

**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE**



“Sistema de monitoreo de pacientes cardíacos en tiempo real, utilizando una aplicación Android con tecnologías Bluetooth y WebSocket”

Patricio Rodríguez Gatica

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO
CIVIL TELEMÁTICO**

PROFESOR GUIA: Marcos Zúñiga B.

PROFESOR CORRESPONDE: Francisco Cabezas B.

PROFESOR CORRESPONDE: Daniel Erraz L.

Agradecimientos

Agradecer es un paso fundamental en todo desarrollo humano, puesto que es de las pocas oportunidades de reflexionar sobre quienes estuvieron y están a nuestro lado en alguna etapa de nuestra vida. Me gustaría destacar que aun cuando agradecer es una vista al pasado, no existe tiempo inconexo en el corazón y los llevo siempre conmigo.

Quiero agradecer a mi familia, mi pareja, amigos, compañeros, profesores y toda persona con quien he tenido contacto en esta etapa universitaria, todos me han formado y son parte de este trabajo de una o de otra manera.

Le dedico este trabajo a quienes siempre creyeron en mí y a quienes aun lo hacen. Porque incluso teniendo un núcleo familiar distinto, el amor y la comprensión siempre estuvieron conmigo: A mi padre Omar Bernales Vega, a mis madres Toya y Mónica Gatica y mis hermanas Bárbara, Elein y Maka. Son mi orgullo y mi ejemplo a seguir.

Por último y no por ello menos importante, a la persona que soportó mis rabietas y jornadas de estrés, quien aun me acompaña y ama de forma extraordinaria, mi Valeria.

Resumen

El presente documento relatará la resolución de un problema real y actual en Chile, a partir de un desafío propuesto en el contexto de las Memorias Multidisciplinarias.

El desafío consiste en el desarrollo de un sistema con la capacidad de monitorear pacientes de forma remota, de bajo costo y con las limitantes geográficas propias de nuestro país, teniendo en mente su aplicación a nivel público del Sistema de Salud. Para esto, se analizaron las distintas opciones existentes en el mercado y se desarrolló una solución a nivel de prototipo funcional que cumpliese con las restricciones ya mencionadas.

Por ser un desafío resuelto de forma multidisciplinaria es importante destacar que el desarrollo en este documento estará enfocado al área informática y de telecomunicaciones asociada a la adquisición, procesamiento, almacenamiento y envío de datos.

El resto del equipo final está compuesto por: Sebastián Castillo actual Ingeniero en Diseño de Productos y Felipe Cordero actual Ingeniero Civil Electrónico, ambos de la misma casa de estudios UTFSM. Ambas memorias complementan la actual en el ámbito correspondiente a sus carreras, pero lógicamente compartiendo su núcleo como proyecto conjunto.

Glosario

- Esquemático : Es una representación pictórica de un circuito electrónico.
Muestra las diferentes componentes del circuito de manera simple y las conexiones de alimentación y señales entre distintos dispositivos.
- PCB : Printed Circuit Board, El circuito impreso se utiliza para conectar eléctricamente a través de las pistas conductoras, y sostener mecánicamente, por medio de la base, un conjunto de componentes electrónicos.
- Hardware : Partes físicas tangibles de un sistema informático.
- Software : Aplicaciones o programas que funcionan en un sistema informático.
- Firmware : Programa informático que establece la lógica de mas bajo nivel que controla los circuitos electrónicos.
- Bootloader : Es un programa que no tiene la totalidad de las funcionalidades para operar un sistema y está diseñado para preparar todo lo que necesita el firmware para ejecutarse.
- IIH : Infección Intra-Hospitalaria

Índice general

1.. <i>Introducción</i>	9
1.1. Memorias multidisciplinaria	9
1.1.1. Felipe Cordero	10
1.1.2. Vanessa Muñoz	10
1.1.3. Patricio Rodríguez	10
1.1.4. Sebastián Castillo	11
1.2. Análisis del desafío	11
1.3. Estado del Arte	12
1.3.1. ViSi Mobile®	13
1.3.2. Qardiocore	14
1.3.3. Nuubo	16
2.. <i>Alternativas de desarrollo</i>	18
2.1. Plataforma de desarrollo	18
2.1.1. Arduino	19
2.1.2. Raspberry	19
2.1.3. Beaglebone	20
2.2. Sensores	21
2.2.1. ECG	21
2.2.2. Temperatura	22
2.2.3. Ritmo Respiratorio	24
2.3. Unidad de movimiento inercial (IMU)	25
2.3.1. MPU-9250	26

2.4.	Comunicación	27
2.4.1.	GPRS shield	27
2.4.2.	Bluetooth BLE shield	29
2.5.	Conclusiones	29
2.5.1.	Plataforma de desarrollo	30
2.5.2.	Electrocardiograma	30
2.5.3.	Temperatura	30
2.5.4.	IMU	31
3..	<i>Selección del microcontrolador</i>	32
3.1.	ATMega328p	32
3.2.	ATMega2560	33
3.3.	ATMega644PA	35
3.4.	Conclusiones	36
4..	<i>Firmware y programación del microcontrolador</i>	38
4.1.	Bootloader	38
4.2.	ISP - In-System Programming	39
4.3.	Programación de un microcontrolador	40
4.3.1.	Programador ISP	41
4.4.	Programación USB	42
4.4.1.	FT232RL	43
4.5.	Programa en Arduino	45
5..	<i>Diseño Electrocardiograma</i>	46
5.1.	DFRobot Heart Rate Monitor	46
5.2.	Biopac MP150 ECG100C	47
5.3.	AD8232	49
5.4.	Comparación entre Biopac y DFRobot	51
5.5.	Eliminando ruido de la fuente	52

<i>6.. Diseño bluetooth</i>	54
6.1. BLEBee	54
6.2. Comunicación UART	56
6.3. Diseño del módulo bluetooth	56
<i>7.. Diseño cargador de batería</i>	58
7.1. Cargador de batería con corriente compartida con carga	59
7.2. MCP73871	59
7.2.1. Power Supply Input (IN)	60
7.2.2. Input Source Type Selection (SEL)	60
7.2.3. Regulación de carga rápida(PROG1)	61
7.2.4. Corriente de término de carga (PROG3)	62
7.2.5. Salida de voltaje a la batería V_{BAT} y sensor de voltaje de batería V_{SENSE}	62
7.2.6. Pines Generales	62
7.2.7. Diseño del esquemático	62
<i>8.. Diseño final</i>	64
8.1. Botón ON/OFF	64
8.2. Esquemático	67
8.2.1. Regulador de voltaje 3.3[V]	67
8.2.2. Conector para sensores	68
8.2.3. Pin header para prototipo	70
<i>9.. Discusión</i>	71
9.1. Transformación de memoria multidisciplinaria a emprendimiento . . .	71
9.1.1. Entrevista a doctores en hospital de Quintero	74
9.1.2. Pitch day seleccionados - Marzo 2018	76
9.2. Tareas Futuras	78

<i>10..Conclusiones</i>	79
-------------------------	----

<i>11..Anexos</i>	80
-------------------	----

ÍNDICE DE FIGURAS

1.1.	Interfaz de usuario ViSi Mobile®	13
1.2.	Modo de uso ViSi Mobile®	14
1.3.	Qardiocore multisensor	15
1.4.	Modo de uso Qardiocore	15
1.5.	nECG Shirt	16
1.6.	Sistema Nuubo	17
2.1.	Placa de desarrollo ECG	21
2.2.	Sensor de temperatura Lilypad	23
2.3.	Sensor de temperatura DS18B20	23
2.4.	Tela Conductiva MedTex	24
2.5.	IMU Sparkfun MPU-9250	26
2.6.	Ejes IMU MPU-9250	27
2.7.	Modulo GPRSbee	28
2.8.	Antena GPRSbee	28
2.9.	Bluetooth RN4020	29
3.1.	Microcontrolador ATMega328p	32
3.2.	Microcontrolador ATMega2560	33
3.3.	Comparación microcontroladores Arduino	34
3.4.	Microcontrolador ATMega644PA	35
4.1.	Configuración ISP para programar un microcontrolador	39
4.2.	Conecotor típico para programación ISP	40

4.3.	Esquemático básico MCU con ISP	41
4.4.	Chip FT232RL	43
4.5.	Conecotor USB 2.0 Micro B	44
5.1.	Forma de onda teórica ECG	46
5.2.	Biopac MP150	47
5.3.	Frecuencias de corte disponibles en Biopac ECG100C	48
5.4.	Diagrama de bloques funcional	49
5.5.	Herramienta para diseño de filtros	50
5.6.	Esquemático ECG DFRobot	50
5.7.	Gráfico Biopac ECG100C vs ECG DFRobot	51
5.8.	ECG en aplicación móvil sin filtro de salida	52
5.9.	Filtgro de salida R8 y C8	53
5.10.	Comparación ECG en aplicación móvil con filtros de salida	53
6.1.	Forma y Pines de puerto XBee	54
6.2.	Distribución de pines BLEBee	55
6.3.	Configuración para comunicación UART	56
6.4.	Esquemático bluetooth diseñado en EagleCAD	57
7.1.	Etapas de carga de baterías Li-ion para una batería de 4.2[V]	58
7.2.	Configuraciones disponibles para carga rápida	61
7.3.	Diseño de cargador de batería en software EagleCAD	63
8.1.	Interruptor de 2 posiciones	64
8.2.	Circuito para botón ON/OFF	65
8.3.	Función de encendido	66
8.4.	Función de apagado	66
8.5.	Velocidades y voltajes recomendados para un microcontrolador AT-Mega2560	67
8.6.	Regulador de voltaje 3.3[V]	68

8.7.	Conecotor para sensores	69
8.8.	Conecotor de 9 pines	69
8.9.	Conección Headers	70
9.1.	Desafío IoT propuesto por el instituto 3IE	72
9.2.	Planificación desafío IoT	73

ÍNDICE DE TABLAS

2.1.	Valores resistencia Tela MedTex en Pectorales	25
2.2.	Valores resistencia Tela MedTex en Plexo	25
2.3.	Valores resistencia Tela MedTex en Estomago	25
3.1.	Comparación microcontroladores	36
4.1.	Programación ISP utilizando Arduino UNO	42
4.2.	Descripción de pines chip FT232RL	43
6.1.	Pines relevantes módulo RN4020	55
7.1.	Descripción de pines MCP73871	60
9.1.	Ranking de dolor Fabián Álvarez	75

1. INTRODUCCIÓN

1.1. *Memorias multidisciplinaria*

La UTFSM ha manifestado, a través de sus planes de desarrollo y ejes estratégicos, la importancia de la formación de los estudiantes en competencias transversales, el fomento de la innovación, el emprendimiento y la vinculación con la industria. Es por esto que surge en la UTFSM el proyecto de Memorias Multidisciplinarias que propone impulsar el desarrollo de una nueva industria tecnológica a través de un programa de formación para la creación sistemática y sustentable de productos de innovación y emprendimientos ligados a tecnología.

Este proyecto de Memorias Multidisciplinarias se desarrolla a través de la proposición de un desafío el cual fue otorgado por el subgerente comercial de la empresa Sistemas Expertos, José Luis Araya. Sistemas Expertos e Ingeniería de Software (SEIS) es una empresa especialista con 10 años de experiencia en el desarrollo e implementación de soluciones tecnológicas para el área de la salud.

El desafío propuesto consiste en ¿Cómo podemos incorporar a bajo costo telemedicina a la salud pública, considerando restricciones económicas y geográficas?. Para esto hubo una conformación de un equipo multidisciplinario quienes desarrollaron durante un año, un plan de negocio, pruebas de concepto y prototipado de la solución con lo cual se pretende formar un emprendimiento. A raíz de las necesidades del desafío propuesto por la empresa Sistemas Expertos, se hace necesaria la incorporación de conocimientos en el ámbito técnico a nivel Hardware, Software, Telecomunicaciones, administración de proyectos, marketing, análisis de consumidor, prototipado y posterior encapsulamiento de la solución. Es por lo anterior que el equipo está compuesto

por cuatro integrantes.

1.1.1. Felipe Cordero

Estudiante de último año de la carrera Ingeniería Civil Electrónica con Mención en Computadores. Ha trabajado en empresas de desarrollo de hardware embebido, tiene un gran interés por crear un emprendimiento y seguir el camino de desarrollo de hardware y software. Su interés en el desafío radica en participar de un proyecto que posee todas las fases de desarrollo de hardware con un cliente desde cero. Al estar relacionado con el área de salud y conectividad permite aportar directamente a mejorar el sistema de salud pública en Chile.

1.1.2. Vanessa Muñoz

Estudiante 5to año de Ingeniería Comercial, 25 años. Colaborado en actividades dentro de la universidad como Preusm y actualmente trabajando por tercer año en la Feria de Empresas y Trabajo USM desempeñándose como Coordinadora General. La principal motivación por escoger este desafío es poder intervenir y mejorar algún área del sistema de la salud pública Chilena, dado que se ha podido presenciar la inefficiencia del servicio en distintas ocasiones. Decide abandonar el grupo por no cumplir los objetivos buscados para su trabajo de tesis.

1.1.3. Patricio Rodríguez

Estudiante de último año en la carrera de Ingeniería Civil Telemática. Ha contribuido en distintos proyectos relacionados a procesamiento de imagen, análisis de redes, simulación, programación, entre otros. Se destaca por su gran motivación y tenacidad a la hora de desempeñar sus tareas, aportando al trabajo en equipo y facilitando la resolución de tareas. Su interés en el desafío recae en la necesidad de conectividad que este conlleva, además de estar ligado al área de la Salud. Área de especial interés considerando la distancia profesional que se puede alcanzar estudiando una carrera de

Ingeniería.

1.1.4. Sebastián Castillo

Estudiante de último año en la carrera de Ingeniería en Diseño de Productos. Participado en actividades relacionadas al voluntariado, desarrollo de proyectos tecnológicos y conservación de la naturaleza. Se perfila como un profesional versátil, comprometido y que considera el trabajo multidisciplinario como fundamental en el desarrollo de soluciones para el mundo actual. El interés en este proyecto se debe a la posibilidad de poder impactar positivamente en la vida de gente con necesidades reales y mejorar, en cierta medida, su calidad de vida a través de la ingeniería, que muchas veces olvida el rol social que puede ejercer

1.2. Análisis del desafío

La empresa Sistemas Expertos ha planteado el desafío: ¿Cómo podemos incorporar a bajo costo telemedicina a la salud pública, considerando restricciones económicas y geográficas?. En donde se da cuenta de la necesidad actual de aplicar las tecnologías existentes en el ámbito de salud, permitiendo de esta forma mejorar la atención. Para conseguir este objetivo se espera el desarrollo de un dispositivo electrónico con capacidad de toma de datos y envío de los mismos. Así, se pueden identificar distintas aristas a considerar, como lo son: Tipo de enfermedades y pacientes a cubrir, tipo de sensores a emplear, tipo de tecnología de comunicación, nivel de interacción con el usuario, entre otros.

Con esto en mente, se debe tomar una decisión con respecto a las enfermedades a medir ya que esto está ligado íntimamente a los sensores a utilizar pudiéndose encontrar entre ellos: electrocardiograma, saturometro, medidor de presión, termómetro, entre otros.

Además de lo anterior, para realizar la comunicación de estos datos de forma remota se contemplan distintas alternativas, entre las que se considera utilizar la infra-

estructura ya presente e implementada en el país, como lo son las antenas celulares conectadas directamente con el dispositivo y también utilizar conexión a internet con un intermediario como un smartphone mediante una conexión bluetooth.

Por último, respecto al nivel de interacción con el usuario, la empresa ha dejado expresa su necesidad de simplicidad en este desarrollo, descartando cualquier interfaz o comunicación directa entre el usuario final y el dispositivo. Si bien dependiendo de la tecnología a emplear esta sugerencia puede cambiar, en una primera instancia se mantiene esta línea de pensamiento en torno al desarrollo completo, intentando así mantener la sencillez en las distintas partes del dispositivo. Permitiendo de este modo reducir los datos a manipular, las interfaces a desarrollar y el riesgo de un mal uso por parte de los usuarios.

1.3. Estado del Arte

En el marco del desarrollo del desafío de Sistemas Expertos, se planteó generar un dispositivo de monitoreo a distancia de pacientes. Para esto se comenzó a estudiar aspectos relacionados con la Telemedicina y sus implicancias en el avance del monitoreo Remoto de Paciente (RPM, por sus siglas en inglés). La Telemedicina es, en principio, la tecnología que permite entregar cuidados médicos a través de la infraestructura de las telecomunicaciones, permitiendo a los médicos diagnosticar o evaluar enfermedades sin la necesidad de un control presencial. Para poder comprender en qué se encuentra la realidad nacional y latinoamericana es de suma importancia revisar algunos casos dónde se apliquen dispositivos de telemedicina bajo la modalidad de monitorear y digitalizar la información, considerando que el objetivo del proyecto se limita a esas dos acciones.

1.3.1. ViSi Mobile®

ViSi Mobile® [1], si bien se utiliza en el cuerpo, es una estación que procesa los datos de otros sensores que van colocados en el cuerpo y que a su vez se conectan al módulo central de procesamiento como se puede observar en la imagen 1 lo que es necesario categorizarlo como un producto modular. Los sensores se encargan de medir pulso, respiración, SpO₂, presión sanguínea continua no invasiva y temperatura de la piel. El principal objetivo es permitir monitorear al paciente de forma continua dentro del hospital, sin intervenir de manera negativa en el flujo de trabajo que allí existe (ViSi Mobile® System, s. f.). ViSi Mobile® se encarga de recopilar los datos que cada sensor pueda otorgar para luego enviarlos de manera simultánea a un smartphone, una plataforma online de monitoreo y además directo a la estación de trabajo del médico a cargo, permitiendo así una atención eficiente.



Fig. 1.1: Interfaz de usuario ViSi Mobile®

Se puede observar en la figura 1.1 la interfaz que puede ver el paciente al utilizar el dispositivo.



Fig. 1.2: Modo de uso ViSi Mobile®

En la imagen de la figura 1.2 se puede observar los sensores conectados al cuerpo que convergen al dispositivo que toma las señales.

1.3.2. Qardiocore

QardioCore[2] es un monitor de electrocardiograma inalámbrico diseñado para mejorar la detección y manejo de las condiciones cardíacas. Seis sensores se encargan de grabar y analizar sobre 20 millones de puntos de datos durante todo el día junto con otros signos vitales. Este dispositivo está orientado a personas con alto nivel de riesgo cardíaco causado por predisposición familiar, historial de ataques al corazón, presión alta, colesterol alto, diabetes o exceso de peso. Monitorea de forma precisa y continua la salud del corazón. El dispositivo graba datos de ECG, pulso, variación de pulso, temperatura corporal, ritmo respiratorio y niveles de estrés. A diferencia de los ECG tradicionales, QardioCore no utiliza gel ni cables para monitorear y funciona entre -20°C y 60°C. Adicionalmente es resistente al agua y su batería dura alrededor de un día.



Fig. 1.3: Qardiocore multisensor

Se puede observar el dispositivo Qardiocore en la figura 1.3 que se conecta a un smartphone para mostrar los datos que se están tomando. Además como se muestra en la figura 1.4 es de simple uso, funciona como un cinturón en el pecho del paciente.



Fig. 1.4: Modo de uso Qardiocore

1.3.3. Nuubo

Nuubo[3] proporciona una nueva perspectiva en la monitorización cardiológica remota e inalámbrica. La plataforma de Nuubo, nECG platform, permite la captura del ECG dinámico a través de un innovador sistema que está basado en textiles biomédicos de nueva generación, y es rentable, remoto, continuo y no invasivo. Además, puede ser utilizado simultáneamente con uno o varios pacientes. La tecnología de electrodos textiles desarrollada por Nuubo simplifica enormemente los incómodos procedimientos tradicionales de conexión de electrodos, reduciéndolos al sencillo acto de vestir la camiseta nECG SHIRT que se muestra en la figura 1.5.



Fig. 1.5: nECG Shirt

El tejido elástico se adapta a los movimientos del paciente, quien puede realizar su actividad física diaria sin estar limitado por cables y sin necesidad de depender de personal médico especializado. Estas características junto con la información de contexto, la actividad física del paciente y su posición/postura, permite el desarrollo de un nuevo rango de soluciones y casos de uso.

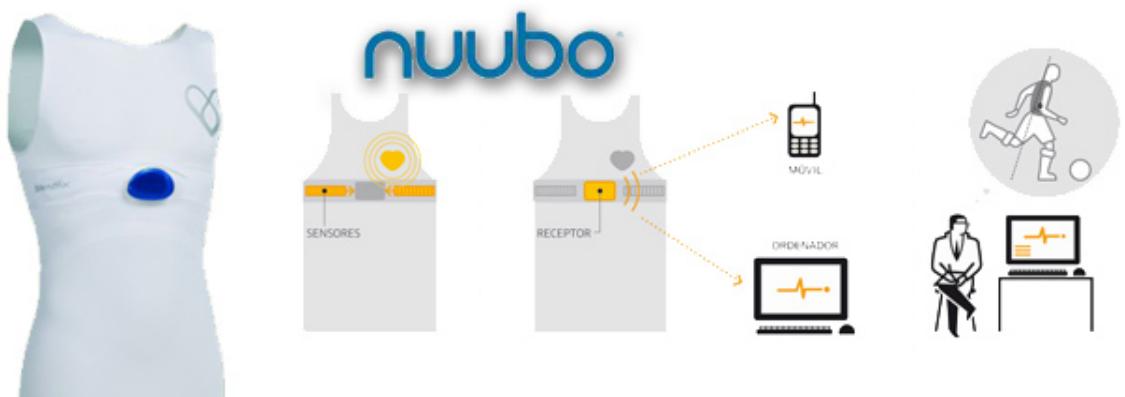


Fig. 1.6: Sistema Nuubo

Como se puede observar en la figura 1.6 la polera toma los datos que son enviados a un dispositivo móvil o un computador para que sea visto por el doctor de manera remota.

2. ALTERNATIVAS DE DESARROLLO

En el presente capítulo se ahondara en las distintas alternativas de diseño que existen para el prototipo con los distintos sensores requeridos además de establecer la comunicación y el envío de la información tomada del paciente. Las etapas para el desarrollo del prototipo constan de: Elección de sistema de procesamiento o unidad central, sensores a utilizar y forma de comunicación inalámbrica.

2.1. *Plataforma de desarrollo*

Al fabricar un prototipo, el desarrollador debe construir el hardware sobre el cual correrá el software del producto que ha diseñado, por lo que debe tomar componentes de diversos proveedores, integrarlos y hacerlos funcionar como un conjunto. Por esa razón se popularizó el uso de plataformas de desarrollo electrónico.

Por lo general, estas son placas que integran microcontroladores, circuitos y componentes electrónicos que le proporcionan diversas capacidades básicas y a partir de esto se puede evaluar la compatibilidad del diseño tanto en hardware como en software antes de enviar a fabricar el producto final.

2.1.1. Arduino

Arduino es una plataforma de desarrollo de bajo costo que permite crear proyectos de base tecnológica de forma sencilla y barata, que consta de entradas análogas, entradas y salidas digitales, PWM, comunicación serial, etc.

Uno de los beneficios de Arduino es que provee módulos de desarrollo de bajo costo para trabajar con integrados y estudiar su funcionamiento y prototipado. Arduino trabaja con una gran variedad microcontroladores AVR que diferencia por modelos dependiendo de las necesidades de proyecto, motivo por el cual varía en precio.

En primera instancia se puede trabajar con un modelo Arduino UNO que es de bajo costo y permite leer señales análogas y traducirlas en su conversor análogo-digital y dependiendo de las necesidades se puede conseguir otro modelo como Arduino Mega que ofrece mayores prestaciones.

2.1.2. Raspberry

Raspberry es una computadora de placa reducida (SBC por sus siglas en inglés) de bajo costo, con el objetivo de estimular la enseñanza de ciencias de la computación, no obstante, es de propiedad registrada para poder mantener el control de la 8 plataforma y no se generan excesivas variantes como es el caso de Arduino. El software que usa es open source, aunque es capaz de ejecutar incluso una versión de Windows 10. Por lo mismo su capacidad de procesar señales es mayor y permite ejecutar proyectos más complejos. No se define si es que pueden o no ser usadas en desarrollos comerciales.

2.1.3. Beaglebone

Beaglebone black es la última iteración de la serie Beaglebone y su versión pequeña. Esencialmente es similar a Raspberry, diferenciándose en cosas como la capacidad para iniciarse sin la necesidad de instalar ningún sistema operativo ya que tiene memoria integrada, no así Raspberry. Adicionalmente cuenta con una cantidad de entradas sustancialmente mayor, por lo que permite hasta el doble de conexiones que su competencia directa. Como si es no fuera suficiente, la arquitectura del procesador que incluye Beaglebone black permite que rinda hasta el doble de rápido que su contraparte en Raspberry pi.

Al igual que su competencia, Beaglebone ofrece mucho mas procesamiento que el necesario por lo que se descarta como una opción para el desarrollo inicial del prototipo, de acuerdo a las necesidades que vayan surgiendo se puede considerar nuevamente como una opción.

2.2. Sensores

Hablando con la contraparte de Sistemas Expertos se decidió, a partir de la información que proveen ellos, que las enfermedades mas comunes son las afecciones cardíacas y también es necesario tener un control de la temperatura de los pacientes a la hora de leer sus signos vitales.

Por otra parte se propuso utilizar una IMU para detectar si algún paciente sufre una caída, este con el fin de emitir una alarma para llamar una ambulancia en caso de ser necesario.

2.2.1. ECG

Electrocardiograma o ECG es el proceso de registrar la actividad eléctrica del corazón en un periodo de tiempo usando electrodos directamente en la piel.

Lo fundamental será buscar un circuito de desarrollo para realizar una prueba de concepto, en la cual se puedan tomar los datos y manejar.

DFRobot Heart Rate Monitor Sensor

El monitor de actividad cardiaca de la empresa DFRobot se usa para medir la actividad eléctrica del corazón con un integrado AD8232[4] que toma señales análogas de los electrodos y utiliza amplificadores para tener una mejor lectura de los datos.

Utilizando un Arduino es posible leer los datos tomados de los electrodos y convertirlos a información digital que puede ser enviada por comunicación serial.

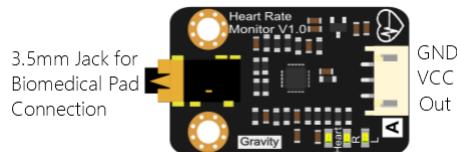


Fig. 2.1: Placa de desarrollo ECG

Como se puede observar en la figura 2.1 posee conexión simple para electrodos y salida análoga, lo que permitirá una rápida prueba de concepto para utilizar este integrado en el diseño del dispositivo final. Además este provee filtros que se van a estudiar mas adelante.

ADS1298

El integrado ADS1298 de la empresa Texas Instrument ofrece un ECG con 8 amplificadores programables de bajo ruido y 8 conversores Análogo-digital de alta resolución.

Utilizado para instrumentación medica y lectura tanto de ECG como EMG (Electromiograma) y EEG (Electroencefalograma).

El integrado ADS1298 es una buena opción para un desarrollo de ECG en el futuro de grado médico, pero es de un precio 10 veces mayor al dispositivo de DFRobot por lo que se va a descartar para el prototipo funcional.

2.2.2. Temperatura

Cuando se requiere realizar alguna medición a un paciente siempre es necesario conocer su temperatura corporal que sirve como información complementaria a los profesionales de la salud es por esto que se evaluarán termistores que permitan la lectura de este dato.

Lilypad Temperature Sensor

Dentro de la tendencia del hardware abierto, uno de los proyectos más destacados es Lilypad Arduino, un conjunto de piezas electrónicas que se pueden coser a los tejidos para darles interactividad con sensores, luces o sonidos.

Entre estos sensores tenemos un sensor de temperatura compuesto por un termistor MCP9700 el cual ofrece una resolución de $\pm 2^{\circ}C$.

La particularidades que ofrece este sensor es ser de muy bajo costo y a su vez es impermeable por lo que permitiría incorporarlo en el wearable de forma permanente sin dañar la componente.

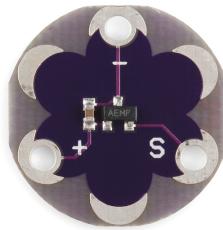


Fig. 2.2: Sensor de temperatura Lilypad

Como se puede observar en la figura 2.2, Lilypad ofrece una PCB impermeable con 3 terminales que permiten utilizar un hilo conductor para coser este a la ropa.

DS18B20

El sensor DS18B20[5] es un termómetro digital que ofrece una medida de 9 a 12 bits de resolución. Se comunica mediante el bus 1-Wire (protocolo de comunicación en serie diseñado por Dallas Semiconductor el cual está basado en un maestro y varios esclavos en una sola linea de datos) lo cual permitiría, en caso de ser necesario, incorporar mas sensores para obtener una medida con mayor precisión. Este termómetro digital ofrece una resolución de $\pm 0,5^{\circ}C$ y a su vez ofrece un formato impermeable en forma cilíndrica como se observa en la figura 2.3.



Fig. 2.3: Sensor de temperatura DS18B20

2.2.3. Ritmo Respiratorio

Para medir el ritmo respiratorio, sensor que la contraparte pidió estudiar utilizando una tela conductora, se consideró el uso de la tela conductiva MedTex, la cual entrega un valor de resistividad en ohms en su estado en reposo y este varía dependiendo de su estiramiento.



Fig. 2.4: Tela Conductiva MedTex

Para estudiar la factibilidad de la tela conductiva que se puede observar en la figura 2.4 se cortó una tira de un tamaño $20x2[cm]$ en estiramiento cero sobre una banda elástica que luego fue cosida como cinturón de pecho. Una vez colocada en cada extremo de la tela conductiva se colocó un caimán conectado a su vez a un multímetro que permitía visualizar variaciones de la resistividad de la tela a partir de su estiramiento.

Resistencia en reposo [Ω]	Resistencia en estiramiento [Ω]	% de variación)
4,8	4,6	0,1420
4,7	4,5	0,1421
4,8	4,7	0,0722

Tab. 2.1: Valores resistencia Tela MedTex en Pectorales

Resistencia en reposo [Ω]	Resistencia en estiramiento [Ω]	% de variación)
4,7	4,6	0,0699
4,7	4,6	0,0699
4,6	4,5	0,0723

Tab. 2.2: Valores resistencia Tela MedTex en Plexo

Resistencia en reposo [Ω]	Resistencia en estiramiento [Ω]	% de variación)
4,7	4,6	0,0699
4,8	4,7	0,0722
4,7	4,5	0,1421

Tab. 2.3: Valores resistencia Tela MedTex en Estomago

Se puede observar en las tablas 2.1, 2.2 y 2.3 las variaciones de resistencias no son constantes ni regulares, el mismo estiramiento a veces no producía la misma variaciones de resistencia. Además por mínimas variaciones en el movimiento también habían variaciones que arruinaban la medición, por lo que esta alternativa no sería viable para medir el ritmo respiratorio.

2.3. Unidad de movimiento inercial (IMU)

Una unidad de movimiento inercial o IMU (del inglés inertial measurement unit), es un dispositivo electrónico que mide la aceleración, inclinación y las fuerzas gravitacionales, usando una combinación de acelerómetros y giroscopios.

2.3.1. MPU-9250

El integrado MPU-9250 es un modulo multi-chip que consiste en 2 integrados en un empaquetado QFN. Este provee un giroscopio de 3 ejes y un acelerómetro de 3 ejes. Este chip provee tres conversores análogo-digital de 16 bits para digitalizar las salidas del giroscopio, acelerómetro y giroscopio de manera independiente.

Sparkfun provee una PCB de desarrollo para realizar pruebas como se muestra en la imagen 2.5.



Fig. 2.5: IMU Sparkfun MPU-9250

Es importante destacar la orientación indicada por el fabricante al momento de diseñar el equipo electrónico que son predefinidas como se puede ver en el caso de la figura 2.5 en la cual se muestran los ejes X, Y y Z tanto para el acelerómetro como para el giroscopio.

Como se observa en la figura 2.6 se muestra además de los ejes de aceleración también las coordenadas de navegación (roll, pitch, yaw).

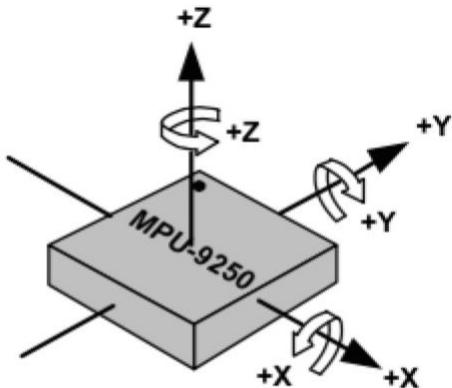


Fig. 2.6: Ejes IMU MPU-9250

2.4. Comunicación

Para la comunicación se han considerado 3 opciones viables, las cuales incluyen redes celulares (GPRS), WiFi y bluetooth.

2.4.1. GPRS shield

Para integrar conexión a redes celulares en el dispositivo es necesario considerar un GPRS shield compatible con socket xbee.

GPRSbee cumple con los requerimientos a un precio no menor (aproximadamente 36.000 CLP).

Se puede observar en la figura 2.7 el módulo disponible para desarrollo.

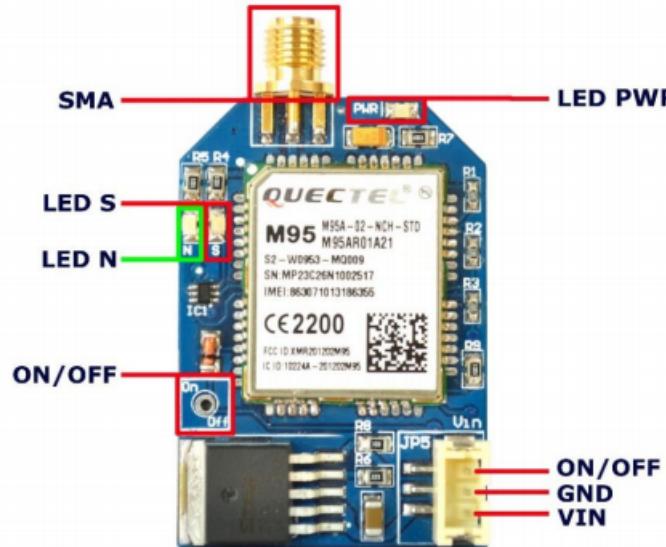


Fig. 2.7: Modulo GPRSbee

Cabe destacar que para poder utilizar este módulo es necesario incluir una antena que se puede ver en la figura 2.8



Fig. 2.8: Antena GPRSbee

Al considerar este módulo se puede concluir que es incompatible con el diseño del wearable ya que la antena es muy grande (aproximadamente 57,40[mm]) lo que sería molesto en el dispositivo final. Otro punto en contra de este módulo es el alto costo y el consumo energía que lo hace incompatible con la autonomía que se desea.

2.4.2. Bluetooth BLE shield

Para integrar bluetooth en el dispositivo se considera un BLEBee el cual ofrece bluetooth versión 4 y comunicación UART mediante un puerto XBEE.

El shield Bluetooth posee un módulo RN4020 el cual ofrece una antena para la comunicación en su misma placa lo que facilita el diseño como se puede observar en la figura 2.9.



Fig. 2.9: Bluetooth RN4020

Es importante destacar que para mejorar el diseño, el fabricante recomienda dejar expuesta la antena para mejorar la comunicación pero esto se va a explicar en otra sección.

2.5. Conclusiones

En esta sección, tomando en cuenta las opciones vistas en el mismo capítulo, se seleccionará las primeras componentes a utilizar para el prototipo funcional y para realizar la prueba de concepto con lo que se va a basar el diseño.

2.5.1. Plataforma de desarrollo

Para la plataforma de desarrollo se va a escoger trabajar con Arduino ya que este posee distintas versiones con distintos costos, los cuales son menores que Raspberry o Beaglebone. Además cabe destacar que el sistema que se quiere desarrollar es toma de datos y envío de información por lo que no se va a requerir tanto procesamiento. Arduino cubre las necesidades en su versión UNO con un microcontrolador ATmega328p, en caso de necesitar uno de mayor capacidad se puede optar por un Arduino Mega.

2.5.2. Electrocardiograma

Para el sensor de electrocardiograma se utilizará el monitor de actividad cardíaca de DFRobot, esto debido a que es la única opción que se puede conseguir en el país para no retrasar el desarrollo. Este sensor es de muy bajo costo (alrededor de 19.500 CLP en MCIElectronics). El integrado ADS1298 es una buena opción como una mejora para una segunda iteración del diseño para mejorar la señal que se puede obtener debido a que este posee mayor tolerancia al ruido. Se debe destacar esta última opción debido a que se debe encargar directamente desde Texas Instruments y esto puede tomar mucho tiempo.

2.5.3. Temperatura

En primera instancia se va a utilizar el sensor Lilypad ya que este está diseñado específicamente para wearables además de que utiliza un hilo conductor para unir sus terminales con la alimentación y la toma de datos. Este sensor tiene un valor aproximado de 3.790 CLP.

Dependiendo de los resultados obtenidos en la primeras pruebas se va a evaluar la segunda alternativa de utilizar el DS18B20 el cual tiene un valor aproximado de 5.900 CLP.

2.5.4. *IMU*

Al buscar las alternativas que existen en el país, todas las opciones de desarrollo usan distintas placas de desarrollo pero utilizan el mismo sensor MPU-9250 por lo que se va a utilizar la placa de Sparkfun MPU-9250 luego de tener el prototipo funcional con los primeros sensores de electrocardiograma y temperatura. Esta placa de desarrollo tiene un valor aproximado de 12.500 CLP.

3. SELECCIÓN DEL MICROCONTROLADOR

Se determinó utilizar la plataforma Arduino por su simplicidad en prototipado y programación, además de ser de fácil acceso y una tecnología escalable.

3.1. *ATMega328p*

Utilizando una placa Arduino Uno se comienza el prototipo del dispositivo que va a cumplir la función de tomar los datos de los sensores análogos. La versión de Arduino Uno utiliza un microcontrolador ATMega328 que posee 32[KB] de memoria de programa y una memoria RAM de 2[Mb]. Este consta de 1 puerto de comunicación UART y posee 32 pines de los cuales 23 pines son programables para entrada/salida como se muestra en la figura 3.1

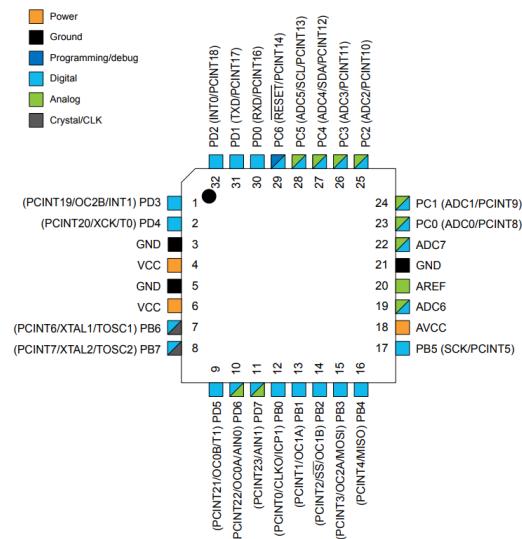


Fig. 3.1: Microcontrolador ATMega328p

3.2. ATMega2560

Contrastando el modelo mostrado anteriormente, la placa Arduino Mega posee un microcontrolador ATMega2560 el cual incluye un aumento en todas sus capacidades. Comenzando con la capacidad de su memoria de programa, que asciende a 250[KB] y una memoria RAM de 8[KB]. Este microcontrolador posee 100 pines lo que incluye un aumento a 4 puertos de comunicación UART, además de aumento de entradas análogas y digitales.

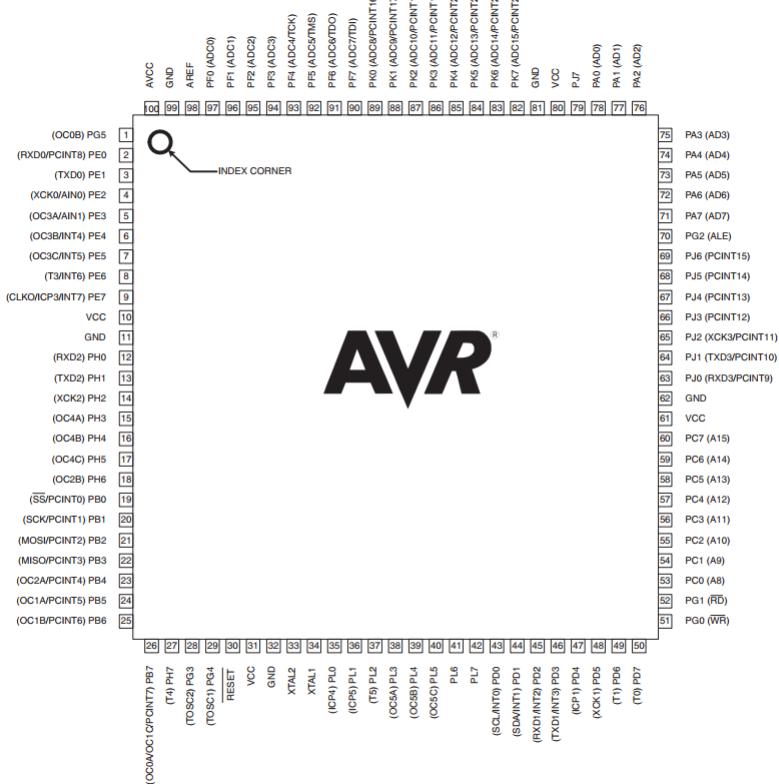


Fig. 3.2: Microcontrolador ATMega2560

Como se puede observar en la figura 3.2, hay un aumento considerable en las prestaciones que ofrece este microcontrolador con respecto al ATMega328p además de su tamaño.

Se va a trabajar en el diseño con este microcontrolador ya que posee 4 interfaces UART de las cuales una de estas siempre está definida para ser utilizada por la programación FTDI, esto se explicará mas adelante.

Esto involucra utilizar un chip con mayores capacidades de las necesarias pero esto es debido a la limitante de los microcontroladores utilizados por Arduino como se puede observar en la figura 3.3.

Arduino Board	Family	Clock	UART	PWM	Digital	Analog	VCC
Duemilanove (328)	ATmega328	16MHz	1	6	14	6	5V
Uno	ATmega328	16MHz	1	6	14	6	5V
Arduino Mega 2560	ATmega2560	16MHz	4	14	54	16	5V
Arduino Mega ADK	ATmega2560	16MHz	4	14	50	16	5V
Arduino Ethernet	ATmega328	16MHz	1	4	9	6	5V
Arduino BT	ATmega328	16MHz	1	6	14	6	5.5V
Arduino Pro Mini 328 5V	ATmega328	16MHz	1	6	14	6	5V
Arduino Nano 3.0	ATmega328	16MHz	1	6	14	8	5V
Arduino Mini	ATmega328	16MHz	1	6	14	8	5V
Arduino Pro 3.3V	ATmega328P	8MHz	1	6	14	6	3.3V
Arduino Pro 5V	ATmega328P	16MHz	1	6	14	6	5V
Arduino Fio	ATmega328P	8MHz	1	6	14	8	3.3V
LilyPad Simple Board	ATmega168P	8MHz	1	5	9	4	2.7-5.5V
LilyPad 328 Main Board	ATmega328P	8MHz	1	6	14	6	2.7-5.5V

Fig. 3.3: Comparación microcontroladores Arduino

Idealmente para el diseño del dispositivo, se utilizará un microcontrolador con 2 puertos UART ya que uno de estos es necesario para la comunicación con el computador y el otro es utilizado para el envío de información mediante bluetooth, pero por necesidad de rapidez para el diseño de prototipo y pruebas de sensores es necesario iterar con una placa ya fabricada para usos generales y disponibles en el mercado. Esto causaría una disminución significativa entre el precio del diseño del producto final.

3.3. ATMega644PA

Considerando la variable de costos es importante destacar una tercera opción para el diseño del dispositivo final. El microcontrolador ATMega644PA posee un menor conteo de pines ya que consta de 44 lo cual disminuye costos y tamaño a la hora del diseño. Con una memoria Flash de 64[KB] y memoria RAM de 4[KB]. Este provee 2 UART para la comunicación y los pines suficientes para sensores como se muestra en la figura 3.4.

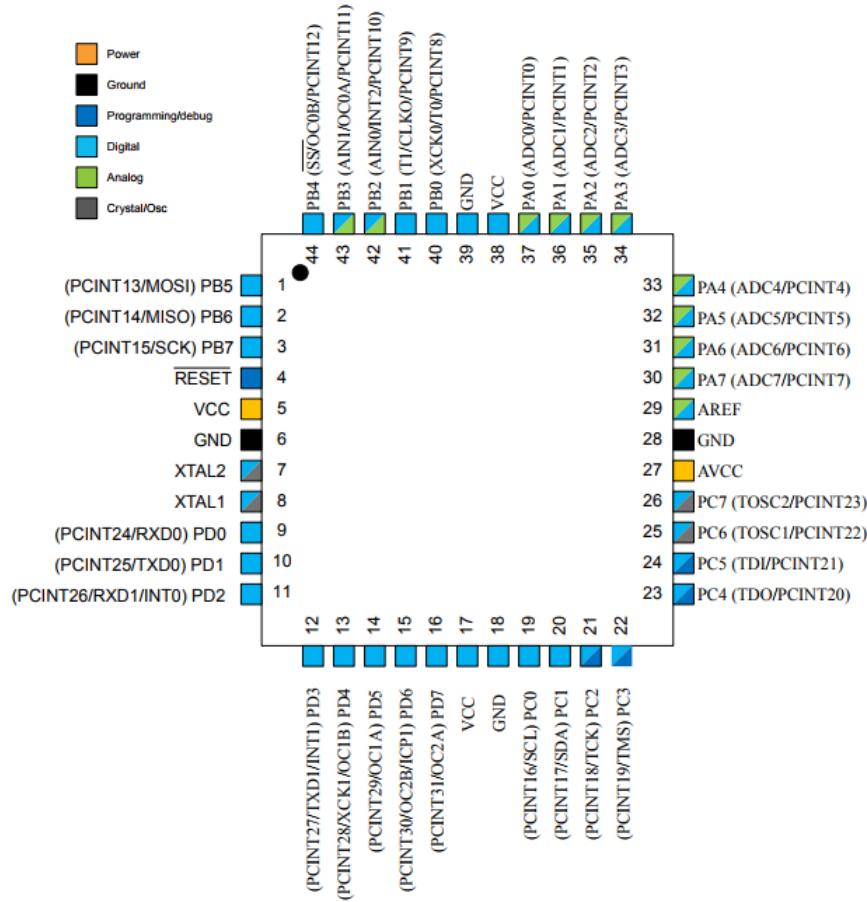


Fig. 3.4: Microcontrolador ATMega644PA

Se puede observar en la imagen la disponibilidad de los puertos UART donde el Pin 9 y Pin 10 corresponden a UART0 considerado para la programación USB. Pin

11 y Pin 12 corresponden a UART1 que será utilizado para la comunicación con el dispositivo bluetooth.

En primera instancia se va a diseñar el dispositivo con el microcontrolador ATmega2560, debido a que existen modelos Arduino con esta misma componente y cumple los requisitos funcionales de poseer mas de 1 puerto UART. No es la opción ideal ya que posee mas prestaciones de las que se necesitan pero permite que se pueda prototipar inmediatamente utilizando un Arduino equivalente, lo cual no se puede hacer con la versión ATMega644PA.

3.4. Conclusiones

Finalmente considerando las 3 opciones elegidas, se puede establecer una decisión final con respecto a cual microcontrolador utilizar.

Microcontrolador	Pines	Precio (USD)
ATMega328p	32	\$2.18
ATMega2560	100	\$12.44
ATMega644PA	44	\$5.24

Tab. 3.1: Comparación microcontroladores

Como se puede observar en la Tabla 1 se tomaron en cuenta los aspectos de tamaño y costos. ATMega328p se considera porque a pesar de que posee un solo puerto UART es posible utilizar otros pines para configurar comunicación serial por software (en comparación con la comunicación por hardware es mucho más lenta) lo cual será muy útil a la hora de hacer pruebas en toma de datos y envío de la información. En cuanto a precio y tamaño es una alternativa ideal ya que minimiza ambos aspectos.

La segunda opción siendo la que ofrece más opciones para sensores y memoria programable es excesivo para lo que se necesita desarrollar, pero se acerca más a los requerimientos necesarios que carece el microcontrolador ATMega328p. Esta opción tiene un el precio más elevado de las 3 opciones y por una diferencia considerable, es

por esto que se utilizará para la primera versión del diseño final enfocado a tener un mínimo producto viable que funcione de la misma manera que el prototipo.

Finalmente la tercera opción siendo la más viable ya que posee un tamaño mucho menor al ATMega2560 y un precio menor a la mitad que este, ofrece todas las capacidades necesarias pero como se muestra en la figura 3.3, no hay en el mercado una placa Arduino con este microcontrolador ni tampoco otra opción que cumpla con el requisito de utilizar 2 UART es por esto que como segunda iteración del producto final se debe buscar un bootloader compatible con el entorno de desarrollo Arduino para el ATMega644PA o trabajar en otra plataforma compatible con ese microcontrolador.

4. FIRMWARE Y PROGRAMACIÓN DEL MICROCONTROLADOR

Los microcontroladores Arduino son usados en el diseño de sistemas embebidos para distintas funciones. El microcontrolador es un pequeño chip que posee pines con funciones de lectura y escritura, memoria, entradas y salidas. Mientras los microcontroladores han sido usados por décadas, los microcontroladores Arduino son utilizados actualmente ya que permiten ejecutar funciones electrónicas sin necesidad de conocer hardware y software integrado en estos.

4.1. *Bootloader*

Dentro de las múltiples definiciones que existen con respecto al bootloader, lo más común es considerarlo como un software o firmware que reside parcialmente en la memoria no volátil del microcontrolador, como la ROM o memoria Flash.

En la práctica el bootloader empieza a funcionar justo después de prender el microcontrolador o después de reiniciarlo.

Este bootloader va a ser el que va a permitir la programación del circuito con el entorno de desarrollo Arduino (Arduino IDE) manejando de mejor manera la toma de datos por medio de las entradas análogas, el procesamiento y la comunicación.

4.2. ISP - In-System Programming

También conocido como programación serial en circuito[6] (ICSP), es la habilidad de algunos dispositivos lógicos programables, microcontroladores y otros circuitos electrónicos de ser programados mientras están instalados en un sistema completo, no es necesario programar el chip antes de instalarlo en el sistema. Esto permite armar un circuito con todo lo que se desea y programarlo en el circuito impreso.

Típicamente los chips que soportan programación ISP tienen circuitería interna que genera el voltaje necesario y permite comunicarse con el programador a través de protocolo serial. La mayoría de los dispositivos lógicos programables usan una variante del protocolo JTAG para ISP para facilitar la integración con procedimientos automatizados de pruebas.

JTAG (Joint Test Action Group) es una interfaz diseñada originalmente para circuitos impresos y es muy útil también como mecanismo para depuración de aplicaciones embebidas, puesto que provee una puerta trasera para acceder al sistema. El módulo de depuración permite al programador corregir errores de código y de lógica de sus sistemas.

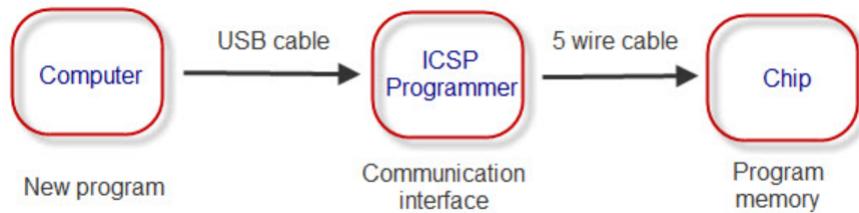


Fig. 4.1: Configuración ISP para programar un microcontrolador

Se observa en la Figura 4.1 un simple diagrama de flujo donde es importante destacar que para poder programar un chip es necesaria una conexión de 5 cables entre un intermediario que viene a ser el programador ICSP (o ISP) y el chip (Microcontrolador).

4.3. Programación de un microcontrolador

Para que el entorno de desarrollo Arduino reconozca el microcontrolador como una placa Arduino, es necesario grabar el bootloader en este. Se escribe el bootloader en la memoria del microcontrolador mediante la comunicación ISP que se muestra en la figura 4.2.

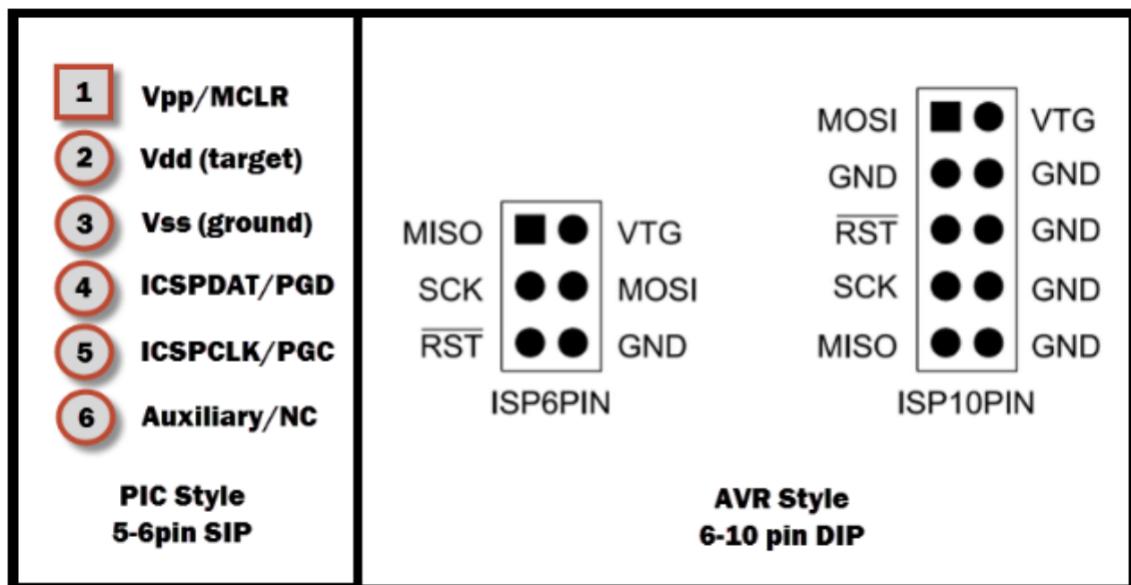


Fig. 4.2: Conector típico para programación ISP

La figura 4.2 muestra el conector ISP que se va a encargar de cargar el bootloader en el microcontrolador. En este caso se está utilizando la tecnología AVR (microcontrolador ATMega) la cual es usada por las placas Arduino y es por esto que se usará la configuración de 6 pines.

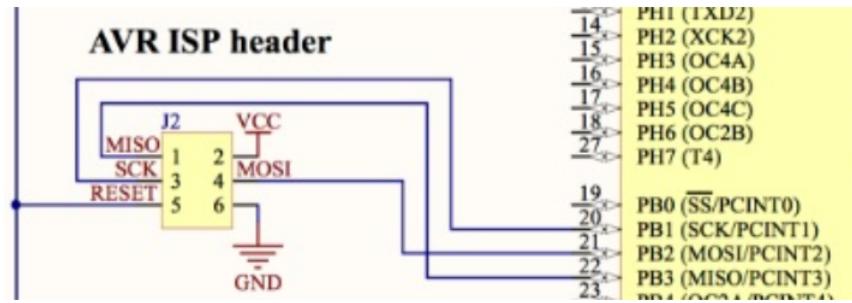


Fig. 4.3: Esquemático básico MCU con ISP

En la figura 4.3 se muestra una parte del esquemático con la configuración básica en el microcontrolador AVR. Se puede observar el diagrama de conexión del bus con el microcontrolador. Esto se puede observar con mayor detalle en la tabla 4.1

4.3.1. Programador ISP

Existen varias alternativas que cumplen esta función y dependen tanto del microcontrolador como del fabricante. Un microcontrolador AVR (Atmel) requiere de un programador STK500 con una interfaz serial RS232 (Existen otros programadores pero cumplen la misma función, el STK500 es el más utilizado y posee mayor compatibilidad). Para programar un microcontrolador de Microchip se requiere de un PICkit. Una segunda alternativa es utilizar un Arduino ya programada que cumpla la función del programador. Estas vienen con un puerto ISP el cual permite cargar el bootloader en el microcontrolador. Respetando la misma conexión que se muestra en la figura 4.3.

AVR ISP	Arduino UNO	ATMega2560
1	MISO	Pin 22
2	VCC	VCC
3	SCK	PIN 20
4	MOSI	Pin 21
5	RESET	Pin 30
6	GND	GND

Tab. 4.1: Programación ISP utilizando Arduino UNO

4.4. Programación USB

La compañía escocesa Future Technology Devices International (FTDI) está especializada en la tecnología USB (Universal Serial Bus). Esta ofrece chips encargados de transformar una conexión USB a un puerto UART y esto será fundamental a la hora de programar el diseño final con el programa en el entorno Arduino.

4.4.1. FT232RL

El Chip FT232RL[7] ofrece una conversión USB-UART y cuenta con 28 pines como se muestra en la figura 4.4.

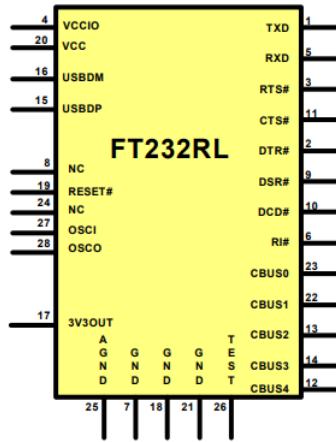


Fig. 4.4: Chip FT232RL

En la tabla 4.2 se muestra la descripción de los pines más importantes para la conexión del FTDI con el microcontrolador y al mismo tiempo con el USB para permitir la programación del chip.

Nº Pin	Nombre	Descripción
1	TXD	Transmisor de datos UART
5	RXD	Receptor de datos UART
4	VCCIO	1.8[V] a 5.25[V] alimentación a la interfaz UART y a los pines CBUS
15	USBDP	Conexión Data+ USB
16	USBDM	Conexión Data- USB
17	3V3OUT	Regulador de voltaje interno.
22	CBUS1	Indicador de funcionamiento RXD
23	CBUS2	Indicador de funcionamiento TXD

Tab. 4.2: Descripción de pines chip FT232RL

Este chip será encargado de conectar la interfaz UART (TX y RX) con el microcontrolador además de permitir la comunicación con la conexión USB (USBDP y USBDM).

Al trabajar con una conexión USB se tiene conexión con VCC y GND desde el computador que está programando y este provee una alimentación de 5[V]. Este voltaje es comúnmente utilizado en los microcontroladores pero además provee un regulador de voltaje interno de 3.3[V] el cual está disponible si se está trabajando con un sistema con menor voltaje. En caso de utilizar voltaje 3.3[V] para todo el sistema, es necesario conectar el pin 3V3OUT a VCCIO utilizando un condensador de 100[nF] como se indica en el manual. De esta forma la alimentación del microcontrolador es la misma que la alimentación FTDI para la programación UART y no se dañan las componentes.

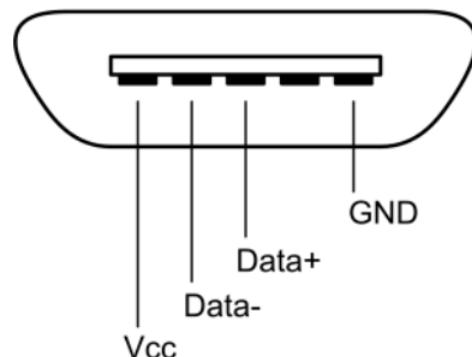


Fig. 4.5: Conector USB 2.0 Micro B

Se muestra en la figura 4.5 las 4 conexiones que posee un conector USB, las cuales son la alimentación VCC, GND y la conexión de datos Data+ (USBDP en FT232) y Data- (USBDM en FT232).

4.5. Programa en Arduino

Para la programación del Arduino se trabajó en conjunto con la parte telemática del grupo. Se realizó la programación de la toma de datos de los sensores. Además de un algoritmo antirebote para pedir toma de muestras en el prototipo que luego fue sustituido por una orden emitida por la aplicación. Este código se puede observar en el anexo. En esta parte se puede observar la división de las tareas en el grupo debido a que esta memoria está enfocada al área de diseño de hardware mayoritariamente por lo que este código no se explicará en mayor profundidad.

5. DISEÑO ELECTROCARDIOGRAMA

El ECG (Electrocardiograma) detecta señales del cuerpo gracias a electrodos que se colocan en la superficie del cuerpo, usualmente acompañados de un gel conductor que elimina interferencia de los músculos, fuente de alimentación, ruido externo, etc. Para que se obtenga una señal de ECG sin distorsiones excesivas, es necesario diseñar filtros que eliminan interferencias antes de analizar la información.

5.1. *DFRobot Heart Rate Monitor*

El monitor de actividad cardíaca de la empresa DFRobot consiste en una placa que consta de un chip AD8232 en su PCB, el cual provee una clara señal de los intervalos PR y QT (Ver Figura 5.1) de un electrocardiograma. Este entrega un valor análogo que puede ser leído por Arduino y con un conversor análogo-digital interpretarse en forma de gráfico.

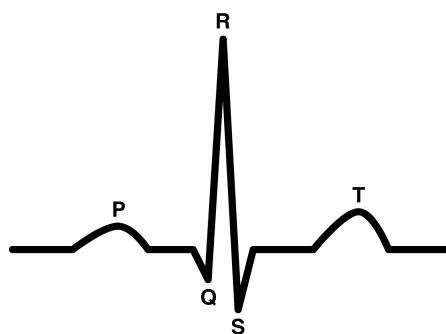


Fig. 5.1: Forma de onda teórica ECG

Se escogió esta placa de desarrollo por disponibilidad de hardware, ya que es la única opción disponible de bajo costo, y que no necesita comprarse fuera del país. Por lo que se tomó la decisión de estudiar su funcionamiento, replicarlo para el diseño del dispositivo final y evaluar posibles mejoras.

5.2. Biopac MP150 ECG100C

Se tuvo la oportunidad de usar el ECG de la empresa Biopac ECG100C, el cual es de mucho mayor costo y tamaño utilizado generalmente para investigación. La idea era ver la forma de onda y comparar las frecuencias de corte utilizadas. La figura 5.2 muestra el aparato encargado de tomar las señales de los electrodos y la figura 5.3 muestra un acercamiento con las posibles frecuencias de corte configurables para los filtros.



Fig. 5.2: Biopac MP150



Fig. 5.3: Frecuencias de corte disponibles en Biopac ECG100C

Cabe destacar el alto precio de este equipo que es de 45.000 USD y grandes dimensiones ya que está diseñado para investigación lo cual servirá para saber donde tiene que apuntar la forma de onda de un ECG de grado médico.

5.3. AD8232

El chip AD8232 es un integrado que permite medir las señales de los electrodos del ECG mediante amplificadores operacionales como se muestra en la figura 5.4.

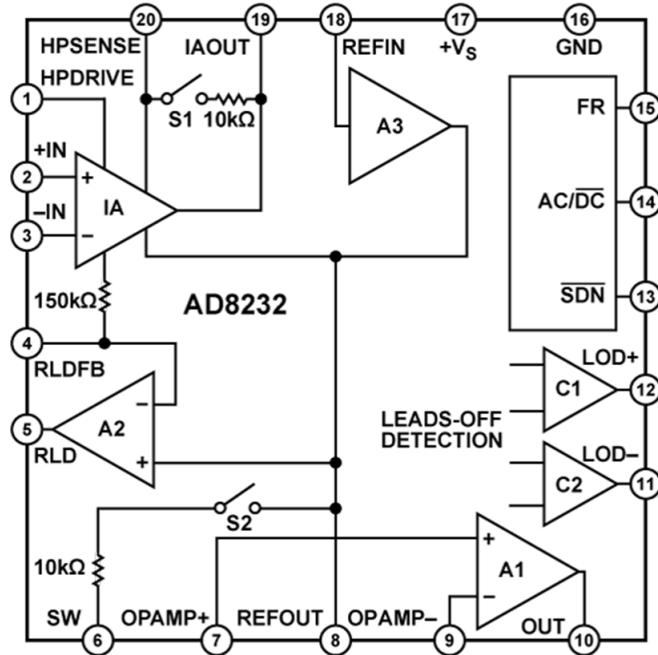


Fig. 5.4: Diagrama de bloques funcional

Esta figura representa el funcionamiento del integrado y el proceso que realiza en su interior desde el punto de vista de los pines. A partir de este diagrama y la herramienta que provee el fabricante para el diseño de filtros, se puede llegar a una configuración del integrado con componentes pasivas para el diseño del circuito impreso como se muestra en la figura 5.5.

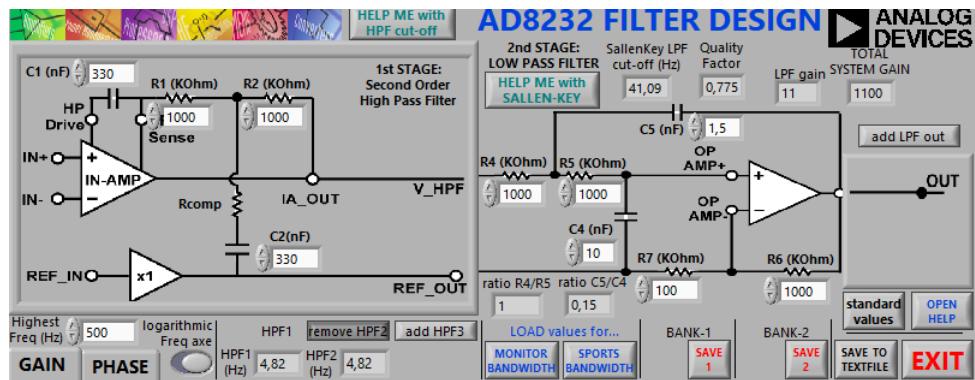


Fig. 5.5: Herramienta para diseño de filtros

Como se puede observar en la figura 5.5, en el lado izquierdo se diseña un filtro pasa alto de segundo orden cuya frecuencia de corte es de 4.82[Hz]. A la derecha se muestra un filtro pasabajo salien-key con frecuencia de corte 41.09[Hz] donde finalmente se tiene una salida hacia el Arduino. Estos valores se obtuvieron al ver el circuito de la PCB de DFRobot (Figura 5.6) y utilizando la herramienta de diseño de filtros. En caso de que se quisieran cambiar las frecuencias de corte se podría hacer un nuevo diseño utilizando el mismo programa pero cambiando y apuntando a frecuencias de corte entre 1-35[Hz] dadas por el dispositivo Biopac.

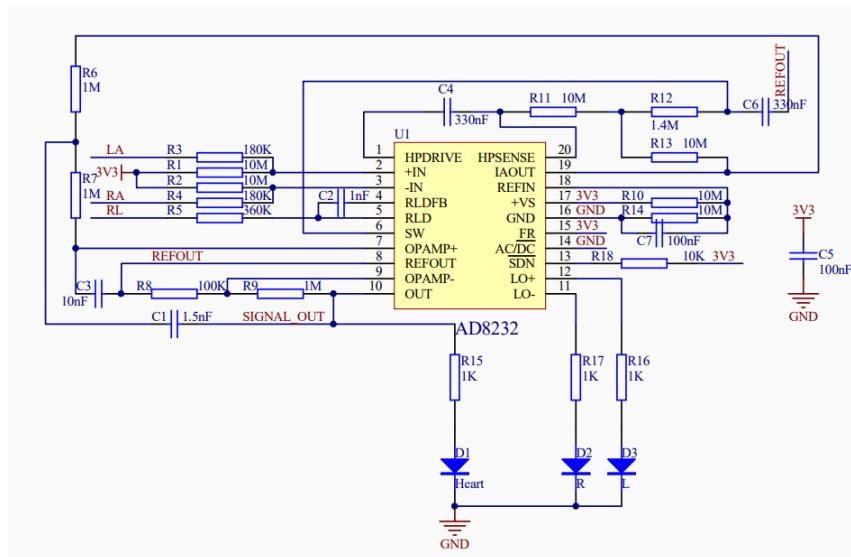


Fig. 5.6: Esquemático ECG DFRobot

5.4. Comparación entre Biopac y DFRobot

Finalmente se pudo contrastar ambas señales, se puede observar en la figura 5.7 la señal de ECG del Biopac y DFRobot colocando los electrodos muy cerca para obtener las señales lo más parecidas posibles y poder compararlas.

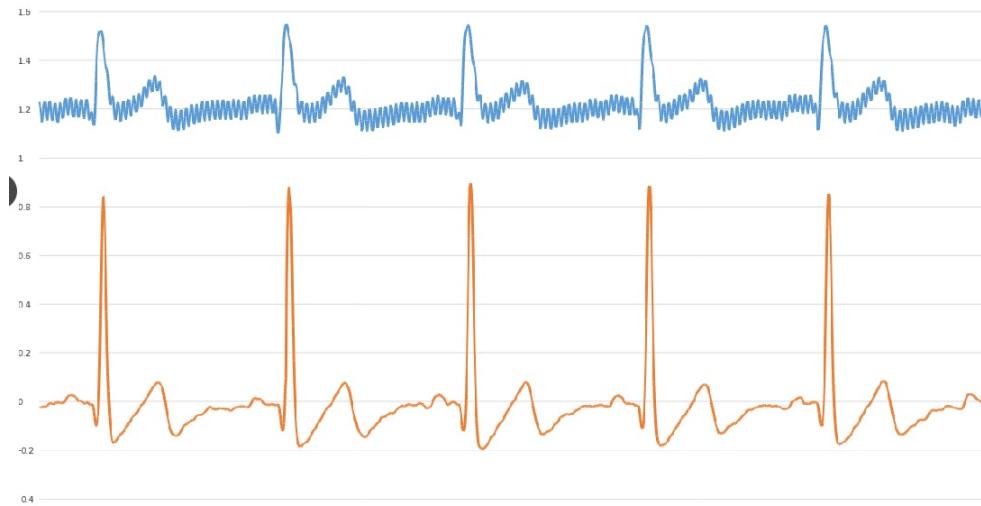


Fig. 5.7: Gráfico Biopac ECG100C vs ECG DFRobot

En el gráfico superior se puede observar la forma de onda del ECG AD8232 en la cual se puede observar notoriamente un ruido de la fuente. En el gráfico inferior se muestra la señal del ECG100C de Biopac, la cual es una señal muy definida a lo cual debería apuntar el diseño del ECG. Se puede concluir de estas imágenes que la forma de onda de ambas señales son parecidas, considerando que el Biopac está diseñado para mostrar la onda completa, en cambio el AD8232 se encarga de mostrar la onda PR y QT. Finalmente se considera que la forma de onda que se tiene sirve para hacer mediciones eliminando el ruido de la fuente ya que es representativa por los peaks que presenta.

5.5. Eliminando ruido de la fuente

Observando la señal que se tiene actualmente en el dispositivo mediante bluetooth en la aplicación móvil se obtiene una señal como se observa en la figura 5.8.

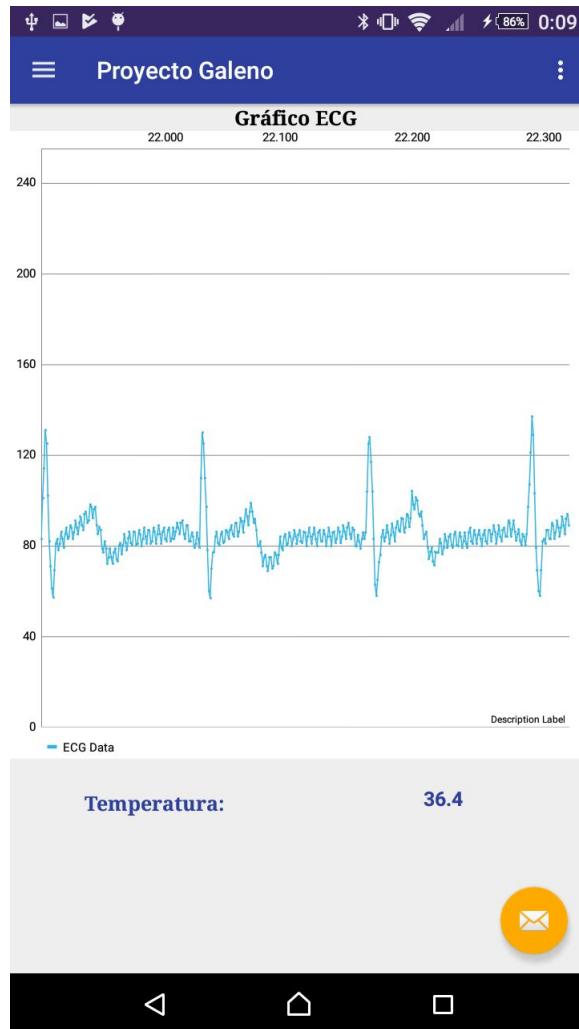


Fig. 5.8: ECG en aplicación móvil sin filtro de salida

Se puede notar en esta imagen que la forma de onda se mantiene al enviarla por bluetooth a la aplicación pero sigue con mucho ruido. Utilizando la herramienta de diseño de filtros para el AD8232 existe una opción para filtrar la salida de la señal final, para esto se considera utilizar un filtro pasabajo para eliminar el ruido como se muestra en la figura 5.9.

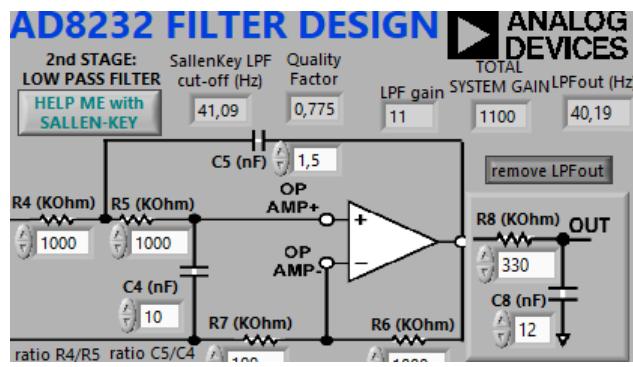


Fig. 5.9: Filtro de salida R8 y C8

Se diseñó con frecuencia de corte de $40,19[\text{Hz}]$ como se muestra en la figura 5.9 dando valores $R8 = 330[\text{k}\Omega]$ y $C8 = 12[\text{nF}]$ para no perder información de la señal y con esto limpiar solamente la señal como se muestra en el resultado de la figura 5.10

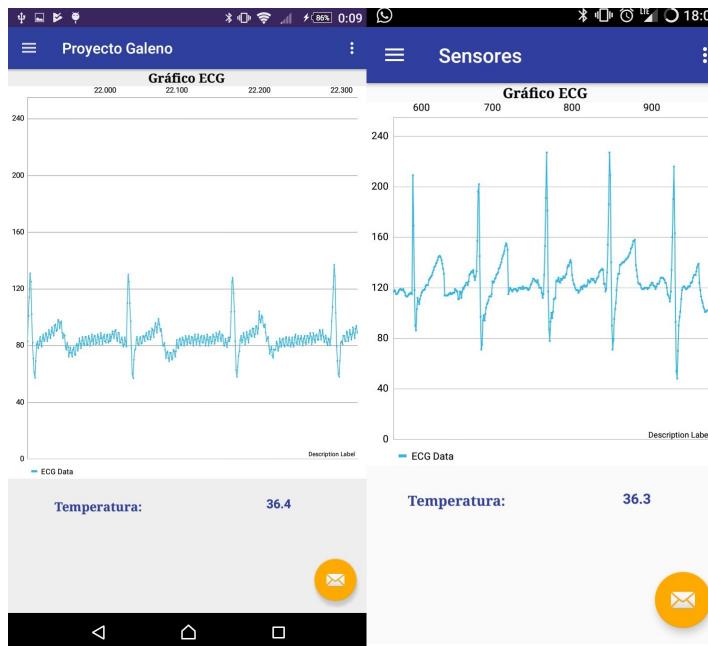


Fig. 5.10: Comparación ECG en aplicación móvil con filtros de salida

Se nota un gran cambio en la forma de la señal, donde se notan de mejor manera, sin ruido de la fuente y sin perder la forma de onda. Esta será la configuración que se utilizará para el diseño final del dispositivo.

6. DISEÑO BLUETOOTH

En esta sección se abordará el hardware utilizado para prototipado y diseño de bluetooth bajo consumo (BLE - bluetooth low energy).

6.1. BLEBee

Para el prototipado se utilizará un módulo bluetooth BLEBee que ofrece una conexión simple a placas Arduino que posean este puerto. BLEBee utiliza un módulo de bluetooth RN4020 el cual ofrece bluetooth versión 4.1.

Mediante interfaz UART, este modulo puede ser configurado para actuar como módulo central o periférico cuando establezca una conexión.

Para conocer mas el funcionamiento de este módulo primero hay que conocer mejor la conexión XBee.

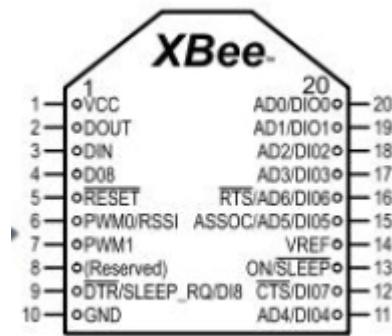


Fig. 6.1: Forma y Pines de puerto XBee

Como se puede observar en la figura 6.1 el puerto XBee ofrece una conexión de 20 pines con usos generales que se utilizan por distintos dispositivos, en este caso

bluetooth.

En el caso de BLEBee la distribución de los pines se puede observar en la figura 6.2



Fig. 6.2: Distribución de pines BLEBee

Comparando la figura 6.1 con la figura 6.2 se puede observar los pines mas relevantes a la hora de diseñar un módulo bluetooth en el dispositivo, siendo los pines mas importantes, y contrastando con el datasheet, los que se muestran en la tabla 6.1

Nº Pin	Nombre	Descripción
5	UART TX	Transmisor UART
6	UART RX	Receptor UART
7	WAKE_ SW	Despertador de modo deep sleep
10	Led Conexión	Led indicador de conexión
12	Led Actividad	Led indicador de actividad

Tab. 6.1: Pines relevantes módulo RN4020

6.2. Comunicación UART

La comunicación UART (Universal Asynchronous Receiver-Transmitter) es un formato de comunicación serial donde el formato y la velocidad de transmisión son configurables. Un UART puede ser un circuito integrado independiente pero en la actualidad vienen incluidos en los microcontroladores.

Para entender mejor como funciona la comunicación serial mediante UART hay que observar la figura 6.3

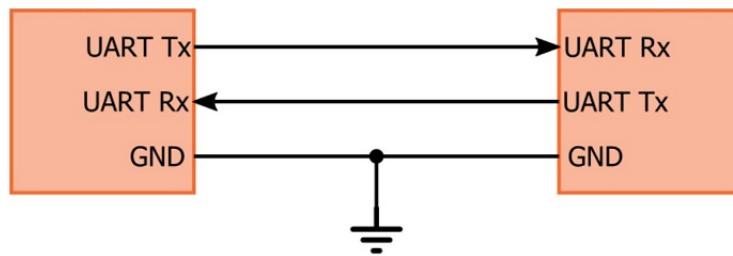


Fig. 6.3: Configuración para comunicación UART

La comunicación UART cuenta de 2 pines TX (transmisor) y RX (receptor) el cual se conecta al Receptor y transmisor del microcontrolador respectivamente.

6.3. Diseño del módulo bluetooth

Finalmente contrastando lo visto anteriormente y con referente a la tabla 6.1 se puede diseñar el esquemático del integrado.

Pin 5 y 6 corresponden a la comunicación UART como se explica en la figura 6.3. Es necesario conectar los pines 10 y 12 con diodos led ya que son pines indicadores, el pin 10 indica el estado de la conexión con un led verde y también se recomienda conectar un led azul al pin 12 para ver el estado de actividad (ej. Si se está enviando información o está dormido).

El pin 7 cumple la función de despertar el módulo bluetooth cuando se encuentre dormido emitiendo desde el microcontrolador una señal de 3,3[V].

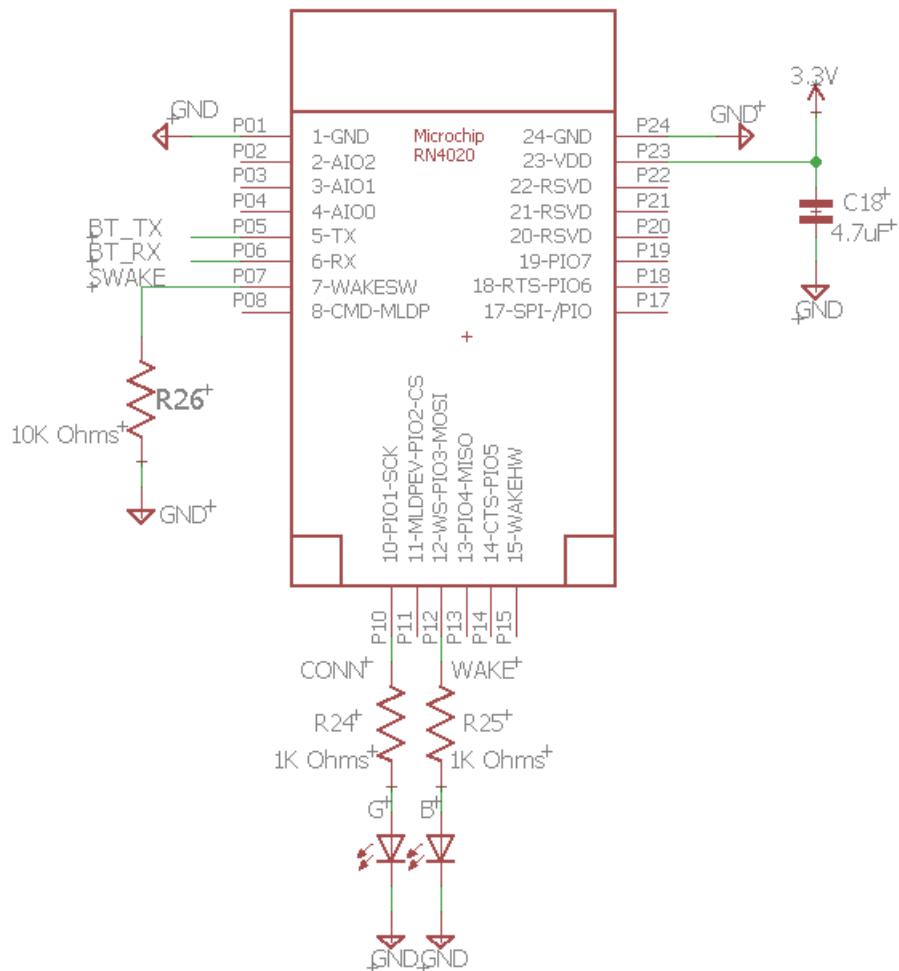


Fig. 6.4: Esquemático bluetooth diseñado en EagleCAD

Como se puede observar en la figura 6.4 se conecta el módulo bluetooth con los pines descritos. Se utiliza un condensador en la alimentación de módulo para regular el voltaje de la entrada.

7. DISEÑO CARGADOR DE BATERÍA

La carga de una batería de Li-ion sigue un perfil diseñado para asegurar la seguridad y la vida de estas sin comprometer su rendimiento. Si una batería de Li-ion es descargada completamente, se aplica una precondición de carga de alrededor del 10 %. Esto previene que la pila suba mucho su temperatura hasta un tiempo que sea aceptable enviar 100 % de la corriente hacia la pila como se puede observar en la figura 7.1.

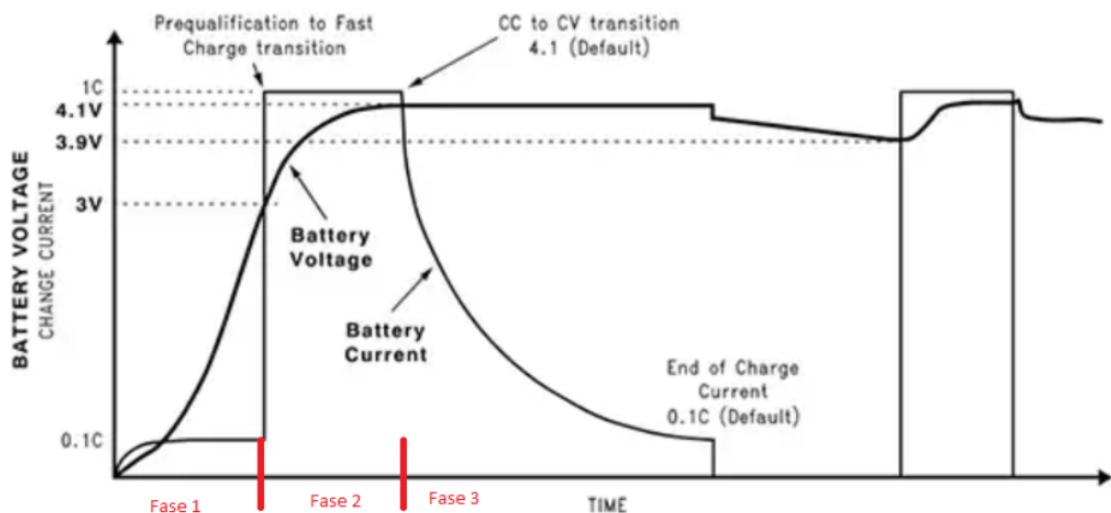


Fig. 7.1: Etapas de carga de baterías Li-ion para una batería de 4.2[V]

Como se puede observar en la figura 7.1, la fase 1 sería la pre-condición de carga del 10 % por un tiempo determinado, luego entra a la fase 2 de corriente al máximo constante y finalmente pasa a la fase 3 donde se mantiene un voltaje constante mientras cae la corriente hasta que termina la carga. Cabe destacar que las corrientes se miden en términos de la variable “C“ la cual se refiere a la corriente que provee la batería,

por ejemplo, una batería de $950[mA]$ posee un $C = 950$. Esto es importante definir para poder realizar los cálculos de las distintas etapas en el proceso de carga.

7.1. Cargador de batería con corriente compartida con carga

En la actualidad, todos los dispositivos electrónicos tienden a ser lo mas simple posible, es por esto que los diseños que se usan ahora poseen una batería interna que no se puede cambiar o no se deban sacar para cargarlas y así facilitar la tarea al usuario.

Esto puede causar un problema debido a la pre-condición que existe en la fase 1 de la figura 7.1. Si el sistema empieza a pedir mucha corriente implicaría que pueda que no se cumpla esa condición y en ese caso la carga (Arduino) estaría descargando la batería en vez de permitir que se cargue, es por esto que existen integrados que pueden otorgar funcionamiento que permite la interacción de estos.

7.2. MCP73871

Considerando un integrado de microchip, el MCP73871[8] otorga una funcionalidad de diseñar un sistema que se encargue de cargar una batería de Li-ion con un sistema conectado, que en este caso sería el Arduino. En la tabla 7.1 se muestra la descripción de los pines que se utilizarán en el diseño.

Nº Pin	Nombre	Función
1, 20	Out	Salida hacia la carga
18, 19	In	Entrada alimentación
13	PROG1	Regulación de corriente de carga rápida
3	SEL	Selección de input
4	PROG2	Límite de corriente entrada USB
12	PROG3	Corriente de término de la carga
14, 15	VBAT	Conexión positiva con la batería
16	VBAT_- SENSE	Sensor de voltaje de la batería

Tab. 7.1: Descripción de pines MCP73871

7.2.1. Power Supply Input (IN)

Este viene a ser la conexión a la alimentación en la cual se puede usar un adaptador a la pared USB además de poder cargarlo con un computador. Cuando se usa con la pared se debe considerar que el rango de voltaje debe estar entre $V_{bat} + 300[mV]$ y $6[V]$. En este caso no habría problema ya que esta conexión USB provee $5[V]$ y la batería que se va a utilizar es de $3,6[V]$.

7.2.2. Input Source Type Selection (SEL)

Este pin cumple una función de selección para carga rápida. Con una entrada lógica baja, el límite de carga es de baja corriente $100[mA]$, en cambio al utilizar una entrada lógica alta, la entrada de corriente llega a ser de $500[mA]$. Este pin tiene una relación con la selección del nivel lógico que tenga el pin "SEL" como se puede observar en la figura 7.2.

When: $SEL = High$

$$I_{FastCharge} = I_{Supply} - I_{SystemLoad}$$

When: $SEL = Low; PROG2 = High$

$$I_{FastCharge} = 500mA - I_{SystemLoad}$$

When: $SEL = Low; PROG2 = Low$

$$I_{FastCharge} = 100mA - I_{SystemLoad}$$

Fig. 7.2: Configuraciones disponibles para carga rápida

Como se puede observar al utilizar "SEL" conectado a tierra permite que "PROG2" pueda ser utilizado cargando a 500[mA] o 100[mA] por lo que se elegirá la entrada de "PROG2" en alto para que se permita de igual manera cargar rápido y utilizar todas las funciones del dispositivo dejando un límite de seguridad de 500[mA].

7.2.3. Regulación de carga rápida(PROG1)

El pin "PROG1" determina el máximo constante de corriente con un resistor entre el pin 13 y tierra. Este además fija el máximo de corriente permitida y la corriente de término. Para calcular se debe considerar la ecuación 7.1.

$$I_{carga} = \frac{1000[V]}{R_{PROG1}[k\Omega]} \quad (7.1)$$

Primero se debe conocer la corriente de carga I_{Carga} , para esto se utiliza la ecuación 7.2.

$$R_{PROG1} = \frac{1000[V]}{I_{carga}} = 1,315[k\Omega] \approx 1,3[k\Omega] \quad (7.2)$$

Despejando esta ecuación se obtiene un resultado aproximado de 1,3[kΩ] que será la resistencia utilizada para el diseño

7.2.4. Corriente de término de carga (PROG3)

El ciclo de carga es terminado cuando, durante una etapa de voltaje constante (Figura 7.1), el promedio de corriente disminuye un límite (generalmente 0.1C) que se fija con un resistor conectado desde el pin "PROG3" hasta V_{ss} el cual puede ser conectado a tierra.

$$R_{PROG3} = \frac{1000[V]}{I_{Termino}} = 10,5[k\Omega] \approx 10[k\Omega] \quad (7.3)$$

7.2.5. Salida de voltaje a la batería V_{BAT} y sensor de voltaje de batería V_{SENSE}

Al conectar el terminal positivo de la batería de Li-ion el pin V_{BAT} cumple la función de cargarla, es recomendable utilizar un capacitor de cerámico en la salida para que asegure la estabilidad de la carga. El pin V_{SENSE} se encarga de entregar el voltaje de salida para que el integrado se pueda realimentar con el valor que está entregando y corregirlo.

7.2.6. Pines Generales

Los pines mencionados anteriormente son los más relevantes con respecto a diseño del dispositivo, hay otros pines que recomienda el fabricante utilizar como referencia para el usuario o el desarrollador los cuales muestran, conectando diodos led, Estado de batería, estado de carga y si se está entregando o no alimentación al dispositivo. Además hay otras funciones que no se utilizan por lo que esos pines son conectados a tierra o al voltaje de alimentación según se especifique.

7.2.7. Diseño del esquemático

Con todos los cálculos anteriores, finalmente se diseña el circuito con el integrado en el programa EagleCAD que será usado para enviar a producción de la PCB como se muestra en la figura 7.3.

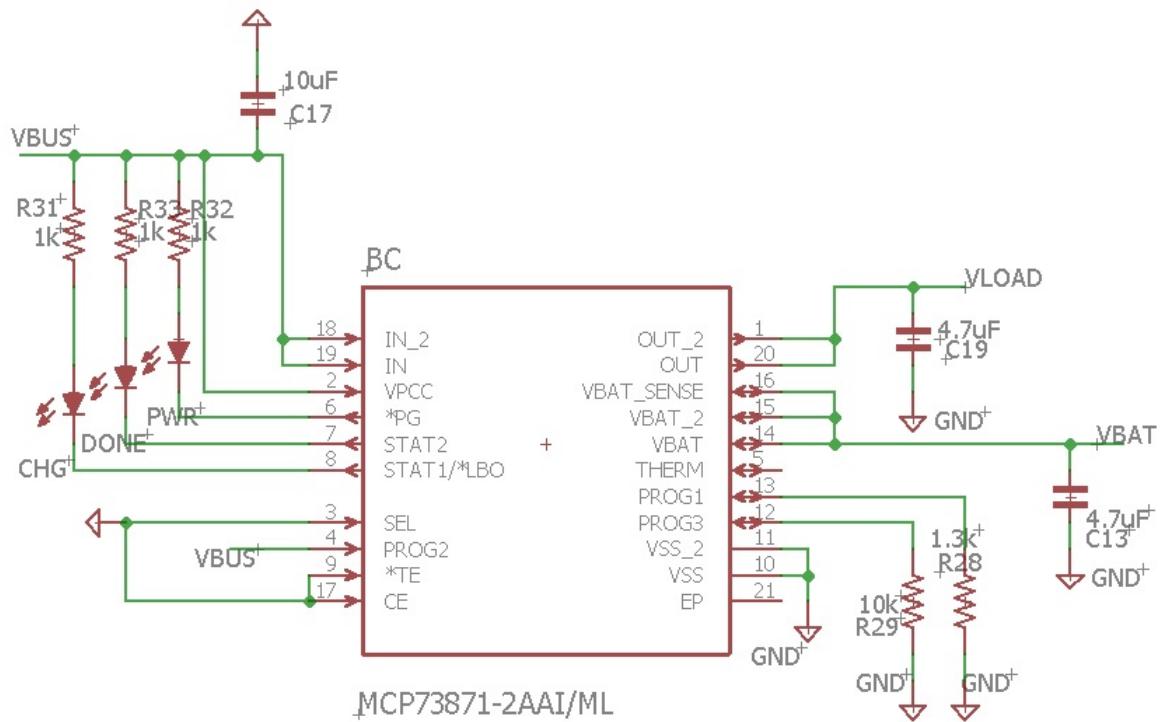


Fig. 7.3: Diseño de cargador de batería en software EagleCAD

Se puede observar los valores diseñados para una batería de $3,7[V]$ de Li-ion con $950[mAh]$ de capacidad. Como se recomendó en V_{BAT} es necesario utilizar condensadores de cerámica para regular las salidas, al igual que en V_{LOAD} el cual viene a ser el que se entrega al Arduino.

Se eligen los valores de "SEL" en $0[V]$ y "PROG2" en $5[V]$ para así limitar la corriente de carga a la diferencia entre $500[mA]$ y la corriente de la carga, tal como se indica en la figura 7.2.

8. DISEÑO FINAL

En este capítulo se van a juntar las partes diseñadas anteriormente para obtener un esquemático final que será adaptado a una PCB utilizando el programa EagleCAD. Además se diseñará un botón ON/OFF con el objetivo de no utilizar interruptores de 2 o mas posiciones.

8.1. Botón ON/OFF

Con las nuevas tecnologías van quedando obsoletos los botones o interruptores de 2 posiciones como se puede observar un ejemplo en la figura 8.1 y también causado por la disminución del tamaño de los dispositivos electrónicos.



Fig. 8.1: Interruptor de 2 posiciones

Estos interruptores ofrecen la funcionalidad que se necesita para un prototipo pero es necesario diseñar un dispositivo final para entregar al cliente. Es por esto nace el requerimiento de que se deba usar un botón único para encender y apagar el dispositivo y esto no es una tarea trivial.

Los requerimientos para este botón es que baje a $0[V]$ cuando se apaga, que sea un único botón y que no requiere ninguna programación.

Un circuito muy utilizado que cumple los requerimientos anteriores se puede observar en la figura 8.2.

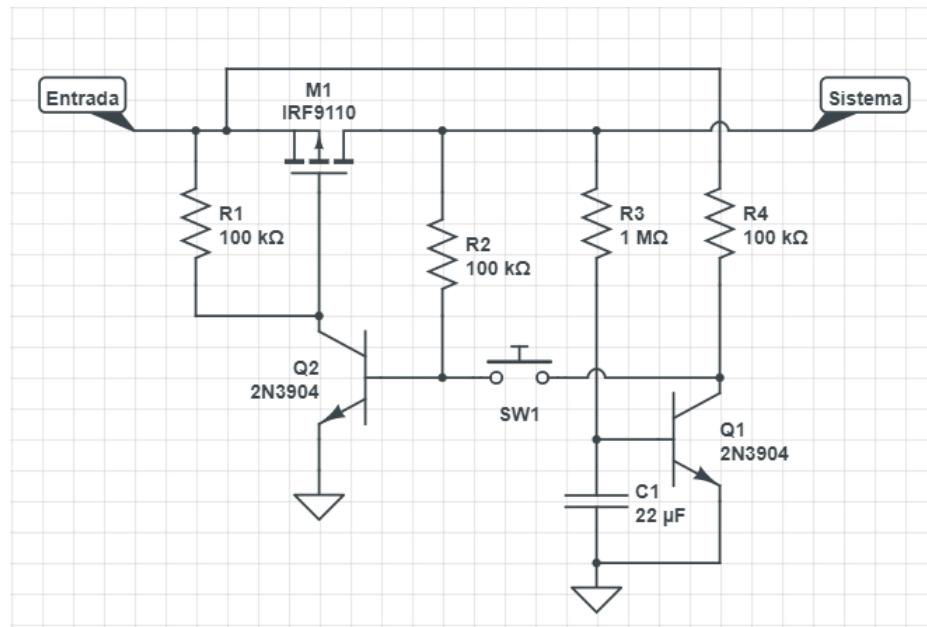


Fig. 8.2: Circuito para botón ON/OFF

Este circuito cumple con los requerimientos de ser simple, funcionar con un solo botón que cumple ambas funciones de encendido y apagado.

Alimentando el circuito con una entrada de $3[V]$ cuando se presiona el botón para encender, se alimenta la base del transistor Q2 lo que provoca que el mosfet M1 permita el paso de la corriente desde "Drain" hasta "Source" esto va a permitir que se alimente el sistema, además va a alimentar el transistor Q1 lo que va a permitir cargar el condensador C1.

El condensador C1 es utilizado para mantener alimentado el transistor Q1 ya que al presionar el botón, una persona normal podría demorar entre $0,1[s]$ a $0,5[s]$ en todo el proceso ya que es humano, por lo que se utiliza para cargarlo en ese tiempo y no se apague automáticamente al soltar el botón.

En una segunda etapa cuando se pulsa el botón para apagar, se corta la base del transistor Q2 y utilizando la resistencia equivalente del Arduino el condensador se va a descargar pasando la corriente hacia el sistema y finalmente se cortará la alimentación. Probando este circuito con un osciloscopio se puede corroborar el funcionamiento de encendido y apagado como se muestra en la figura 8.3

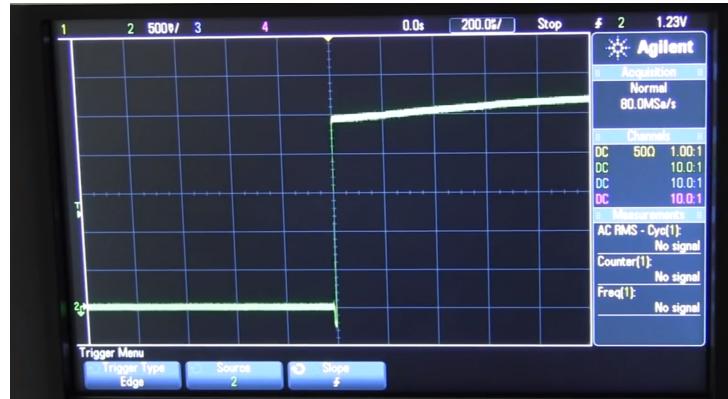


Fig. 8.3: Función de encendido

Esto muestra lo poco que demora el circuito en pasar de un estado a otro y permite mantenerse encendido.

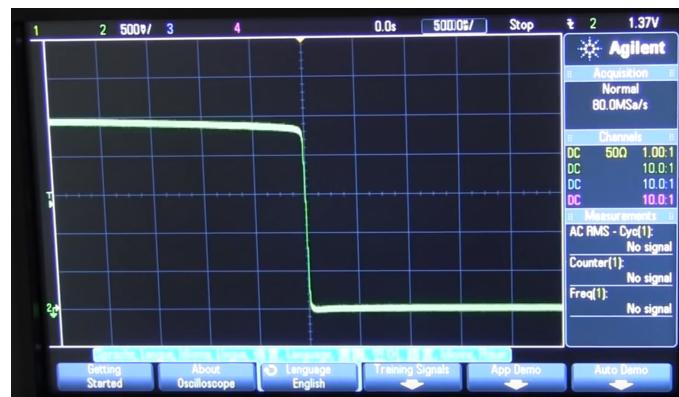


Fig. 8.4: Función de apagado

En la figura 8.4 se puede observar que también se demora poco tiempo en descargarse, debido a que está conectado a una carga muy grande y que pide mucha corriente, lo que hace que el tiempo de descarga del condensador C1 sea mucho menor.

8.2. Esquemático

El diseño final dispositivo se puede considerar como 7 módulos en conjunto entre los cuales se encuentra el microcontrolador, ECG, cargador de batería, botón ON/OFF, FTDI, Bluetooth y regulador de voltaje los cuales resultan en un esquemático final que se puede observar en los anexos.

Los módulos mas grandes se explicaron en los capítulos anteriores, en esta sección se explicará el regulador de voltaje y los diseños finales para el dispositivo.

8.2.1. Regulador de voltaje 3.3[V]

Para proveer energía al microcontrolador y al sistema entero, es necesario un regulador de voltaje de bajo ruido que cumpla la función de evitar grandes variaciones que pueden provocar fallas en la PCB.

Todos los integrados utilizados tienen un rango aceptable de funcionamiento que varía entre los 2 – 5[V] aproximadamente. En este intervalo los valores comúnmente utilizados son 3,3[V] y 5[V].

La alimentación es un factor determinante a la hora de diseñar un microcontrolador ya que al utilizar un resonador para sincronizar su reloj, dependiendo del voltaje que se provee, este funciona a distintas velocidades como se puede observar en la figura 8.5.

Speed Grade:

- ATmega640V/ATmega1280V/ATmega1281V:
 - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
- ATmega2560V/ATmega2561V:
 - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
- ATmega640/ATmega1280/ATmega1281:
 - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
- ATmega2560/ATmega2561:
 - 0 - 16MHz @ 4.5V - 5.5V

Fig. 8.5: Velocidades y voltajes recomendados para un microcontrolador ATMega2560

Como se puede observar, es posible configurar el microcontrolador de 2[MHz] con voltaje entre 1,8[V] y 5,5[V]. Además una velocidad de 8[MHz] se puede conse-

uir con un voltaje entre 2,7 y 5,5[V]. Finalmente se puede utilizar una velocidad de 16[MHz] con un voltaje entre 4,5[V] y 5,5[V].

Como una de las condiciones es que el dispositivo sea de pequeño tamaño y peso, se utilizará una batería de 4,3[V] de 950[mAh] y es por esto que se va a bajar la velocidad a 8[MHz] para utilizar una alimentación de 3,3[V].

Se utilizará un integrado MIC5219 el cual ofrece un regulador de bajo ruido que funciona a 3,3[V] y permitirá energizar el dispositivo. En la figura 8.6 se puede observar el integrado y su configuración típica.

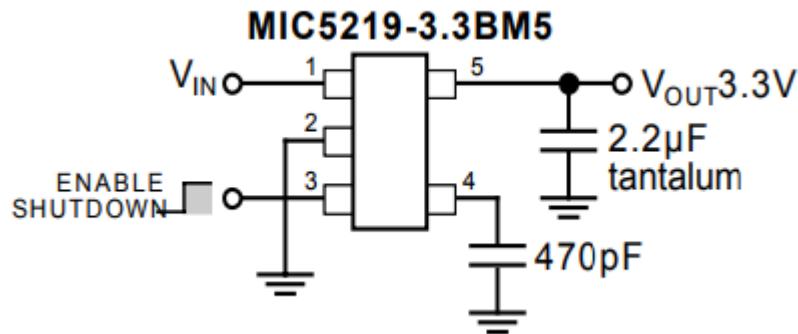


Fig. 8.6: Regulador de voltaje 3.3[V]

Para entender este diagrama es necesario destacar V_{in} que viene a ser la alimentación de la batería de Li-ion, V_{out} es la salida de 3,3[V] hacia la carga que involucra el microcontrolador y todas las componentes del sistema.

El pin 2 es la tierra del sistema y el pin 3 ofrece una función para habilitar y deshabilitar el regulador la cual no se va a utilizar. Finalmente el pin 4 ofrece reducción de ruido conectandolo a tierra con un condensador de 470[pF].

8.2.2. Conector para sensores

Al trabajar con sensores, es necesario facilitar la tarea para el diseñador de producto de conectar las entradas al sistema con el vestible, es por esto que se va a reducir

todos los sensores a un solo conector. Para esto, considerando que se utilizará el ECG y 2 sensores de temperatura para promediar su valor y obtener mayor precisión, se necesitan 5 conectores para los datos y además alimentación y tierra.

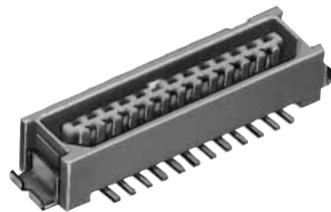


Fig. 8.7: Conector para sensores

El conector que se utilizará se puede observar en la figura 8.7. La particularidad que ofrecen estos tipos de conectores es que poseen entre 9 y 51 contactos.

Para el diseño se van utilizar 9 contactos los cuales constan de 3 conectores para los 3 electrodos de ECG, 2 para sensores de temperatura y 4 de alimentación (Voltaje y tierra para ECG independientes de la alimentación de los sensores de temperatura) como se puede observar en la figura 8.8.

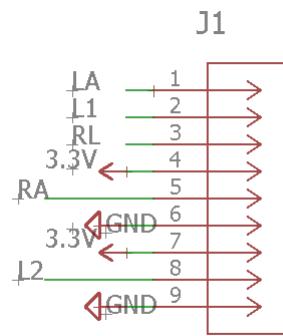


Fig. 8.8: Conector de 9 pines

8.2.3. Pin header para prototipo

Finalmente, como el microcontrolador ofrece 16 pines análogos, 54 digitales y 14 PWM se dejarán headers en la primera iteración del dispositivo para que se pueda seguir prototipando y agregar mas sensores en el futuro, donde se pueda trabajar en el mismo dispositivo en vez de utilizar uno independiente.

También se va a dejar disponible pines para comunicación UART (2 utilizadas en el dispositivo y 2 libres) y también los pines SDA y SCL para comunicación I2C.



Fig. 8.9: Conección Headers

Como se puede notar en la figura 8.9 estos pines ofreceran alimentación y comunicación UART en cada header. 3 pines analogos (A3, A4 y A5), los 2 pines de conexión I2C (SDA y SCL) y 5 pines digitales (PA0, PA1, PA2, PB4 y PB5).

9. DISCUSIÓN

Este trabajo tuvo el propósito de buscar una solución al desafío propuesto por la empresa Sistemas Expertos. Como grupo multidisciplinario se tomó la decisión de no dejar el trabajo e investigación realizada solo como un prototipo por lo que se llevó al área del emprendimiento postulando a fondos para desarrollar mejor la idea y proponer el modelo de negocios desarrollado durante el año.

9.1. Transformación de memoria multidisciplinaria a emprendimiento

Al finalizar el diseño del dispositivo y tener un prototipo funcional como grupo de memorias multidisciplinarias se decidió continuar trabajando en el desafío con una alianza estratégica con la empresa Sistemas Expertos.

Se realizaron entrevistas con profesionales de la salud en el área de Quintero y se postuló a un torneo de emprendimiento del instituto 3IE de la universidad Santa María. El Instituto 3IE tiene como principal objetivo apoyar el desarrollo y aceleración de emprendimientos de base tecnológica, con mérito innovador y alto potencial de escalamiento, a través de su Programa de Incubación, cuya principal propuesta de valor se basa en vinculación temprana con el sector productivo o empresarial, la validación técnico y comercial de las soluciones (productos y/o servicios), la gestión comercial, y el financiamiento temprano para hacer despegar los negocios.

Este Programa, se implementa bajo la línea de financiamiento Torneos de Emprendimiento Tecnológico, apoyado por CORFO y ejecutado por el Instituto 3IE, y que busca instalar capacidades y conocimientos en tecnologías habilitantes para el desarrollo de

nuevas soluciones tecnológicas que logren un impacto positivo en la competitividad y eficiencia del sector industrial o público.



Fig. 9.1: Desafío IoT propuesto por el instituto 3IE

Este concurso posee una planificación del proceso completo que se puede observar en la figura 9.2



Fig. 9.2: Planificación desafío IoT

9.1.1. Entrevista a doctores en hospital de Quintero

Se realizaron entrevistas a 2 profesionales de la salud en el hospital de quintero para evaluar la factibilidad del emprendimiento y ver el problema desde el punto de vista de quienes trabajan en el sector público.

Primero se realizó la entrevista a Fabián Álvarez, medico general de zona y cuyo cargo es jefe de urgencia. También se realizó una entrevista al doctor Cristian Mondaca, Jefe del programa de salud cardiovascular y jefe de farmacia.

Entrevista a Fabián Álvarez

En la entrevista comenta que los adultos sanos realizan controles 1 vez al año y se hacen exámenes preventivos. Además existe un programa de salud cardiovascular para personas mayores de 18 años, según guías clínicas .^{En}foque de riesgo cardiovascular” del MINSAL (2015), cada 3, 6 y 12 meses en los cuales se hacen exámenes de laboratorio, electrocardiograma, se mide presión arterial, peso y estatura. Los problemas que se identificaron fueron la poca disponibilidad de exámenes y recursos humanos para atender a la demanda programada. Además implementan telemedicina por medio de fotos para dermatología donde le envían estas a algún especialista, teleasistencia broncopulmonar mediante videoconferencia y teleasistencia para electrocardiograma el cual intenta no usarse demasiado.

Debido a la sobre demanda del sistema de urgencias, los profesionales de la salud deciden hacer una categorización de pacientes desde C1 a C5 donde C5 significa que requiere asistencia inmediata y C1 puede ser que el paciente vuelva otro día.

Finalmente se hizo una encuesta donde el médico debía dar un valor a las siguientes afirmaciones según un ranking de dolor entre 1 y 10 como se muestra en la tabla 9.1

Afirmación	Nota de Dolor
Muchos Recursos por IIH	3-4
Necesidad de control mas rápido en urgencias	8
Reclamo por el servicio	10
Reclamos internos de los médicos	10
Bajos recursos para mejorar atención	10

Tab. 9.1: Ranking de dolor Fabián Álvarez

Entrevista a Cristian Mondaca

Como jefe del programa de salud cardiovascular, da énfasis en el riesgo cardiovascular con controles cada 3, 6 y 12 meses y destaca que muchas veces se atrasa a los pacientes que van mejorando pese a que no se debería hacer debido a la disponibilidad de horas médicas y la cantidad de pacientes. En caso de mayor necesidad se tiene contemplado el uso de una hora extra al mes siguiente por exámenes nuevos.

Se mostró un vídeo con el prototipo funcional realizado en este trabajo de tesis y el médico hace la observación de que para realizar controles se requiere hacer un ECG completo, es decir, se necesitan las 6 precordiales y las 3 tradicionales (2 electrodos en brazos y 1 en pierna).

En la entrevista destaca que para realizar pruebas en pacientes reales se debe hablar con el director del hospital y desarrollar un consentimiento informado de parte del paciente.

Un dato importante que resalta es la cantidad de pacientes en el programa de salud cardiovascular el cual alcanza un total de 1900 incluyendo hipertiroidismo, de los cuales 1500 son de mayor riesgo.

Entre las ideas propuestas por el médico se destaca la necesidad de tener una base de datos con historial de electrocardiogramas ya que al realizar exámenes tradicionales estos se imprimen en un papel milimétrico y son otorgados a los pacientes, los cuales generalmente pierden esta información para la próxima consulta lo que hace más

difícil un diagnóstico preciso.

Finalmente se hace énfasis en que el hospital de Quintero es un hospital de baja complejidad y no hay especialistas, por lo que cuando hay alguna anomalía se deriva a un cardiólogo o médico internista para realizar un informe y diagnosticar.

Cuando ocurre un infarto, es enviado en ambulancia directo a Viña del mar para que sea atendido por un especialista.

9.1.2. Pitch day seleccionados - Marzo 2018

Para realizar el Pitch para el concurso se utilizó lo aprendido en los módulos de memoria y las clases impartidas por el desafío del instituto 3IE en los cuales dividen el pitch en 3 partes.

La primera parte es hablar del dolor, identificar cuántos tipos de cliente se ven afectados y mostrarlos en términos objetivos (cifras), con esto se debe presentar cuáles son los analgésicos internos, externos y cómo se evitan este dolor y finalmente presentar una promesa de valor en la cual se explica, de manera simple, qué logra nuestra solución. Para esto se recopiló información y se identificó que solo para el año 2016 fallecieron alrededor de 25,000 personas esperando atención médica en alguno de los 203 hospitales públicos chilenos. De esos, 18,423 (74 % aproximadamente) corresponden a personas sobre 65 años. Esto solo refleja un universo mucho mayor, cercano a 1,661,826 de personas que estuvieron en lista de espera y la única forma de mitigar esto es por medio de categorización de pacientes por su nivel de riesgo, lo cual no disminuye las listas, sino que da prioridad a aquellos más urgentes.

La segunda parte consiste en el servicio propiamente tal, donde se plantea reducir las listas de espera y tener una gestión eficiente del capital humano donde los profesionales atienden donde y a quien corresponde en base a un historial. Proyecto Galeno ofrece una pollera inteligente con sensores de electrocardiograma y temperatura conectada a un dispositivo Android que a su vez envía la información a un sitio web donde muestra en tiempo real la información y donde también se puede controlar el comportamiento del ecosistema.

En la tercera parte se muestra el plan de vuelo donde se dan a conocer los hitos relevantes del proyecto, el plan de negocio y el equipo. La obtención de dinero es posible a través de un sistema de arriendo mensual de los dispositivos y la plataforma en cada hospital, según las necesidades de cada uno. Como en caso concreto se tiene al Hospital de Quintero con alrededor de 34 camas disponibles y un ejemplo de pacientes con patologías cardiovasculares, pudiendo proyectar al menos entre 2000 y 6000 USD mensuales (considerando entre 10 y 30 dispositivos a 200 USD cada uno).

Finalmente se cierra presentando el equipo y presentando nuestro aliado estratégico Sistemas Expertos, empresa que posee presencia en el mercado de la atención pública con 17 centros asistenciales a lo largo de Chile en el servicio de software para digitalización de la información.

En esta fase concluye la participación de Proyecto Galeno en el desafío IoT debido a que el jurado considera que la inversión en el desarrollo y la validación del dispositivo para obtener la certificación de grado médico es muy grande, se propone abordar este mismo problema externalizando el desarrollo con un dispositivo que ya tenga validación médica buscando aliados estratégicos para poder desarrollar a partir de esto. Además la presencia de Accuhealth en el mercado ofrece el mismo servicio con una mayor gama de sensores a un precio un poco superior al calculado por este proyecto, lo que hace muy difícil entrar al mercado de la salud privada y no consideran que entrar en el sistema de salud público sea una buena oportunidad.

9.2. Tareas Futuras

La primera versión de un dispositivo electrónico es solo el comienzo de un largo camino en el desarrollo ya que es donde se puede evaluar de mejor manera el funcionamiento del diseño y evaluar cuales son los puntos a mejorar o cambiar.

En primera instancia se desea integrar la IMU en la siguiente iteración utilizando los headers de comunicación I2C que se diseñaron para prototipado y evaluar la posibilidad de cambiar el integrado amplificador para la señal del ECG a uno con menor ruido y mayor precisión.

Además se diseñó un modulo de botón ON/OFF y cargador de batería de forma independiente, lo cual se puede sustituir por un solo integrado lo cual disminuiría mucho el tamaño, pero esta idea fue descartada para la primera versión debido a que aumenta mucho el costo de fabricación debido a su empaquetado lo cual requiere un servicio adicional para soldarlo en la PCB final.

10. CONCLUSIONES

Al ver el esquemático terminado y las piezas seleccionadas se puede notar que se cumplen los requisitos propuestos en un principio. Realizando un listado de costos de componentes se obtiene un valor de $42USD$ total. Además el costo de fabricación de una PCB de tamaño menor a $10x10[cm^2]$ es de aproximadamente $7USD$. Lo que lleva a un costo total del dispositivo de $49USD$ contando solamente componentes electrónicas.

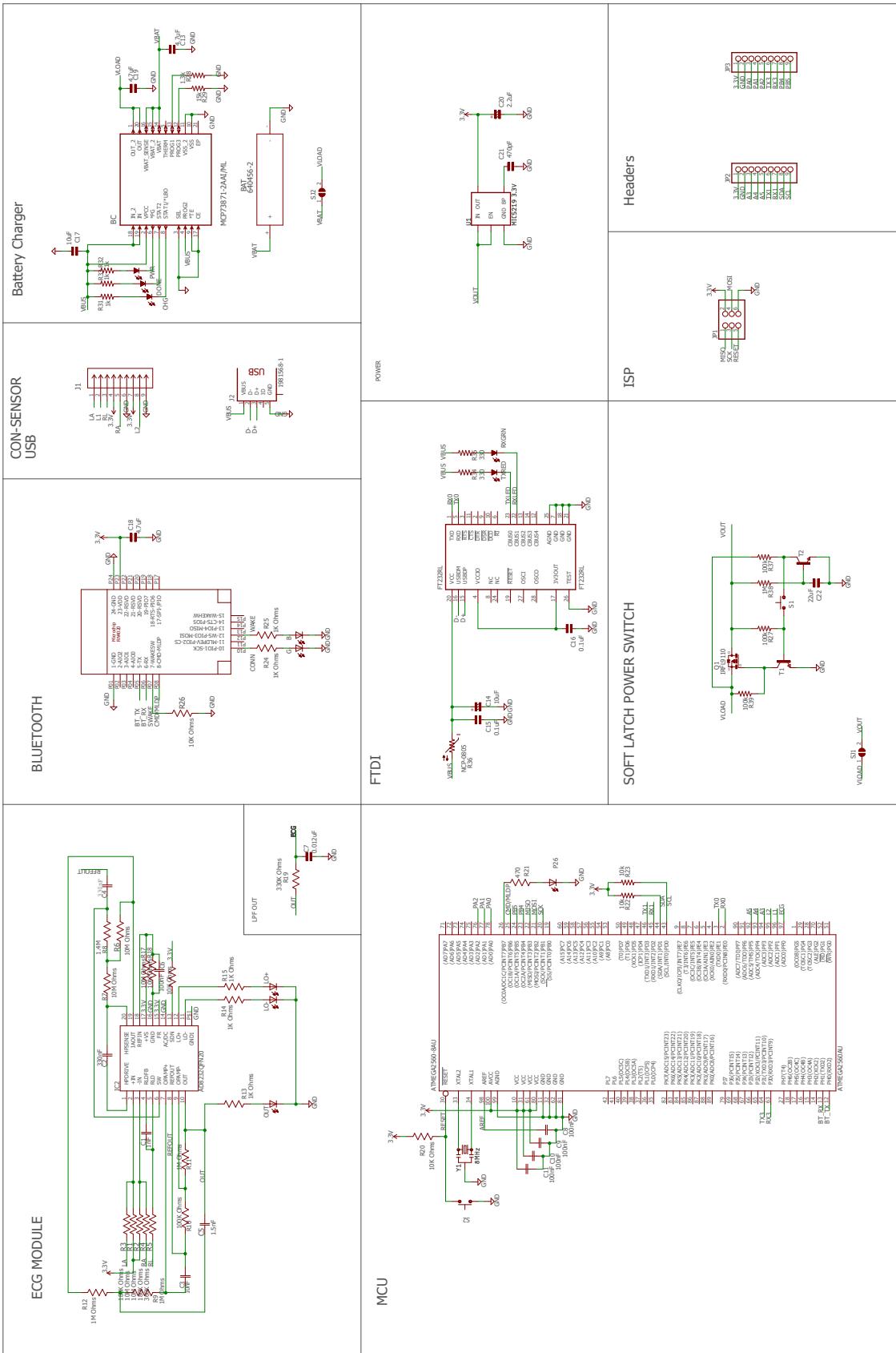
Es importante destacar que esta primera versión del dispositivo es solo un comienzo en la larga etapa de desarrollo, luego de producir esta primera placa se debe realizar las pruebas correspondientes y verificación de integridad de la placa.

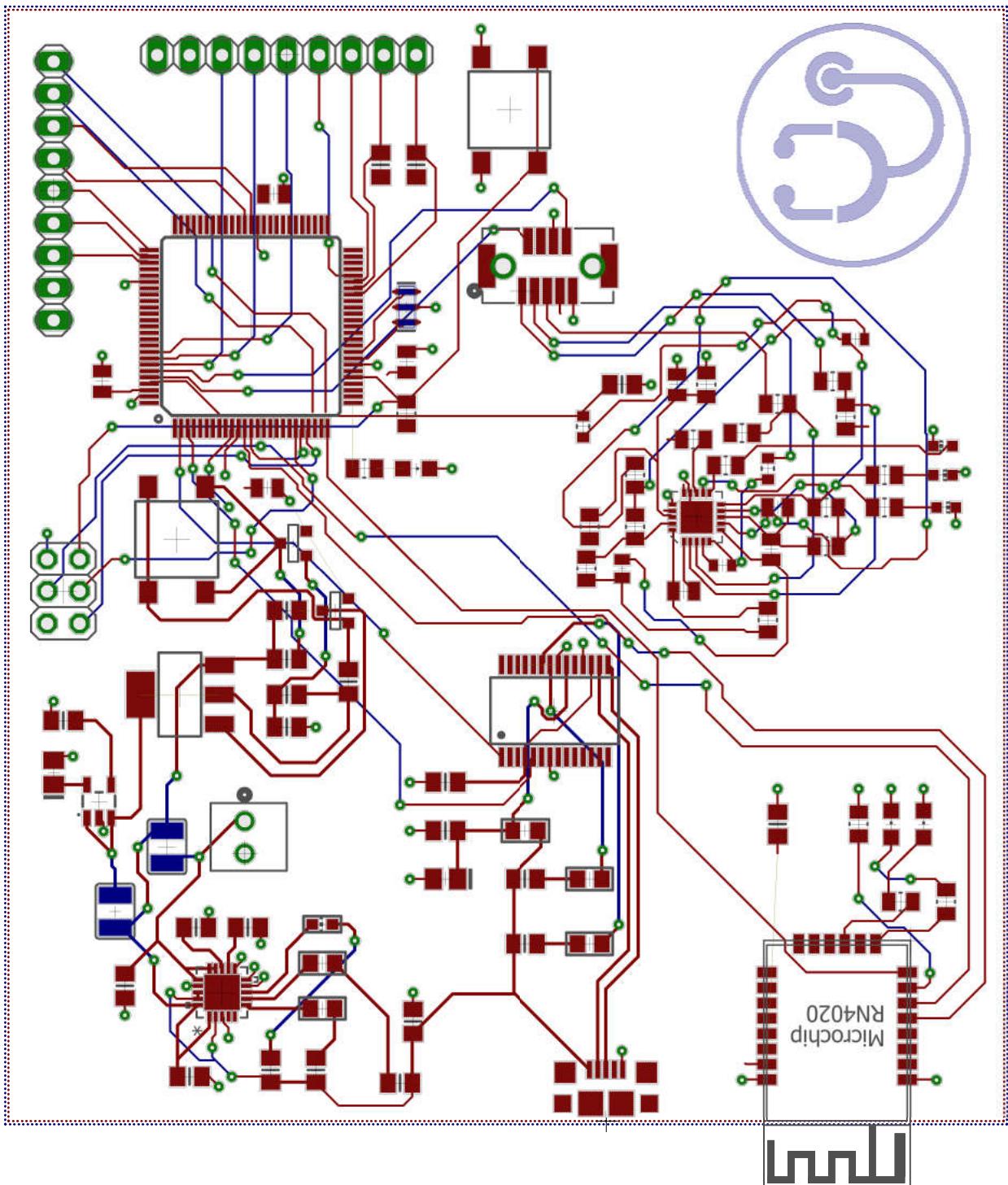
El diseño final de la PCB se puede observar en los anexos.

Además se aprendió a diseñar un dispositivo electrónico desde su prototipo en una placa de desarrollo, donde fue necesario leer constantemente las hojas de datos de todas las componentes a utilizar y seleccionar cuales son las mejores, que al mismo tiempo sean compatibles y coherente con el diseño que se está llevando a cabo.

También es importante destacar la experiencia que se obtuvo con el concurso de emprendimiento. Se aprendió mucho del mundo del emprendimiento al presentar el trabajo realizado. Además se tuvo la oportunidad de observar los trabajos propuestos por los otros emprendedores y estudiar los criterios de evaluación. Se dió mucha importancia al modelo de negocios y las alianzas estratégicas que tenía cada participante, mas que al trabajo realizado y estado actual de la idea o prototipo. Muchos de los seleccionados ya tenían potenciales clientes y alianzas estratégicas que los hacía mejores candidatos a ganar el dinero para el emprendimiento.

11. ANEXOS





The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)
- Menu Bar:** Archivo Editar Programa Herramientas Ayuda
- Toolbar:** Includes icons for Save, Open, Print, and others.
- Project Name:** BLE2_2
- Code Area:** Displays the following C++ code for an Arduino sketch:

```
#include <DallasTemperature.h>
#include <OneWire.h>

#include <SoftwareSerial.h>
#include <string.h>
SoftwareSerial mySerial(3, 2); /* RX, TX */

#define MAX_STRING_LEN 50 /* Máximo largo a leer desde Serial */
#define ECG_SIZE 20 /* Cantidad de datos por envío para ECG */
#define CANTIDADAT 100 /* Cantidad de datos de T a promediar */
#define FRECUENCIAECG 300 /* Frecuencia ECG aproximada al entero más cercano */
#define ONE_WIRE_BUS 5

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);

boolean test = true;

char buffer_char[MAX_STRING_LEN]; /* Almacenamiento máximo para lectura desde Serial */

String UUID_S = "66ecf52ce19f1le48a001681e6b88ec1"; /* Random UUID */
String UUID_ECG = "66ecf194e19f1le48a001681e6b88ec2"; /* Random UUID */
String UUID_T = "f6327dc6a8144e14b68036cf626cba58"; /* Random UUID */
String UUID_C = "962e8b4bedfc4688ac09dd8476ed2dc6"; /* Random UUID */

const int ECGPin = A1; /* Pin para ECG */
const int TPin = A5; /* Pin para T */

const int tiempoAntirebote = 10; /*tiempo antirebote*/
int estadoBoton;
int estadoBotonAnterior;

int ct = 0;
int subbuf;
String Hex = "";
boolean sub = false;
```

The screenshot shows the Arduino IDE interface with the title bar "BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main window displays the code for the "BLE2_2" sketch. The code is written in C++ and defines various variables and functions. It includes comments explaining the purpose of each variable. A section of the code is enclosed in a block comment starting with "/*-----Función Antirebote-----*/".

```
boolean detenerEnvio = false;
boolean iniciarEnvio = true;
int periodoECG = 0;
int periodoT = 0;
int totalesECG = 0;
int totalesT = 0;
boolean enviarECG = false; /* Usada para activar/Desactivar envio ECG data */
boolean enviarT = false; /* Usada para activar/Desactivar envio T data */
int contadorECG = 0; /* Usada para contar datos a enviar de ECG */
int contadorT = 0; /* Usada para contar datos a enviar de T */
int TValue = 0;
String sensor = "00";

int veces = 0; /* Usada para repetir una cantidad x de envios */
String stringValueECG = ""; /* Usado para almacenar valores antes de un envio de ECG */
String stringValueT = ""; /* Usado para almacenar valores antes de un envio de T */
int T = 0; /* Usado para almacenar los valores random de T en testing */
String randomTS = ""; /* String asociado a la T random por testing*/
unsigned int tiempo = 0; /* Control de tiempo minucioso */

/***************************************************************************
 *-----Función Antirebote-----
 **************************************************************************/
boolean antirebote (int pin) {
    int contador = 0;
    boolean estado; // guarda el estado del boton
    boolean estadoAnterior; // guarda el ultimo estado del boton

    do {
        estado = digitalRead (pin);
        if (estado != estadoAnterior ){ // comparamos el estado actual
            contador = 0; // reiniciamos el contador
            estadoAnterior = estado;
        }
        else{
            contador = contador +1; // aumentamos el contador en 1
        }
        delay (1);
    }
}
```

BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)

Archivo Editar Programa Herramientas Ayuda

BLE2_2

```
while (contador < tiempoAntirebote);
return estado;
}

void setup() {
    periodoECG = 9000/FRECUENCIAECG;
    periodoT = 5500/CANTIDADT;
    Serial.begin(19200);
    while (! Serial);      /* No comenzar hasta que el canal Serial se encuentre disponible */
    mySerial.begin(19200);  /* Valores probados: 2400 9600 19200, defecto: 115200 */
    while (! mySerial);    /* No comenzar hasta que el canal Serial se encuentre disponible */
    //pinMode(boton, INPUT); /* Boton de control para enviar datos */

    /*-----SCRIPT FOR BLUETOOTH-----*/
    /* Permite almacenar un set de instrucciones ejecutables por medio de Flags @Flag asociados a */
    /* un comando específico tipo 'WR,X' con X como un valor preestablecido para un cierto FLAG */
    /*-----*/

    mySerial.println("WC");delay(100);          /* Limpiar cualquier Script de la memoria */
    mySerial.println("ww");delay(100);          /* Limpiar cualquier Script de la memoria */
    mySerial.println("@PW_ON");delay(10);       /* Activar enviando un 'WR,0' por mySerial */
    mySerial.println("SF,1");delay(10);          /* Reset */
    mySerial.println("S-,Galenol");delay(10);    /* Nombre del Dispositivo */
    mySerial.println("SB,2");delay(10);          /* Baudios a utilizar */
    mySerial.println("SP,4");delay(10);          /* Potencia de Tx */
    mySerial.println("SR,24062000");delay(10);    /* Características permitidas */
    mySerial.println("SS,00000001");delay(10);    /* Tipo de Servicios (privado) */
    mySerial.println("R,1");delay(10);           /* Reiniciar */
    mySerial.println("WP");delay(100);           /* Detener Ejecución Script */
    mySerial.println("ww");delay(100);           /* Comenzar a agregar a Script */
    mySerial.println("@CONN");delay(10);         /* Activar enviando un 'WR,3' por mySerial */
    mySerial.println("B");delay(10);              /* Pedir vincular con dispositivo conectado */
    mySerial.println("WP");delay(100);           /* Detener Ejecución Script */
    mySerial.println("ww");delay(100);           /* Comenzar a agregar a Script */
    mySerial.println("@DISCON");delay(10);        /* Activar enviando un 'WR,3' por mySerial */
    mySerial.println("A,0064,2710");delay(10);    /* Activar reconocimiento cada 100 ms por 10 s */
```

The screenshot shows the BLE2_2 Arduino IDE interface. The title bar reads "BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for save, run, upload, and download. The main area contains the following C++ code:

```
BLE2_2
mySerial.println("WP");delay(100); /* Detener Ejecución Script */
mySerial.println("\x1B");delay(100); /* Inserta ESC para finalizar escritura de Script */

/*-----SCRIPT FOR BLUETOOTH-----*/
/*-----PRIVATE SERVICE AND CHARACTERISTIC ASSOCIATED-----*/

mySerial.println("PZ");delay(100); /* Limpiar servicios privados */

/* Servicio principal contenedor de las características */
mySerial.println("PS,"+UUID_S);delay(100); /*PS,UUID*/
/* ECG Sensor */
mySerial.println("PC,"+UUID_ECG+,10,14,01");delay(100); /* PC, UUID_PC, CHARACTERISTIC_PROP, SI

/* Temperature Sensor */
mySerial.println("PC,"+UUID_T+,10,03,01");delay(100);

/* Control */
mySerial.println("PC,"+UUID_C+,04,01,01");delay(100);

mySerial.println("K");delay(100); /* Desconectarse al inicio */

/*-----PRIVATE SERVICE AND CHARACTERISTIC ASSOCIATED-----*/
/*-----Inicialización-----*/

/* Valores iniciales para las características de ECG y T */
mySerial.println("SUW,"+UUID_ECG+,00000000000000000000000000000000F0");
mySerial.println("SUW,"+UUID_T+,0000F0");
/*-----*/
```



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)
- Menu Bar:** Archivo Editar Programa Herramientas Ayuda
- Toolbar:** Includes icons for Save, Run, Stop, Upload, and Download.
- Sketch Name:** BLE2_2
- Code Content:** The code is a C++ program for an Arduino. It includes comments explaining the initialization, data transmission, and button handling logic. The code uses the Serial port for communication and manages two data streams: ECG and T.

```
/*
-----Inicialización-----
*******/

void loop() {
    if (tiempo == 10000) {
        tiempo = 0;
    }
    tiempo = tiempo + 1;
    /*
-----ENVÍO DE DATOS-----
    */
    /* Envio consecutivo de datos por medio de las 2 características: ECG y T
    * En consideración:
    *   ECG: Cada dato con resolución de 8 bits, almacenados como hexadecimal de 2 caracteres,
    *   es decir, 1 byte en total por dato.
    *   T: Resolución de 8 bits, con valores oscilantes entre 150 y 170 para luego ser interpretados
    *   como 350 y 370 solo con el fin de que el rango de valores queda en 8 bits [0 - 255]
    *   El DELAY establecido dentro del for será el límite de frecuencia con el cual se
    *   enviarán los datos.
    *
    * Para el ECG: Con un valor de 5ms se alcanzará una tasa de: cada 100 ms (20 * 5ms) un
    * almacenamiento de 20 datos, los cuales serán enviados a una frecuencia de 10 Hz (100ms).
    * Logrando así una tasa real de 200 datos por segundo.
    *
    * Para la T: Con un valor de 5 ms, se demorará 100 ms (20 * 5ms) en almacenar 10 datos, luego
    * se esperan 10 repeticiones de lo anterior (veces%10 == 0) para alcanzar 100 datos.
    * Para terminar enviándolos previo promedio de datos.
    */
    /*
    estadoBoton = digitalRead(boton);
    if (estadoBoton != estadoBotonAnterior) {      //si hay cambio con respecto al estado
        if (antirebote (boton)){                  //chequemos si esta preionado y si lo esta
            test = 1;
            if (test == 1){
                Serial.println("APAGAR");
                test = 0;
            }
        }
    }
}
```

The screenshot shows the BLE2_2 Arduino IDE interface. The title bar reads "BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main area displays the following C++ code:

```
else {
    Serial.println("ENCENDER");
    test = 1;
}
}

/*
 * TOMA DE DATOS ECG */
if (enviarECG && tiempo%periodoECG == 0){
    int ECGValue = analogRead(ECGPin);
    ECGValue = ECGValue/4;
    if(ECGValue>16{
        if (test)stringValueECG = stringValueECG + String(150, HEX);
        else stringValueECG = stringValueECG + String(ECGValue, HEX);
        contadorECG = contadorECG + 1;
    }
}

/*
 * ENVIO DE DATOS ECG */
if (enviarECG && contadorECG == 20){
    //Serial.println("Envio ECG");
    totalesECG = totalesECG + 1;
    mySerial.println("SUW,"+UUID_ECG+","+stringValueECG);
    stringValueECG = "";
    contadorECG = 0;
}

/*
 * TOMA DE DATOS T */
if (enviarT && tiempo%periodoT == 0){
    //TValue = analogRead(TPin);
    //TValue = ((TValue*(5/1024.0))-0.5)*100;
    //if(test == 1) T = T + 37;
    //else T = T + 37;
    contadorT = contadorT + 1;
}

/*
 * ENVIO DE DATOS T */
```

The screenshot shows the BLE2_2 Arduino IDE interface. The title bar reads "BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". The toolbar has icons for save, build, upload, and download. The main area displays the following C++ code:

```
BLE2_2

if(enviarT && contadorT == CANTIDADT){
    //Serial.println("Envio T");
    sensors.requestTemperatures();
    TValue = (sensors.getTempCByIndex(0)+2.5)*100;
    if(test) TValue = 3700;

    //T=T/100;
    //stringValueT = String(T,HEX);
    if(TValue>16){
        totalesT = totalesT + 1;
        stringValueT = String(TValue,HEX);
        while (stringValueT.length() < 6) stringValueT = "0" + stringValueT;
        mySerial.println("SUW,"+UUID_T+","+stringValueT);
    }
    //Serial.println("SUW,"+UUID_T+","+stringValueT);
    //mySerial.println("SUW,"+UUID_T+",000025");
    stringValueT = "";
    T=0;
    contadorT = 0;
}

/* LIMPIEZA DE VARIABLES */
/*
if (!enviarECG && tiempo%10000 == 0){
    contadorECG = 0;
    stringValueECG = "";
}
if (!enviarT && tiempo%10000 == 0){
    contadorT = 0;
    T = 0;
} */

/*****************************************-----ENVÍO DE DATOS-----*/
/*****************************************-----LECTURA DE SERIALES-----*/
```

```
BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)
Archivo Editar Programa Herramientas Ayuda
BLE2_2
/*
 * Doble lectura de Serial para debugging con uso de char[]
 */

int MavailableBytes = mySerial.available();
if (MavailableBytes > 1){
    for(int i=0; i<MavailableBytes; i++){
        {
            char c = mySerial.read();
            buffer_char[i] = c;
            if (i == MavailableBytes-1){
                buffer_char[i+1] = '\0';
            }
        }
        int len = strlen(buffer_char);
        //Serial.println(len);
        if (len == 3){
            sub = true;
            subbuf = len-3;
            buffer_char[len-1]='\0';
        }
        else if (len == 4 || len==5 || len==7){
            sub = true;
            subbuf = len-4;
            buffer_char[len-2]='\0';
        }
        else if(len==6){
            sub = true;
            subbuf = len-2;
        }
        if(sub){
            /*Serial.print(strlen(buffer_char[subbuf]));
            Serial.print("-");
            Serial.print(buffer_char[subbuf]);
            Serial.print("|");
            Serial.print("01");
            Serial.print("-");
            Serial.println(strlen("01")); */
        }
    }
}
```

The screenshot shows the Arduino IDE interface with the title bar "BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main window displays the code for the "BLE2_2" sketch. The code is written in C++ and handles various sensor control logic based on character input from a serial buffer.

```
if(detenerEnvio && strcmp(&buffer_char[subbuf], "00") == 0){
    periodoT = 5500/CANTIDADT;
    Serial.println("Apagar ambos!");
    iniciarEnvio = true;
    enviarECG = false;
    enviarT = false;
    Hex = String(totalesECG, HEX);
    while (Hex.length() < 4) Hex = "0" + Hex;
    mySerial.println("SUW,"+UUID_ECG+,xFFFFFFFFFFFFFFFFFFFFFF" + Hex);
    totalesECG = 0;
    //delay(10);
    Hex = String(totalesT, HEX);
    while (Hex.length() < 4) Hex = "0" + Hex;
    mySerial.println("SUW,"+UUID_T+,FF"+Hex);
    totalesT = 0;
    detenerEnvio = false;
}
else if(iniciarEnvio && strcmp(&buffer_char[subbuf], "11") == 0){
    sensor = "11";
    periodoT = periodoT/3;
    Serial.println("Encender ambos!");
    mySerial.println("SUW,"+UUID_ECG+,AAAAAAAAAAAAAAA" + sensor);
    mySerial.println("SUW,"+UUID_T+,AAAA" + sensor);
    iniciarEnvio = false;
    enviarECG = true;
    enviarT = true;
    detenerEnvio = true;
}
else if(iniciarEnvio && strcmp(&buffer_char[subbuf], "10") == 0){
    sensor = "10";
    Serial.println("Encender ECG!");
    mySerial.println("SUW,"+UUID_ECG+,AAAAAAAAAAAAAAA" + sensor);
    iniciarEnvio = false;
    enviarECG = true;
    enviarT = false;
    detenerEnvio = true;
}
else if(iniciarEnvio && strcmp(&buffer_char[subbuf], "01") == 0){
    sensor = "01";
}
```

The screenshot shows the Arduino IDE interface with the title bar "BLE2_2 Arduino 1.8.5 (Windows Store 1.8.10.0)". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main window displays the code for the sketch "BLE2_2".

```
ct = ct + 1;
if(ct == 2){
    ct = 0;
    Serial.println("Encender T!");
    mySerial.println("SUW,"+UUID_T+,AAAA"+sensor);
    iniciarEnvio = false;
    enviarECG = false;
    enviarT = true;
    detenerEnvio = true;
}
}
}
sub = false;
//Serial.print(buffer_char); /* Usado para mostrar lo que responde el BT */
}

int SavailableBytes = Serial.available();
if (SavailableBytes > 0){
    for(int i=0; i<SavailableBytes; i++)
    {
        char c = Serial.read();
        buffer_char[i] = c;
        if (i == SavailableBytes-1){
            buffer_char[i+1] = '\0';
        }
    }
    mySerial.print(buffer_char);
}

if (tiempo%3000 == 0){
    mySerial.print("SUR,"+UUID_C+"\n\0");
}
delayMicroseconds(100);
}
```

Bibliografía

- [1] Sotera Wireless, ViSi Mobile®System, rev. 05 marzo 2018,
<http://www.soterawireless.com/visi-mobile/>
- [2] Qardio Inc., QardioCore, rev. 05 marzo 2018,
<https://www.getqardio.com/es/qardiocore-wearable-ecg-ekg-monitor-iphone/>
- [3] Nuubo, Nuubo wearable ECG, rev. 05 marzo 2018, hypesref[nuubo]
<https://www.nuubo.com/producto>
- [4] Analog Devices, “Single-Lead Heart Rate Monitor Front End”, Rev. B Marzo 2017, <http://www.analog.com/media/en/technical-documentation/data-sheets/AD8232.pdf>
- [5] Maxim Integrated, “Programable Resolution 1-Wire Digital Thermometer”, rev Enero 2015, <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>
- [6] Microchip, “In-Circuit Serial Programming (ICSP) Guide”, rev Enero 2015, <http://ww1.microchip.com/downloads/en/devicedoc/30277d.pdf>
- [7] FTDI Chip, “Future Technology Devices International Ltd. FT232R USB UART IC”, rev 18 noviembre 2015, http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf
- [8] Microchip, “Stand Alone System Load Sharing and Li Ion Battery Charge Management Controller”, rev Septiembre 2013, <http://ww1.microchip.com/downloads/en/DeviceDoc/20002090D.pdf>