular.

En todos los casos la temperatura medida fue la de uno de los integrantes del grupo quien se colocó el sensor de temperatura debajo del brazo. Considerando que se trata de un sistema biológico a temperatura razonablemente constante, la repetitividad fue evaluada tomando 10 medidas sucesivas en 8 ensayos diferentes.

Por otra parte, antes y después de los ensayos separados en un tiempo de 2hrs, se mide también la temperatura corporal del mismo integrante utilizando un termómetro estándar de mercurio en la misma zona corporal.
Esta temperatura registrada en ambos casos fue de **36.6 ºC.**

Se calcula la incertidumbre asociada a cada resultado de acuerdo a las relaciones mostradas en el Apéndice A. Se adjunta en el Apéndice B las impresiones de pantallas del celular correspondientes a los ensayos.

5.2. Ensayos Sensor - Pasarela

Para conocer cómo influye la separación entre el Sistema Sensor y el Sistema Pasarela se procedió de la siguiente manera:

Ensayo 1: Sensor a 0cm encima y paralelo a la Pasarela

Se dejó fija la Pasarela apoyada sobre una superficie plana (una mesa), y se colocó apoyada sobre la misma el Sistema Sensor. Se relevaron los valores mostrados en la tabla 15.

ENSAYO 1		
POSICIÓN	**DATO x_i (ºC)**	**Media**
	36,5	36,5 ºC
Encima de la Pasarela	36,5	
	36,5	**Incertidumbre Tipo A**
	36,5	0,00 ºC
	36,5	
	36,5	**Incertidumbre Expandida (factor de cobertura 2)**
	36,5	
	36,5	0,00 ºC
	36,5	
	36,5	

Tabla 15 - Datos obtenidos en el ensayo 1

RESULTADO:	**36,5 ± 0,0 ºC**
ERROR RELATIVO:	**0,27 %**

Ensayo 2: Sensor a 1cm encima y paralelo a la Pasarela

Con la Pasarela fija en la misma superficie plana que en el ensayo anterior, se coloca el Sistema Sensor en un plano paralelo al de esta superficie plana y separado 1 cm del mismo. Es decir, un sistema encima del otro pero separados 1 cm.
Los datos relevados se muestran en la tabla 16.

ENSAYO 2		
POSICIÓN	**DATO x_i (ºC)**	**Media**

	36,5	36,5 ºC
Encima a 1 cm de la Pasarela	36,5	
	36,5	**Incertidumbre Tipo A**
	36,5	0,00 ºC
	36,5	
	36,5	**Incertidumbre Expandida (factor de cobertura 2)**
	36,5	
	36,5	0,00 ºC
	36,5	
	36,5	

Tabla 16 - Datos obtenidos en el ensayo 2

RESULTADO:	36,5 ± 0,0 ºC
ERROR RELATIVO:	0,27 %

Ensayo 3: Sensor a 2,5 cm encima y paralelo a la Pasarela

A partir del ensayo anterior, se continúan separando los planos, siendo siempre paralelos entre sí. En este ensayo la separación entre ambos sistemas es de 2,5 cm. En la tabla 17 se pueden ver las medidas obtenidas.

ENSAYO 3		
POSICIÓN	**DATO x_i (ºC)**	**Media**
	36,6	36,6 ºC
Encima a 2,5 cm de la Pasarela	36,6	
	36,6	**Incertidumbre Tipo A**
	36,6	0,00 ºC
	36,6	
	36,6	**Incertidumbre Expandida (factor de cobertura 2)**
	36,6	
	36,6	0,00 ºC
	36,6	
	36,6	

Tabla 17 - Datos obtenidos en el ensayo 3

RESULTADO:	36,6 ± 0,0 ºC

ERROR RELATIVO:	0,00 %

Ensayo 4: Sensor a 5,5 cm encima y paralelo a la Pasarela

Se sigue aumentando la separación entre los planos paralelos al igual que en los ensayos anteriores. Esta vez la separación es de 5,5 cm. En la tabla 18 se ven los valores obtenidos.

ENSAYO 4		
POSICIÓN	**DATO x_i (ºC)**	**Media**
	36,6	36,7 ºC
Encima a 5,5 cm de la Pasarela	36,6	
	36,7	**Incertidumbre Tipo A**
	36,7	0,0189 ºC
	36,7	
	36,7	**Incertidumbre Expandida (factor de cobertura 2)**
	36,7	
	36,7	0,0377 ºC
	36,7	
	36,7	

Tabla 18 - Datos obtenidos en el ensayo 4

RESULTADO:	36,7 ± 0,04 ºC
ERROR RELATIVO:	0,22 %

Ensayo 5: Sensor a 7,5 cm encima y paralelo a la Pasarela

Se repiten los ensayos anteriores con una separación de 7,5cm entre los planos paralelos. En la tabla 19 se muestran los valores relevados.

ENSAYO 5		
POSICIÓN	**DATO x_i (ºC)**	**Media**
	36,7	36,8 ºC
Encima a 7,5 cm de la Pasarela	36,7	
	36,7	**Incertidumbre Tipo A**
	36,8	0,0216 ºC
	36,8	
	36,8	

	36,8	**Incertidumbre Expandida (factor de cobertura 2)**
	36,8	0,0432 ºC
	36,8	
	36,8	

Tabla 19 - Datos obtenidos en el ensayo 5

RESULTADO:	36,8 ± 0,04 ºC
ERROR RELATIVO:	0,46 %

Ensayo 6: Sensor a 9,4 cm encima y paralelo a la Pasarela

Se continuaron separando los planos entre sí (manteniéndose siempre paralelos) y midiendo la respuesta del sistema.

Al llegar a una separación entre ambos de 9,4 cm, el sistema dio error. Es decir, acorde a lo esperado, en la aplicación del celular se recibió el código de error pertinente que indica que el lector no pudo obtener la respuesta del Sistema Sensor.

Esto era lo esperado puesto que la antena provista en el **TRF7960EVM,** la cual es una antena rectangular con una diagonal de 2”**,** tiene un alcance de lectura de aproximadamente 4”. [17]

Ensayo 7: Sensor a 45º con respecto a la superficie de la mesa y encima de la Pasarela

Se procede a estudiar cómo influye el ángulo entre las antenas de la pasarela y el sensor. El primer ensayo se realiza con la pasarela sobre una mesa fija e inmóvil y el sensor en un plano a 45º de la superficie de la mesa y pegado a la pasarela. Al igual que en los ensayos anteriores, se relevaron 10 valores los cuales se muestran en la tabla 20.

ENSAYO 7		
POSICIÓN	**DATO x_i (ºC)**	**Media**
	36,6	36,5 ºC
En plano a 45º de la Pasarela	36,5	
	36,5	**Incertidumbre Tipo A**
	36,5	0,0141 ºC
	36,5	
	36,5	**Incertidumbre Expandida (factor de cobertura 2)**
	36,5	
	36,5	0,0283 ºC
	36,5	
	36,5	

RESULTADO:	36,5 ± 0,03 ºC
ERROR RELATIVO:	0,25 %

Tabla 20 - Datos obtenidos en el ensayo 7

Ensayo 8: Sensor a 90º con respecto a la superficie de la mesa y encima de la Pasarela

Se coloca el sensor a en un plano perpendicular a la mesa y sobre la Pasarela. Al relevar las medidas nuevamente se obtiene en la aplicación del celular el código de error. El error es esperado debido a que no existe acoplamiento entre antenas que estén perpendiculares entre sí.

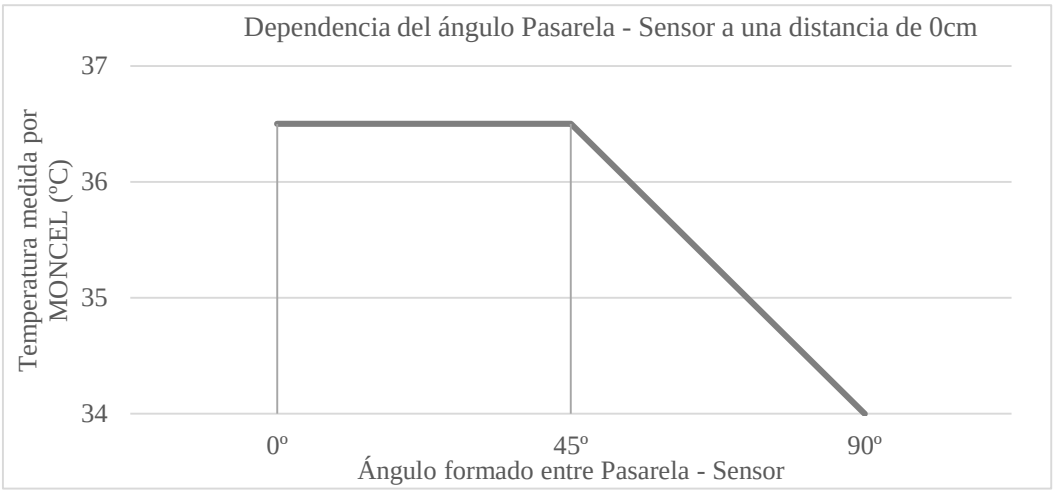


Figura 26 - Gráfica de Temperatura medida por MONCEL vs. Ángulo formado entre Pasarela y sensor a una distancia de 0cm entre sí

5.3. Ensayo Pasarela - Celular

Si bien la dependencia de esta distancia no es la más crítica para el correcto funcionamiento del sistema, se procedió a estudiar el alcance máximo entre el celular y la pasarela donde los datos seguían siendo medidos por el sensor y recibidos por el Celular.

Para esto se dejó fijo el conjunto Pasarela – Sensor apoyado sobre una superficie plana (una mesa), y con el celular conectado a la Pasarela y recibiendo continuamente los datos medidos (con una cadencia de 10 segundos), se lo fue alejando hasta encontrar la distancia a la cual dejaba de recibirlos.

El requerimiento de MONCEL para esta distancia quedó establecido en 4m. Sin embargo, la distancia máxima obtenida fue muy superior a esto alcanzando un valor de 16.2m.

Distancia Máxima obtenida Pasarela - Celular	**16,2 m.**

Cabe destacar de todos modos, que si bien hasta esa distancia la comunicación Pasarela – Celular funcionaba bien, no era posible iniciar la comunicación estando tan separados.
Para poder establecer la comunicación se requería estar a al menos 4m uno del otro.

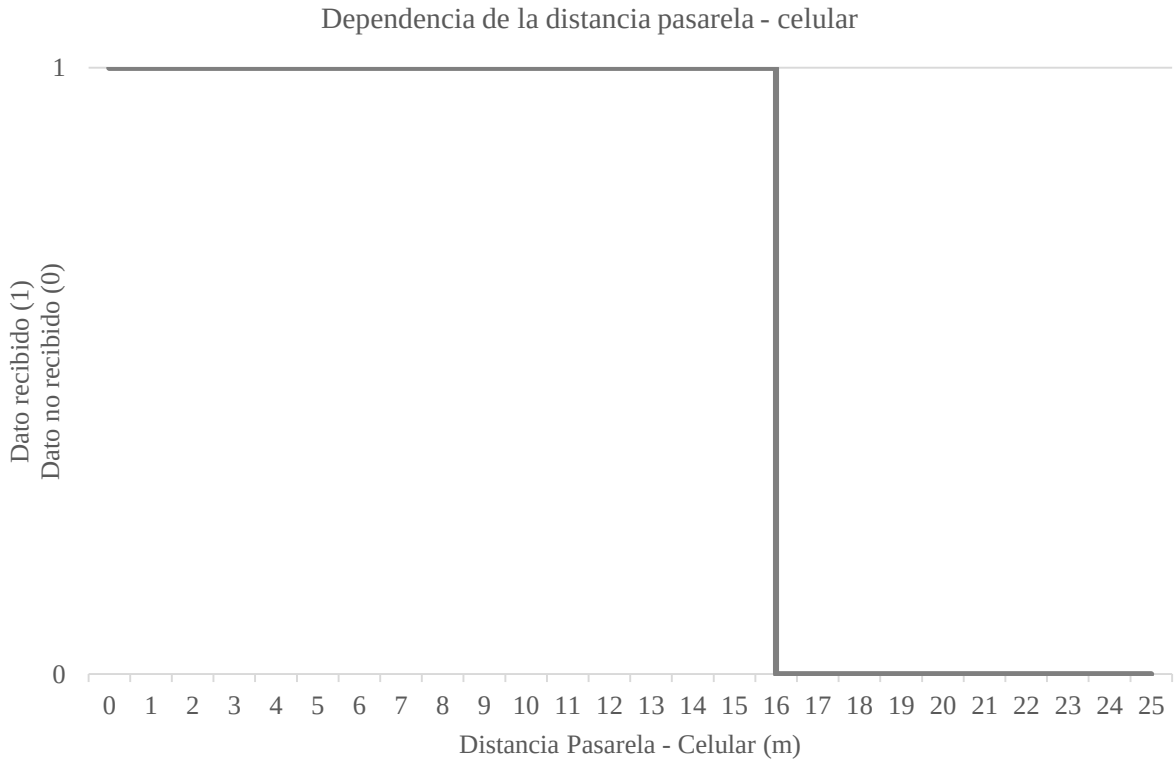


Figura 27 - Gráfica distancia sensor - pasarela

5.4. Dependencia del valor sensado en función de la distancia

A partir de los valores obtenidos, podemos observar que la medida registrada depende a la distancia a la que se encuentra la pasarela del sensor. Si bien éste valor varía solo un 0,8% correspondiente a los 0.3ºC de diferencia entre el valor sensado a ≈0cm (36.5º) y a la distancia máxima de 9.4cm (36.8ºC), MONCEL debe asegurar un rango donde la medida obtenida junto con su incertidumbre no dependa de la distancia entre la pasarela y el sensor.

Para estudiar esta dependencia, se registraron 14 medidas del voltaje Vfield y Vreg variando la distancia entre la pasarela y el sensor.
Vfield es el voltaje generado en el sistema sensor a partir del campo inducido por la pasarela.
Vreg es la salida regulada de Vfield y es el voltaje utilizado para alimentar los distintos componentes del MLX90129.

Considerando el diseño descrito en el Capítulo 3, Vreg es además el voltaje que alimenta el puente de Weathstone donde se encuentra el termistor, afectando el valor del ΔVsensor a la entrada del amplificador PGA1.

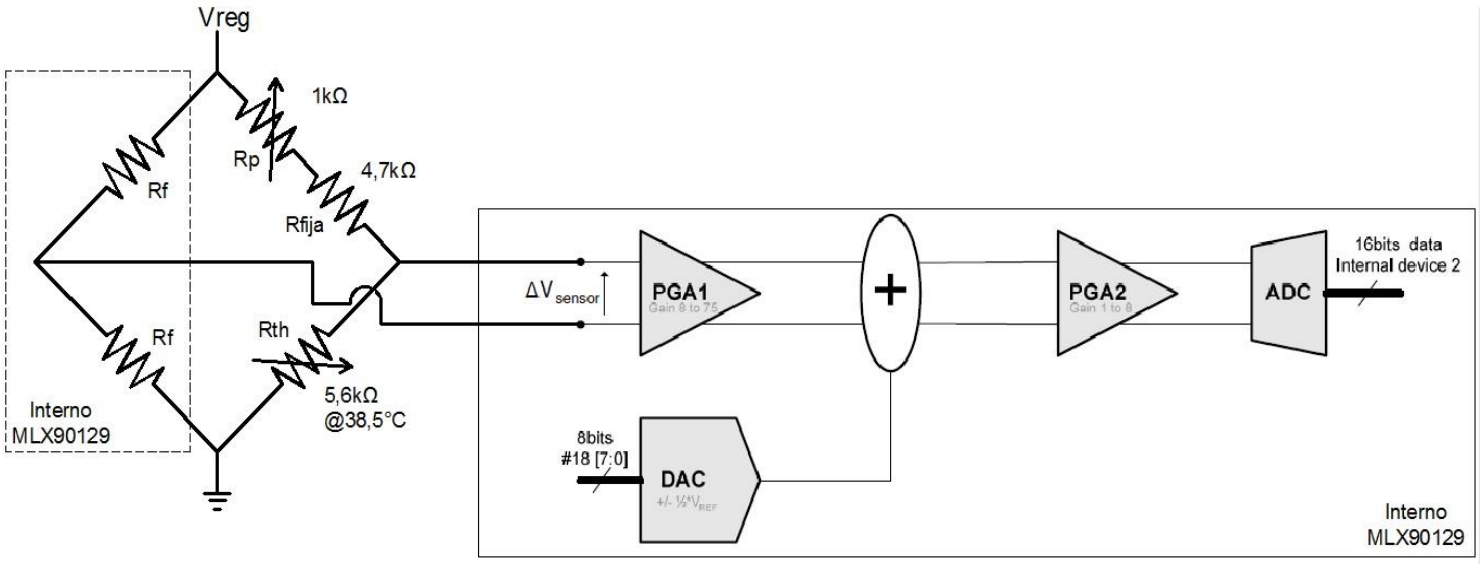


Figura 28 - Conexión del circuito del sistema sensor

A continuación se realizaron 14 muestras de Vfield y Vreg, variando la distancia entre el sensor y la pasarela en un plano paralelo entre ambos desde 0 a 14cm.

Las medidas obtenidas fueron las siguientes:

Distancia Pasarela - Sensor (cm)	Vreg (V)	Vfield (V)
0	2.94	4.9
1	2.94	4.87
2	2.94	4.78
3	2.94	4.7
5	2.94	4.45
6.5	2.94	3.2
7.5	2.42	2.84
8.5	2	2.53
9	1.76	2.1
9.4	1.5	1.74
10.5	1.15	1.45
12	0.8	1
13.5	0.12	0.99
14	0.03	0.96

Tabla 21 - Medidas de Vreg y Vfield registradas en función de la distancia entre pasarela y sensor

Como se puede observar en los datos obtenidos, el voltaje Vreg permanece constante hasta los 6.5cm, por lo que podemos asumir que dentro de este rango, la medida no dependerá de las distancia sino de la propia incertidumbre del termistor que es de ±0.1ºC de acuerdo al fabricante.

La Figura 23 muestra la media de cada uno de los ensayos realizados en función de la distancia. A los 7.5cm, el valor promedio registrado fue de 36.8ºC y es donde se encuentra la mayor variación con respecto a la temperatura corporal medida por termómetro de 36.6ºC.
Para esta distancia, el valor de Vreg varió casi un 18%, afectando la medida de la resistencia del termistor y comprometiendo el dato sensado.

A partir de los 7.5cm, el error continúa agravándose hasta llegar al voltaje mínimo de funcionamiento. Este voltaje es de 2.1V especificado por el fabricante para el MLX90129, por lo que a distancias mayores a ≈ 8.5cm el sistema dejará de sensar.
Recordando el ensayo 6, se obtuvo un comando de error cuando se intentó realizar la medida a una distancia de 9.4cm, verificando de esta manera el alcance máximo especificado por el fabricante.

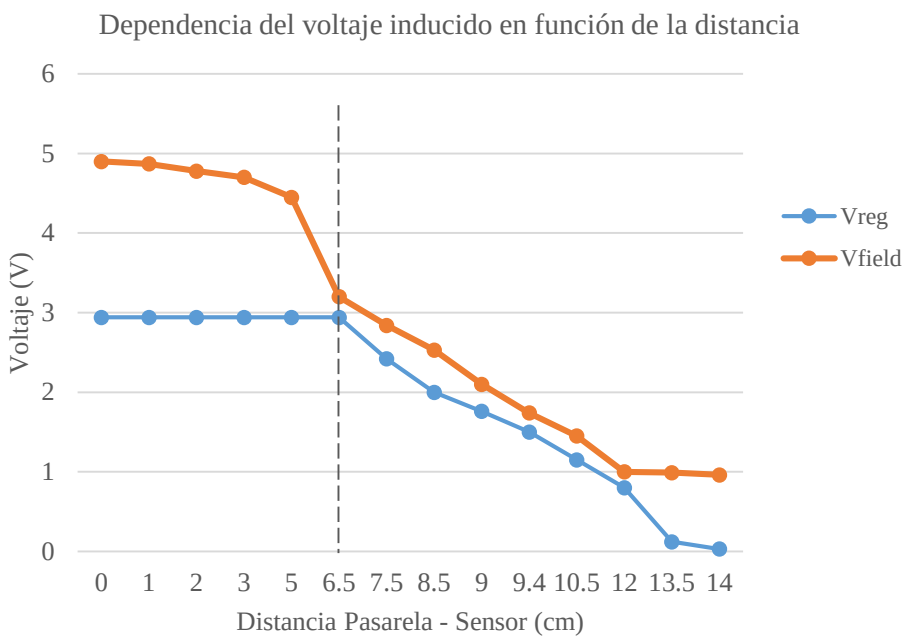


Figura 29 – Gráfica Vreg y Vfield en función de la distancia entre la pasarela y el sensor. Valores tomados de la Tabla 21.

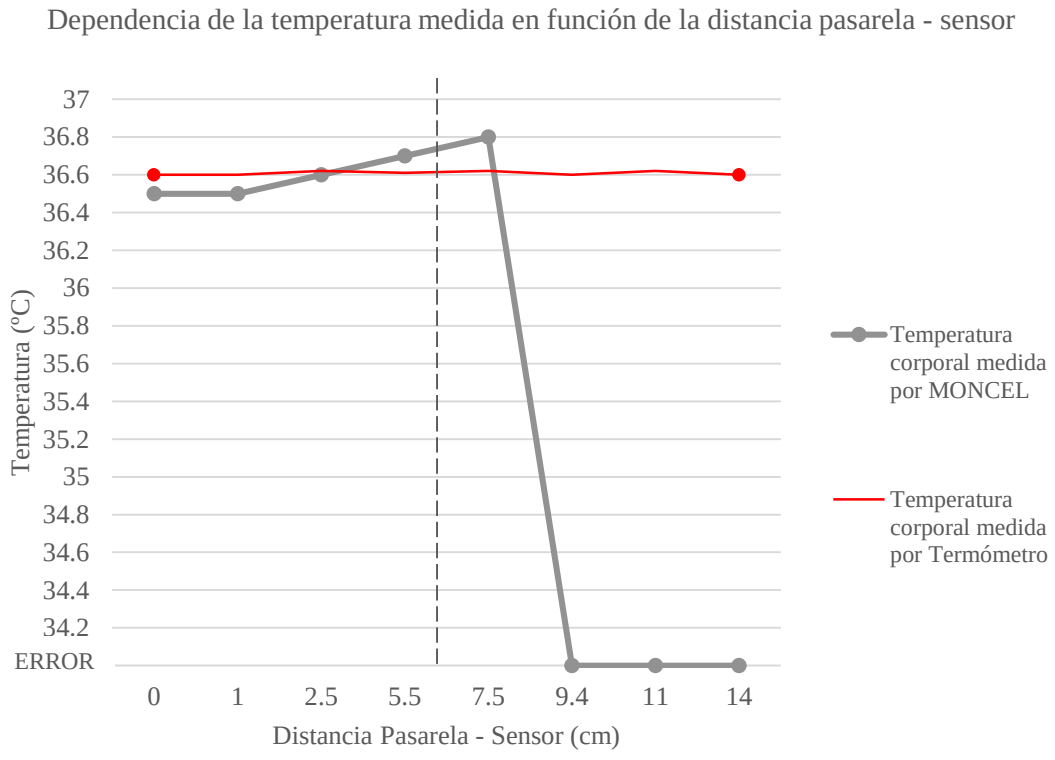


Figura 30 - Gráfica Temperatura corporal medida por MONCEL y medida por termómetro en función de la distancia pasarela sensor

De esta manera, elegimos el rango de **[0..6,5cm]** donde el voltaje Vreg permanece constante en 2.94V y la medida sólo depende de la variación que pueda existir debido al margen de error que exista en el termistor.

Considerando este rango de distancia, la temperatura corporal promedio registrada por los ensayos varió en un rango de **[36,5ºC…36,65ºC]**, logrando obtener una media de 36,57ºC, 0,03ºC menos de la temperatura medida por el termómetro.
Se alcanzó un error de ≈ **±0.1ºC** limitado por la incertidumbre del propio termistor.

Posición	**Distancia máxima**
Sensor paralelo a superficie y alejado de Pasarela	**6,5 cm.**

Por otra parte, cabe destacar que el ángulo del sensor con respecto a la pasarela no tiene mayor influencia sobre el resultado final y se sigue obteniendo una medida sensada de 36,6ºC. Contar con esa seguridad implica que no debería ser un inconveniente si la pasarela que se encuentra pegada al cuerpo encima del sensor varía su ángulo pero se mantiene en el mismo plano paralelo al plano del sensor. En cambio, sí es importante destacar que el sistema no funcionará si la Pasarela se encuentra igualmente sobre el sensor pero situándose en un plano perpendicular al de la piel en donde se halla implantado el sensor.

Solución a la variación de la medida con la distancia en el rango [6,5cm…8,5cm]

A partir de una distancia de 8,5cm entre la pasarela y el sensor, se despliega en pantalla del celular un ERROR debido a el voltaje inducido en el sensor no es suficiente para realizar una medida. Entre los 0cm y 6,5cm, el voltaje permanece aproximadamente constante por lo que el error en la medida depende de la propia incertidumbre del termistor como se analizó en la sección anterior.

En el rango de [6,5cm…8,5cm] donde el voltaje inducido decrece, no se puede discriminar si el cambio en el valor obtenido se debe a el cambio en la distancia o a un cambio real en la variable monitoreada, resultando en perdida de confiabilidad del sistema. Debemos asegurar entonces un mecanismo que arroje un ERROR en la pantalla del celular de la misma manera que si se encontrara a una distancia mayor.

El integrado MLX90129 cuenta con una opción que permite configurar uno de sus sensores internos para detectar la intensidad del campo inducido en la antena del mismo. A través de este valor podemos deducir si la distancia está dentro del rango aceptado o no, y así generar un mensaje de alerta para que se corrija la distancia.

Para detectar si el campo inducido es suficiente, es necesario definir el umbral mínimo al cual el MLX90129 se encuentra correctamente energizado. El procedimiento para definir este umbral fue colocar el sensor y la pasarela a una distancia de 6,5cm, corroborando que el voltaje inducido continuaba siendo constante de 2,94V y se procedió a leer el sensor interno configurado para obtener la intensidad de campo inducido. Una vez obtenido este valor que resultó ser de 6A00, antes y después de realizar cada medida de temperatura se compara el campo con este valor, y si menor se descarta el dato y devuelve un mensaje de ERROR al celular. De esta manera se logra garantizar la validez de los datos en cualquier rango de distancia, obteniendo una buena confiabilidad del sistema.

Capítulo 6
Gestión del Proyecto

6.1. WBS

Una Estructura de Descomposición del Trabajo, también conocida por su nombre en inglés Work Breakdown Structure o WBS, es en gestión de proyectos una descomposición jerárquica orientada al entregable del trabajo a ser ejecutado por el equipo de proyecto.
El WBS definido en el Plan del Proyecto en Abril 2013 fue el siguiente:

- Definiciones previas y compras:
  - Tecnología de enlace
  - Selección de plataforma de celulares: Android o Multiplataforma
  - Investigación sobre tecnologías de alimentación de energía
  - Elección de fuente de alimentación
  - Definir microcontroladores y radios
  - Compras de los elementos especificados
- Desarrollo de aplicación en el celular
  - Manejo de la comunicación con el dispositivo
  - Definir la interfaz usuario
  - Manejo de alarmas propias y a terceros
- Fuente de alimentación del biosensor
  - Elección del sensor
  - Desarrollo de la fuente
  - Se agregarán tareas una vez concluido el objetivo de Definiciones previas y compras
- Comunicación entre el celular y el dispositivo embebido
  - Diagrama de bloques
  - Arquitectura de la solución
  - Elección de plataforma de desarrollo
  - Elección de Radio
  - Pruebas de envío y recepción de los datos usando la tecnología definida
  - Muestreo del sensor
- Especificaciones físicas del diseño
  - Definiciones del diseño físico

- Gabinete
- Interconexión de módulos
- Ubicación de la antena

6.2. Cronograma detallado

El cronograma definido en el Plan de Proyecto se representó de la siguiente manera:

ID	Nombre	Duración	Comienzo	Fin	Recursos	Antecesores
282	**1 - Definiciones y Compras**	**80**	**05/04/2013**	**01/08/2013**		
283	1.1 - Definir tecnologías de enlace	49	05/04/2013	14/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	
284	1.2 - Definir plataforma: Andriod vs Multiplataforma	7	16/05/2013	24/05/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	
285	1.3 - Definir plataforma de desarrollo Aplicación Celular	10	27/05/2013	07/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	284
286	1.4 - Definir microcontroladores y moduladores	4	17/06/2013	21/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	283
287	1.5 - Definir antenas	4	17/06/2013	21/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	283
288	1.6 - Definir sensor	4	17/06/2013	21/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	
289	1.7 - Definir plataformas de desarrollo embebido	5	24/06/2013	28/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	286
290	1.8 - Definir fuentes de alimentación	5	24/06/2013	28/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	288
291	1.9 - Comprar microcontroladores, moduladores y antenas	27	24/06/2013	31/07/2013	Juan Martín Ortega,Ramiro Barrón	286;287
292	1.10 - Comprar componentes fuentes de alimentación	22	01/07/2013	31/07/2013	Andrea Cukerman,Juan Martín Ortega	290
352	1.11 - Comprar Biosensor	27	24/06/2013	31/07/2013	Andrea Cukerman,Ramiro Barrón	288
348	1.12 - Documentación	34	13/06/2013	01/08/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	285;289;291;292;352
303	**2 - Aplicación Pasarela**	**125**	**01/08/2013**	**27/01/2014**		**289**
304	2.1 - Familiarizarse con plataforma de desarrollo	10	01/08/2013	14/08/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	
305	2.2 - Realizar diagrama de bloques	5	21/08/2013	27/08/2013	Juan Martín Ortega,Ramiro Barrón	
306	2.3 - Definir arquitectura de SW	3	21/08/2013	23/08/2013	Andrea Cukerman,Ramiro Barrón	

307	2.4 - Desarrollar la aplicación embebida	90	28/08/2013	02/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	304;305;306
309	2.5 - Pruebas y debug	16	03/01/2014	27/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	307
308	2.6 - Documentación	20	27/12/2013	27/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	309
301	**3 - Hardware Pasarela**	**178**	**20/05/2013**	**29/01/2014**		
302	3.1 - Definir arquitectura de HW y montaje de sus componentes	5	01/08/2013	07/08/2013	Andrea Cukerman,Juan Martín Ortega	290
310	3.2 - Estudiar consumo energético	29	20/05/2013	28/06/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	
311	3.3 - Definir diseño físico de pasarela	20	08/08/2013	04/09/2013	Ramiro Barrón	302
312	3.4 - Fabricar o comprar gabinete	84	05/09/2013	02/01/2014	Andrea Cukerman	311
313	3.5 - Fabricar o comprar PCB para montaje de componentes	104	08/08/2013	02/01/2014	Juan Martín Ortega	302
314	3.6 - Probar HW	8	03/01/2014	15/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	312;313
315	3.7 - Documentación	10	16/01/2014	29/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	314
316	**4 - Aplicación Sistema Biosensor**	**57**	**01/08/2013**	**18/10/2013**		**288**
327	4.1 - Familiarizarse con plataforma de desarrollo	10	01/08/2013	14/08/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	289
325	4.2 - Definir arquitectura de SW	10	01/08/2013	14/08/2013	Juan Martín Ortega,Ramiro Barrón	289
323	4.3 - Realizar diagrama de bloques	5	02/08/2013	08/08/2013	Andrea Cukerman,Juan Martín Ortega	282;286;288
329	4.4 - Desarrollar la aplicación embebida	30	15/08/2013	25/09/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	323;325;327
319	4.5 - Pruebas y debug	7	26/09/2013	04/10/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	329
321	4.6 - Documentación	12	03/10/2013	18/10/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	319
330	**5 - Hardware Sistema Biosensor**	**67**	**01/08/2013**	**01/11/2013**		**286;287;288**

331	5.1 - Estudiar transmisión inalámbrica de energía	30	01/08/2013	11/09/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	
339	5.2 - Definir arquitectura de HW y montaje de sus componentes	20	01/08/2013	28/08/2013	Ramiro Barrón	
343	5.3 - Definir diseño físico de Sistema Biosensor	15	29/08/2013	18/09/2013	Andrea Cukerman	339
337	5.4 - Fabricar o comprar PCB para montaje Sistema Biosensor	30	29/08/2013	09/10/2013	Juan Martín Ortega	291;292;339;352
578	5.5 - Fabricar o comprar el diseño físico definido	20	19/09/2013	16/10/2013	Andrea Cukerman	343
335	5.6 - Testear	10	10/10/2013	23/10/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	337
341	5.7 - Documentación	67	01/08/2013	01/11/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	335
293	**6 - Aplicación en el celular**	**63**	**01/11/2013**	**31/01/2014**		**285**
294	6.1 - Aprendizaje lenguaje de programación	15	01/11/2013	21/11/2013	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	
295	6.2 - Diagrama de módulos	4	01/11/2013	06/11/2013	Juan Martín Ortega	
296	6.3 - Definir interfaz de usuario	5	01/11/2013	07/11/2013	Andrea Cukerman	
297	6.4 - Definir criterios de manejo de alarmas	2	01/11/2013	04/11/2013	Ramiro Barrón	
298	6.5 - Desarrollar aplicación	35	22/11/2013	14/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	294;295;296;297
299	6.6 - Probar aplicación	13	15/01/2014	31/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	298
300	6.7 - Documentación	40	04/12/2013	31/01/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	295;296;297;298;299
345	7 - Unificar Documentación	80	06/11/2013	28/02/2014	Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón	300;308;315;321;341;348
125	**Hito 1**	**0**	**16/09/2013**	**16/09/2013**	**Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón**	
127	**Hito 2**	**0**	**17/02/2014**	**17/02/2014**	**Andrea Cukerman,Juan Martín Ortega,Ramiro Barrón**	

Tabla 22 - Cronograma detallado de la gestión del proyecto

Uno de los principales errores dentro de la planificación del proyecto fue distinguir el hardware del software tanto en la Pasarela como en el Sistema Sensor.
Tanto es así que en particular el Sistema Sensor adquirido fue un circuito integrado sin capacidad de programación, por lo cual la planificación prevista resultó obsoleta.

6.3. Análisis de riesgos

Elementos de análisis

Resulta interesante comprar el Análisis de riegos definidos en el Plan de Proyecto en Abril 2013 con la realidad durante el curso de desarrollo.

El Análisis definido fue el siguiente:

Hardware

Al ser este un proyecto que involucra Hardware, el riesgo natural que se presenta es la eventual falla o rotura de alguno de los componentes. Este riesgo se incrementa aún más dado que su origen puede deberse a muchos factores. Esto hace que, dependiendo de cual sea, pueda llegar a ser crítico.
Por ejemplo, si llegara a quemarse o a fallar un micro el día antes de la defensa del proyecto.
Probabilidad de ocurrencia: Muy probable
Nivel de impacto: Alto

Comunicación Inalámbrica

Una parte del proyecto consiste en lograr comunicar los datos provenientes del sensor a una aplicación cargada en un teléfono celular. Esta comunicación se pretende se haga de forma inalámbrica.
Podría llegar a pasar que por algún motivo esta comunicación inalámbrica no pueda llegar a establecerse.
Probabilidad de ocurrencia: Moderada
Nivel de impacto: Alto

Alimentación

Se pretende diseñar y fabricar un dispositivo el cual, no sólo haga de nexo entre el teléfono celular y el sensor, sino que además, sea capaz también de alimentar al sistema del biosensor logrando así que el mismo sea pasivo, y por ende, menos invasivo para el ser humano.
Podría acontecer que no se logre el correcto diseño del mismo no siendo éste entonces, capaz de alimentar al biosensor de forma correcta. O pudiera pasar, que para alimentar correctamente al biosensor se requiera de antenas o una batería excesivamente grandes haciendo esto, poco portable y práctico al dispositivo lo cual contradice uno de los objetivos planteados para el proyecto.
Probabilidad de ocurrencia: Moderada
Nivel de impacto: Bajo

Compras

Al tratarse de un proyecto que involucra Hardware y al tenerse el mismo que comprar muy probablemente en el exterior, puede pasar que todos o alguno de los componentes llegue a manos de este equipo de trabajo después de la fecha prevista.
Dependiendo de que componente se trate y de que tan después de esa fecha lo haga, será más o menos crítico.
Probabilidad de ocurrencia: Moderada
Nivel de impacto: Medio

Software

Podría llegar a ocurrir que una vez avanzado el proyecto, se detectara que el lenguaje de programación elegido o la arquitectura de software seleccionada no son los más adecuados para el mismo. Sin dudas el tener que ir para atrás y reprogramar buena parte del código impactaría notoriamente en los tiempos pre establecidos.

Probabilidad de ocurrencia: Poco probable

Riesgo	**Probabilidad de Ocurrencia**	**Nivel de Impacto**
Hardware	Muy probable	Alto
Comunicación Inalámbrica	Moderada	Alto
Alimentación	Moderada	Medio
Compras	Moderada	Medio
Software	Poco probable	Medio

		Probabilidad de Ocurrencia		
		Poco probable	Moderado	Muy probable
Nivel de Impacto	Ninguno			
	Bajo			
	Medio	Software	Compras/Alimentación	
	Alto		Com. Inalámbrica	Hardware
	Extremo			

Nivel de impacto: Medio

Tabla 23 - Tabla de Riesgos del proyecto

Plan de respuesta

Se evaluarán uno a uno los riesgos intentando que los mismos no queden en la "zona roja", y de ser posible, tampoco en "la amarilla".

Hardware

Este riesgo se consideró como muy probable debido a la cantidad de elementos de este tipo que se estarán manejando, y se consideró como de impacto alto dado que dependiendo de lo que pueda fallar, podría quedar hasta pausado el proyecto.

Para minimizar el impacto del mismo, se estarán comprando varias unidades de cada elemento. En particular, de aquellos considerados críticos como ser microcontroladores y antenas.
De esta forma se logrará mover el impacto de alto a bajo o incluso idealmente, ninguno si es que, dado los costos, se consigue tener respaldo de todos los componentes.

Se debió agregar como plan de contingencia la búsqueda de elementos alternativos de hardware y considerar además del riesgo eventual del daño, el asociado a fallas en la descripción de los dispositivos.

Comunicación Inalámbrica

Al depender esta comunicación íntegramente del diseño y fabricación de un dispositivo por parte del equipo de trabajo, la probabilidad de que algo falle estará siempre.
Como forma de evitar que el proyecto quede trunco por falta de lograr esta comunicación, se establecerá como contingencia el uso de una comunicación cableada entre el sistema del biosensor y el teléfono celular.
De esta forma se logra bajar el impacto de este riesgo de alto a ninguno.

No fue necesario debido a la elección de la tecnología de enlace utilizada

Alimentación

Como plan de contingencia para este caso, y con el fin de hacer hincapié en el desarrollo de la aplicación y de la comunicación entre el teléfono celular y el biosensor, se optará llegado este caso por el uso de un sistema biosensor activo en lugar de pasivo.
De esta forma se logra bajar el impacto de este riesgo de medio a bajo.

No fue necesario. El estudio previo demostró que el sistema sensor puede cumplir su función sin necesidad de una batería externa.

Compras

Parte de las medidas para minimizar esta eventualidad ya fueron tomadas al realizar la gestión de tiempos en donde se consideraron tiempos “excesivos” para contemplar estos casos.
De todos modos, como medida adicional, también se establecerán ciertas tareas, a priori posteriores, de modo de ir avanzando en otras cosas y no perder tanto tiempo.
Con estas medidas adicionales, se logrará bajar el impacto de este riesgo de medio a bajo.

El tiempo de demora en entrega en las compras no fue la mayor inconveniencia sino el material comprado en cuestión no era el esperado.

Software

Se considera que este riesgo es muy poco probable y de hecho no queda ni en la zona roja ni en la amarilla.
Esto se debe en gran parte a que los lenguajes de programación y la plataforma de desarrollo a utilizar están muy vinculados al hardware a elegir para lo cual se consultará también a los tutores.
Si bien es poco probable que se tenga que cambiar todo y re programarlo, sí cabe destacar que no hay manera de mover este riesgo para que tenga un impacto menor al que ya tiene.

Una vez analizados todos los riesgos precedentes, se llega a las siguientes tablas:

Riesgo	**Probabilidad de Ocurrencia**	**Nivel de Impacto**
Hardware	Muy probable	Bajo
Comunicación Inalámbrica	Moderada	Ninguno
Alimentación	Moderada	Bajo
Compras	Moderada	Bajo
Software	Poco probable	Medio

		Probabilidad de Ocurrencia		
		Poco probable	Moderado	Muy probable
Nivel de Impacto	Ninguno		Com. Inalámbrica	
	Bajo		Compras/Alimentación	Hardware
	Medio	Software		
	Alto			
	Extremo			

Tabla 24 - Tabla de Riesgos adecuada al Plan de Respuesta

6.4. Dificultades encontradas

Las fechas reales de los capítulos definidos fueron las siguientes:

1- Definiciones y Compras. Fecha fin real: 01/02/2014.
Comentarios:
Se debió separar las definiciones de la selección de compras.
El estudio previo de la tecnología finalizó en Julio 2013, y las primeras compras se efectuaron la primer semana de Setiembre 2013. Durante el desarrollo del proyecto, el hardware adquirido para la Pasarela tuvo que ser modificado debido a que era necesario programar en un microprocesador externo a la placa de desarrollo adquirida para lograr establecer una comunicación con el Sistema Sensor, no especificado correctamente en la descripción de la hoja de datos al momento de la compra. Considerando el tiempo limitado de entrega del proyecto se decidió adquirir un nuevo dispositivo, y consecuentemente aguardar su entrega, familiarizarse con el mismo, lo que resultó en un atraso en la planificación en 2 meses.

2- Pasarela (se incluye Hardware y Software). Fecha fin real: 15/06/2014.
Comentarios:
Como se menciona anteriormente, no se debió diferenciar entre el Hardware y Software en la Pasarela. La relación de dependencia entre ambos en el proyecto es tan fuerte, que resulta imposible desarrollar una aplicación sin contar con los requerimientos hardware predefinidos.
Tampoco se debió separar el proyecto en los 3 bloques a desarrollar, sino en la funcionalidad de los enlaces de comunicación entre los mismos. De esta manera se hubiese separado en objetivos específicos de la forma: Comunicación Pasarela – Celular, Comunicación Pasarela – Sistema Sensor, Unión de ambos proyectos.

3- Sistema Sensor (se incluye Hardware y Software). Fecha fin real: 01/05/2014.
Comentarios:
No hubo mayores inconvenientes en el Sistema Sensor.
La calibración del sensor del termistor tuvo ciertas dificultades considerando que uno de los criterios de éxito del proyecto era lograr una precisión de 0.1ºC, por lo que los valores de los registros utilizados debieron ser configurados de manera muy exacta para poder alcanzar este valor.

4- Aplicación de celular. Fecha fin real: 01/06/2014.
Comentarios:
El desarrollo de la aplicación de celular resultó el punto más sencillo del proyecto por la facilidad de información disponible en la web. Si bien no se encontraba especificado el desarrollo ejecutable de las excepciones mencionadas en el Capitulo 3 dentro del alcance de la aplicación, el desarrollo es de suma importancia y deberá efectuarse en la posible continuación del proyecto.

Otras dificultades encontradas durante el desarrollo del proyecto que derivaron directamente en el atraso en la planificación del mismo fue la fuerte dedicación horaria laboral presente en los 3 integrantes del grupo. Durante el curso de gestión de proyectos, la jornada laboral de 2 de los 3 integrantes era de 2 horas diarias menor a la realizada durante el primer semestre 2014, dificultando el horario de las reuniones.

Para finalizar, el primer semestre del año 2013 fue dedicado exclusivamente al estudio de la tecnología de Radio frecuencia, lo que derivó en un mayor entendimiento de la temática y en una mejor selección de compras dentro de lo posible. Sin embargo, la dedicación al estudio fue tal que al no haber trabajado sobre lo tangible sin obtener algún resultado en concreto, hizo que el grupo se desmotivara al punto que se decidió adelantar el desarrollo de la aplicación de celular siendo el capítulo más independiente del proyecto.

6.5. Dedicación horaria

A continuación se representan en dos figuras la dedicación horaria del proyecto.
La Tabla 25 muestra una estimación de las horas semanalmente por integrante.
Los valores no son exactos, pero intentan demostrar la influencia de las actividades semanales durante el trascurso del proyecto.

DATOS								
Semana	Mes	Fecha	Andrea	Ramiro	Juan Martín	Equipo	Por mes y por Equipo	Observaciones
1	ABRIL	8 al 14	10	11	10	31		Curso: Tarea 1
2		15 al 21	12	11	12	35	117	Curso: Tarea 2
3		22 al 28	6	12	9	27		Curso: Tarea 3
4		29 al 5	7	9	8	24		Curso: Plan de Proyecto
5	MAYO	6 al 12	6	5	4	15		Curso: Plan de Proyecto/Parciales
6		13 al 19	10	9	10	29	106	Curso: Plan de Proyecto
7		20 al 26	11	10	9	30		
8		27 al 2	9	11	12	32		
9	JUNIO	3 al 9	10	11	9	30		
10		10 al 16	11	9	10	30		
11		17 al 23	8	10	9	27	114	
12		24 al 30	10	9	8	27		
13	JULIO	1 al 7	8	7	6	21		Parciales
14		8 al 14	7	5	5	17		Parciales
15		15 al 21	9	8	8	25	108	Exámenes
16		22 al 28	6	6	7	19		Exámenes
17		29 al 4	9	9	8	26		Exámenes
18	AGOSTO	5 al 11	12	9	10	31		
19		12 al 18	13	9	8	30		
20		19 al 25	16	11	16	43	148	
21		26 al 1	13	16	15	44		
22	SETIEMBRE	2 al 8	13	12	12	37		
23		9 al 15	16	15	16	47	154	Hito 1
24		16 al 22	10	13	12	35		
25		23 al 29	11	10	14	35		
26	OCTUBRE	30 al 6	7	3	3	13		Parciales
27		7 al 13	13	14	19	46		
28		14 al 20	16	16	17	49	184	
29		21 al 27	10	15	11	36		Lab0 Control
30		28 al 3	12	15	13	40		Lab1 Control
31	NOVIEMBRE	4 al 10	13	13	17	43		Lab2 Control
32		11 al 17	12	17	14	43		Lab3 Control
33		18 al 24	16	16	13	45	176	Info Control
34		25 al 1	14	15	16	45		Parciales
35	DICIEMBRE	2 al 8	18	16	12	46		Parciales/Exámenes
36		9 al 15	21	18	16	55		Exámenes
37		16 al 22	23	20	18	61	184	Exámenes
38		23 al 29	12	4	6	22		
39	ENERO	30 al 5	5	3	4	12		
40		6 al 12	19	18	18	55		
41		13 al 19	17	21	20	58	235	
42		20 al 26	18	19	22	59		
43		27 al 2	16	16	19	51		
44	FEBRERO	3 al 9	18	20	17	55		Exámenes
45		10 al 16	16	15	18	49		Hito 2/Exámenes
46		17 al 23	15	14	19	48	202	Exámenes
47		24 al 2	15	19	16	50		Exámenes
48	MARZO	3 al 9	14	15	13	42		
49		10 al 16	21	18	18	57		
50		17 al 23	18	19	16	53	208	
51		24 al 30	16	20	20	56		
52	ABRIL	31 al 6	16	21	18	55		
53		7 al 13	17	16	18	51		
54		14 al 20	16	19	20	55	207	
55		21 al 27	10	9	9	28		
56		28 al 4	5	7	6	18		
57	MAYO	5 al 11	15	16	15	46		
58		12 al 18	25	25	25	75		Parciales
59		19 al 25	16	4	4	24	250	
60		26 al 1	35	35	35	105		
61	JUNIO	2 al 8	12	15	15	42		
62		9 al 15	4	15	15	34		
63		16 al 22	0	4	4	8	84	
64		23 al 29				0		
		Totales	819	832	826	**2477**		

Tabla 25 - Dedicación horaria semanal por integrante

Las siguientes figuras comparan la dedicación horaria real contra las horas nominales previstas de trabajo a razón de 15 horas semanales.

RESUMEN - Horas nominales a 15HRS semanales			
Mes	**Horas Nominales**	**Horas Reales**	**Porcentaje**
Abril	180	117	65.0%
Mayo	180	106	58.9%
Junio	180	114	63.3%
Julio	225	108	48.0%
Agosto	180	148	82.2%
Setiembre	180	154	85.6%
Octubre	225	184	81.8%
Noviembre	180	176	97.8%
Diciembre	180	184	102.2%
Enero	225	235	104.4%
Febrero	180	202	112.2%
Marzo	180	208	115.6%
Abril	225	207	92.0%
Mayo	180	250	138.9%
Junio	135	84	62.2%
Total	2835	**2393**	84.4%

Tabla 26 - Resumen de Horas nominales vs Horas Reales

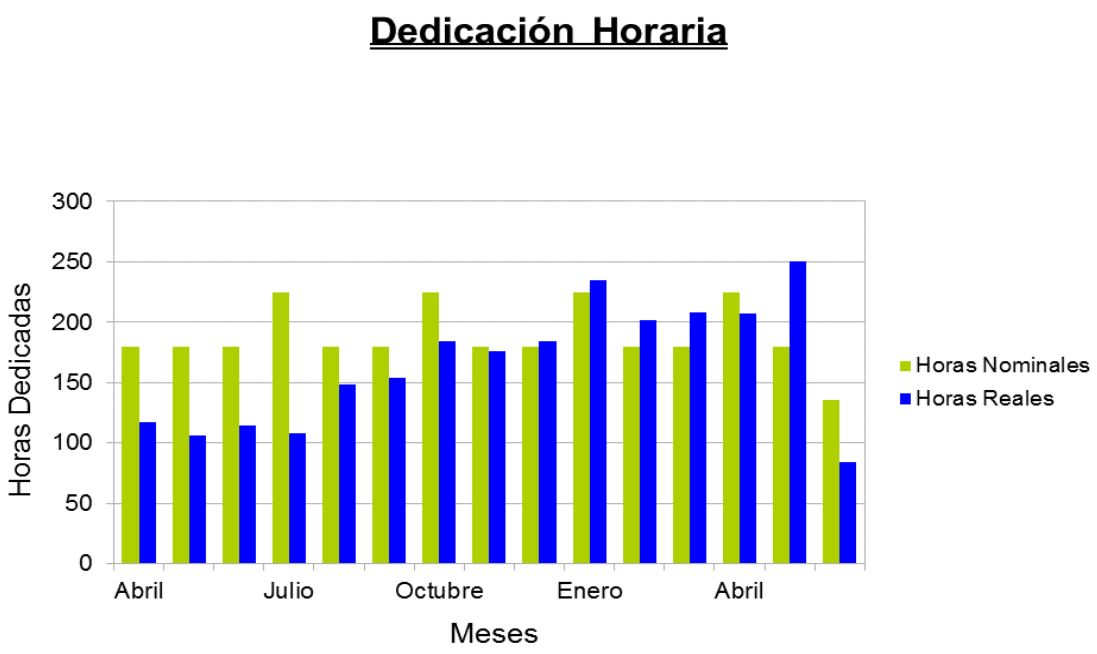


Figura 31 - Gráfica Dedicación Horaria

6.6. Costos del proyecto

Los costos se pueden dividir en dos grandes categorías: costos vinculados a la compra de material de Hardware y la dedicación horaria grupal en el transcurso del proyecto.

Descripción	Cantidad	Precio	Total
Hardware			
EVB 90121 - EVALUATION BOARD FOR MLX90121	1	U$S 138.62	U$S 138.62
EVB 90129 - EVALUATION BOARD FOR MLX90129	1	U$S 132.41	U$S 132.41
MLX90129 - IC SENSOR TAG 13.56MHZ	5	U$S 4.56	U$S 22.80
MLX90121 - IC RFID TXRX 13.56MHZ	5	U$S 6.89	U$S 35.43
SHIPPING CHARGES		U$S 43.90	U$S 43.90
SALES TAX		U$S 19.68	U$S 19.68
TRF7960EVM - EVALUATION MODULE FOR TRF7960	1	U$S 100	U$S 100
BT MODULE	1	U$S 8	U$S 8
	Total		**U$S 499.50**
	IVA 22 %		**0**
	Total General		**U$S 499.50**

Descripción	Cantidad	Precio	Total
Horas Hombre			
Dedicación horaria dedicada durante el proyecto	2.396	U$S 100	U$S 23.960
(No se distingue las horas entre los tres integrantes del grupo)			
(El precio de la hora se fijó de acuerdo a las pautas del mercado a U$S100 la hora)			
	Total		**U$S 23.960**
	IVA 22 %		**0**
	Total General		**U$S 23.960**

TOTAL	U$S 24.459

Tabla 27 - Costos del proyecto

No hubo costos de software asociados.

Se adjuntan las facturas en el CD.

Capítulo 7
Proyecto de Producción

7.1. Introducción

Para lograr disminuir el tamaño del sensor y abaratar los costos, se plantea a continuación una proyección de producción de MONCEL.

Considerando que el sistema sensor deberá ir dentro del cuerpo, se rediseña el tamaño de la antena en base a las recomendaciones del fabricante como se especifica en la Application Note Antenna Design for MLX90129 [19].

7.2.- Adaptación de señal y Antena en el Sensor

Diseño de antena

Los pasos para el diseño del circuito implementado se describen en punto 3.5 Interconexión de componentes.

La frecuencia de resonancia de un circuito LC está definida por la ecuación:

$$f_{resonancia} = \frac{1}{2\pi\sqrt{LC}}$$

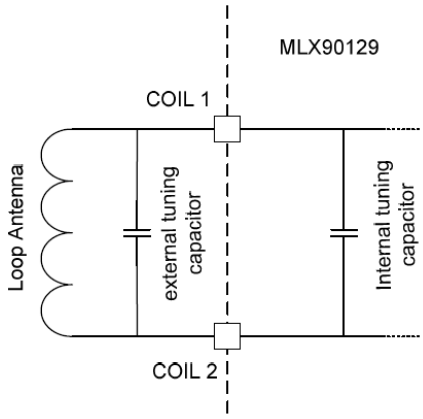


Figura 32 – Modelo de Antena de MLX90129 tomada de "MLX90129 Datasheet" pag. 4 [19]

La frecuencia de la portadora de del estándar ISO 15693 es de 13.56MHz, por lo que se diseñó la antena para que resuene a esta frecuencia.
Considerando la capacitancia interna del MLX90129 y la frecuencia de resonancia podemos obtener el valor de inductancia que necesitamos para nuestro diseño. Para poder mejorar la sintonía de nuestro resonador, se le agrega una capacitancia variable de 45pF al resonador LC.

Finalmente se obtiene:

$$L = \frac{1}{(2\pi * f_{resonancia})^2 * C} = \frac{1}{(2\pi * 13.56 * 10^6)^2 * (75 * 10^{-12} + C_{Variable})}$$

$$L = \begin{cases} C_{Variable} = 0 \Rightarrow L = 1.837uHy \\ C_{Variable} = 45pF \Rightarrow L = 1.148uHy \end{cases}$$

Luego se diseñó la inductancia utilizando la ecuación:

$$L = \frac{A^2 n^2}{30A - 11D}\mu Hy \qquad donde \qquad A = \frac{D + N(w + s)}{2}$$

$$con \begin{cases} D: Diámetro\ interior \\ s: espacio\ entre\ pistas \\ w: Ancho\ de\ pista \\ N: Número\ de\ vueltas \end{cases}$$

$$Considerando \begin{cases} D = 0.38in \\ s = 0.02in \\ w = 0.02in \\ N = 6 \end{cases} \xrightarrow{obtenemos} L = 1.355\mu Hy$$

Energía

Siguiendo con lo recomendado en la hoja de datos de la placa de evaluación, se colocó un diodo zener entre Vfield y Vss para evitar sobre voltajes. En paralelo, se colocó un capacitor para acumular la energía del lector de radio frecuencia.

Valores de componentes

Nombre	**Valor**
DZ	4.7V
C1	100nF
C2	0 - 47pF
Rp	0 – 10kΩ
Rth	0 – 10kΩ

Tabla 28 - Resumen de valores para proyecto de producción

Diseño del PCB

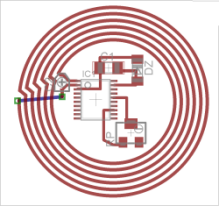
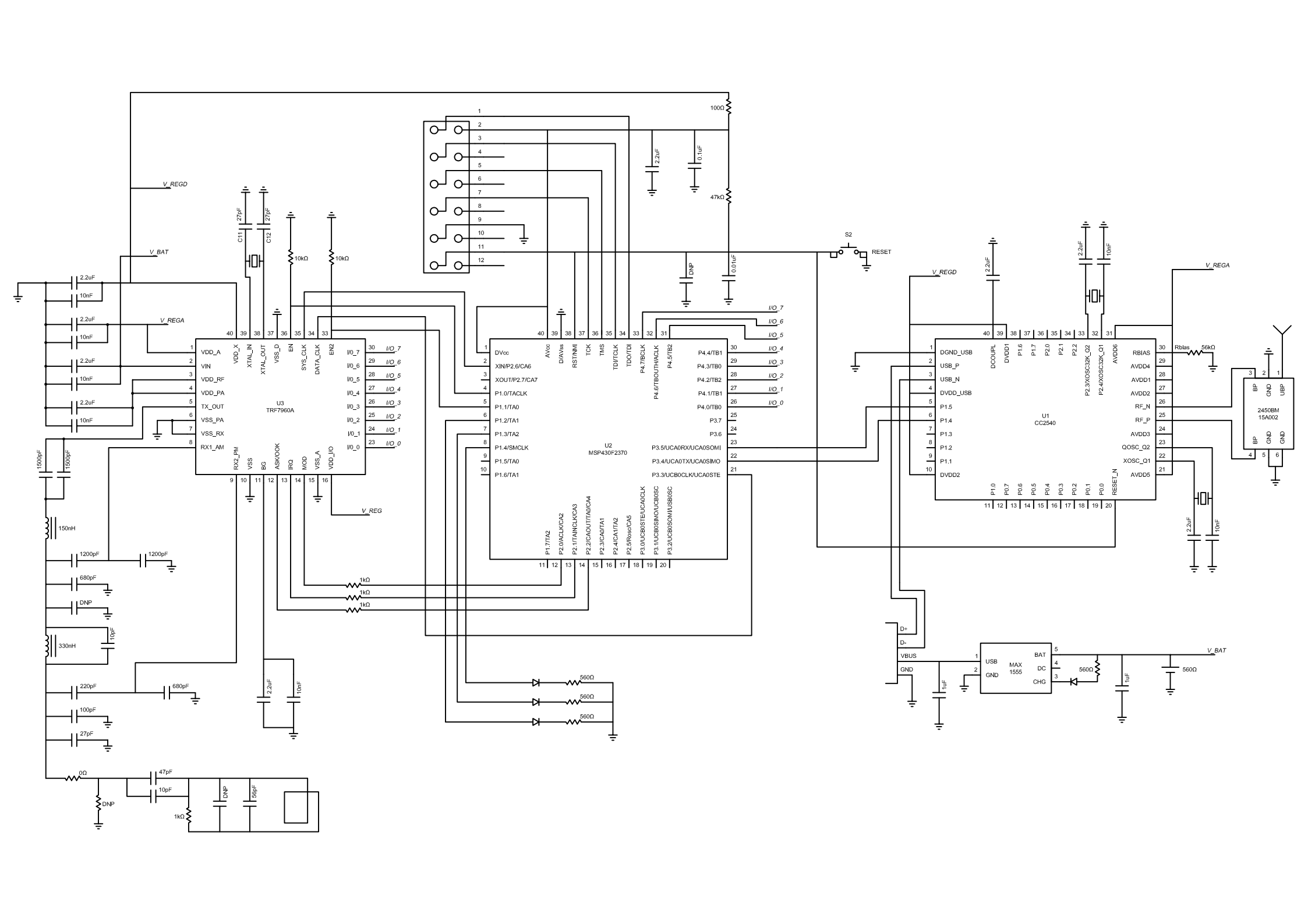


Figura 33 - PCB generado

7.3.- Esquemático de Pasarela

En la siguiente figura se muestra una versión preliminar del esquemático de la Pasarela. Cuenta con un MSP430F2370 (microcontrolador que gestión la transferencia), TRF7960A (RF frontend), CC2540 (bluetooth frontend), MAX1555 (cargador batería de litio) y el 2450BM15A002 (filtro de 2.4GHz).


V_REGD
V_BAT
V_REGA
V_REG
U3
TRF7960A
VDD_A
VIN
VDD_RF
VDD_PA
TX_OUT
VSS_PA
VSS_RX
RX1_AM
VDD_X
XTAL_IN
XTAL_OUT
VSS_D
EN
SYS_CLK
DATA_CLK
EN2
I/0_7
I/0_6
I/0_5
I/0_4
I/0_3
I/0_2
I/0_1
I/0_0
RX2_PM
VSS
BG
ASK/OOK
IRQ
MOD
VSS_A
VDD_I/O
U2
MSP430F2370
DVcc
XIN/P2.6/CA6
XOUT/P2.7/CA7
P1.0/TACLK
P1.1/TA0
P1.2/TA1
P1.3/TA2
P1.4/SMCLK
P1.5/TA0
P1.6/TA1
AVcc
DVss
RST/NMI
TCK
TMS
TDI/TCLK
TDO/TDI
P4.7/BCLK
P4.6/TBOUTH/ACLK
P4.5/TB2
P4.4/TB1
P4.3/TB0
P4.2/TB2
P4.1/TB1
P4.0/TB0
P3.7
P3.6
P3.5/UCA0RX/UCA0SOMI
P3.4/UCA0TX/UCA0SIMO
P3.3/UCB0CLK/UCA0STE
P1.7/TA2
P2.0/ACLK/CA2
P2.1/TAINCLK/CA3
P2.2/CAOUT/TA0/CA4
P2.3/CA0/TA1
P2.4/CA1/TA2
P2.5/Rosc/CA5
P3.0/UCB0STE/UCA0CLK
P3.1/UCB0SIMO/UCB0SC
P3.2/UCB0SOMI/USB0SC
U1
CC2540
DGND_USB
USB_P
USB_N
DVDD_USB
P1.5
P1.4
P1.3
P1.2
P1.1
DVDD2
DCOUPL
DVDD1
P1.6
P1.7
P2.0
P2.1
P2.2
P2.3/XOSC32K_Q2
P2.4/XOSC32K_Q1
AVDD6
RBIAS
AVDD4
AVDD1
AVDD2
RF_N
RF_P
AVDD3
QOSC_Q2
XOSC_Q1
AVDD5
RESET_N
P1.0
P0.7
P0.6
P0.5
P0.4
P0.3
P0.2
P0.1
P0.0
Rblas
56kΩ
2450BM
15A002
BP
GND
UBP
MAX
1555
USB
GND
BAT
DC
CHG
D+
D-
VBUS
S2
RESET
100Ω
47kΩ
10kΩ
27pF
C11
C12
2.2uF
0.1uF
0.01uF
DNP
10nF
1500pF
150nH
1200pF
680pF
330nH
220pF
100pF
27pF
0Ω
47pF
10pF
56pF
1kΩ
560Ω
1uF

7.3. Estimación de costos

De acuerdo a las horas insumidas en el proyecto, y las compras asociadas al hardware, el costo de la unidad de MONCEL es de **U$S 24.500**.

Relevando en forma primaria informalmente en el mercado uruguayo, el costo de fabricar 10 unidades del sistema sensor es de $U 2.200, lo que significa un costo de U$S 9.5 cada unidad. Si a esto le sumamos el costo del MLX90129 llegamos a **U$S 14** por unidad de sistema sensor.

A esto debemos sumarle el costo de la fabricación de la Pasarela, que debe incluir el costo del módulo BT, la fabricación del PCB, el MSP430, TRF7960 y la batería. Si asumimos que el costo de fabricación del PCB para 10 unidades de pasarela es el mismo que para el sensor, y al igual que en el caso anterior no varían los costos de los elementos adquiridos, cada unidad de pasarela se estima en **U$S 40** aproximadamente.

Los costos se resumen en la Tabla 29:

Concepto	**1**	**10**	**100**	**1000**
Costo de desarrollo	U$S 24.500	U$S 2.450	U$S 245	U$S 25
Sistema Sensor (*)	Incluido	U$S 14	U$S 13	U$S 8
Pasarela(*)	Incluido	U$S 40	U$S 37	U$S 22
Horas de armado (U$S 100 la hora) (**)	Incluido	U$S 3.000	U$S 500	U$S 100
TOTAL por unidad aprox.	**U$S 24.500**	**U$S 5.500**	**U$S 800**	**U$S 160**

Tabla 29 - Proyección de costos de producción en escalas crecientes

(*) Con 10 unidades, el costo desciende 10% con respecto al de la unidad.
Con 100 unidades, el costo desciende 25% con respecto al de la unidad
Con 1000 unidades, el costo desciende 50%, con respecto al de la unidad
Criterios tomados de Digikey a la fecha de Julio 2014 [20]

(**) De las 2.400 horas dedicadas al proyecto, se estima que aproximadamente un 10% de las horas fueron dedicadas a la compra de los componentes, a la interconexión de los mismos y a la calibración del sensor, lo que equivale a 240hrs de trabajo. Si asumimos que en producción de 10 componentes la compra, conexionado y fabricación del PCB ya están contemplados, los tiempos se producción se pueden reducir a 30hrs de trabajo para la fabricación de 1 sensor. Si con práctica se logra diseñar una metodología tanto para la calibración como para el testeo del sensor, las horas de trabajo pueden disminuir hasta 5hrs y 1hrs para el armado de 1 unidad en un lote de 10 y 1 unidad en un lote de 1000 respectivamente.

Capítulo 8
Conclusiones

La diabetes es una enfermedad crónica debido a una insuficiente producción de la hormona insulina por parte del páncreas, resultando en altos valores de azúcares en la sangre. El monitoreo continuo de estos valores puede prevenir daños en la visión, infecciones en la piel, malformación de articulaciones y resulta imprescindible para evitar una posible hipo o hiperglucemia.
Dentro de este marco se desarrolló MONCEL, un sistema sensor electrónico que debe funcionar sin batería y con características subcutáneas, capaz de trasmitir continuamente al celular del paciente los valores de la variable fisiológica medida.

Estas características implicaron la necesidad de un estudio sobre las diferentes tecnologías de alimentación externas, área en la cual ningún integrante del equipo contaba con la experiencia. Esto derivó en cuatro meses en el transcurso del proyecto dedicados al estudio del enlace, descartando elecciones que en un primer momento parecían acordes a la tarea pero que luego profundizando la investigación no cumplían con las especificaciones requeridas como ser tamaño de antena, permeabilidad en la piel, entre otras.

Finalmente se concluyó en la selección de la **Tecnología RF de 13.56MHz** para la comunicación entre la Pasarela y el Sistema Sensor. Esta frecuencia es compatible con la ubicación subcutánea.

Se alcanzó así el objetivo de pasividad del sensor debajo de la piel, siendo alimentado únicamente por la fuente de energía proveniente de la Pasarela y trabajando en una banda de frecuencias que permite que el sensor cuente con características subcutáneas.

Para poder determinar el rango máximo de funcionamiento, se procedió a realizar 8 ensayos de medidas en donde se registraron valores de temperaturas y del voltaje generado a partir del campo inducido en el sistema sensor.

Se determinó rango de distancia máximo entre el sensor y la pasarela donde la medida se mantiene confiable. El alcance resultó de **6,5cm** donde el voltaje inducido permanece constante en 2.94V y la medida sólo depende de la variación que pueda existir debido al margen de error que exista en el termistor.

La temperatura corporal promedio registrada por los ensayos varió en un rango de [36,5ºC…36,7ºC], logrando obtener una media de 36,6ºC, igual a la temperatura medida por el termómetro de mercurio.

Más allá de esta distancia, el sistema fue diseñado para bloquear la medida en cado de registrarse un “campo débil”, o sea superior al límite.

Se alcanzó un error de ≈ **±0.1ºC** limitado por la incertidumbre del propio termistor.

Las distancias de funcionamiento estan acotadas por dos parámetros: entre el sensor y pasarela máximo de 6cm y no más de **16m** entre la pasarela y el celular.

Con respecto al requerimiento de distancia entre el celular y el sensor, la distancia máxima de 4m requerida queda contemplada en el alcance del enlace Bluetooth entre la pasarela y el celular.

La aplicación de celular por su parte debía contar con la funcionalidad de alertar al usuario si el dato recibido excede el rango normal predefinido, contando además con un diario de registros.

Esto último implica por un lado administrar la recepción de los datos en la aplicación del celular desde la Pasarela, y por otro el desarrollo de software propio de la aplicación para cumplir con los niveles de alerta definidos.

Con respecto a la recepción de los datos, la tecnología de comunicación entre pasarela y celular elegida fue la trasmisión por **Bluetooth**. Considerando la seguridad en el enlace, su popularidad en el mercado, el consumo energético, y su facilidad de implementación, concluimos que la elección fue acertada cumpliendo con los requerimientos definidos. Adicionalmente, la tecnología permitió superar el criterio de distancia mínima compensando el limitado alcance de comunicación entre Pasarela y el Sistema Sensor.

En la aplicación de celular se logró cumplir con los **requerimientos de alerta**, notificando a pantalla al paciente y enviando un mensaje de texto al celular predefinido.
Adicionalmente, se almacena la información recibida por bluetooth en una **base de datos** propia de Android y se imprime en la pantalla inicial los valores de la medida.

Si bien no se encontraba dentro del alcance del proyecto, se gestionó el sistema de preferencias del usuario, donde el paciente es capaz de definir el número de celular de alerta, activar y desactivar esta función, encender el Bluetooth para permitir la conexión y definir el rango de notificaciones.

Por otra parte, Android es un sistema libre de distribución y código abierto, lo que implicó que el desarrollo no llevó ningún costo asociado. Al no contar con experiencia en la plataforma de la aplicación resultó imprescindible la característica de open source, habilitándonos el acceso a portales de información y blogs para consultas durante el desarrollo del proyecto.

Desde el punto de vista energético, se logró estimar la duración de la batería de 1300mAh en **3 días** para un período de muestreo de 5 minutos. Se observó que el tiempo entre las tomas no influye en el consumo energético debido al alto consumo en el estado de Stand By entre las muestras.

Este estado queda determinado en un 72% por el consumo que implica dejar el módulo de Bluetooth activo. Si apagamos el Bluetooth durante la etapa de reposo y lo volvemos a encender cada vez que se necesita trasmitir, se logra para el mismo periodo de muestro triplicar la duración de la batería. Se pudo visualizar que es conveniente siempre apagar y encender el Bluetooth para periodos mayores a 10s.

A lo largo del proyecto se detectaron puntos débiles a ser tenidos en cuenta por futuras revisiones, siendo apropiadamente documentados en los capítulos correspondientes.

Algunas posibles mejoras que sugerimos para la continuación del proyecto si fuese elegido por otro grupo de trabajo son:

- Nuevas funcionalidades de la aplicación de celular:
  - Utilización del GPS para notificar las coordenadas en caso de emergencia si el dato excede el rango
  - Parametrización de manejo de los tiempos en la aplicación
  - Generación de la información clínica en formato CDA para la historia electrónica
- Implementación del proyecto de producción
- Integración del sensor de glicemia al proyecto
- Mejora en eficiencia de consumo energético de Pasarela apagando el Bluetooth en estado de stand by
- Posibilidad de integración física de la Pasarela al celular

La innovación de MONCEL consiste en una trasferencia de datos continua desde un sensor pasivo y su integración con los celulares del mercado. MONCEL permite emprender un camino de desarrollo que esperamos llegue a colaborar con una mejor calidad de vida durante el monitoreo continuo de la enfermedad en los pacientes que padecen de diabetes.

Bibliografía

[1] *"American Diabetes Association®."* [Online]. Available: http://www.diabetes.org/. [Accessed: 10-Jul-2014].
See more at: Living With Diabetes > Treatment and Care > Blood Glucose Testing - http://www.diabetes.org/living-with-diabetes/treatment-and-care/blood-glucose-control/?loc=lwd-slabnav#sthash.JVOLJ6aP.dpuf
See more at: Diabetes Basics > Statistics - http://www.diabetes.org/diabetes-basics/statistics/?loc=db-slabnav

[2] *"International Diabetes Federation"* [Online]. Available: http://www.idf.org/.[Accessed: 10-Jul-2014].
See more at: Home > diabetesatlas - http://www.idf.org/diabetesatlas

[3] International Diabetes Federation, *"International Diabetes Federation: Diabetes Atlas. "* Sixth Edition, Brussels, Belguim, pp. 1–103, 2013

[4] V. Chawla and D. S. Ha, *"An overview of passive rfid"* Communications Magazine IEEE (Volume:45 , Issue: 9) , Virginia Polytech. Inst. & State Univ., Blacksburg, pp. 11–17 2007.

[5] K. Finkenzeller, "*RFID Handbook: Fundamentals and applications in contactless smart cards, radio frequency identification and near field communication"*, Third Edition, John Wiley and Sons Ltd Publication, Munich, Germany, pp. 350–480, 2010.

[6] G. Benelli and A. Pozzebon, *"RFID Under Water: Technical Issues and Applications."*, in *"Radio Frequency Identification from System to Application"* , First Edition, University of Siena, Department of Information Engineering, Intech ®, Siena, Italy, pp. 380-396, 2007 .

[7] Texas Instruments Inc., *"NFC ISO15693 Sensor Transponder Datasheet."* Rev 3, Dallas, Texas, pp. 1-7, 2012

[8] ISO Central Secretariat, *"Identification cards - Contactless integrates circuit(s) cards - Vicinity cards - Part 3: Anticollision and transmission protocol."* ISO/IEC FCD 15693-3, Geneva, Switzerland, pp. 14-46, 2000.

[9] Melexis Microelectronics Systems, *"MLX90129 13.56MHZ sensor tag / Datalogger IC Datasheet"* Rev 9, Ieper, Belgium, pp. 1–60, 2012

[10] Melexis Microelectronics Systems, *"MLX90121 13.56MHZ RFID Transciever Datasheet"* Rev 10, Ieper, Belgium, pp. 1–27, 2012

[11] Texas Instruments Inc., *"TRF7960A Multi-Protocol fully integrated 13.56 MHZ RFID reader/writer IC Datasheet"* Rev 3, Dallas, Texas, pp. 1-70, 2013

[12] *"Basics | Bluetooth Technology Website."* [Online]. Available: http://www.bluetooth.com [Accessed: 10-Jul-2014].
See more at: What is Bluetooth Technology > Technology Basics - http://www.bluetooth.com/Pages/Basics.aspx

[13] Core Electonics, *"Core Electronics User Guide, JY-MCU Bluetooth to UART Wireless Serial Port Module for Arduino"* Greenhills, Australia, pp. 1-8

[14] *"Titanium - Biogenesis."* [Online]. Available: http://www.biogenesis.com.uy/es/productos/temperatura/titanium.html. [Accessed: 04-Aug-2014].

[15] *"Android Developers."* [Online]. Available: http://developer.android.com. [Accessed: 10-Jul-2014].
See more at: Develop > API Guides > Introduction > App Fundamentals - http://developer.android.com/intl/es/guide/components/fundamentals.html.

[16] Melexis Microelectronics Systems, *"Application Note MLX90129 Acquisition Chain"* Rev 1, Ieper, Belgium, pp. 1–10, 2011.

[17] J. Schillinger, *"Application Note Antenna Matching for the TRF7960 RFID Reader,"* Texas Instruments Inc, Dallas, Texas, pp. 1–8, 2013.

[18] D. C. Baird, *"Experimentation: An introduction to Measurement theory and experiment design: Chapter 2 Measurement and Uncertainty"* Third Edition, Prentice Hall, INC, New York, United States of America. 1995

[19] Melexis Microelectronics Systems, *"Application Note MLX90129 Antenna Design"* Rev 1, Ieper, Belgium, pp. 1–8, 2010.

[20] *" MLX90129RGO-CAA-000-RE Melexis Technologies NV | DigiKey Costs"* [Online]. Available: http://www.digikey.com/product-detail/en/MLX90129RGO-CAA-000-RE/MLX90129RGO-CAA-000-RECT-ND/3721091. [Accessed: 04-Aug-2014]

Apéndices

Apéndice A – Cálculo de Incertidumbre

El cálculo de la incertidumbre Tipo A se calcula mediante la siguiente relación, donde xi es el valor de cada medida, x es el promedio de los valores registrados, y n es la cantidad de repeticiones efectuadas, que en todos los ensayos es igual a 10.

$$u(x_i) = \sqrt{\frac{\sum_{k=1}^{n} (x_i - \bar{x})^2}{n(n-1)}} \ .$$

Utilizando un factor de cobertura de k=2 que produce un nivel de confianza de un 95%, se calcula la incertidumbre expandida. [18]

Apéndice B – Impresiones de pantallas de pruebas

Ensayo 1: Sensor encima de Pasarela

Ensayo 3: Sensor a 2,5 cm de la Pasarela

Ensayo 4: Sensor a 5.5 cm de la Pasarela

Ensayo 5: Sensor a 7.5 cm de la Pasarela

Ensayo 6: Sensor a 9.4 cm de la Pasarela

Ensayo 7: Sensor a 45º con respecto a la superficie de la mesa y pegado a la Pasarela

Ensayo 8: Sensor a 90º con respecto a la superficie de la mesa y encima de la Pasarela

Apéndice C – Paper

MONCEL

Transmission of a physiological variable from a subcutaneous passive device to the cellphone

Ramiro Barrón, Andrea Cukerman, Juan Martin Ortega

Universidad de la República
Montevideo, Uruguay

In diabetic patients, measurement of blood glucose levels is the main source of information to ensure adequate control of the disease. At the present day, patients have two options monitoring systems: the commonly used glucose meters where they must remember to draw a blood sample for examination, and insulin pumps that have subcutaneous device measurement which transmit the data wirelessly to an external receiver or "beeper". These patients should periodically remove the implant to replace the battery with the discomfort that this process entails. [1]

MONCEL is specially designed to meet these needs, with a subcutaneous device that works without batteries and is capable of continuously transmitting the patient's sensed values to his or hers mobile. Then, if the data received exceeds a predefined range, the app will alert the user and text an emergency contact, storing the information for further consultation.
This way, MONCEL achieves a continuous monitoring of the measured variable reaching an accuracy up to 0.1ºC in body temperatures, maximum range of 10m between the sensor and the mobile, and costs of production of less than USD 160 per unit in one thousand batch.

I. FUNCTIONAL SPECIFICATION SYSTEM

The project is divided into 3 main blocks shown in Figure 1:

1. System Sensor: integrated circuit connected to the sensor.
The system features will be continually sensing the variable of interest without interrupting the patient's activities and without their own source of energy thus avoiding extraction to remove your battery sensor. While subcutaneous development is excluded from the scope of the project, the need for this feature is decisive in the election of communication technologies.
It works with a temperature sensor mechanism simulating the operation of the system as if there were glucose sensors.

2. Gangway: External embedded system that feeds the sensor system, receives data and re transmits to the mobile.
Its main function is to intercommunicate the Sensor System and mobile application. It should provide enough energy for the sensor system to succeed in obtaining the measure, and be able to read and interpret the data and re transmit it to the mobile. It will have a separate power supply.

3. Application Mobile: single block of user interaction.

The user must download the app and it will be the only mechanism of interaction and system configuration. The application receives the data through the gangway and stores in a corresponding data base. If the value exceeds the range defined by the user, it will handle an alert system in the mobile itself and via text messages to third parties to prevent a possible complication

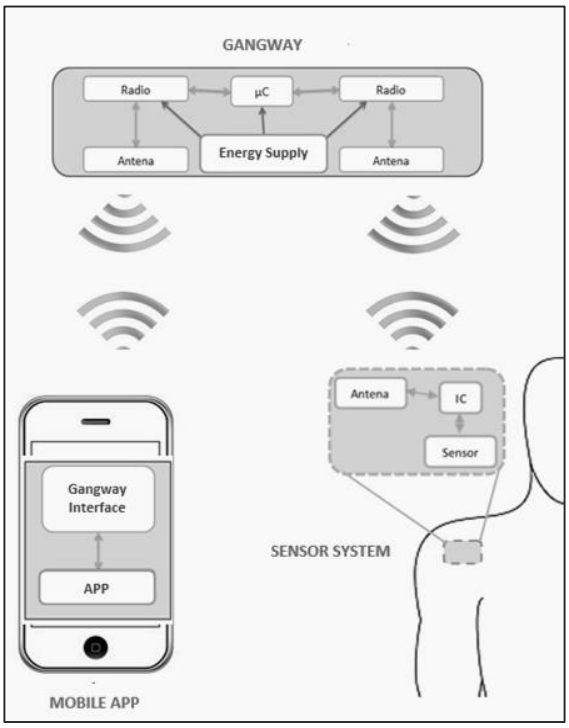


Figure 1 - Block diagram of MONCEL. Sensor System: integrated circuit connected to sensor, Gangway: External embedded system that feeds the sensor system, receives data and re transmits to the mobile. Application Mobile: single block of user interaction.

II. DESIGN CONSIDERATIONS

Given the dependence of the passive sensor system and the gangway, MONCEL uses RF (Radio Frequency) technology as a means of communication between the two.
The technology involves the communication of a transponder or tag containing the information to be transmitted, along with its corresponding reader or transceiver that sends the RF signal to the tag transmit the stored data. [2]

Within the 3 possible radio frequencies studied bands, HF is chosen due to the advantages in the Table 1.

Regarding the communication technology between the gangway and the cellphone, it is used Bluetooth LE. This technology is present in most of the latest generation mobile devices since 2000. Their operation is not dependent on the cellular network, only limited by the distance between the mobile device and the gateway device that can reach up to 10m depending on the capacity of the cellular model user. [3]

Tecnology	Frequency	Range	Skin permeability	Antenna´s size	Transponder
LF	3 - 300kHz	< 50cm	YES	Too big for implantable devices	Passive/Active
HF	3 - 30MHz	~ 1m	YES	Recomended	Passive/Active
UHF	300MHz - 3GHz	< 9m	NO	Small	Active

Table 1 - Radio frequency selection

III. TESTING AND RESULTS

By acquiring transponder *MLX90129* [4] from Melexis, transceiver *TRF7960* [5] from Texas Instruments, *JYMCUBLUETOOTH* from Core Electronics, MONCEL began testing its operation with a thermistor as biosensor.

There were 80 measurements achieved using the thermistor as a body thermometer. Previously and after the trials during a time span of 2hrs, the body temperature measured by a conventional mercurial thermometer was 36.6ºC. By moving the gangway away from the sensor system in a parallel plane between both elements, we observed that the measurement obtained depended on this distance. The reason for this is because the voltage generated in the sensor system from the field induced by the gateway decreases when de distance between both grows as shown in Figure 2. The measurement depends on this voltage as it feeds the Wheathstone bridge design for the sensor´s calibration system.

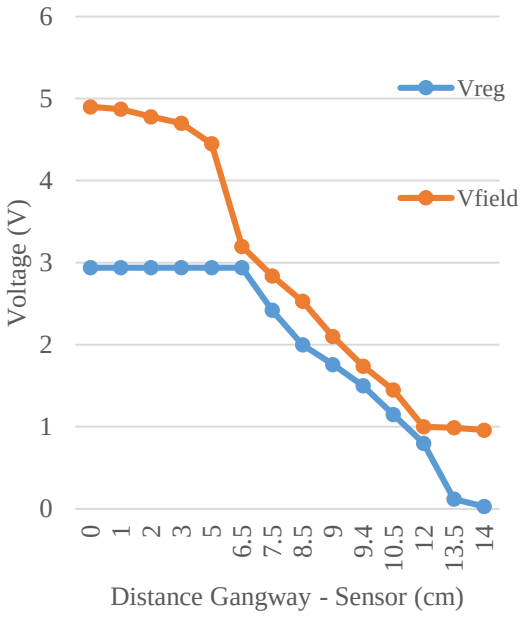


Figure 2 – Vfield -induced voltage in sensor system- and Vreg –Vfield regulated, voltage used to feed sensor system- vs. distance between gangway and sensor system

Considering that MONCEL must ensure a its operation in a range where the sensed value doesn't depend on the distance between its elements, we chose a range between [0…6.5cm] between the gangway and the sensor system because the results show that the voltage induced in this range doesn't vary with distance.

Within this range, the measurements obtained vary [36.5ºC … 36.65ºC], yielding an average of 36.57ºC just 0.03ºC under the temperature measured by the thermometer.

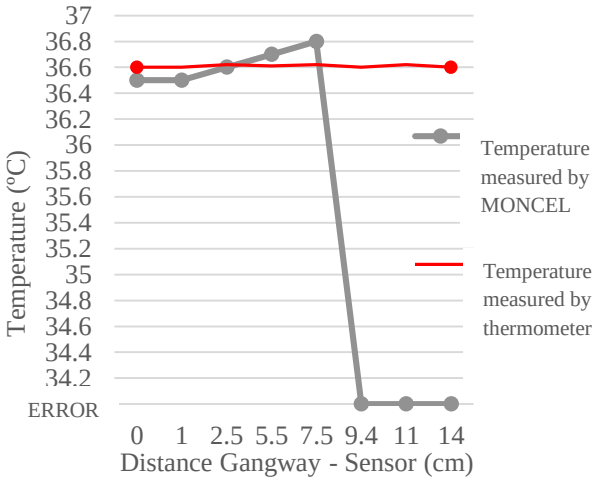


Figure 3 - Temperature measured by MONCEL and Temperature measured by thermometer vs. distance between gangway and sensor system

To ensure that the variation of the supply due to weak electromagnetic fields between [6.5cm…8.5cm] doesn't corrupts the sensor value, weak field detection can be implemented on the sensor tag. This method is based on an internal block and requires a simple calibration. By sensing the field at a distance of 6.5cm and verifying that this value is lower than the field sensed during the measurement, MONCEL can alert off screen on user´s mobile to bring closer the gangway to the senor tag. This can assure the data validity no matter the distance between the blocks.

IV. FUTURE PRODUCTION

MONCEL projects a redesign of the sensor´s antenna reaching size of 1cmx1cm as shown in Figure 2:

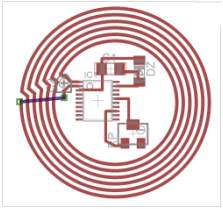


Figure 4 - PCB of sensor tag

Finally, the following chart projects an estimation of the cost of building 1 unit of MONCEL in growing scales.

Concept	1	10	100	1000
Develpment cost	U$S 24.500	U$S 2.450	U$S 245	U$S 25
Sensor system	Included	U$S 14	U$S 13	U$S 11
Gangway	Included	U$S 30	U$S 27	U$S 24
Reinforced hours	Included	U$S 3.000	U$S 500	U$S 100
TOTAL per unit	**U$S 24.500**	**U$S 5.500**	**U$S 785**	**U$S 160**

Table 2 - Costs projected in growing scales

V. CONCLUSIONS

The following results were successfully achieved:

- Maximum range between sensor and gangway: 6.5cm
- Maximum range between gangway and cellphone: 16.2m
- Accuracy ± 0.1ºC
- Stored data using SQLite by Android in mobile
- Alert dialog box and text messages to emergency contact when the value exceeds the predefined range by user
- Duration of battery in gangway of 3 days transmitting continuously every 5 minutes

MONCEL innovation is a continuous transfer of data from a passive sensor to a mobile phone application. MONCEL is a contribution to a better quality of life for the continuous monitoring of physiological variables such as temperature or glycaemia for people with diabetes.

VI. BIBILOGRAPHY

[1] *"American Diabetes Association®."* [Online]. Available: http://www.diabetes.org/.
See more at: Living With Diabetes > Treatment and Care > Blood Glucose Testing http://www.diabetes.org/living-with-diabetes/treatment-and-care/blood-glucose-control/?loc=lwd-slabnav#sthash.JVOLJ6aP.dpuf
See more at: Diabetes Basics > Statistics - http://www.diabetes.org/diabetes-basics/statistics/?loc=db-slabnav

[2] K. Finkenzeller, "*RFID Handbook: Fundamentals and applications in contactless smart cards, radio frequency identification and near field communication*", Third Edition, John Wiley and Sons Ltd Publication, Munich, Germany, pp. 350–480, 2010.

[3] *"Basics | Bluetooth Technology Website."* [Online]. Available: http://www.bluetooth.com
See more at: What is Bluetooth Technology > Technology Basics – http://www.bluetooth.com/Pages/Basics.aspx

[4] Melexis Microelectronics Systems, *"MLX90129 13.56MHZ sensor tag / Datalogger IC Datasheet"* Rev 9, Ieper, Belgium, pp. 1–60, 2012

[5] Texas Instruments Inc., *"TRF7960A Multi-Protocol fully integrated 13.56 MHZ RFID reader/writer IC Datasheet"* Rev 3, Dallas, Texas, pp. 1-70, 2013

Apéndice D – Documentación de Software Pasarela

La documentación del código del Software Pasarela fue generada utilizando el programa de generación de código DOXYGEN

MONCEL

1.0

Índice general

Capítulo 1

Índice de estructura de datos

1.1. Estructura de datos

Lista de estructuras con una breve descripción:

Capítulo 2

Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 3

Documentación de las estructuras de datos

3.1. Referencia de la Estructura buffer

`#include <uart.h>`

Campos de datos

- char datos [TAM]
- int indice

3.1.1. Descripción detallada

Define el formato de los buffers, tanto de transmisión como de recepción.

3.1.2. Documentación de los campos

3.1.2.1. char datos[TAM]

Arreglo de caracteres donde se almacenan los datos.

3.1.2.2. int indice

Índice para recorrer el buffer.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- uart.h

3.2. Referencia de la Estructura shell_command

`#include <shell_commands.h>`

Campos de datos

- char * nombre
- char * descripcion
- void(* funcion)()

3.2.1. Descripción detallada

Define el formato de los comandos.

3.2.2. Documentación de los campos

3.2.2.1. char* descripcion

String que describe la funcionalidad del comando.

3.2.2.2. void(* funcion)()

Puntero a la función que implementa el comando.

3.2.2.3. char* nombre

String para el nombre del comando.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- shell_commands.h

Capítulo 4

Documentación de archivos

4.1. Referencia del Archivo aux_functions.c

Archivo que contiene funciones auxiliares que se utilízan en los distintos módulos de la aplicación. Incluye funciones para comparar dos cadenas de forma insensible a mayúsculas y minúscular, y para convertir un entero en una cadena.

```
#include <string.h>
#include <ctype.h>
#include "aux_functions.h"
```

Funciones

- int stricmp (char *str1, char *str2)

  Compara la cadena apuntada por str1 con la cadena apuntada por str2 sin sensibilidad a mayúsculas.
- char * itoa (int integer)

  Convierte el entero integer en una cadena de caracteres.

4.1.1. Descripción detallada

Archivo que contiene funciones auxiliares que se utilízan en los distintos módulos de la aplicación. Incluye funciones para comparar dos cadenas de forma insensible a mayúsculas y minúscular, y para convertir un entero en una cadena.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.1.2. Documentación de las funciones

4.1.2.1. char * itoa (int *integer*)

Convierte el entero integer en una cadena de caracteres.

Parámetros

integer	Entero a convertir.

Devuelve

Retorna un puntero que apunta a la cadena convertida.

4.1.2.2. int stricmp (char * *str1*, char * *str2*)

Compara la cadena apuntada por str1 con la cadena apuntada por str2 sin sensibilidad a mayúsculas.

Parámetros

str1	Cadena a comparar.
str2	Cadena a comparar.

Devuelve

Retorna un número entero indicando la relación entre las dos cadenas. Cero si ambas cadenas son iguales, mayor que cero si la cadena apuntada por str1 es mayor que la apuntada por str2, y menor que cero en el caso contrario al anterior.

4.2. Referencia del Archivo aux_functions.h

Define la interfaz pública del módulo aux_functions.

Funciones

- int stricmp (char *str1, char *str2)

  Compara la cadena apuntada por str1 con la cadena apuntada por str2 sin sensibilidad a mayúsculas.
- char * itoa (int integer)

  Convierte el entero integer en una cadena de caracteres.

4.2.1. Descripción detallada

Define la interfaz pública del módulo aux_functions.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.2.2. Documentación de las funciones

4.2.2.1. char* itoa (int *integer*)

Convierte el entero integer en una cadena de caracteres.

Parámetros

integer	Entero a convertir.

Devuelve

Retorna un puntero que apunta a la cadena convertida.

4.2.2.2. int stricmp (char * *str1*, char * *str2*)

Compara la cadena apuntada por str1 con la cadena apuntada por str2 sin sensibilidad a mayúsculas.

Parámetros

str1	Cadena a comparar.
str2	Cadena a comparar.

Devuelve

Retorna un número entero indicando la relación entre las dos cadenas. Cero si ambas cadenas son iguales, mayor que cero si la cadena apuntada por str1 es mayor que la apuntada por str2, y menor que cero en el caso contrario al anterior.

4.3. Referencia del Archivo moncel.c

Módulo que contiene el programa principal del sistema. Desde su bucle infinito se van chequando las banderas e invocando según estas a las funciones pertinentes de los diferentes módulos.

```
#include "msp430f2370.h"
#include <stdlib.h>
#include <stdio.h>
#include "trf7960.h"
#include "types.h"
#include "timer.h"
#include "shell.h"
#include "uart.h"
#include "queue.h"
```

Funciones

- void inicializar (void)
- void main (void)

Variables

- char buf [300]
- char enable = 0
- char i_reg = 0x01
- char irq_flag = 0x00
- char rx_error_flag = 0x00
- s08_t rxtx_state = 1
- char error = 0x00
- char * respuesta
- char * respuestaFinal

- char uid [8] = {0xE0,0x1F,0xA0,0x84,0x14,0x56,0x27,0x43}
- int sensingTime

  Función que modifica la ventana de tiempo entre las mediciones. Se espera la unidad de tiempo venga dada como parámetro (s, m, h).
- int sendingTime

  Función que modifica la ventana de tiempo entre las notificaciones. Se espera la unidad de tiempo venga dada como parámetro (s, m, h).
- buffer RXbuffer

  Variable tipo buffer para la recepción.

4.3.1. Descripción detallada

Módulo que contiene el programa principal del sistema. Desde su bucle infinito se van chequando las banderas e invocando según estas a las funciones pertinentes de los diferentes módulos.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.3.2. Documentación de las funciones

4.3.2.1. void inicializar (void)

4.3.2.2. void main (void)

4.3.3. Documentación de las variables

4.3.3.1. char buf[300]

4.3.3.2. char enable = 0

4.3.3.3. char error = 0x00

4.3.3.4. char i_reg = 0x01

4.3.3.5. char irq_flag = 0x00

4.3.3.6. char* respuesta

4.3.3.7. char* respuestaFinal

4.3.3.8. char rx_error_flag = 0x00

4.3.3.9. RXbuffer

Variable tipo buffer para la recepción.

4.3.3.10. s08_t rxtx_state = 1

4.3.3.11. sendingTime

Función que modifica la ventana de tiempo entre las notificaciones. Se espera la unidad de tiempo venga dada como parámetro (s, m, h).

Variable tipo int que especifica el intervalo, en segundos, entre envíos de los datos al celular. Por defecto, se inicializa en 1 minuto.

Parámetros

argc	Cantidad de parámetros.
argv	Parámetros del comando.

4.3.3.12. sensingTime

Función que modifica la ventana de tiempo entre las mediciones. Se espera la unidad de tiempo venga dada como parámetro (s, m, h).

Variable tipo int que especifica el intervalo, en segundos, entre notificaciones. Por defecto, se inicializa en 1 minuto.

Parámetros

argc	Cantidad de parámetros.
argv	Parámetros del comando.

4.3.3.13. char uid[8] = {0xE0,0x1F,0xA0,0x84,0x14,0x56,0x27,0x43}

4.4. Referencia del Archivo msp430f2370.c

Archivo que contiene funciones particulares del micro MSP430F2370.

```
#include "trf7960.h"
#include "types.h"
#include "msp430f2370.h"
```

Funciones

- void delay (u32_t n_ms)

  Función usada para crear delays. 23/11/2010 RP Texas Instruments.
- void Msp430f23x0OscSel (char mode)

  Función usada para elegir el modo del oscilador. 23/11/2010 RP Texas Instruments Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega.

4.4.1. Descripción detallada

Archivo que contiene funciones particulares del micro MSP430F2370.

Autor

Texas Instruments Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.4.2. Documentación de las funciones

4.4.2.1. void delay (u32_t *n_ms*)

Función usada para crear delays. 23/11/2010 RP Texas Instruments.

Parámetros

n_ms	Long sin signo que indica la cantidad de milisegundos a retardar.

4.4.2.2. void Msp430f23x0OscSel (char *mode*)

Función usada para elegir el modo del oscilador. 23/11/2010 RP Texas Instruments Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega.

Parámetros

mode	Char con el modo a configurar, a saber, DCO u oscilador de cristal.

4.5. Referencia del Archivo msp430f2370.h

Archivo que contiene las declaraciones particulares al micro MSP430F2370.

```
#include <MSP430x23x0.h>
#include "types.h"
```

'defines'

- #define ENABLE_INTERRUPTS _EINT()
- #define TRF_WRITE P4OUT
- #define TRF_READ P4IN
- #define TRF_DIR_IN P4DIR = 0x00
- #define TRF_DIR_OUT P4DIR = 0xFF
- #define TRF_FUNC P4SEL = 0x00
- #define ENABLE_SET P1DIR |= BIT0
- #define TRF_ENABLE P1OUT |= BIT0
- #define TRF_DISABLE P1OUT &= ~BIT0
- #define CLK_P_OUT_SET P3DIR |= BIT3
- #define CLK_ON P3OUT |= BIT3
- #define CLK_OFF P3OUT &= ~BIT3
- #define MOD_SET P2DIR |= BIT0;
- #define MOD_ON P2OUT |= BIT0
- #define MOD_OFF P2OUT &= ~BIT0;
- #define IRQ_PIN_SET P2DIR &= ~BIT1;
- #define IRQ_PIN BIT1
- #define IRQ_PORT P2IN
- #define IRQ_ON P2IE |= BIT1
- #define IRQ_OFF P2IE &= ~BIT1
- #define IRQ_EDGE_SET P2IES &= ~BIT1

- #define IRQ_CLR P2IFG = 0x00
- #define IRQ_REQ_REG P2IFG
- #define LED_PORT_SET P1DIR |= 0xFC;
- #define LED_ALL_OFF P1OUT &= ∼0xFC;
- #define LED_ALL_ON P1OUT |= 0xFC;
- #define LED_POWER_ON P1OUT |= BIT7;
- #define LED_POWER_OFF P1OUT &= ∼BIT7;
- #define LED_14443A_ON P1OUT |= BIT6;
- #define LED_14443A_OFF P1OUT &= ∼BIT6;
- #define LED_14443B_ON P1OUT |= BIT5;
- #define LED_14443B_OFF P1OUT &= ∼BIT5;
- #define LED_15693_ON P1OUT |= BIT4;
- #define LED_15693_OFF P1OUT &= ∼BIT4;
- #define LED_OPEN1_ON P1OUT |= BIT3;
- #define LED_OPEN1_OFF P1OUT &= ∼BIT3;
- #define LED_OPEN2_ON P1OUT |= BIT2;
- #define LED_OPEN2_OFF P1OUT &= ∼BIT2;
- #define SLAVE_SELECT_PORT_SET P3DIR |= BIT0;
- #define SLAVE_SELECT_HIGH P3OUT |= BIT0;
- #define SLAVE_SELECT_LOW P3OUT &= ∼BIT0;
- #define OOK_DIR_IN P2DIR &= ∼BIT2;
- #define OOK_DIR_OUT P2DIR |= BIT2
- #define OOK_OFF P2OUT &= ∼BIT2
- #define OOK_ON P2OUT |= BIT2
- #define COUNT_VALUE TACCR0
- #define START_COUNTER_A TACTL |= MC1
- #define STOP_COUNTER_A TACTL &= ∼(MC0 + MC1)
- #define START_COUNTER_B TBCTL |= MC_2
- #define STOP_COUNTER_B TBCTL &= ∼(MC0 + MC1)
- #define COUNT_1ms 0x069F
- #define DELAY_1ms 6780

Funciones

- void delay (u32_t n_ms)

  Función usada para crear delays. 23/11/2010 RP Texas Instruments.
- void Msp430f23x0OscSel (char mode)

  Función usada para elegir el modo del oscilador. 23/11/2010 RP Texas Instruments Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega.

4.5.1. Descripción detallada

Archivo que contiene las declaraciones particulares al micro MSP430F2370.

Autor

Texas Instruments Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.5.2. Documentación de los 'defines'

4.5.2.1. #define CLK_OFF P3OUT &= ∼BIT3

4.5.2.2. #define CLK_ON P3OUT |= BIT3

4.5.2.3. #define CLK_P_OUT_SET P3DIR |= BIT3

4.5.2.4. #define COUNT_1ms 0x069F

4.5.2.5. #define COUNT_VALUE TACCR0

4.5.2.6. #define DELAY_1ms 6780

4.5.2.7. #define ENABLE_INTERRUPTS _EINT()

4.5.2.8. #define ENABLE_SET P1DIR |= BIT0

4.5.2.9. #define IRQ_CLR P2IFG = 0x00

4.5.2.10. #define IRQ_EDGE_SET P2IES &= ∼BIT1

4.5.2.11. #define IRQ_OFF P2IE &= ∼BIT1

4.5.2.12. #define IRQ_ON P2IE |= BIT1

4.5.2.13. #define IRQ_PIN BIT1

4.5.2.14. #define IRQ_PIN_SET P2DIR &= ∼BIT1;

4.5.2.15. #define IRQ_PORT P2IN

4.5.2.16. #define IRQ_REQ_REG P2IFG

4.5.2.17. #define LED_14443A_OFF P1OUT &= ∼BIT6;

4.5.2.18. #define LED_14443A_ON P1OUT |= BIT6;

4.5.2.19. #define LED_14443B_OFF P1OUT &= ∼BIT5;

4.5.2.20. #define LED_14443B_ON P1OUT |= BIT5;

4.5.2.21. #define LED_15693_OFF P1OUT &= ∼BIT4;

4.5.2.22. #define LED_15693_ON P1OUT |= BIT4;

4.5.2.23. #define LED_ALL_OFF P1OUT &= ∼0xFC;

4.5.2.24. #define LED_ALL_ON P1OUT |= 0xFC;

4.5.2.25. #define LED_OPEN1_OFF P1OUT &= ∼BIT3;

4.5.2.26. #define LED_OPEN1_ON P1OUT |= BIT3;

4.5.2.27. #define LED_OPEN2_OFF P1OUT &= ∼BIT2;

4.5.2.28. #define LED_OPEN2_ON P1OUT |= BIT2;

4.5.2.29. #define LED_PORT_SET P1DIR |= 0xFC;

4.5.2.30. #define LED_POWER_OFF P1OUT &= ∼BIT7;

4.5.2.31. #define LED_POWER_ON P1OUT |= BIT7;

4.5.2.32. #define MOD_OFF P2OUT &= ∼BIT0;

4.5.2.33. #define MOD_ON P2OUT |= BIT0

4.5.2.34. #define MOD_SET P2DIR |= BIT0;

4.5.2.35. #define OOK_DIR_IN P2DIR &= ∼BIT2;

4.5.2.36. #define OOK_DIR_OUT P2DIR |= BIT2

4.5.2.37. #define OOK_OFF P2OUT &= ∼BIT2

4.5.2.38. #define OOK_ON P2OUT |= BIT2

4.5.2.39. #define SLAVE_SELECT_HIGH P3OUT |= BIT0;

4.5.2.40. #define SLAVE_SELECT_LOW P3OUT &= ∼BIT0;

4.5.2.41. #define SLAVE_SELECT_PORT_SET P3DIR |= BIT0;

4.5.2.42. #define START_COUNTER_A TACTL |= MC1

4.5.2.43. #define START_COUNTER_B TBCTL |= MC_2

4.5.2.44. #define STOP_COUNTER_A TACTL &= ∼(MC0 + MC1)

4.5.2.45. #define STOP_COUNTER_B TBCTL &= ∼(MC0 + MC1)

4.5.2.46. #define TRF_DIR_IN P4DIR = 0x00

4.5.2.47. #define TRF_DIR_OUT P4DIR = 0xFF

4.5.2.48. #define TRF_DISABLE P1OUT &= ∼BIT0

4.5.2.49. #define TRF_ENABLE P1OUT |= BIT0

4.5.2.50. #define TRF_FUNC P4SEL = 0x00

4.5.2.51. #define TRF_READ P4IN

4.5.2.52. #define TRF_WRITE P4OUT

4.5.3. Documentación de las funciones

4.5.3.1. void delay (u32_t *n_ms*)

Función usada para crear delays. 23/11/2010 RP Texas Instruments.

Parámetros

n_ms	Long sin signo que indica la cantidad de milisegundos a retardar.

4.5.3.2. void Msp430f23x0OscSel (char *mode*)

Función usada para elegir el modo del oscilador. 23/11/2010 RP Texas Instruments Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega.

Parámetros

mode	Char con el modo a configurar, a saber, DCO u oscilador de cristal.

4.6. Referencia del Archivo queue.c

Archivo que contiene las funciones que manejan la cola de datos.

```
#include <stdio.h>
#include "uart.h"
#include "queue.h"
```

Funciones

- void initializeQueue (void)

  Función que inicializa la cola de datos poniendo 0xFF en todo el array.
- void resetQueue (void)

  Función que resetea la cola de datos. Setea el Head, Tail y la bandera fullQueue en 0.
- int getQueueHead (void)

  Función que devuelve la posición de la Cabecera.
- int getQueueTail (void)

  Función que devuelve la posición del último valor de la cola de datos.
- void addDataToQueue (char ∗newData)

  Función que agrega un dato a la cola.
- void sendQueue (void)

  Función que envía a través de la UART todos los datos almacenados en la cola de datos.

4.6.1. Descripción detallada

Archivo que contiene las funciones que manejan la cola de datos.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.6.2. Documentación de las funciones

4.6.2.1. addDataToQueue (char ∗ *newData*)

Función que agrega un dato a la cola.

Parámetros

newData	puntero al nuevo dato a añadir.

4.6.2.2. getQueueHead (void)

Función que devuelve la posición de la Cabecera.

Devuelve

Retorna el valor de la Cabecera.

4.6.2.3. getQueueTail (void)

Función que devuelve la posición del último valor de la cola de datos.

Devuelve

Retorna el valor de la posición del último valor de la cola de datos.

4.6.2.4. initializeQueue (void)

Función que inicializa la cola de datos poniendo 0xFF en todo el array.

4.6.2.5. void resetQueue (void)

Función que resetea la cola de datos. Setea el Head, Tail y la bandera fullQueue en 0.

4.6.2.6. sendQueue (void)

Función que envía a través de la UART todos los datos almacenados en la cola de datos.

4.7. Referencia del Archivo queue.h

Archivo que define la interfaz pública de la cola de datos.

'defines'

- #define MAX_QUEUE 300

  Constante que define el tamaño de la cola de datos.

Funciones

- void initializeQueue (void)

  Función que inicializa la cola de datos poniendo 0xFF en todo el array.
- void resetQueue (void)

  Función que resetea la cola de datos. Setea el Head, Tail y la bandera fullQueue en 0.
- int getQueueHead (void)

  Función que devuelve la posición de la Cabecera.

- int getQueueTail (void)

  Función que devuelve la posición del último valor de la cola de datos.
- void addDataToQueue (char ∗newData)

  Función que agrega un dato a la cola.
- void sendQueue (void)

  Función que envía a través de la UART todos los datos almacenados en la cola de datos.

4.7.1. Descripción detallada

Archivo que define la interfaz pública de la cola de datos.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.7.2. Documentación de los 'defines'

4.7.2.1. #define MAX_QUEUE 300

Constante que define el tamaño de la cola de datos.

Definición de constantes privadas MAX_QUEUE 300

4.7.3. Documentación de las funciones

4.7.3.1. void addDataToQueue (char ∗ *newData*)

Función que agrega un dato a la cola.

Parámetros

newData	puntero al nuevo dato a añadir.

4.7.3.2. int getQueueHead (void)

Función que devuelve la posición de la Cabecera.

Devuelve

Retorna el valor de la Cabecera.

4.7.3.3. int getQueueTail (void)

Función que devuelve la posición del último valor de la cola de datos.

Devuelve

Retorna el valor de la posición del último valor de la cola de datos.

4.7.3.4. void initializeQueue (void)

Función que inicializa la cola de datos poniendo 0xFF en todo el array.

4.7.3.5. void resetQueue (void)

Función que resetea la cola de datos. Setea el Head, Tail y la bandera fullQueue en 0.

4.7.3.6. void sendQueue (void)

Función que envía a través de la UART todos los datos almacenados en la cola de datos.

4.8. Referencia del Archivo shell.c

Intérprete de comandos. Recibe una cadena de carecteres que especifica un comando. El comando está especificado por el nombre y además permite recibir parámetros separados por espacios. Al recibir un string lo procesa para obtener los parámetros y el nombre de comando. Luego llama a la función correspondiente.

```
#include <string.h>
#include <ctype.h>
#include "shell_commands.h"
#include "shell.h"
#include "aux_functions.h"
#include "uart.h"
```

'defines'

- #define MAX_PAR 3

  Constante que define la máxima cantidad posible de parámetros esperados para un comando.

Funciones

- void shell_exec (char *str)

  Función que procesa el string recibido para obtener el comando y sus parámetros. Luego, busca y ejecuta el comando pasado. De no encontrarse el comando devuelve un mensaje de error.

Variables

- char * command_par [MAX_PAR]

  Variable tipo char donde se guardarán el comando, y los parámetros pasados al mismo.*

- unsigned int number_par

  Variable tipo int que indica la cantidad de argumentos que se pasan en la cadena de caracteres.

- shell_command commands []

  Variable tipo shell_command donde están listados todos los comandos disponibles. El arreglo debe terminar en {0,0,0}.

4.8.1. Descripción detallada

Intérprete de comandos. Recibe una cadena de carecteres que especifica un comando. El comando está especificado por el nombre y además permite recibir parámetros separados por espacios. Al recibir un string lo procesa para obtener los parámetros y el nombre de comando. Luego llama a la función correspondiente.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.8.2. Documentación de los 'defines'

4.8.2.1. #define MAX_PAR 3

Constante que define la máxima cantidad posible de parámetros esperados para un comando.

Definición de constantes privadas MAX_PAR 3

4.8.3. Documentación de las funciones

4.8.3.1. void shell_exec (char * *str*)

Función que procesa el string recibido para obtener el comando y sus parámetros. Luego, busca y ejecuta el comando pasado. De no encontrarse el comando devuelve un mensaje de error.

Parámetros

str	Cadena con el comando a buscar.

4.8.4. Documentación de las variables

4.8.4.1. command_par[MAX_PAR]

Variable tipo char* donde se guardarán el comando, y los parámetros pasados al mismo.

Declaración de variables privadas

4.8.4.2. commands[ ]

Variable tipo shell_command donde están listados todos los comandos disponibles. El arreglo debe terminar en {0,0,0}.

4.8.4.3. number_par

Variable tipo int que indica la cantidad de argumentos que se pasan en la cadena de caracteres.

4.9. Referencia del Archivo shell.h

Define la interfaz pública del módulo shell.

Funciones

- void shell_exec (char *str)

  Función que procesa el string recibido para obtener el comando y sus parámetros. Luego, busca y ejecuta el comando pasado. De no encontrarse el comando devuelve un mensaje de error.

4.9.1. Descripción detallada

Define la interfaz pública del módulo shell.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.9.2. Documentación de las funciones

4.9.2.1. void shell_exec (char * *str*)

Función que procesa el string recibido para obtener el comando y sus parámetros. Luego, busca y ejecuta el comando pasado. De no encontrarse el comando devuelve un mensaje de error.

Parámetros

str	Cadena con el comando a buscar.

4.10. Referencia del Archivo shell_commands.c

Módulo donde se definen los comandos y la lista de los mismos con el puntero que apunta a la función y su descripción.

```
#include <string.h>
#include "aux_functions.h"
#include "shell_commands.h"
#include "timer.h"
#include "shell.h"
#include "uart.h"
```

Funciones

- void sensingTime (unsigned int argc, char **argv)
- void sendingTime (unsigned int argc, char **argv)
- void showConfig (unsigned int argc, char **argv)

  Función que envía al celular los parámetros configurados.

Variables

- char alarm_reply [250]

  Variable tipo arreglo de caracteres usada para guardar el mensaje a enviar al usuario.

- shell_command commands [ ]

  Variable tipo shell_command donde están listados todos los comandos disponibles. El arreglo debe terminar en {0,0,0}.

4.10.1. Descripción detallada

Módulo donde se definen los comandos y la lista de los mismos con el puntero que apunta a la función y su descripción.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.10.2. Documentación de las funciones

4.10.2.1. void sendingTime (unsigned int *argc,* char ** *argv*)

4.10.2.2. void sensingTime (unsigned int *argc,* char ** *argv*)

4.10.2.3. showConfig (unsigned int *argc,* char ** *argv*)

Función que envía al celular los parámetros configurados.

Parámetros

argc	Cantidad de parámetros.
argv	Parámetros del comando.

4.10.3. Documentación de las variables

4.10.3.1. alarm_reply[250]

Variable tipo arreglo de caracteres usada para guardar el mensaje a enviar al usuario.

4.10.3.2. shell_command commands[ ]

Valor inicial:

```
=
      {
    {"SENSAR","Configura tiempo entre mediciones.",sensingTime},
        {"ENVIAR","Configura tiempo entre envíos al celular.",sendingTime},
        {"CONFIG", "Muestra configuracion actual.", showConfig},

    {0,0,0}
      }
```

Variable tipo shell_command donde están listados todos los comandos disponibles. El arreglo debe terminar en {0,0,0}.

4.11. Referencia del Archivo shell_commands.h

Define la interfaz pública del módulo shell_commands. Se definen estructuras para listar los comandos y los usuarios.

Estructuras de datos

- struct shell_command

Funciones

- void sensingTime (unsigned int argc, char **argv)
- void sendingTime (unsigned int argc, char **argv)
- void showConfig (unsigned int argc, char **argv)

  Función que envía al celular los parámetros configurados.

4.11.1. Descripción detallada

Define la interfaz pública del módulo shell_commands. Se definen estructuras para listar los comandos y los usuarios.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.11.2. Documentación de las funciones

4.11.2.1. void sendingTime (unsigned int *argc,* char ** *argv*)

4.11.2.2. void sensingTime (unsigned int *argc,* char ** *argv*)

4.11.2.3. void showConfig (unsigned int *argc,* char ** *argv*)

Función que envía al celular los parámetros configurados.

Parámetros

argc	Cantidad de parámetros.
argv	Parámetros del comando.

4.12. Referencia del Archivo timer.c

Módulo encargado del control de tiempos del sistema. El mismo se usa tanto para medir y setear el tiempo entre las distintas notificaciones, como también para implementar los time out en caso de que el sistema tenga algún problema imprevisto.

```
#include "timer.h"
#include <stdlib.h>
#include "uart.h"
#include "trf7960.h"
#include "msp430f2370.h"
```

Funciones

- void setsensingTime (char *time, char *unit)

  Función pública que setea el intervalo entre mediciones.
- int getsensingTime ()

  Función pública que devuelve el intervalo entre mediciones.
- void setsendingTime (char *time, char *unit)

  Función pública que setea el intervalo entre envíos de los datos hacia el celular.
- int getSendingTime ()

  Función pública que devuelve el intervalo configurado para el envío de los datos hacia el celular.
- int getFlagSensar ()

  Función pública que devuelve la bandera que indica tiempo entre mediciones expirado.
- int getFlagEnviar ()

  Función pública que devuelve la bandera que indica tiempo entre envíos al celular expirado.
- void resetFlagSensar ()

  Función que baja la bandera de tiempo entre mediciones.
- void resetFlagEnviar ()

  Función que baja la bandera de tiempo entre envíos al celular.
- void configTimerA (void)

  Función que se encarga de configurar el TIMER A para usar con el cristal externo de 13,56MHz. Se usa para todo lo referente a las transacciones por RF con el Tag.
- void configTimerB (void)

  Función que se encarga de configurar el TIMER B. El mismo se usa para medir los tiempos tanto entre mediciones como también entre notificaciones.
- void controlCounter ()

  Funcion que controla los tiempos transcurridos tanto entre mediciones como entre envíos de datos al celular. Si alguno expiró, resetea dicho contador y levanta la bandera correspondiente.
- void controlTimeout ()

  Funcion que implementa el timeout del envío de mensajes.
- __interrupt void Msp430f23x0TimerAHandler (void)

  ISR del Timer de time outs (TA): maneja los time outs para el envío/recepción por RF.
- __interrupt void timerB (void)

  ISR del Timer B (TB) que controla los tiempos tanto entre mediciones como de envío de los datos hacia el celular.

Variables

- int cuentaPeriodos = 0
- int cuentaPeriodos1 = 0
- char irq_flag
- char i_reg

4.12.1. Descripción detallada

Módulo encargado del control de tiempos del sistema. El mismo se usa tanto para medir y setear el tiempo entre las distintas notificaciones, como también para implementar los time out en caso de que el sistema tenga algún problema imprevisto.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.12.2. Documentación de las funciones

4.12.2.1. void configTimerA (void)

Función que se encarga de configurar el TIMER A para usar con el cristal externo de 13,56MHz. Se usa para todo lo referente a las transacciones por RF con el Tag.

4.12.2.2. void configTimerB (void)

Función que se encarga de configurar el TIMER B. El mismo se usa para medir los tiempos tanto entre mediciones como también entre notificaciones.

4.12.2.3. void controlCounter ( )

Funcion que controla los tiempos transcurridos tanto entre mediciones como entre envíos de datos al celular. Si alguno expiró, resetea dicho contador y levanta la bandera correspondiente.

4.12.2.4. void controlTimeout ( )

Funcion que implementa el timeout del envío de mensajes.

4.12.2.5. int getFlagEnviar ( )

Función pública que devuelve la bandera que indica tiempo entre envíos al celular expirado.

Devuelve

Valor de la bandera.

4.12.2.6. int getFlagSensar ( )

Función pública que devuelve la bandera que indica tiempo entre mediciones expirado.

Devuelve

Valor de la bandera.

4.12.2.7. int getSendingTime ( )

Función pública que devuelve el intervalo configurado para el envío de los datos hacia el celular.

Devuelve

Devuelve el tiempo especificado en minutos.

4.12.2.8. int getsensingTime ( )

Función pública que devuelve el intervalo entre mediciones.

Devuelve

Devuelve el tiempo especificado en minutos.

4.12.2.9. __interrupt void Msp430f23x0TimerAHandler (void)

ISR del Timer de time outs (TA): maneja los time outs para el envío/recepción por RF.

4.12.2.10. void resetFlagEnviar ( )

Función que baja la bandera de tiempo entre envíos al celular.

4.12.2.11. void resetFlagSensar ( )

Función que baja la bandera de tiempo entre mediciones.

4.12.2.12. void setsendingTime (char * *time*, char * *unit*)

Función pública que setea el intervalo entre envíos de los datos hacia el celular.

Parámetros

time	Cadena que especifica el intervalo a setear.
unit	Cadena que especifica la unidad en que está el tiempo dado por el otro parámetro

4.12.2.13. void setsensingTime (char * *time*, char * *unit*)

Función pública que setea el intervalo entre mediciones.

Parámetros

time	Cadena que especifica el intervalo a setear.
unit	Cadena que especifica la unidad en que está el tiempo dado por el otro parámetro

4.12.2.14. __interrupt void timerB (void)

ISR del Timer B (TB) que controla los tiempos tanto entre mediciones como de envío de los datos hacia el celular.

4.12.3. Documentación de las variables

4.12.3.1. int cuentaPeriodos = 0

4.12.3.2. int cuentaPeriodos1 = 0

4.12.3.3. char i_reg

4.12.3.4. char irq_flag

4.13. Referencia del Archivo timer.h

Archivo que define la interfaz pública del módulo timer.

'defines'

- #define MINUTE_TO_SECOND 60

Constante que define la cantidad de segundos en un minuto.

- #define HOUR_TO_SECOND 3600

  Constante que define la cantidad de segundos en una hora.

Funciones

- void configTimerA (void)

  Función que se encarga de configurar el TIMER A para usar con el cristal externo de 13,56MHz. Se usa para todo lo referente a las transacciones por RF con el Tag.
- void configTimerB (void)

  Función que se encarga de configurar el TIMER B. El mismo se usa para medir los tiempos tanto entre mediciones como también entre notificaciones.
- void setsensingTime (char ∗time, char ∗unit)

  Función pública que setea el intervalo entre mediciones.
- int getsensingTime ()

  Función pública que devuelve el intervalo entre mediciones.
- void setsendingTime (char ∗time, char ∗unit)

  Función pública que setea el intervalo entre envíos de los datos hacia el celular.
- int getsendingTime ()
- int getFlagSensar ()

  Función pública que devuelve la bandera que indica tiempo entre mediciones expirado.
- int getFlagEnviar ()

  Función pública que devuelve la bandera que indica tiempo entre envíos al celular expirado.
- void resetFlagSensar ()

  Función que baja la bandera de tiempo entre mediciones.
- void resetFlagEnviar ()

  Función que baja la bandera de tiempo entre envíos al celular.
- void controlCounter ()

  Funcion que controla los tiempos transcurridos tanto entre mediciones como entre envíos de datos al celular. Si alguno expiró, resetea dicho contador y levanta la bandera correspondiente.
- void controlTimeout ()

  Funcion que implementa el timeout del envío de mensajes.

4.13.1. Descripción detallada

Archivo que define la interfaz pública del módulo timer.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.13.2. Documentación de los 'defines'

4.13.2.1. #define HOUR_TO_SECOND 3600

Constante que define la cantidad de segundos en una hora.

HOUR_TO_SECOND 3600

4.13.2.2. #define MINUTE_TO_SECOND 60

Constante que define la cantidad de segundos en un minuto.

Definición de constantes privadas MINUTE_TO_SECOND 60

4.13.3. Documentación de las funciones

4.13.3.1. void configTimerA (void)

Función que se encarga de configurar el TIMER A para usar con el cristal externo de 13,56MHz. Se usa para todo lo referente a las transacciones por RF con el Tag.

4.13.3.2. void configTimerB (void)

Función que se encarga de configurar el TIMER B. El mismo se usa para medir los tiempos tanto entre mediciones como también entre notificaciones.

4.13.3.3. void controlCounter ( )

Funcion que controla los tiempos transcurridos tanto entre mediciones como entre envíos de datos al celular. Si alguno expiró, resetea dicho contador y levanta la bandera correspondiente.

4.13.3.4. void controlTimeout ( )

Funcion que implementa el timeout del envío de mensajes.

4.13.3.5. int getFlagEnviar ( )

Función pública que devuelve la bandera que indica tiempo entre envíos al celular expirado.

Devuelve

Valor de la bandera.

4.13.3.6. int getFlagSensar ( )

Función pública que devuelve la bandera que indica tiempo entre mediciones expirado.

Devuelve

Valor de la bandera.

4.13.3.7. int getsendingTime ( )

4.13.3.8. int getsensingTime ( )

Función pública que devuelve el intervalo entre mediciones.

Devuelve

Devuelve el tiempo especificado en minutos.

4.13.3.9. void resetFlagEnviar ( )

Función que baja la bandera de tiempo entre envíos al celular.

4.13.3.10. void resetFlagSensar ( )

Función que baja la bandera de tiempo entre mediciones.

4.13.3.11. void setsendingTime (char * *time*, char * *unit*)

Función pública que setea el intervalo entre envíos de los datos hacia el celular.

Parámetros

time	Cadena que especifica el intervalo a setear.
unit	Cadena que especifica la unidad en que está el tiempo dado por el otro parámetro

4.13.3.12. void setsensingTime (char * *time*, char * *unit*)

Función pública que setea el intervalo entre mediciones.

Parámetros

time	Cadena que especifica el intervalo a setear.
unit	Cadena que especifica la unidad en que está el tiempo dado por el otro parámetro

4.14. Referencia del Archivo trf7960.c

Archivo que contiene las funciones que manejan el lector TRF7960A.

```
#include "trf7960.h"
#include "types.h"
#include "msp430f2370.h"
#include "queue.h"
#include "uart.h"
#include "timer.h"
```

Funciones

- void communicationSetup (void)

  Función que configura la comunicación paralelo entre el MSP y el TRF7960.
- void directCommand (char *pbuf)

  Función que envía el comando pasado como parámetro directamente al TRF7960.
- void initialSettings (void)

  Función que inicializa el TRF7960 poniendo el reloj en 6.78MHz.
- void ISR (char *irq_status)

  Función que maneja las interrupciones del TRF7960. Determina el motivo de la interrupción leyendo el IRQ Status Register y realiza las acciones requeridas.
- __interrupt void portB (void)
- void rawWrite (char *pbuf, char length)

  Función que escribe directamente en el FIFO del TRF7960.
- void readSingle (char *pbuf, char number)

Función que lee los registros pasados como parámetro del TRF7960. Escribe el resultado en el propio string pasado.

- void readCont (char ∗pbuf, char length)

  Función que lee la cantidad dada de registros del TRF7960 a partir de la dirección pasada en ∗pbuf. Escribe el resultado en el propio string pasado.
- void readIrqStatus (char ∗pbuf)

  Función que lee el registro IRQ STATUS.
- void writeSingle (char ∗pbuf, char length)

  Función que escribe los registros pasados como parámetros.
- void writeCont (char ∗pbuf, char length)

  Función que escribe los registros pasados como parámetros. Al igual que su análoga readCont, escribe a partir del registro pasado como parámetro y a las direcciones consecutivas.
- void writeIsoControl (char iso_control)

  Función que escribe en el registro ISO CONTROL quien determina el protocolo a usar y sus parámetros (e.g. high data rate, ISO 15693, etc.)
- void reset (void)

  Función que envía al TRF7960 el comando de RESET.
- void resetIrqStatus (void)

  Función que borra el registro IRQ STATUS.
- void turnRfOn (void)

  Función que enciende la parte de RF del TRF7960.
- void stopDecoders (void)

  Función que pone la parte digital del receptor de RF en modo reset para evitar interrupciones en ambientes ruidosos.
- void runDecoders (void)

  Función que habilita nuevamente la parte digital del receptor de RF.
- void startCondition (void)

  Función que genera la condición de inicio de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.
- void stopCondition (void)

  Función que genera la condición de finalización de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.
- void stopContinuous (void)
- void cargarUID (char ∗pbuffer)

  Función que carga el identificador específico del TAG a usar.
- void consultarTag (char ∗pbuf, char length)

  Función que se encarga de interrogar al TAG enviándole el comando pasado como parámetro. Espera por la respuesta del TAG o indica un error si hubo time out.
- void sensar (char sensor)

  Función que se encarga de interrogar al TAG para saber el valor del sensor especificado en el parámetro. Maneja las respuestas intermedias recibidas del TAG. Se encarga también de despertar al TRF7960 antes de iniciar la interrogación, así también como de ponerlo en modo de bajo consumo mientras espera y luego de finalizada la consulta. Espera por la respuesta del TAG o indica un error ya sea si hubo time out o si el valor obtenido no es correcto por encontrarse demasiado apartados tag y lector.
- void enterMode1_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 1 de bajo consumo (5mA típico) En el mismo, tanto TX como RX de RF están apagadas.
- void enterMode2_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 2 de bajo consumo (10.5mA típico) En el mismo, TX está apagada, pero la RX por RF está activa.
- void enterMode3_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 3 de consumo (70mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, pero se transmite con media potencia.
- void enterMode4_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 4 de consumo (130mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, y se transmite a máxima potencia.

Variables

- char command [2]
- char direct_mode = 0
- char buf [300]
- char i_reg
- char irq_flag
- char rx_error_flag
- s08_t rxtx_state
- char stand_alone_flag
- char error
- char uid [8]

4.14.1. Descripción detallada

Archivo que contiene las funciones que manejan el lector TRF7960A.

Autor

Texas Instruments - Peter Reiser 2/12/2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.14.2. Documentación de las funciones

4.14.2.1. void cargarUID (char * *pbuffer*)

Función que carga el identificador específico del TAG a usar.

Parámetros

pbuffer	cadena con el identificador a cargar.

4.14.2.2. communicationSetup (void)

Función que configura la comunicación paralelo entre el MSP y el TRF7960.

4.14.2.3. void consultarTag (char * *pbuf*, char *length*)

Función que se encarga de interrogar al TAG enviándole el comando pasado como parámetro. Espera por la respuesta del TAG o indica un error si hubo time out.

Parámetros

*pbuf	string con el comando y sus parámetros a enviar al TAG
length	cantidad de datos a enviar al TAG

Autor

RP - Texas Instruments - 2/12/2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.14.2.4. directCommand (char * *pbuf*)

Función que envía el comando pasado como parámetro directamente al TRF7960.

Parámetros

∗*pbuf*	Puntero a la cadena que contiene el comando.

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.14.2.5. void enterMode1_5V (void)

Función que se encarga de poner el TRF7960 en el modo 1 de bajo consumo (5mA típico) En el mismo, tanto TX como RX de RF están apagadas.

4.14.2.6. void enterMode2_5V (void)

Función que se encarga de poner el TRF7960 en el modo 2 de bajo consumo (10.5mA típico) En el mismo, TX está apagada, pero la RX por RF está activa.

4.14.2.7. void enterMode3_5V (void)

Función que se encarga de poner el TRF7960 en el modo 3 de consumo (70mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, pero se transmite con media potencia.

4.14.2.8. void enterMode4_5V (void)

Función que se encarga de poner el TRF7960 en el modo 4 de consumo (130mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, y se transmite a máxima potencia.

4.14.2.9. initialSettings (void)

Función que inicializa el TRF7960 poniendo el reloj en 6.78MHz.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.10. ISR (char ∗ *irq_status*)

Función que maneja las interrupciones del TRF7960. Determina el motivo de la interrupción leyendo el IRQ Status Register y realiza las acciones requeridas.

Parámetros

irq_status	string con el valor del ISR Status Register

Autor

RP - Texas Instruments

4.14.2.11. __interrupt void portB (void)

4.14.2.12. rawWrite (char * *pbuf,* char *length*)

Función que escribe directamente en el FIFO del TRF7960.

Parámetros

∗pbuf	string con los datos a escribir en el FIFO
length	cantidad de datos a enviar.

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.14.2.13. readCont (char * *pbuf,* char *length*)

Función que lee la cantidad dada de registros del TRF7960 a partir de la dirección pasada en ∗pbuf. Escribe el resultado en el propio string pasado.

Parámetros

∗pbuf	string con la dirección del primer registro a leer.
length	cantidad de registros a leer

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.14.2.14. readIrqStatus (char * *pbuf*)

Función que lee el registro IRQ STATUS.

Parámetros

∗pbuf	string donde guarda el valor

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.15. readSingle (char * *pbuf*, char *number*)

Función que lee los registros pasados como parámetro del TRF7960. Escribe el resultado en el propio string pasado.

Parámetros

**pbuf*	string con los datos a escribir en los registros.
number	cantidad de registros

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.14.2.16. void reset (void)

Función que envía al TRF7960 el comando de RESET.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.17. void resetIrqStatus (void)

Función que borra el registro IRQ STATUS.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.18. void runDecoders (void)

Función que habilita nuevamente la parte digital del receptor de RF.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.19. void sensar (char *sensor*)

Función que se encarga de interrogar al TAG para saber el valor del sensor especificado en el parámetro. Maneja las respuestas intermedias recibidas del TAG. Se encarga también de despertar al TRF7960 antes de iniciar la interrogación, así también como de ponerlo en modo de bajo consumo mientras espera y luego de finalizada la consulta. Espera por la respuesta del TAG o indica un error ya sea si hubo time out o si el valor obtenido no es correcto por encontrarse demasiado apartados tag y lector.

Parámetros

sensor	string que indica el sensor a consultar.

Autor

RP - Texas Instruments - 2/12/2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.14.2.20. void startCondition (void)

Función que genera la condición de inicio de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.21. void stopCondition (void)

Función que genera la condición de finalización de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.

Función que genera la condición de finalización de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960, cuando se usa el modo continuo de envío de datos.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.22. void stopContinuous (void)

4.14.2.23. void stopDecoders (void)

Función que pone la parte digital del receptor de RF en modo reset para evitar interrupciones en ambientes ruidosos.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.24. void turnRfOn (void)

Función que enciende la parte de RF del TRF7960.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.25. void writeCont (char * *pbuf,* char *length*)

Función que escribe los registros pasados como parámetros. Al igual que su análoga readCont, escribe a partir del registro pasado como parámetro y a las direcciones consecutivas.

Parámetros

**pbuf*	string donde se pasa la dirección del primer registro a escribir seguida de los valores para ese registro y para los que le siguen.
length	variable donde se pasa el número de registros a escribir más uno.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.26. void writeIsoControl (char *iso_control*)

Función que escribe en el registro ISO CONTROL quien determina el protocolo a usar y sus parámetros (e.g. high data rate, ISO 15693, etc.)

Parámetros

iso_control	valor a configurar en el registro.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.2.27. void writeSingle (char * *pbuf,* char *length*)

Función que escribe los registros pasados como parámetros.

Parámetros

∗pbuf	string donde se pasa la dirección del registro seguida del valor a escribirle.
length	variable donde se pasa el DOBLE del número de registros a escribir.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.14.3. Documentación de las variables

4.14.3.1. char buf[300]

4.14.3.2. char command[2]

4.14.3.3. char direct_mode = 0

4.14.3.4. char error

4.14.3.5. char i_reg

4.14.3.6. char irq_flag

4.14.3.7. char rx_error_flag

4.14.3.8. s08_t rxtx_state

4.14.3.9. char stand_alone_flag

4.14.3.10. char uid[8]

4.15. Referencia del Archivo trf7960.h

Archivo que contiene la interfaz pública que maneja el lector TRF7960A.

```
#include <stdlib.h>
```

'defines'

- #define IDLE 0x00

  ==== TRF796x definitions ==
- #define SOFT_INIT 0x03
- #define INITIAL_RF_COLLISION 0x04
- #define RESPONSE_RF_COLLISION_N 0x05
- #define RESPONSE_RF_COLLISION_0 0x06
- #define RESET 0x0F
- #define TRANSMIT_NO_CRC 0x10
- #define TRANSMIT_CRC 0x11
- #define DELAY_TRANSMIT_NO_CRC 0x12
- #define DELAY_TRANSMIT_CRC 0x13

- #define TRANSMIT_NEXT_SLOT 0x14
- #define CLOSE_SLOT_SEQUENCE 0x15
- #define STOP_DECODERS 0x16
- #define RUN_DECODERS 0x17
- #define CHECK_INTERNAL_RF 0x18
- #define CHECK_EXTERNAL_RF 0x19
- #define ADJUST_GAIN 0x1A
- #define CHIP_STATE_CONTROL 0x00
- #define ISO_CONTROL 0x01
- #define ISO_14443B_OPTIONS 0x02
- #define ISO_14443A_OPTIONS 0x03
- #define TX_TIMER_EPC_HIGH 0x04
- #define TX_TIMER_EPC_LOW 0x05
- #define TX_PULSE_LENGTH_CONTROL 0x06
- #define RX_NO_RESPONSE_WAIT_TIME 0x07
- #define RX_WAIT_TIME 0x08
- #define MODULATOR_CONTROL 0x09
- #define RX_SPECIAL_SETTINGS 0x0A
- #define REGULATOR_CONTROL 0x0B
- #define IRQ_STATUS 0x0C
- #define IRQ_MASK 0x0D
- #define COLLISION_POSITION 0x0E
- #define RSSI_LEVELS 0x0F
- #define SPECIAL_FUNCTION 0x10
- #define RAM_START_ADDRESS 0x11
- #define NFCID 0x17
- #define NFC_TArGET_LEVEL 0x18
- #define NFC_TARGET_PROTOCOL 0x19
- #define TEST_SETTINGS_1 0x1A
- #define TEST_SETTINGS_2 0x1B
- #define FIFO_STATUS 0x1C
- #define TX_LENGTH_BYTE_1 0x1D
- #define TX_LENGTH_BYTE_2 0x1E
- #define FIFO 0x1F

Funciones

- void communicationSetup (void)

  Función que configura la comunicación paralelo entre el MSP y el TRF7960.
- void directCommand (char ∗pbuf)

  Función que envía el comando pasado como parámetro directamente al TRF7960.
- void initialSettings (void)

  Función que inicializa el TRF7960 poniendo el reloj en 6.78MHz.
- void ISR (char ∗irq_status)

  Función que maneja las interrupciones del TRF7960. Determina el motivo de la interrupción leyendo el IRQ Status Register y realiza las acciones requeridas.
- void rawWrite (char ∗pbuf, char length)

  Función que escribe directamente en el FIFO del TRF7960.
- void readSingle (char ∗pbuf, char number)

  Función que lee los registros pasados como parámetro del TRF7960. Escribe el resultado en el propio string pasado.
- void readCont (char ∗pbuf, char length)

  Función que lee la cantidad dada de registros del TRF7960 a partir de la dirección pasada en ∗pbuf. Escribe el resultado en el propio string pasado.

- void readIrqStatus (char *pbuf)

  Función que lee el registro IRQ STATUS.
- void writeSingle (char *pbuf, char length)

  Función que escribe los registros pasados como parámetros.
- void writeCont (char *pbuf, char length)

  Función que escribe los registros pasados como parámetros. Al igual que su análoga readCont, escribe a partir del registro pasado como parámetro y a las direcciones consecutivas.
- void writeIsoControl (char iso_control)

  Función que escribe en el registro ISO CONTROL quien determina el protocolo a usar y sus parámetros (e.g. high data rate, ISO 15693, etc.)
- void reset (void)

  Función que envía al TRF7960 el comando de RESET.
- void resetIrqStatus (void)

  Función que borra el registro IRQ STATUS.
- void turnRfOn (void)

  Función que enciende la parte de RF del TRF7960.
- void stopDecoders (void)

  Función que pone la parte digital del receptor de RF en modo reset para evitar interrupciones en ambientes ruidosos.
- void runDecoders (void)

  Función que habilita nuevamente la parte digital del receptor de RF.
- void startCondition (void)

  Función que genera la condición de inicio de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.
- void stopCondition (void)

  Función que genera la condición de finalización de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.
- void stopContinuous (void)
- void cargarUID (char *pbuffer)

  Función que carga el identificador específico del TAG a usar.
- void consultarTag (char *pbuf, char length)

  Función que se encarga de interrogar al TAG enviándole el comando pasado como parámetro. Espera por la respuesta del TAG o indica un error si hubo time out.
- void sensar (char sensor)

  Función que se encarga de interrogar al TAG para saber el valor del sensor especificado en el parámetro. Maneja las respuestas intermedias recibidas del TAG. Se encarga también de despertar al TRF7960 antes de iniciar la interrogación, así también como de ponerlo en modo de bajo consumo mientras espera y luego de finalizada la consulta. Espera por la respuesta del TAG o indica un error ya sea si hubo time out o si el valor obtenido no es correcto por encontrarse demasiado apartados tag y lector.
- void enterMode1_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 1 de bajo consumo (5mA típico) En el mismo, tanto TX como RX de RF están apagadas.
- void enterMode2_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 2 de bajo consumo (10.5mA típico) En el mismo, TX está apagada, pero la RX por RF está activa.
- void enterMode3_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 3 de consumo (70mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, pero se transmite con media potencia.
- void enterMode4_5V (void)

  Función que se encarga de poner el TRF7960 en el modo 4 de consumo (130mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, y se transmite a máxima potencia.

4.15.1. Descripción detallada

Archivo que contiene la interfaz pública que maneja el lector TRF7960A.

Autor

Texas Instruments - Peter Reiser 2/12/2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.15.2. Documentación de los 'defines'

4.15.2.1. #define ADJUST_GAIN 0x1A

4.15.2.2. #define CHECK_EXTERNAL_RF 0x19

4.15.2.3. #define CHECK_INTERNAL_RF 0x18

4.15.2.4. #define CHIP_STATE_CONTROL 0x00

4.15.2.5. #define CLOSE_SLOT_SEQUENCE 0x15

4.15.2.6. #define COLLISION_POSITION 0x0E

4.15.2.7. #define DELAY_TRANSMIT_CRC 0x13

4.15.2.8. #define DELAY_TRANSMIT_NO_CRC 0x12

4.15.2.9. #define FIFO 0x1F

4.15.2.10. #define FIFO_STATUS 0x1C

4.15.2.11. #define IDLE 0x00

==== TRF796x definitions ==

4.15.2.12. #define INITIAL_RF_COLLISION 0x04

4.15.2.13. #define IRQ_MASK 0x0D

4.15.2.14. #define IRQ_STATUS 0x0C

4.15.2.15. #define ISO_14443A_OPTIONS 0x03

4.15.2.16. #define ISO_14443B_OPTIONS 0x02

4.15.2.17. #define ISO_CONTROL 0x01

4.15.2.18. #define MODULATOR_CONTROL 0x09

4.15.2.19. #define NFC_TArGET_LEVEL 0x18

4.15.2.20. #define NFC_TARGET_PROTOCOL 0x19

4.15.2.21. #define NFCID 0x17

4.15.2.22. #define RAM_START_ADDRESS 0x11

4.15.2.23. #define REGULATOR_CONTROL 0x0B

4.15.2.24. #define RESET 0x0F

4.15.2.25. #define RESPONSE_RF_COLLISION_0 0x06

4.15.2.26. #define RESPONSE_RF_COLLISION_N 0x05

4.15.2.27. #define RSSI_LEVELS 0x0F

4.15.2.28. #define RUN_DECODERS 0x17

4.15.2.29. #define RX_NO_RESPONSE_WAIT_TIME 0x07

4.15.2.30. #define RX_SPECIAL_SETTINGS 0x0A

4.15.2.31. #define RX_WAIT_TIME 0x08

4.15.2.32. #define SOFT_INIT 0x03

4.15.2.33. #define SPECIAL_FUNCTION 0x10

4.15.2.34. #define STOP_DECODERS 0x16

4.15.2.35. #define TEST_SETTINGS_1 0x1A

4.15.2.36. #define TEST_SETTINGS_2 0x1B

4.15.2.37. #define TRANSMIT_CRC 0x11

4.15.2.38. #define TRANSMIT_NEXT_SLOT 0x14

4.15.2.39. #define TRANSMIT_NO_CRC 0x10

4.15.2.40. #define TX_LENGTH_BYTE_1 0x1D

4.15.2.41. #define TX_LENGTH_BYTE_2 0x1E

4.15.2.42. #define TX_PULSE_LENGTH_CONTROL 0x06

4.15.2.43. #define TX_TIMER_EPC_HIGH 0x04

4.15.2.44. #define TX_TIMER_EPC_LOW 0x05

4.15.3. Documentación de las funciones

4.15.3.1. void cargarUID (char * *pbuffer*)

Función que carga el identificador específico del TAG a usar.

Parámetros

pbuffer	cadena con el identificador a cargar.

4.15.3.2. void communicationSetup (void)

Función que configura la comunicación paralelo entre el MSP y el TRF7960.

4.15.3.3. void consultarTag (char * *pbuf*, char *length*)

Función que se encarga de interrogar al TAG enviándole el comando pasado como parámetro. Espera por la respuesta del TAG o indica un error si hubo time out.

Parámetros

*pbuf	string con el comando y sus parámetros a enviar al TAG
length	cantidad de datos a enviar al TAG

Autor

RP - Texas Instruments - 2/12/2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.15.3.4. void directCommand (char * *pbuf*)

Función que envía el comando pasado como parámetro directamente al TRF7960.

Parámetros

*pbuf	Puntero a la cadena que contiene el comando.

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.15.3.5. void enterMode1_5V (void)

Función que se encarga de poner el TRF7960 en el modo 1 de bajo consumo (5mA típico) En el mismo, tanto TX como RX de RF están apagadas.

4.15.3.6. void enterMode2_5V (void)

Función que se encarga de poner el TRF7960 en el modo 2 de bajo consumo (10.5mA típico) En el mismo, TX está apagada, pero la RX por RF está activa.

4.15.3.7. void enterMode3_5V (void)

Función que se encarga de poner el TRF7960 en el modo 3 de consumo (70mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, pero se transmite con media potencia.

4.15.3.8. void enterMode4_5V (void)

Función que se encarga de poner el TRF7960 en el modo 4 de consumo (130mA típico) Es un modo activo. En el mismo TX y RX están ambas activas, y se transmite a máxima potencia.

4.15.3.9. void initialSettings (void)

Función que inicializa el TRF7960 poniendo el reloj en 6.78MHz.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.10. void ISR (char * *irq_status*)

Función que maneja las interrupciones del TRF7960. Determina el motivo de la interrupción leyendo el IRQ Status Register y realiza las acciones requeridas.

Parámetros

irq_status	string con el valor del ISR Status Register

Autor

RP - Texas Instruments

4.15.3.11. void rawWrite (char * *pbuf,* char *length*)

Función que escribe directamente en el FIFO del TRF7960.

Parámetros

*pbuf	string con los datos a escribir en el FIFO
length	cantidad de datos a enviar.

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.15.3.12. void readCont (char * *pbuf,* char *length*)

Función que lee la cantidad dada de registros del TRF7960 a partir de la dirección pasada en *pbuf. Escribe el resultado en el propio string pasado.

Parámetros

∗pbuf	string con la dirección del primer registro a leer.
length	cantidad de registros a leer

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.15.3.13. void readIrqStatus (char ∗ *pbuf*)

Función que lee el registro IRQ STATUS.

Parámetros

∗pbuf	string donde guarda el valor

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.14. void readSingle (char ∗ *pbuf,* char *number*)

Función que lee los registros pasados como parámetro del TRF7960. Escribe el resultado en el propio string pasado.

Parámetros

∗pbuf	string con los datos a escribir en los registros.
number	cantidad de registros

Autor

RP - Texas Instruments

Fecha

24/11/2010

4.15.3.15. void reset (void)

Función que envía al TRF7960 el comando de RESET.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.16. void resetIrqStatus (void)

Función que borra el registro IRQ STATUS.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.17. void runDecoders (void)

Función que habilita nuevamente la parte digital del receptor de RF.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.18. void sensar (char *sensor*)

Función que se encarga de interrogar al TAG para saber el valor del sensor especificado en el parámetro. Maneja las respuestas intermedias recibidas del TAG. Se encarga también de despertar al TRF7960 antes de iniciar la interrogación, así también como de ponerlo en modo de bajo consumo mientras espera y luego de finalizada la consulta. Espera por la respuesta del TAG o indica un error ya sea si hubo time out o si el valor obtenido no es correcto por encontrarse demasiado apartados tag y lector.

Parámetros

sensor	string que indica el sensor a consultar.

Autor

RP - Texas Instruments - 2/12/2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.15.3.19. void startCondition (void)

Función que genera la condición de inicio de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.20. void stopCondition (void)

Función que genera la condición de finalización de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960.

Función que genera la condición de finalización de una comunicación para el modo de conexión paralelo entre el MSP y el TRF7960, cuando se usa el modo continuo de envío de datos.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.21. void stopContinuous (void)

4.15.3.22. void stopDecoders (void)

Función que pone la parte digital del receptor de RF en modo reset para evitar interrupciones en ambientes ruidosos.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.23. void turnRfOn (void)

Función que enciende la parte de RF del TRF7960.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.24. void writeCont (char * *pbuf,* char *length*)

Función que escribe los registros pasados como parámetros. Al igual que su análoga readCont, escribe a partir del registro pasado como parámetro y a las direcciones consecutivas.

Parámetros

**pbuf*	string donde se pasa la dirección del primer registro a escribir seguida de los valores para ese registro y para los que le siguen.

length	variable donde se pasa el número de registros a escribir más uno.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.25. void writeIsoControl (char *iso_control*)

Función que escribe en el registro ISO CONTROL quien determina el protocolo a usar y sus parámetros (e.g. high data rate, ISO 15693, etc.)

Parámetros

iso_control	valor a configurar en el registro.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.15.3.26. void writeSingle (char * *pbuf,* char *length*)

Función que escribe los registros pasados como parámetros.

Parámetros

*pbuf	string donde se pasa la dirección del registro seguida del valor a escribirle.
length	variable donde se pasa el DOBLE del número de registros a escribir.

Autor

RP - Texas Instruments

Fecha

2/12/2010

4.16. Referencia del Archivo types.h

Archivo que contine la definición de los tipos de datos más usados.

'typedefs'

- typedef unsigned short u16_t
- typedef unsigned long u32_t
- typedef signed char s08_t
- typedef signed short s16_t
- typedef signed long s32_t

4.16.1. Descripción detallada

Archivo que contine la definición de los tipos de datos más usados.

Autor

Peter Reiser - Texas Instruments.

Fecha

2 Diciembre 2010

4.16.2. Documentación de los 'typedefs'

4.16.2.1. typedef signed char s08_t

4.16.2.2. typedef signed short s16_t

4.16.2.3. typedef signed long s32_t

4.16.2.4. typedef unsigned short u16_t

4.16.2.5. typedef unsigned long u32_t

4.17. Referencia del Archivo uart.c

Archivo con las funciones que se encargan de manejar la UART para la comunicación con el módulo BT.

```
#include <string.h>
#include <stdio.h>
#include "msp430f2370.h"
#include "types.h"
#include "uart.h"
#include "trf7960.h"
```

Funciones

- void set_eofl (char end_of_line)

  Función que especifica cual será el caracter a considerar como fin de trama.
- char get_eofl (void)

  Función que devuelve el caracter de fin de trama configurado.
- void cargarTXbuffer (char ∗string1)

  Función que carga en el buffer de transmisión el string a enviar. Envía el primer caracter a enviar y habilita las interrupciones de transmisión.
- void uartSetup (void)

  Función que configura los pines correspondientes del puerto 3 para poder usarlos como UART.
- void putChar (char tx_char)

  Función que envía un único caracter a través de la UART.
- void sendString (char ∗pstr)

  Función que envía un string a través de la UART.
- __interrupt void ISR_Rx ()

  ISR de RX: copia al buffer de RX el caracter recibido y setea bandera en caso de detectar el final de la trama.
- __interrupt void ISR_Tx ()

ISR de TX: envia siguiente caracter del buffer de TX. Si es el último deshabilita interrupciones y resetea indice.

- int getFlagRX ()

  Función que indica si se ha recibido una cadena por la UART.

- int getFlagTX ()
- void resetFlagRX ()

  Función que baja la bandera de recepción de la UART.

Variables

- buffer RXbuffer

  Variable tipo buffer para la recepción.

4.17.1. Descripción detallada

Archivo con las funciones que se encargan de manejar la UART para la comunicación con el módulo BT.

Autor

Peter Reiser - Texas Instruments Setiembre 2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.17.2. Documentación de las funciones

4.17.2.1. void cargarTXbuffer (char * *string1*)

Función que carga en el buffer de transmisión el string a enviar. Envía el primer caracter a enviar y habilita las interrupciones de transmisión.

Parámetros

string1	Cadena con el texto a enviar.
string1	cadena con el string a enviar.

4.17.2.2. char get_eofl (void)

Función que devuelve el caracter de fin de trama configurado.

4.17.2.3. int getFlagRX ( )

Función que indica si se ha recibido una cadena por la UART.

Función que obtiene el valor de la bandera de transmisión.

Devuelve

Retorna el valor de la bandera de recepción.
Retorna el valor de la bandera de transmisión.

4.17.2.4. int getFlagTX ( )

4.17.2.5. __interrupt void ISR_Rx ( )

ISR de RX: copia al buffer de RX el caracter recibido y setea bandera en caso de detectar el final de la trama.

4.17.2.6. __interrupt void ISR_Tx ( )

ISR de TX: envia siguiente caracter del buffer de TX. Si es el último deshabilita interrupciones y resetea indice.

4.17.2.7. putChar (char *tx_char*)

Función que envía un único caracter a través de la UART.

Parámetros

tx_char	Caracter a enviar.

4.17.2.8. void resetFlagRX ( )

Función que baja la bandera de recepción de la UART.

4.17.2.9. sendString (char * *pstr*)

Función que envía un string a través de la UART.

Parámetros

pstr	Cadena a enviar.

4.17.2.10. void set_eofl (char *end_of_line*)

Función que especifica cual será el caracter a considerar como fin de trama.

Parámetros

end_of_line	Final de la trama.
end_of_line	caracter que indicará el caracter de final de la trama.

4.17.2.11. uartSetup (void)

Función que configura los pines correspondientes del puerto 3 para poder usarlos como UART.

4.17.3. Documentación de las variables

4.17.3.1. buffer RXbuffer

Variable tipo buffer para la recepción.

4.18. Referencia del Archivo uart.h

Define la interfaz pública del módulo uart.

Estructuras de datos

- struct buffer

'defines'

- #define BAUD0 109
- #define BAUD1 0
- #define BAUD0EN 0x2B
- #define BAUD1EN 0x00
- #define TAM 128

Funciones

- void cargarTXbuffer (char *string1)

  Función que carga en el buffer de transmisión el string a enviar. Envía el primer caracter a enviar y habilita las interrupciones de transmisión.
- void set_eofl (char end_of_line)

  Función que especifica cual será el caracter a considerar como fin de trama.
- char get_eofl (void)

  Función que devuelve el caracter de fin de trama configurado.
- void resetFlagRX ()

  Función que baja la bandera de recepción de la UART.
- int getFlagRX ()

  Función que indica si se ha recibido una cadena por la UART.
- int getFlagTX ()
- void putChar (char TXchar)

  Función que envía un único caracter a través de la UART.
- void sendString (char *pstr)

  Función que envía un string a través de la UART.
- void uartSetup (void)

  Función que configura los pines correspondientes del puerto 3 para poder usarlos como UART.

4.18.1. Descripción detallada

Define la interfaz pública del módulo uart.

Autor

Peter Reiser - Texas Instruments Setiembre 2010 Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

4.18.2. Documentación de los 'defines'

4.18.2.1. #define BAUD0 109

Definición de constantes privadas

4.18.2.2. #define BAUD0EN 0x2B

4.18.2.3. #define BAUD1 0

4.18.2.4. #define BAUD1EN 0x00

4.18.2.5. #define TAM 128

Constante que define el tamaño de los buffers, tanto de recepción como de transmisión.

4.18.3. Documentación de las funciones

4.18.3.1. void cargarTXbuffer (char * *string1*)

Función que carga en el buffer de transmisión el string a enviar. Envía el primer caracter a enviar y habilita las interrupciones de transmisión.

Parámetros

string1	Cadena con el texto a enviar.
string1	cadena con el string a enviar.

4.18.3.2. char get_eofl (void)

Función que devuelve el caracter de fin de trama configurado.

4.18.3.3. int getFlagRX ( )

Función que indica si se ha recibido una cadena por la UART.

Función que obtiene el valor de la bandera de transmisión.

Devuelve

Retorna el valor de la bandera de recepción.
Retorna el valor de la bandera de transmisión.

4.18.3.4. int getFlagTX ( )

4.18.3.5. void putChar (char *tx_char*)

Función que envía un único caracter a través de la UART.

Parámetros

tx_char	Caracter a enviar.

4.18.3.6. void resetFlagRX ( )

Función que baja la bandera de recepción de la UART.

4.18.3.7. void sendString (char * *pstr*)

Función que envía un string a través de la UART.

Parámetros

pstr	Cadena a enviar.

4.18.3.8. void set_eofl (char *end_of_line*)

Función que especifica cual será el caracter a considerar como fin de trama.

Parámetros

end_of_line	Final de la trama.
end_of_line	caracter que indicará el caracter de final de la trama.

4.18.3.9. void uartSetup (void)

Función que configura los pines correspondientes del puerto 3 para poder usarlos como UART.

Apéndice E – Documentación de Software Android

La documentación del código del Software Android fue generada utilizando el programa de generación de código DOXYGEN

MONCEL APP

1

Generado por Doxygen 1.8.7

Índice general

Capítulo 1

Indice de namespaces

1.1. Paquetes

Aquí van los paquetes con una breve descripción (si etá disponible):

Capítulo 2

Indice jerárquico

2.1. Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Capítulo 3

Índice de clases

3.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 4

Indice de archivos

4.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 5

Documentación de namespaces

5.1. Paquetes com

Paquetes

- package example

5.2. Paquetes com.example

Paquetes

- package superapp

5.3. Paquetes com.example.superapp

Clases

- class BtService
- class BuildConfig
- class DataBaseWrapper
- class MainActivity
- class R
- class Register
- class RegisterOperations
- class UserSettingActivity

5.3.1. Descripción detallada

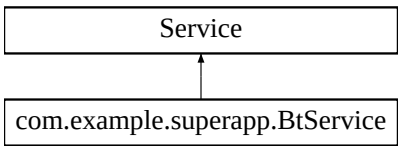
Automatically generated file. DO NOT MODIFY

Capítulo 6

Documentación de las clases

6.1. Referencia de la Clase com.example.superapp.BtService

Diagrama de herencias de com.example.superapp.BtService



Métodos públicos

- int onStartCommand (Intent intent, int flags, int startId)

  Implementa onStartComand de la super clase Service.
- void addRegister (double value, long date)

  Agrega un registro de temperatura en la tabla de registros.
- void AnalizarDato (float temperature)

  Analiza el dato recibido. En caso de estar fuera de rango envia alerta a tercero y/o genera una notificación.
- boolean compareValue (float value)

  Verifica si el valor leído del sensor está dentro del rango de temperatura deseado.
- void Send (String message)

  Escribe en el buffer de salida de Bluetooth un mensaje para enviar.
- IBinder onBind (Intent intent)

  Implementa onBind de la clase service.

Atributos públicos

- ArrayAdapter< String > adapter
- LocalBroadcastManager broadcaster

Atributos públicos estáticos

- static final String EXTRA_MESSAGE = "com.example.superapp.MESSAGE"

6.1.1. Descripción detallada

Definición en la línea 58 del archivo BtService.java.

6.1.2. Documentación de las funciones miembro

6.1.2.1. com.example.superapp.BtService.addRegister (double *value,* long *date*)

Agrega un registro de temperatura en la tabla de registros.

Parámetros

value	Temperatura en ℃ del registro.
date	Fecha del registro.

Definición en la línea 153 del archivo BtService.java.

6.1.2.2. com.example.superapp.BtService.AnalizarDato (float *temperature*)

Analiza el dato recibido. En caso de estar fuera de rango envia alerta a tercero y/o genera una notificación.

Parámetros

temperature	Valor de temperatura a comparar.

Definición en la línea 223 del archivo BtService.java.

6.1.2.3. com.example.superapp.BtService.compareValue (float *value*)

Verifica si el valor leído del sensor está dentro del rango de temperatura deseado.

Parámetros

value	Temperatura en ℃ del valor leído.

Devuelve

Retorna "True" si temperatura fuera de rango.

Definición en la línea 264 del archivo BtService.java.

6.1.2.4. IBinder com.example.superapp.BtService.onBind (Intent *intent*)

Implementa onBind de la clase service.

Definición en la línea 411 del archivo BtService.java.

6.1.2.5. com.example.superapp.BtService.onStartCommand (Intent *intent,* int *flags,* int *startId*)

Implementa onStartComand de la super clase Service.

Definición en la línea 92 del archivo BtService.java.

6.1.2.6. com.example.superapp.BtService.Send (String *message*)

Escribe en el buffer de salida de Bluetooth un mensaje para enviar.

Parámetros

message	Mensaje a enviar.

Definición en la línea 274 del archivo BtService.java.

6.1.3. Documentación de los datos miembro

6.1.3.1. ArrayAdapter<String> com.example.superapp.BtService.adapter

Definición en la línea 68 del archivo BtService.java.

6.1.3.2. LocalBroadcastManager com.example.superapp.BtService.broadcaster

Definición en la línea 84 del archivo BtService.java.

6.1.3.3. final String com.example.superapp.BtService.EXTRA_MESSAGE = "com.example.superapp.MESSAGE" [static]

Definición en la línea 66 del archivo BtService.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/src/com/example/superapp/BtService.java

6.2. Referencia de la Clase com.example.superapp.BuildConfig

Atributos públicos estáticos

- static final boolean DEBUG = true

6.2.1. Descripción detallada

Definición en la línea 4 del archivo BuildConfig.java.

6.2.2. Documentación de los datos miembro

6.2.2.1. final boolean com.example.superapp.BuildConfig.DEBUG = true [static]

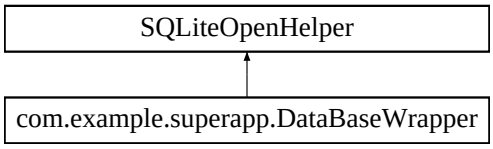
Definición en la línea 5 del archivo BuildConfig.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/gen/com/example/superapp/BuildConfig.java

6.3. Referencia de la Clase com.example.superapp.DataBaseWrapper

Diagrama de herencias de com.example.superapp.DataBaseWrapper



Métodos públicos

- DataBaseWrapper (Context context)

  Constructor de objetos de la clase. Utilizado para crear nuevas instancias.
- void onCreate (SQLiteDatabase db)

  Llamada cuando se crea la base por primera vez. Crea las tablas y \ valores iniciales.
- void onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)

  Llamada cuando se necesita renovar la base de datos.

Atributos públicos estáticos

- static final String REGISTERS = "Registers"
- static final String REGISTER_ID = "_id"
- static final String REGISTER_DATE = "_date"
- static final String REGISTER_VALUE = "_value"

6.3.1. Descripción detallada

Definición en la línea 26 del archivo DataBaseWrapper.java.

6.3.2. Documentación del constructor y destructor

6.3.2.1. com.example.superapp.DataBaseWrapper.DataBaseWrapper (Context *context*)

Constructor de objetos de la clase. Utilizado para crear nuevas instancias.

Parámetros

context	Contexto donde se creará la nueva instancia.

Definición en la línea 47 del archivo DataBaseWrapper.java.

6.3.3. Documentación de las funciones miembro

6.3.3.1. com.example.superapp.DataBaseWrapper.onCreate (SQLiteDatabase *db*)

Llamada cuando se crea la base por primera vez. Crea las tablas y \ valores iniciales.

Parámetros

db	Base de datos a crear.

Definición en la línea 58 del archivo DataBaseWrapper.java.

6.3.3.2. com.example.superapp.DataBaseWrapper.onUpgrade (SQLiteDatabase *db,* int *oldVersion,* int *newVersion*)

Llamada cuando se necesita renovar la base de datos.

Parámetros

db	Base de datos a renovar.
oldVersion	Versión de la base de datos vieja.
newVersion	Versión de la nueva base de datos.

Definición en la línea 70 del archivo DataBaseWrapper.java.

6.3.4. Documentación de los datos miembro

6.3.4.1. final String com.example.superapp.DataBaseWrapper.REGISTER_DATE = "_date" [static]

Definición en la línea 30 del archivo DataBaseWrapper.java.

6.3.4.2. final String com.example.superapp.DataBaseWrapper.REGISTER_ID = "_id" [static]

Definición en la línea 29 del archivo DataBaseWrapper.java.

6.3.4.3. final String com.example.superapp.DataBaseWrapper.REGISTER_VALUE = "_value" [static]

Definición en la línea 31 del archivo DataBaseWrapper.java.

6.3.4.4. final String com.example.superapp.DataBaseWrapper.REGISTERS = "Registers" [static]

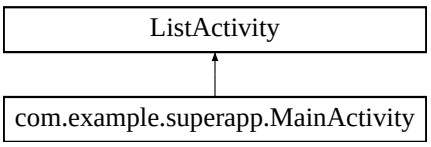
Definición en la línea 28 del archivo DataBaseWrapper.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/src/com/example/superapp/DataBaseWrapper.java

6.4. Referencia de la Clase com.example.superapp.MainActivity

Diagrama de herencias de com.example.superapp.MainActivity



Métodos públicos

- boolean onCreateOptionsMenu (Menu menu)

  Muestra en pantalla el menu solicitado.
- boolean onOptionsItemSelected (MenuItem item)

  Gestiona accion a realizar cuando se selecciona un item.

Atributos públicos

- ArrayAdapter< String > adapter
- Intent bluetooth

Atributos públicos estáticos

- static final String EXTRA_MESSAGE = "com.example.superapp.MESSAGE"

Métodos protegidos

- void onCreate (Bundle savedInstanceState)

  Implementa onCreate de la super clase Activity.
- void onStart ()

  Implementa onStart de la super clase Activity.
- void onResume ()

  Implementa onResume de la super clase Activity.
- void onPause ()

  Implementa onPause de la super clase Activity.
- void onStop ()

  Implementa onStop de la super clase Activity.
- void onDestroy ()

  Implementa onDestroy de la super clase Activity.

6.4.1. Descripción detallada

Definición en la línea 40 del archivo MainActivity.java.

6.4.2. Documentación de las funciones miembro

6.4.2.1. com.example.superapp.MainActivity.onCreate (Bundle *savedInstanceState*) [protected]

Implementa onCreate de la super clase Activity.

Definición en la línea 59 del archivo MainActivity.java.

6.4.2.2. com.example.superapp.MainActivity.onCreateOptionsMenu (Menu *menu*)

Muestra en pantalla el menu solicitado.

Parámetros

menu	Menu a mostrar en pantalla.

Definición en la línea 185 del archivo MainActivity.java.

6.4.2.3. com.example.superapp.MainActivity.onDestroy ( ) [protected]

Implementa onDestroy de la super clase Activity.

Definición en la línea 149 del archivo MainActivity.java.

6.4.2.4. com.example.superapp.MainActivity.onOptionsItemSelected (MenuItem *item*)

Gestiona accion a realizar cuando se selecciona un item.

Parámetros

item	Item seleccionado.

Definición en la línea 197 del archivo MainActivity.java.

6.4.2.5. com.example.superapp.MainActivity.onPause ( ) [protected]

Implementa onPause de la super clase Activity.

Definición en la línea 130 del archivo MainActivity.java.

6.4.2.6. com.example.superapp.MainActivity.onResume ( ) [protected]

Implementa onResume de la super clase Activity.

Definición en la línea 110 del archivo MainActivity.java.

6.4.2.7. com.example.superapp.MainActivity.onStart ( ) [protected]

Implementa onStart de la super clase Activity.

Definición en la línea 99 del archivo MainActivity.java.

6.4.2.8. com.example.superapp.MainActivity.onStop ( ) [protected]

Implementa onStop de la super clase Activity.

Definición en la línea 139 del archivo MainActivity.java.

6.4.3. Documentación de los datos miembro

6.4.3.1. ArrayAdapter<String> com.example.superapp.MainActivity.adapter

Definición en la línea 48 del archivo MainActivity.java.

6.4.3.2. Intent com.example.superapp.MainActivity.bluetooth

Definición en la línea 52 del archivo MainActivity.java.

6.4.3.3. final String com.example.superapp.MainActivity.EXTRA_MESSAGE = "com.example.superapp.MESSAGE" [static]

Definición en la línea 44 del archivo MainActivity.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/src/com/example/superapp/MainActivity.java

6.5. Referencia de la Clase com.example.superapp.R

Clases

- class **attr**

- class **dimen**
- class **drawable**
- class **id**
- class **layout**
- class **menu**
- class **string**
- class **style**
- class **xml**

6.5.1. Descripción detallada

Definición en la línea 10 del archivo R.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/gen/com/example/superapp/R.↩ java

6.6. Referencia de la Clase com.example.superapp.Register

Métodos públicos

- long getId ()

  Retorna identificador único del registro.
- void setId (int id)

  Setea identificador único del registro.
- String getDate ()

  Retorna timestamp del registro.
- void setDate (String date)

  Setea timestamp del registro.
- Double getValue ()

  Retorna valor del registro.
- void setValue (Double value)

  Setea valor del registro.
- String toString ()

  Función utilizada para obtener contenido de registro en una cadena de caracteres.

6.6.1. Descripción detallada

Definición en la línea 26 del archivo Register.java.

6.6.2. Documentación de las funciones miembro

6.6.2.1. com.example.superapp.Register.getDate ( )

Retorna timestamp del registro.

Devuelve

Timestamp del registro.

Definición en la línea 55 del archivo Register.java.

6.6.2.2. com.example.superapp.Register.getId ( )

Retorna identificador único del registro.

Devuelve

ID del registro.

Definición en la línea 37 del archivo Register.java.

6.6.2.3. com.example.superapp.Register.getValue ( )

Retorna valor del registro.

Devuelve

Valor del registro.

Definición en la línea 73 del archivo Register.java.

6.6.2.4. com.example.superapp.Register.setDate (String *Date*)

Setea timestamp del registro.

Parámetros

date	Timestamp del registro.

Definición en la línea 64 del archivo Register.java.

6.6.2.5. com.example.superapp.Register.setId (int *id*)

Setea identificador único del registro.

Parámetros

id	ID del registro.

Definición en la línea 46 del archivo Register.java.

6.6.2.6. com.example.superapp.Register.setValue (Double *Value*)

Setea valor del registro.

Parámetros

value	Valor del registro.

Definición en la línea 82 del archivo Register.java.

6.6.2.7. com.example.superapp.Register.toString ( )

Función utilizada para obtener contenido de registro en una cadena de caracteres.

Devuelve

String indicando valor y timestamp del registro.

Definición en la línea 92 del archivo Register.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/src/com/example/superapp/Register.↩ java

6.7. Referencia de la Clase com.example.superapp.RegisterOperations

Métodos públicos

- RegisterOperations (Context context)

  Crea instancia de la clase DataBaseWrapper.
- void open () throws SQLException

  Establece la conexion con la base de datos.
- void close ()

  Cierra la conexion con la base de datos.
- Register addRegister (Double value, String date)

  Agrega registro a la base de datos.
- void deleteRegistro (Register reg)

  Elimina registro de la base de datos.
- List< String > getAllRegisters ()

  Devuelve una lista con los registros de la base de datos.

6.7.1. Descripción detallada

Definición en la línea 30 del archivo RegisterOperations.java.

6.7.2. Documentación del constructor y destructor

6.7.2.1. com.example.superapp.RegisterOperations.RegisterOperations (Context *context*)

Crea instancia de la clase DataBaseWrapper.

Parámetros

context	Contexto de la actividad.

Definición en la línea 42 del archivo RegisterOperations.java.

6.7.3. Documentación de las funciones miembro

6.7.3.1. com.example.superapp.RegisterOperations.addRegister (Double *value,* String *date*)

Agrega registro a la base de datos.

Parámetros

value	Valor del registro.
date	Fecha del registro.

Devuelve

Registro guardado en la base de datos.

Definición en la línea 69 del archivo RegisterOperations.java.

6.7.3.2. com.example.superapp.RegisterOperations.close ( )

Cierra la conexion con la base de datos.

Definición en la línea 58 del archivo RegisterOperations.java.

6.7.3.3. com.example.superapp.RegisterOperations.deleteRegistro (Register *reg*)

Elimina registro de la base de datos.

Parámetros

reg	Registro a eliminar.

Definición en la línea 90 del archivo RegisterOperations.java.

6.7.3.4. com.example.superapp.RegisterOperations.getAllRegisters ( )

Devuelve una lista con los registros de la base de datos.

Devuelve

Lista con los registros de la base de datos.

Definición en la línea 102 del archivo RegisterOperations.java.

6.7.3.5. com.example.superapp.RegisterOperations.open ( ) throws SQLException

Establece la conexion con la base de datos.

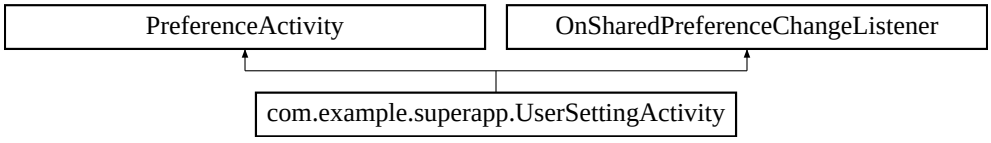
Definición en la línea 50 del archivo RegisterOperations.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/src/com/example/superapp/RegisterOperations.java

6.8. Referencia de la Clase com.example.superapp.UserSettingActivity

Diagrama de herencias de com.example.superapp.UserSettingActivity



Métodos públicos

- void onCreate (Bundle savedInstanceState)
- void onSharedPreferenceChanged (SharedPreferences sharedPreferences, String key)

Métodos protegidos

- void onResume ()
- void onPause ()

6.8.1. Descripción detallada

Definición en la línea 30 del archivo UserSettingActivity.java.

6.8.2. Documentación de las funciones miembro

6.8.2.1. com.example.superapp.UserSettingActivity.onCreate (Bundle *savedInstanceState*)

Definición en la línea 37 del archivo UserSettingActivity.java.

6.8.2.2. com.example.superapp.UserSettingActivity.onPause ( ) [protected]

Definición en la línea 62 del archivo UserSettingActivity.java.

6.8.2.3. com.example.superapp.UserSettingActivity.onResume ( ) [protected]

Definición en la línea 52 del archivo UserSettingActivity.java.

6.8.2.4. com.example.superapp.UserSettingActivity.onSharedPreferenceChanged (SharedPreferences *sharedPreferences,* String *key*)

Definición en la línea 72 del archivo UserSettingActivity.java.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/src/com/example/superapp/UserSettingActivity.java

Capítulo 7

Documentación de archivos

7.1. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/AndroidManifest.xml

7.2. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/bin/AndroidManifest.xml

7.3. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/gen/com/example/superapp/BuildConfig.java

Clases

- class com.example.superapp.BuildConfig

Paquetes

- package com.example.superapp

7.4. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/gen/com/example/superapp/R.java

Clases

- class com.example.superapp.R
- class **com.example.superapp.R.attr**
- class **com.example.superapp.R.dimen**
- class **com.example.superapp.R.drawable**
- class **com.example.superapp.R.id**
- class **com.example.superapp.R.layout**
- class **com.example.superapp.R.menu**
- class **com.example.superapp.R.string**
- class **com.example.superapp.R.style**
- class **com.example.superapp.R.xml**

Paquetes

- package com.example.superapp

7.5. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/layout/activity_config.xml

7.6. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/layout/activity_display_message.xml

7.7. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/layout/activity_main.xml

7.8. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/layout/activity_ver_registros.xml

7.9. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/layout/preferences.xml

7.10. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/menu/config.xml

7.11. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/menu/configuracion.xml

7.12. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/menu/display_message.xml

7.13. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/menu/main.xml

7.14. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/menu/ver_registros.xml

7.15. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/values/dimens.xml

7.16. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/values-sw600dp/dimens.xml

7.17. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/values-sw720dp-land/dimens.xml

7.18. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/values/strings.xml

7.19. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/values/styles.xml

7.20. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/values-v11/styles.xml

7.21. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/values-v14/styles.xml

7.22. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/res/xml/settings.xml

7.23. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/src/com/example/superapp/BtService.java

Clases

- class com.example.superapp.BtService

Paquetes

- package com.example.superapp

7.24. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/src/com/example/superapp/DataBaseWrapper.java

Clase que ayuda manejar la creación y manejar las versiones \ de la base de datos.

Clases

- class com.example.superapp.DataBaseWrapper

Paquetes

- package com.example.superapp

7.24.1. Descripción detallada

Clase que ayuda manejar la creación y manejar las versiones \ de la base de datos.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

Definición en el archivo DataBaseWrapper.java.

7.25. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/src/com/example/superapp/MainActivity.java

Servicio de Bluetooth. Se encarga de establecer la conexion, crear un handler para manejar los datos entrantes para su posterior analizis. Es el encargado de crear las notificaciones y enviar los mensajes de alerta.

Clases

- class com.example.superapp.MainActivity

Paquetes

- package com.example.superapp

7.25.1. Descripción detallada

Servicio de Bluetooth. Se encarga de establecer la conexion, crear un handler para manejar los datos entrantes para su posterior analizis. Es el encargado de crear las notificaciones y enviar los mensajes de alerta.

Actividad inicial del sistema. Despliega en pantalla los \ últimos registros que se obtuvieron.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

Definición en el archivo MainActivity.java.

7.26. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/src/com/example/superapp/Register.java

Clase para la creación de registros a guardar en la base de datos. Define \ la estructura de los registros, y brinda funciones para desplegarlos en \ pantalla.

Clases

- class com.example.superapp.Register

Paquetes

- package com.example.superapp

7.26.1. Descripción detallada

Clase para la creación de registros a guardar en la base de datos. Define \ la estructura de los registros, y brinda funciones para desplegarlos en \ pantalla.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

Definición en el archivo Register.java.

7.27. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/↩ SuperApp_vfranco/src/com/example/superapp/RegisterOperations.java

Clase utilizada para realizar operaciones sobre la base de datos.

Clases

- class com.example.superapp.RegisterOperations

Paquetes

- package com.example.superapp

7.27.1. Descripción detallada

Clase utilizada para realizar operaciones sobre la base de datos.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

Definición en el archivo RegisterOperations.java.

7.28. Referencia del Archivo C:/Users/Ramiro/Desktop/ANDROID_PROY_TEMPORAL/SuperApp_vfranco/src/com/example/superapp/UserSettingActivity.java

Genera una activity a partir de un .xml donde se configuran \ las preferencias del sistema.

Clases

- class com.example.superapp.UserSettingActivity

Paquetes

- package com.example.superapp

7.28.1. Descripción detallada

Genera una activity a partir de un .xml donde se configuran \ las preferencias del sistema.

Autor

Andrea Cukerman, Ramiro Barrón, Juan Martín Ortega

Fecha

Junio 2014

Definición en el archivo UserSettingActivity.java.