

**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 GUÍA:

Marcos Zúñiga B.

PROFESOR CORRESPONENTE:

Francisco Cabezas B.

PROFESOR CORRESPONENTE:

Daniel Erraz L.

Julio - 2018

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 aún 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 aún 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 aún 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

- 2G : Segunda generación de telefonía móvil.
- RF : Radio Frecuencia.
- Wi-Fi : Proviene del término Wireless Fidelity. Corresponde a la norma IEEE 802.11 que define los estándares de conectividad inalámbrica para transmisión de datos entre dispositivos.
- IDE : De sus siglas en inglés: Entorno de desarrollo integrado.
- SO : Sistema Operativo.
- Framework : Estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, librerías y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes.
- SCRUM : Metodología de desarrollo ágil caracterizado principalmente por: Adaptabilidad a cambios, solapamiento de fases de desarrollo y foco en resultados incrementales.
- Sprint : Iteración recurrente utilizada en la metodología ágil SCRUM, corresponde a bloques temporales cortos y fijos.
- UART : De sus siglas en inglés: Transmisor-Receptor Asíncrono Universal, es el dispositivo que controla los puertos y dispositivos serie.
- UUID : Identificador único universal o universally unique identifier (UUID) es un número de 16 bytes (128 bits - Ejemplo: 550e8400-e29b-41d4-a716-446655440000).
- RFCOMM : De sus siglas en inglés: Comunicación por radio frecuencia, es un conjunto simple de protocolos de transporte, construido sobre el protocolo L2CAP. El protocolo está basado en el estándar ETSI TS 07.10.

- Backend** : Capa posterior de un servicio informático, la cual no está pensada para el usuario y tiene como fin sustentar el servicio como proveer datos o lógica.
- IP** : Una dirección IP es un número que identifica, de manera lógica y jerárquica, a una Interfaz en red. La más común es la IPv4 aunque también existe la IPv6 y su única diferencia es la cantidad de bits empleados: 32 y 128 bits respectivamente.
- Hosting** : Servicio de alojamiento en la web.
- HTTP** : Protocolo de transferencia de hipertexto, es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.
- Instancia** : En el contexto informático entiéndase como la "materialización" de cierta abstracción como lo puede ser un modelo o esquema.
- Puerto** : Número de 16 bits, usado por el protocolo host-a-host para identificar a qué protocolo de más alto nivel o programa de aplicación (proceso) debe entregar los mensajes de entrada.
- Dominio** : Nombre que sirve de identificador para una dirección IP.
- DNS** : El sistema de nombres de dominio (DNS, por sus siglas en inglés, Domain Name System) es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada. Este sistema asocia información variada con nombre de dominio asignado a cada uno de los participantes.
- Proxy** : Agente o sustituto autorizado para actuar en nombre de otra máquina o sistema.
- TCP** : Protocolo de control de transmisión, es uno de los protocolos fundamentales en Internet.

- Socket : Concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada..
- WebSocket : Tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP.
- Baudio : Unidad de medida de la velocidad de transmisión de señales que se expresa en símbolos por segundo.

Índice general

1.. <i>Introducción</i>	8
1.1. Memorias multidisciplinaria	8
1.2. Equipo	9
1.2.1. Felipe Cordero	9
1.2.2. Vanessa Muñoz	9
1.2.3. Patricio Rodríguez	10
1.2.4. Sebastián Castillo	10
1.3. Desafío	11
2.. <i>Estado del arte</i>	12
2.1. ViSi Mobile®	12
2.2. Qardiocore®	14
2.3. Nuubo®	15
3.. <i>Arquitectura de la solución</i>	17
3.1. Adquisición de datos	18
3.2. Comunicación	19
3.3. Servicio web	20
4.. <i>Alternativas de desarrollo</i>	21
4.1. Plataforma de desarrollo	21
4.1.1. Arduino	22
4.1.2. Raspberry	22
4.1.3. Beaglebone	23

4.2.	Sensores	24
4.2.1.	ECG	24
4.2.2.	Temperatura	25
4.2.3.	Ritmo Respiratorio	27
4.2.4.	Unidad de movimiento inercial (IMU)	28
4.3.	Comunicación	30
4.3.1.	Celular directa: GPRS/GSM shield	33
4.3.2.	Celular indirecta: Bluetooth BLE shield	34
4.4.	Conclusiones	35
4.4.1.	Plataforma de desarrollo	35
4.4.2.	Electrocardiograma	35
4.4.3.	Temperatura	36
4.4.4.	IMU	36
4.4.5.	Comunicación	36
5..	<i>Sistema de telecomunicaciones</i>	37
5.1.	Redes móviles, Bluetooth y Android	37
5.2.	Perfiles Bluetooth	39
5.3.	Razones para Android	40
5.4.	Comparativa desarrollo híbrido	41
5.5.	Prueba de concepto	43
6..	<i>Implementación de la solución de lado del servidor</i>	44
6.1.	Requerimientos	44
6.1.1.	Tipos de alojamiento Web	44
6.1.2.	Requerimientos del proyecto	46
6.2.	Hosting VPS	47
6.3.	Servidor Tornado	49
6.4.	MongoDB	49
6.5.	Nginx	49

6.6. DNS No-IP	49
7.. <i>Implementación de la solución de lado del cliente Android</i>	50
7.1. Servicios en Android	50
7.2. Implementación de servicios y su comunicación en Android en función de los requerimientos del proyecto	51
8.. <i>Configuración RN4020</i>	52
8.1. BLE (Bluetooth 4.0)	52
8.2. Características disponibles y selección, autoconexión Bluetooth (desde Android)	58
9.. <i>Integración de las componentes de la solución</i>	59
9.1. MLDP RN4020	59
9.2. Implementación de código ejemplo en Android	60
9.3. Arduino actuando de servidor con RN4020	60
10.. <i>Prototipo funcional V1</i>	61
10.1. Configuración final RN4020	61
10.2. Análisis de rendimiento por codificación	62
10.3. Elementos en Android	63
10.3.1. RecyclerView	64
10.3.2. Códigos QR	65
10.3.3. Fragment	66
10.3.4. MPAndroidChart	67
10.3.5. Shared Preference	68
10.3.6. Inicio automático de Servicio y otros	68
11.. <i>Prototipo funcional V2</i>	69
11.1. Control de sensores por Android	69
11.2. Renovación de servidor Arduino	71

11.3. Análisis de resolución y frecuencia para ECG	72
11.4. Base de datos local (comparativa Realm, SQLite)	74
11.5. Modelo tentativo para Realm	76
<i>12..Prototipo final</i>	78
12.1. Paquetes de control Bluetooth	78
12.2. Implementación de Realm	79
12.3. Servicio web	80
12.4. Selección de usuario, tipo de cuentas y muestras	83
<i>13..Discusión</i>	86
<i>14..Conclusión</i>	87
<i>15..Anexos</i>	88

ÍNDICE DE FIGURAS

2.1.	Interfaz de usuario ViSi Mobile®	13
2.2.	Modo de uso ViSi Mobile®	13
2.3.	Qardiocore® multisensor	14
2.4.	Modo de uso Qardiocore	15
2.5.	nECG Shirt	16
2.6.	Sistema Nuubo	16
3.1.	Arquitectura referente	17
4.1.	Placa de desarrollo ECG	24
4.2.	Sensor de temperatura Lilypad	26
4.3.	Sensor de temperatura DS18B20	26
4.4.	Tela Conductiva MedTex	27
4.5.	IMU Sparkfun MPU-9250	29
4.6.	Ejes IMU MPU-9250	30
4.7.	Modulo GPRSbee	33
4.8.	Antena GPRSbee	33
4.9.	Bluetooth RN4020	34
5.1.	Mercado compartido mundial de sistemas operativos móviles 2017-2018 [18]	40
5.2.	Arquitectura de la prueba de concepto, elaboración propia	43
6.1.	Primera versión de la página web	48

8.1. Proceso de anuncio de un dispositivo.	53
8.2. Transacciones GATT.	54
8.3. Orden de objetos en transacciones GATT.	55
8.4. Topología de red de dispositivos conectados.	57
 10.1. Resultados, pruebas de configuración ECG	62
10.2. Uso de librerías en Android	63
10.3. Uso de RecyclerView	64
10.4. QR con MAC de dispositivo Bluetooth	65
10.5. Menú lateral comprendido por distintos Fragment	66
10.6. Gráfico en tiempo real con datos de ECG por Bluetooth	67
 11.1. Control de peticiones por Handler y repetición	70
11.2. ECG a 300 [Hz] y 8 bit de resolución	72
11.3. ECG a 300 [Hz] y 10 bit de resolución	72
11.4. ECG a 1.000 [Hz] y 10 bit de resolución	73
11.5. Benchmark frente a SQLite con el uso de distintas librerías	74
11.6. Comparativa en escrituras a la base de datos	75
11.7. Comparativa en lecturas a la base de datos	75
 12.1. Clases de Realm utilizados para los datos de ECG	79
12.2. Tecnologías empleadas en el lado del servidor	82
12.3. Pantalla principal de un usuario normal	83
12.4. Pantalla principal de un usuario con acceso total	83
12.5. Medición de ambos sensores en tiempo real	84
12.6. Medición de ambos sensores en tiempo real por página Web	84
12.7. Medición de ambos sensores en tiempo real por aplicación Android . .	85

ÍNDICE DE TABLAS

4.1. Valores resistencia Tela MedTex en Pectorales	28
4.2. Valores resistencia Tela MedTex en Plexo	28
4.3. Valores resistencia Tela MedTex en Estomago	28
4.4. Comparativa tecnologías / requerimientos	32
5.1. Comparativa de desarrollo nativo, elaboración propia	41
5.2. Comparativa de desarrollo híbrido, elaboración propia	41
5.3. Desarrollo nativo versus híbrido, elaboración propia	42
11.1. Tabla Paciente: Datos personales del paciente	76
11.2. Tabla Mediciones: Al iniciar una medición almacena los datos relacionada a esta por posible retransmisión necesaria hacia el servidro web	76
11.3. Tabla ECG: Al iniciar una medición de ECG	77
11.4. Tabla Datos ECG: Se escribe con cada dato, pero asociado a una medición(Fecha_hora_min_seg)	77
11.5. Tabla T: Al iniciar una medición de T	77
11.6. Tabla Datos T: Se escribe con cada dato, pero asociado a una medición(Fecha_hora_min_seg)	77

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.

1.2. Equipo

1.2.1. 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.

1.2.2. 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 ineficiencia del servicio en distintas ocasiones. Decide abandonar el grupo por no cumplir los objetivos buscados para su trabajo de tesis.

1.2.3. 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.

1.2.4. 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

1.3. 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 infraestructura 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.

2. 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.

2.1. *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. Esto lo logra haciendo uso de una red existente de Wi-Fi y encriptación WPA2 para la seguridad en la comunicación[2].



Fig. 2.1: Interfaz de usuario ViSi Mobile®

Se puede observar en la figura 2.1 la interfaz que puede ver el paciente al utilizar el dispositivo.



Fig. 2.2: Modo de uso ViSi Mobile®

En la imagen de la figura 2.2 se puede observar los sensores conectados al cuerpo que convergen al dispositivo que toma las señales.

2.2. Qardiocore®

QardioCore® [3] 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. Respecto a las especificaciones técnicas, es capaz de funcionar con una frecuencia de 600 muestras por segundo y una resolución de 16 bit, apoyándose en comunicación Bluetooth 4.0 y plataforma exclusiva iOS (9.0 o superior)[4].



Fig. 2.3: Qardiocore® multisensor

Se puede observar el dispositivo Qardiocore en la figura 2.3 que se conecta a un smartphone para mostrar los datos que se están tomando.

Además como se muestra en la figura 2.4 es de simple uso, funciona como un cinturón en el pecho del paciente.



Fig. 2.4: Modo de uso Qardiocore

2.3. Nuubo®

Nuubo® [5] 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. Se basa en tecnología Bluetooth v2.0 + EDR (PC y móvil), con una frecuencia de 250 muestras por segundo y 12 bit de resolución[6]. 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 2.5.



Fig. 2.5: 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.

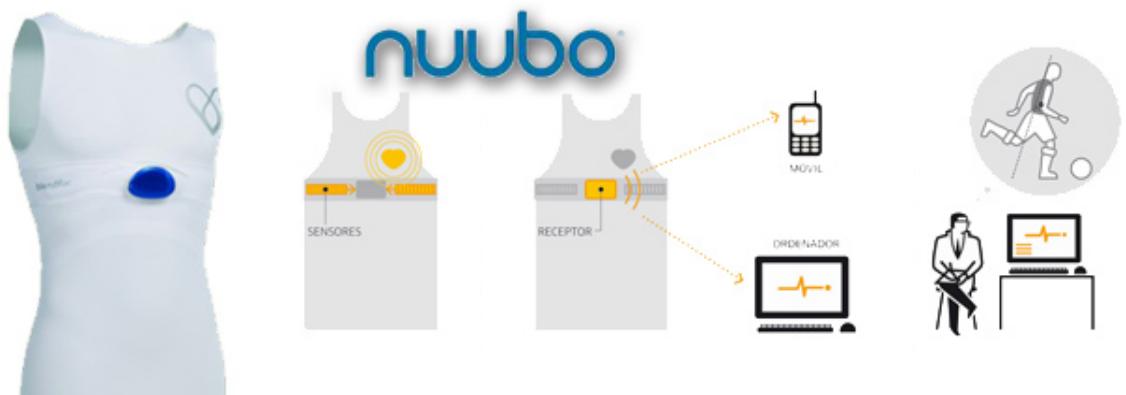


Fig. 2.6: Sistema Nuubo

Como se puede observar en la figura 2.6 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.

3. ARQUITECTURA DE LA SOLUCIÓN

Luego de analizar las necesidades del proyecto y el estado actual de la industria frente al desafío, el siguiente paso es establecer una arquitectura base con la cual definir las partes más relevantes del sistema. En la figura 3.1 se pueden apreciar las 3 secciones más importantes y detalladas más adelante. Cabe destacar que es transversal la necesidad de herramientas que permitan un rápido despliegue, con el fin de generar pruebas constantemente.

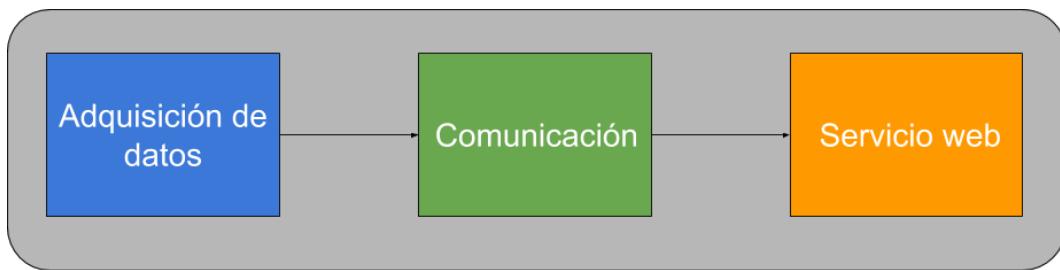


Fig. 3.1: Arquitectura referente

3.1. Adquisición de datos

Una primera necesidad del proyecto es la obtención y procesamiento de los datos, en una primera instancia se omitirá la definición del almacenamiento, permitiendo enfocar los esfuerzos en la selección de sensores, su interconexión y la plataforma que sustente su funcionamiento. Algunos de los requisitos en este apartado son:

1. **Variedad:** Considerando la gama de enfermedades que se podrían cubrir, es importante contemplar una plataforma que permita trabajar con gran cantidad sensores.
2. **Comodidad:** A raíz de que este apartado es el único que tendrá contacto con el paciente es importante pensar en el confort ofrecido, descartando opciones que afecten este apartado, como placas demasiado grandes o pesadas.
3. **Flexibilidad:** Al estar en un proceso iterativo en búsqueda de opciones, un factor a considerar es la flexibilidad que nos puedan ofrecer las distintas opciones, permitiéndonos realizar cambios importantes sin afectar en gran medida las decisiones ya tomadas.

3.2. Comunicación

Dado los requerimientos del proyecto, como lo son las restricciones geográficas que impone Chile y un dispositivo de bajo costo, es necesario contemplar alternativas que no impliquen demasiada infraestructura (o que hagan uso de infraestructura ya disponible) y que además posean la penetración (o el potencial) necesaria dado el territorio nacional. Dentro de las características relevantes en este apartado, podemos encontrar:

1. **Gran cobertura:** Considerando la envergadura inicial del proyecto, Chile, es de vital importancia que la tecnología a utilizar permita generar conexiones en la mayor parte del territorio nacional.
2. **Alta disponibilidad:** Se requiere que la tecnología a emplear permita establecer conexiones a lo largo del tiempo, presentando pocas o de preferencia nulas desconexiones o incapacidades de conexión.
3. **Escalabilidad:** Si bien es un aspecto dependiente de los anteriores requerimientos, es relevante considerarlo por separado como la medida que representa la capacidad de atender una gran cantidad de conexiones.

3.3. Servicio web

El sistema completo requiere además de los puntos anteriores, de un servicio web acorde con las necesidades del proyecto. El cual le de soporte y lo dote de mayores prestaciones, completando así un ecosistema completo en función del desafío. Entre los puntos más relevantes de este apartado se consideran:

1. **Baja latencia:** Este proyecto se desarrolla en un marco con pacientes y posibles estados críticos de los mismos, es por ello que el tiempo de respuesta es fundamental en el servicio que se pretende ofrecer.
2. **Alta concurrencia** Actualmente es común que las conexiones a servicios web tengan una alta demanda y larga duración, aumentando la concurrencia notablemente. Lo anterior es lo que se espera de un monitoreo, el cual debe ser constante en el tiempo (o al menos en una cierta ventana).
3. **Seguridad:** Los datos que utilizará el sistema son totalmente privados y la protección de estos junto con los datos de conexión son un eje fundamental en la elección de tecnología a emplear, o en su defecto usar capas de seguridad anexas para brindar esta funcionalidad que se considera base.

4. ALTERNATIVAS DE DESARROLLO

En el presente capítulo se ahondará 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.

4.1. *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.

4.1.1. 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.

4.1.2. 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.

4.1.3. 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.

4.2. 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.

4.2.1. 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[7] 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.

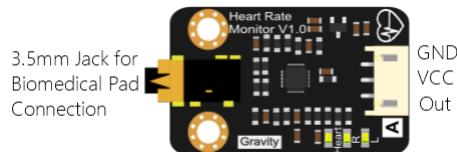


Fig. 4.1: Placa de desarrollo ECG

Como se puede observar en la figura 4.1 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.

4.2.2. 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.

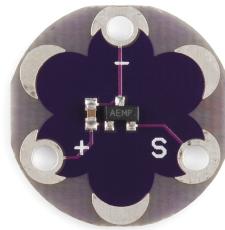


Fig. 4.2: Sensor de temperatura Lilypad

Como se puede observar en la figura 4.2, Lilypad ofrece una PCB impermeable con 3 terminales que permiten utilizar un hilo conductor para coser este a la ropa.

DS18B20

El sensor DS18B20[13] 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 4.3.



Fig. 4.3: Sensor de temperatura DS18B20

4.2.3. 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. 4.4: Tela Conductiva MedTex

Para estudiar la factibilidad de la tela conductiva que se puede observar en la figura 4.4 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. 4.1: 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. 4.2: 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. 4.3: Valores resistencia Tela MedTex en Estomago

Se puede observar en las tablas 4.1, 4.2 y 4.3 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.

4.2.4. 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.

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 4.5.



Fig. 4.5: 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 4.5 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 4.6 se muestra además de los ejes de aceleración también las coordenadas de navegación (roll, pitch, yaw).

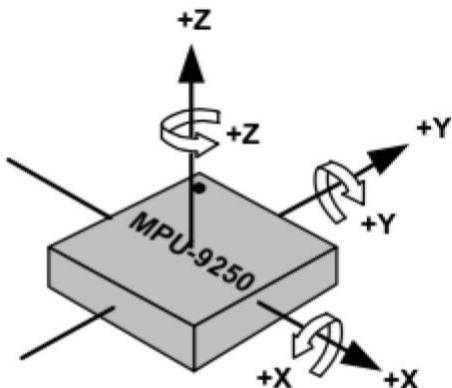


Fig. 4.6: Ejes IMU MPU-9250

4.3. Comunicación

Para el proyecto se consideran distintas alternativas de conexión, las cuales deben seguir ciertos aspectos relevantes según las especificaciones del dispositivo a implementar.

1. **Alta disponibilidad:** Se requiere que la tecnología a emplear permita establecer conexiones a lo largo del tiempo, presentando pocas o de preferencia nulas desconexiones o incapacidades de conexión.
2. **Gran cobertura:** Considerando la envergadura inicial del proyecto, Chile, es de vital importancia que la tecnología a utilizar permita generar conexiones en la mayor parte del territorio nacional.
3. **Bajo costo:** Dentro de los requerimientos del proyecto se encuentra el desarrollo e implementación a bajo costo de la solución final, por tanto la tecnología de comunicación a emplear debe seguir esta directriz para ser seleccionada.

4. **Baja complejidad:** Considerando que el dispositivo en cuestión debiera ser lo más autónomo y sencillo de configurar, es relevante considerar tecnologías de comunicación que no requieran de complejas operaciones para su uso e implementación.
5. **Escalabilidad:** Si bien es un aspecto dependiente de los anteriores requerimientos, es relevante considerarlo por separado como la medida que representa la capacidad de atender una gran cantidad de conexiones (usuarios en definitiva).

En base a lo anterior, se hace un análisis rápido de diferentes alternativas que podrían utilizarse en el proyecto:

1. **Antenas de RF:**[9] Comunicación generada en las bandas situadas entre los 3 kilohercios (KHz) y 300 Gigahercios (GHz). Esta tecnología incluye distintas otras tecnologías como las redes celulares, pero en este apartado se especifica el uso de bandas no utilizadas por esta y otras tecnologías. Permitiendo una conexión directa de antena a antena a una frecuencia específica a determinar.
2. **Comunicación satelital:**[10] Comunicación por medio de ondas electromagnéticas transmitidas gracias a la presencia en el espacio de satélites artificiales situados en órbita alrededor de la Tierra. Dentro de esta tecnología se pueden encontrar dos grandes clasificaciones: Satélites activos y Satélites pasivos, los cuales se diferencian por la amplificación o de las señales antes de ser reenviadas a la Tierra, respectivamente.
3. **Redes Wi-Fi:**[11] También llamada WLAN (Wireless lan, red inalámbrica) o estándar IEEE 802.11, es una de las tecnologías de comunicación inalámbrica mediante ondas más utilizada hoy en día. Existen distintas variantes de este estándar de comunicación, entre los que se destacan 802.11g y 802.11n, por su uso actual en dispositivos comerciales.

4. **Redes celulares:**[12] Consiste en una red de celdas cada una con su propio transmisor, conocidas como estación base. Ampliamente utilizadas en la actualidad, lográndose encontrar hasta 7 compañías distintas que ofrecen sus servicios en Chile: Movistar, Entel, WOM, Claro, Virgin, VTR, SIMPLE.

- a) **Comunicación directa:** Tipo de comunicación en donde el dispositivo posee la capacidad de conectarse, registrarse y hacer uso completo de la infraestructura proporcionada por distintas compañías.
- b) **Comunicación indirecta:** Tipo de comunicación con la cual el dispositivo requiere de un paso intermedio de comunicación para generar la conexión a la red requerida, para este paso se puede destacar el uso de Bluetooth para la comunicación con otro dispositivo con la capacidad de conectarse de forma directa a las redes celulares.

Luego de caracterizar las distintas tecnologías disponibles para su uso en el proyecto, se procede a analizar sus cualidades en función de las 4 especificaciones anteriores:

Tecnología	Disponibilidad	Cobertura	Costo	Complejidad	Escalable
Antenas RF	Alta	Baja	Alta	Alta	Baja
Satelital	Alta	Alta	Alta	Alta	Baja
Wi-Fi	Alta	Baja	Bajo	Media	Alta
Directa	Alta	Alta	Medio	Baja	Alta
Indirecta	Alta	Alta/Baja	Bajo	Media	Alta/Media

Tab. 4.4: Comparativa tecnologías / requerimientos

A raíz de lo anterior se destaca el uso de tecnologías con redes celulares por su lineamiento con el proyecto. La tecnología Wi-Fi se descarta por ser de baja cobertura (en una primera instancia y pensando a nivel nacional) y esto a su vez ser un ámbito crítico para el proyecto. A continuación se presentan alternativas para la plataforma Arduino en torno a las tecnologías ya mencionadas.

4.3.1. Celular directa: GPRS/GSM 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 4.7 el módulo disponible para desarrollo.

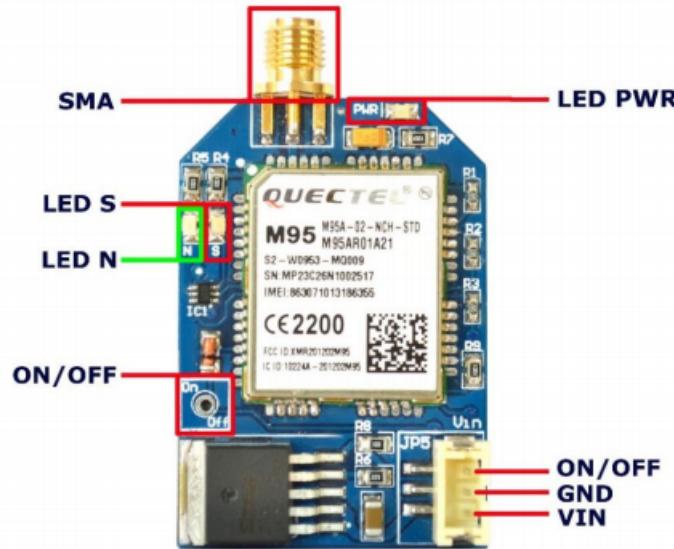


Fig. 4.7: Modulo GPRSBee

Cabe destacar que para poder utilizar este módulo es necesario incluir una antena que se puede ver en la figura 4.8



Fig. 4.8: 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.

4.3.2. Celular indirecta: Bluetooth BLE shield

Para integrar Bluetooth en el dispositivo se considera un BLEBee el cual ofrece Bluetooth versión 4.1 y comunicación UART mediante un puerto XBEE.

El shield Bluetooth posee un módulo RN4020[8] el cual ofrece una antena para la comunicación en su misma placa lo que facilita el diseño como se puede observar en la figura 4.9.



Fig. 4.9: Bluetooth RN4020

Es importante destacar que para mejorar el diseño, el fabricante recomienda dejar expuesta la antena y se destaca la comunicación UART para las configuraciones futuras del sistema.

4.4. Conclusiones

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

4.4.1. 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.

4.4.2. 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.

4.4.3. 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.

4.4.4. 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.

4.4.5. Comunicación

Entre las alternativas mencionadas, se opta por utilizar en primera instancia el GPRS/GSM shield, el cual posee un costo aproximado de 43.980 CLP. Cabe mencionar que en una segunda instancia se utilizaría el chip RN4020, el cual posee un precio aproximado de 20.000 CLP.

Si bien el costo es menor, es relevante la complejidad que agrega la segunda opción (al agregar un actor como lo puede ser un teléfono inteligente). Aunque la flexibilidad que brinda esta segunda opción y ciertos aspectos de la primera son las que obligan este cambio.

5. SISTEMA DE TELECOMUNICACIONES

Se determinó utilizar la plataforma Arduino por su simplicidad en prototipado y programación, además de ser de fácil acceso y una tecnología escalable. En base a esto se decide comenzar a trabajar en el apartado de comunicación.

5.1. *Redes móviles, Bluetooth y Android*

Como se comentó anteriormente, la primera elección de tecnología para la comunicación fue la de utilizar redes móviles directamente, esto por medio del módulo GPRS Shield. Este es un módulo de comunicación 2G compatible con socket XBee para la Arduino UNO (un socket XBee dota a una placa Arduino de la capacidad de comunicarse en forma inalámbrica [14]). Para comodidad se escogió una variante de Arduino UNO llamado PICARO+, la cual posee entre otras modificaciones el socket XBee integrado.

Para esta comunicación se debe hacer uso tanto de la placa principal, el chip de comunicación y una antena, estas últimas 2 mencionadas en las figuras 4.7 y 4.8.

Dentro de las características del GPRS se encuentran el emplear una tarjeta SIM, conector SMA y el chip Quectel M95. En cuanto a la antena nos encontramos con cuatri-banda:

1. **GSM/850E: 824 a 894 [MHz]**
2. **GSM: 880 a 960 [MHz]**
3. **DCS: 1710 a 1880 [MHz]**
4. **PCS: 1850 a 1990 [MHz]**

Si bien puede parecer cuestionable el utilizar tecnología 2G, es importante considerar que chip provee de hasta 85.6 [kbps] y diversos protocolos de comunicación. Con lo cual al año 2017 (se espera deshabilitar las redes 2G en el mediano plazo para dar paso a nuevas tecnologías) sirve como prueba de concepto dado su bajo costo y el acercamiento que ofrece a los comandos AT, los cuales son los empleados para controlar chips de este tipo. Luego de comenzado el proceso de configuración, se encontraron diversos problemas con esta elección:

1. **Dimensiones:** Dado que este módulo esta contemplado para operar en conjunto con la placa principal, se hace engorroso el tener una antena de casi 6 [cm] y de gran grosor adosado al cuerpo del paciente.
2. **Consumo energético:** Este módulo hace necesario el uso de una fuente de alimentación externa de mayor capacidad (9[V] aproximadamente) respecto a la necesidad base de la placa (3.3[V]), lo que conlleva a usar un cargador externo y en su momento a una batería de mayor capacidad.
3. **Antigüedad de comandos:** Los comandos Hayes (también llamados AT [15]) son un conjunto de comandos empleados en la configuración y parametrización de módems, su uso data de al menos 1990 y en cierto punto dejaron de usarse para dar paso a controladores específicos.
4. **Tasas de transferencias:** A raíz de un estudio preliminar en tasas de transferencia se estableció que alrededor de 150 datos por segundo debían ser enviados (esto en función de las tasas de operación de un ECG común[16]), por tanto el usar esta tecnología obliga a emplear 3G como mínimo.
5. **Costo:** En comparación a otras tecnologías indirectas de redes móviles como lo puede ser el Bluetooth, la inversión necesaria es mayor y su flexibilidad bastante menor.

Por todo lo anterior, se pasa a una segunda iteración en busca de emplear tecnología Bluetooth y un intermediario para llegar a las redes celulares.

5.2. Perfiles Bluetooth

Bluetooth [17] es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) creado por Bluetooth Special Interest Group, Inc. que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz y data de 1994 y actualmente se encuentra en su versión 5.0. La versión a emplear en este proyecto es la 4.0 llamada BLE (Bluetooth Low Energy) que se detalla en capítulos adelante.

Un perfil Bluetooth es la especificación de una interfaz de alto nivel para su uso entre dispositivos Bluetooth. Para utilizar Bluetooth, un dispositivo debe implementar alguno de los perfiles soportados. Los perfiles son descripciones de comportamientos generales que los dispositivos pueden utilizar para comunicarse, formalizados para favorecer un uso unificado. La forma de utilizar las capacidades de Bluetooth se basa, por tanto, en los perfiles que soporta cada dispositivo. Los perfiles permiten la manufatura de dispositivos que se adapten a sus necesidades.

A la fecha existen más de 27 perfiles Bluetooth, pero durante el desarrollo del proyecto se emplearon solo dos: SPP y GATT, los cuales se detallarán en su momento.

Las principales diferencias entre estos últimos dos perfiles son: GATT pertenece al estándar introducido en la versión 4.0 (desde ahora BLE) mientras que SPP en la versión 2.1. BLE está pensado para operar con un consumo energético inferior que versiones anteriores, posee mayor velocidad en el establecimiento de la conexión y está pensado para la transferencia de pequeñas cantidades de datos. Excepto por el último punto se puede observar una notoria superioridad de GATT (BLE) frente a SPP, pero como se verá más adelante, las tasas de transferencias obtenidas con GATT son lo suficientemente buenas como para escogerla en este proyecto.

5.3. Razones para Android

Para seleccionar el intermediario entre la comunicación Bluetooth y las redes móviles se decidió un teléfono inteligente, por sus capacidades de cómputo, gran accesibilidad y flexibilidad al ofrecer un entorno de desarrollo propio de su sistema operativo. Ahora bien, para seleccionar el sistema operativo se recurre a su penetración en el mercado y como se puede observar en la figura 5.1 Android se alza como el gigante en el mercado.

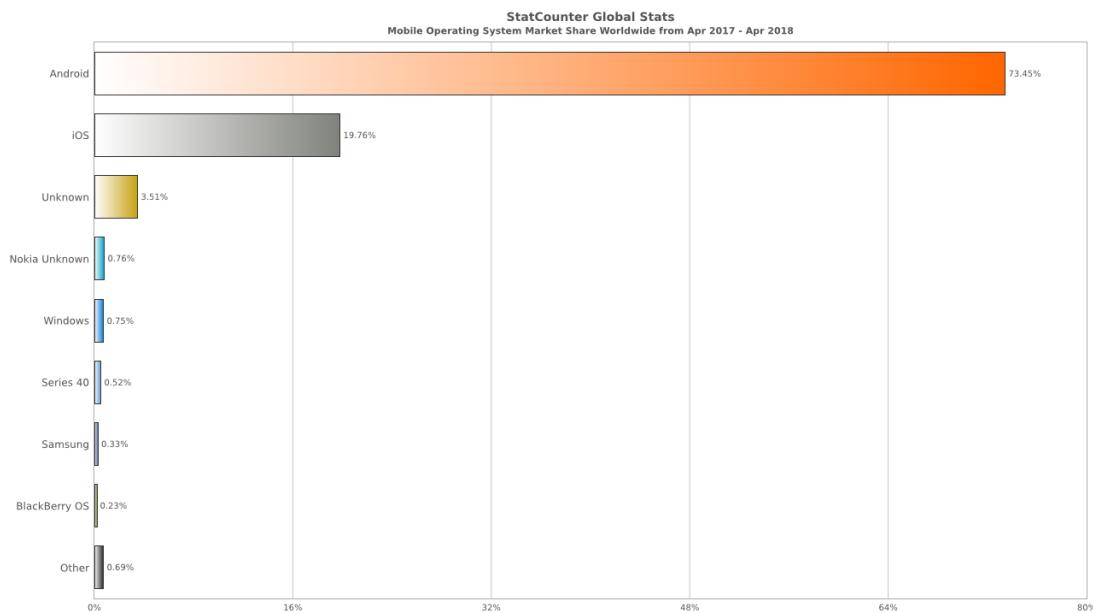


Fig. 5.1: Mercado compartido mundial de sistemas operativos móviles 2017-2018 [18]

Además de lo anterior, se ha de considerar el entorno de desarrollo y el ecosistema que rodea al sistema operativo en cuestión. En el caso de Android, se trabaja principalmente con Android Studio (Para sistemas Linux, Windows y Mac), en lenguaje Java o Kotlin, con una comunidad activa de desarrolladores y una gran cantidad de librerías a disposición.

Por último se considera la accesibilidad de los terminales, con lo cual es de conocimiento general que Apple posee precios más elevados que los dispositivos Android. Por lo tanto se concluye que Android es la mejor alternativa en estos momentos.

5.4. Comparativa desarrollo híbrido

Para el desarrollo móvil actual existen dos grandes aproximaciones, las cuales pasan principalmente por el uso de entornos de desarrollo que permiten el despliegue en más de un sistema operativo (llamados híbridos) o uno determinado con un solo código fuente.

Primero, en el ámbito nativo (un código fuente para un despliegue único):

OS	Lenguaje	IDE	Plataforma
Android	Java, Kotlin	Android Studio, Eclipse	Linux, Mac, Windows
iOS	Objective-C, Swift	XCode	Mac

Tab. 5.1: Comparativa de desarrollo nativo, elaboración propia

Continuando con el ámbito híbrido, se destacan 3 grandes competidores:

Framework	Tipo de resultado	Lenguaje
Ionic	No nativo	JavaScript (AngularJS)
Reac Native	Nativo	JavaScript (React)
Flutter	Nativo	Dart

Tab. 5.2: Comparativa de desarrollo híbrido, elaboración propia

Por último se analizan ventajas y desventajas de ambas aproximaciones al desarrollo móvil, cabe destacar que Flutter aún en 2018 se encuentra en fase beta, pero se considera por las grandes prestaciones que presenta (por lo que no se considerarán sus ventajas en la siguiente tabla), además se establece un marco en donde se espera obtener un desarrollo tanto para iOS como para Android:

Característica	Nativo	Híbrido
Rendimiento	Máximo	Suficiente
Actualizaciones SO	Sin retraso	Con retraso
Librerías	Extenso	Acotado
Control	Total	Parcial
Tiempo de desarrollo	Alto	Medio/bajo
Cantidad de código	Alto	Mínimo
Diversidad de código	Total	Unificado
Complejidad	Alta	Baja

Tab. 5.3: Desarrollo nativo versus híbrido, elaboración propia

Como se puede observar en la tabla 5.3, el mayor potencial para el desarrollo híbrido es cuando no se tienen funcionalidades demasiado específicas (que requieran librerías especiales), no se requiere gran rendimiento, no se utilizarán las últimas características de seguridad del SO y el tiempo es primordial.

Si bien se podría considerar el uso híbrido, se espera que la aplicación haga uso de alto poder de procesamiento, utilice librerías específicas (como graficar en tiempo real como se verá más adelante) y se tenga el mayor control posible de todos los procesos. Por lo tanto se descarta el uso de entornos híbridos para el desarrollo, en desmedro de la compatibilidad con dispositivos Apple.

5.5. Prueba de concepto

Para comenzar el desarrollo e iniciar los sucesivos Sprint (metodología SCRUM), se hizo uso del perfil SPP de Bluetooth, incluído en su versión 2.1 + EDR (2004) y que permite comunicación bidireccional. Es uno de los perfiles fundamentales de Bluetooth al tener un comportamiento muy parecido a los de la comunicación serial (como la usada en conexiones RS-232 o UART). Está basado en el protocolo RF-COMM y emula una linea serial, para su uso se utilizan dos actores, uno que actúa como servidor y otro que actúa como cliente. El primero queda a la espera de alguna conexión entrante (visible), luego por medio de una búsqueda y el uso de un UUID el segundo genera una conexión para comenzar a intercambiar datos.

El objetivo es generar una prueba de concepto por el cual se usara a una aplicación Android como puente para llevar información internet.

Para esto se implementó la siguiente arquitectura:



Fig. 5.2: Arquitectura de la prueba de concepto, elaboración propia

Como se puede observar en la figura 5.2 el servidor Bluetooth ()desarrollado en Java) fue implementado en computador (Windows), mientras que la aplicación Android básica permite el escaneo, selección y conexión con el servidor Bluetooth. Esto último sin utilizar librerías externas.

La prueba resultó exitosa, pudiendo enviar información (cadenas de texto) desde el computador hasta una página web previamente configurada, la cual se detallará en el siguiente capítulo. Cabe destacar que el uso de este perfil Bluetooth fue solo por simplicidad y próximamente se hará uso de un perfil acorde al proyecto.

6. IMPLEMENTACIÓN DE LA SOLUCIÓN DE LADO DEL SERVIDOR

Como se comentó en el final del capítulo anterior, se ha hecho uso de una página web básica en un servidor Linux contratado y configurado completamente. Esto con el fin de poder ir realizando pruebas y al mismo tiempo tener un espacio preparado para el backend de la aplicación. En este capítulo se ahondará en su adquisición y características generales.

6.1. Requerimientos

Desde un inicio se tuvo en cuenta la necesidad de tener conexión a un servidor capaz de ofrecer el almacenamiento de datos, procesamiento de los mismos y despliegue de ellos a los profesionales de la salud, es por ello que en un inicio se hizo búsqueda de una página web con dichas características. Pero, rápidamente se hace necesario distinguir los servicios ofrecidos para el desarrollo de backend con acceso universal (dirección ip pública).

6.1.1. Tipos de alojamiento Web

El hosting[19] (alojamiento web en español) consiste en un servicio prestado a través de internet, por el cual se permite el almacenamiento (a través del uso de un servidor) de distintos contenidos en la web.

Para comprender a cabalidad de qué trata, es necesaria una distinción y definición previa de la palabra servidor en nuestro contexto: Se pueden definir dos grandes formas de servidor, uno físico y uno informático (comúnmente llamado servidor web). Por una parte, el servidor físico provee de recursos electrónicos como lo puede ser: Capacidad de memoria ram, disco duro, procesamiento, entre otros. Por otro lado, el servidor informático.^{es} aquel **programa** con la capacidad de procesar peticiones web y opcionalmente (habitual) responderlas.

En los últimos años, el mercado de este tipo de servicios ha ido en aumento y por tanto se han generado distintas alternativas de obtenerlo. Existiendo más de una docena de formatos distintos con los cuales se ofrece, pero en este apartado solo veremos los más relevantes para el proyecto (dejando de lado servicios específicos como el de correos electrónicos, entre otros).

1. **Alojamiento compartido:** En este alojamiento se permite compartir un mismo servidor físico entre distintos clientes y sus páginas web. Entre sus ventajas está el corto periodo de tiempo para la puesta en marcha y el costo, aunque entre sus principales desventajas se encuentra el rendimiento general, la poca flexibilidad de configuración y la disminución en seguridad y estabilidad.
2. **Servidor virtual (VPS):** Servicio en el cual se comparten los recursos de hardware, pero se aíslan las distintas instancias a través de máquinas virtuales. Entre sus ventajas está el control completo del sistema al ofrecer comúnmente una base solo de sistema operativo, programas base y dirección IP pública. Su principal desventaja es un bajo rendimiento bajo alta carga.
3. **Servidor dedicado:** Alojamiento en el cual se emplea una computadora con todos sus recursos al servicio de un único cliente, permitiendo tercerizar el cuidado de la máquina y la conexión a internet de la misma. Entre sus principales ventajas está el control casi absoluto del servicio a configurar y el alto rendimiento al no compartir el hardware. Su principal desventaja es el costo asociado a este servicio tan personalizado.

4. **Alojamiento en la nube:** Basado en las tecnologías más innovadoras, permite el uso de múltiples máquinas actuar como un sistema unificado. Dotando así de gran capacidad de expansión (crecimiento en tiempo real). Su principal desventaja al igual que el anterior, es su alto costo.

Como se puede observar entre las opciones disponibles, existen diferencias tanto a nivel técnico como económico que hacen necesario un análisis más específico de los requerimientos del proyecto.

6.1.2. Requerimientos del proyecto

El proyecto requiere de una serie de características a nivel de Backend, entre las que destacan:

Bajo costo

Alta flexibilidad

Alto rendimiento

Al estar en una etapa de desarrollo se considera irrelevante la capacidad de cómputo y de ancho de banda ofrecido. Así y considerando las opciones de la anterior sección se destaca la opción de un alojamiento de tipo VPS, con el cual poder acceder a una máquina virtual con todas las prestaciones de un sistema operativo (como lo es acceso a socket e instalación manual de herramientas).

6.2. Hosting VPS

A partir de los requerimientos del proyecto se hace notoria la posible falta de rendimiento al escoger el alojamiento VPS, es por esto que al momento de gestionar el contrato con una prestadora de servicio se priorizó aquellas que tuvieran almacenamiento SSD (Disco en estado sólido en español), aumentando así notoriamente el rendimiento de la máquina virtual. Además de seleccionar un proveedor local del servicio tanto por temas de soporte, como por el tiempo de respuesta en las peticiones hechas desde el mismo territorio nacional. Además de lo anterior, se destaca la necesidad de un servidor con la opción de manejar los puertos del equipo y no solo el 80 (destinado a peticiones web).

De esta forma se logra acceder a una máquina con las siguientes características:

Empresa: OpenCloud en Chile por \$6.000 pesos mensuales.

IP: xxx.xxx.xxx.xxx

Ubicación: Chile.

Ancho de banda: 60 [Mb/s]

Tráfico: 2 [TB] (Bajada + Subida) mensuales.

Sistema Operativo: Ubuntu v16.04 - 64 bit

Procesador: Monoprocesador a 2.2 [GHz]

Ram: 2 [GB]

Disco Duro: 20 [GB]

En una primera instancia se maneja comunicación síncrona a través de peticiones HTTP comunes y experimentación con WebSocket (detallado más adelante). Proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP), permitiendo una comunicación entre el servidor y el navegador, método empleado para refrescar de forma automática la página principal en caso de algún ingreso nuevo de datos. Para esto cada 100 [ms] se envía desde el navegador una marca de tiempo correspondiente al último dato recibido, en el servidor si ésta marca no corresponde al último dato existente, se envía una actualización de la cantidad de datos y el último dato actual, además de la nueva marca de tiempo.

A través de una petición GET se entregan los datos a ingresar:

http://galenoproject.sytes.net/data/?name=ECG&data=8&save=yes

En el ejemplo anterior se agregará un documento a la base de datos con los parámetros name y data, el parámetro save es utilizado para controlar si se debe o no almacenar dichos datos.

Luego, a partir de la dirección: *http://galenoproject.sytes.net/* se puede observar:

Datos:

Cantidad de datos: 58 | Último dato: 87

Fig. 6.1: Primera versión de la página web

Lo cual corresponde a la cantidad de datos actual en la base de datos y el último dato recibido.

En las siguientes secciones se detalla la configuración del equipo y las herramientas empleadas hasta el momento.

6.3. Servidor Tornado

Framework de desarrollo web basado en Python, ofrece alto rendimiento, gran escalabilidad y ofrece hasta 10.000 usuarios conectados bajo conexiones continuas gracias a su tratamiento asíncrono de las entradas y salidas del sistema. Utilizado para procesar solicitudes como entrada o salida de datos.

Implementado: Python v2.7.12, Tornado v4.5.1

6.4. MongoDB

Base de datos no relacional destacada por su escalabilidad, rendimiento y gran disponibilidad. Utilizado para almacenar los datos recopilados y su posterior lectura.

Implementado: MongoDB v3.2.13

6.5. Nginx

Servidor Web/Proxy Inverso de alto rendimiento. Utilizado para redirigir de forma interna peticiones entrantes a distintos servidores que se están comunicando en distintos puertos de forma interna.

Implementado: Nginx v1.10.0

6.6. DNS No-IP

Compañía que provee DNS dinámicos por pago o gratuitos con ciertas limitaciones, como lo pueden ser las alternativas de dominios disponibles. Utilizado para enmascarar la ip pública del hosting y acceder a través de un nombre de dominio.

Implementado: galenoproject.systes.net

7. IMPLEMENTACIÓN DE LA SOLUCIÓN DE LADO DEL CLIENTE ANDROID

7.1. *Servicios en Android*

La necesidad de utilizar servicios para el proyecto es que no necesariamente se requerirá de interacción con el usuario para ciertas tareas, como el acoplamiento automático del dispositivo y el celular, la búsqueda de servicios Bluetooth o la comunicación del servidor hacia y desde el dispositivo. Es por esta razón que un subproceso no cumple con lo requerido, ya que solo podría existir dentro del tiempo en que algún usuario haga uso de la aplicación (la mantenga abierta).

Los Servicios[20] en Android se pueden comportar de tres maneras a grandes rasgos:

Servicio iniciado: Proceso en segundo plano con una tarea preestablecida y que al cabo de ejecutar se detendrá, normalmente no retorna algo y no interactúa con la actividad que lo generó.

Servicio de enlace: Proceso en segundo plano que se enlaza de componentes para existir, permite tareas que retornan valores y es capaz de interactuar con una o más componentes, incluso bajo comunicación entre procesos (IPC). Se destruye si la no hay componentes enlazados.

Servicio iniciado y de enlace: Servicio que puede ejecutarse de forma indefinida (tarea preestablecida) y que permite enlaces de componentes para interactuar y comunicarse. Es la unión de ambas características anteriores en un mismo servicio.

7.2. Implementación de servicios y su comunicación en Android en función de los requerimientos del proyecto

Para el apartado asociado a los servicios en Android se comienzan las pruebas con el fin de conseguir las necesidades del proyecto: Autoconexión y envío de datos por internet. Ambos procesos deben ser llevados a cabo sin intervención del usuario y de forma permanente en cuanto la aplicación esté instalada y configurada a un equipo (placa principal con módulos de sensores).

Para esto se trabaja con la idea de un servicio que contenga dos procesos (no necesariamente hilos, dado que hay que detallar el funcionamiento del dispositivo BLE para la comunicación de sus datos). Por esta razón se analiza el funcionamiento de las clases Services e Intent Services, en donde su diferencia radica en el objetivo de su ejecución: Services para ejecución indefinida hasta su detención manual e Intent Services para ejecución de una tarea concreta (contempla la creación propia de un hilo de ejecución definida).

Luego para el tema de comunicación con el servicio existen distintas alternativas: AIDL, Binder y Messenger. AIDL usado para comunicación entre procesos de forma primitiva, Binder para comunicación solo con la aplicación contenedora del servicio (ventaja de poder acceder directamente a métodos públicos del servicio) y Messenger que es una forma de comunicación entre procesos con estructura basada en AIDL pero de mayor nivel y facilidad de uso por medio de un mensajero. Esta tarea queda en espera para trabajar con el dispositivo BLE, configurando sus parámetros e integrando su funcionamiento al módulo central de procesamiento Arduino, aunque se deja esbozado el servicio a necesitar, mezclando un servicio de la clase Services creado solo una vez para luego que las próximas ejecuciones de la aplicación se enlacen a este servicio (definiendo así un servicio indefinido contenedor y manipulador de la interacción con el bluetooth a través de IBinder).

8. CONFIGURACIÓN RN4020

8.1. *BLE (Bluetooth 4.0)*

El módulo de comunicación consiste en el chip RN4020 de MicroChip, el cual opera bajo el estándar BLE (Bluetooth Low Energy). La tecnología BLE consiste en un nuevo estándar de Bluetooth a partir de su versión 4.0, en la cual se optimiza el tamaño y consumo de potencia de los dispositivos, dentro de su implementación cabe mencionar distintos actores y definiciones con los cuales opera:

-GAP (Generic Access Profile): Dentro de la GAP se definen varios tipos de roles, pero estos están implementados bajo el concepto en que los dispositivos BLE pueden actuar con rol Central o un rol Periférico. Los dispositivos con rol periférico son pequeños, de baja potencia y que pueden conectarse a un dispositivo central mucho más potente. Los dispositivos periféricos pueden ser: monitor de frecuencia cardíaca, una tarjeta de proximidad BLE, etc. Por lo general, los dispositivos con rol Central usualmente son teléfono móvil, tableta o PC los cuales poseen mucha más potencia de procesamiento y memoria.

-GATT (Generic Attribute Profile): Define de qué modo dos dispositivos Bluetooth de bajo consumo transfieren sus datos, haciendo usos de los conceptos de Servicios y Características. Además se hace uso de un protocolo de datos genéricos llamados Attribute Protocol (ATT) en el cual se almacenan los servicio, características y datos relacionados en una simple tabla de consultas en cuya entrada se utiliza un ID de 16 bits. Una vez que ya se encuentra establecida la comunicación entre los dos dispositivos, entra en juego los servicios y atributos definidos por GATT, es decir, que ya

tuvo que haber concluido la etapa de Anuncio y asociación de los dispositivos gobernado por las definiciones establecidas por GAP. Cabe mencionar que los servicios y características descritas en la GATT, la conexión entre el dispositivo periférico y el central es exclusiva, es decir, que solo puede haber una comunicación a la vez. De esta manera, para que otro dispositivo pueda conectarse al periférico, primero debe desconectarse de su enlace actual.

-Anunciación de un dispositivo: Hay dos modos diferentes de realizar el envío de datos en un anuncio GAP : Advertising Data y Scan Response. Los datos enviados en cada uno de los anuncios forman el llamado Payload del mensaje. Ambas cargas son idénticas y pueden almacenar hasta 31 bytes de datos, pero sólo los mensajes de advertising son obligatorios para la norma, ya que ésta contiene siempre la información útil que constantemente se está transmitiendo desde el dispositivo periférico al dispositivo central. Los datos de los mensajes Scan response actualmente son opcionales y son utilizados para enviar una secuencia de datos secundaria a pedido del dispositivo central. Esto permite a los desarrolladores de equipos ingresar datos adicionales que se entregan en el momento de realizar el anuncio, como indicar el nombre del dispositivo, por ejemplo.

En la figura 8.1 se puede visualizar el proceso de anuncio por parte de un dispositivo periférico a un equipo central y de cómo entran en juego los mensajes del tipo Advertising data y Scan data.

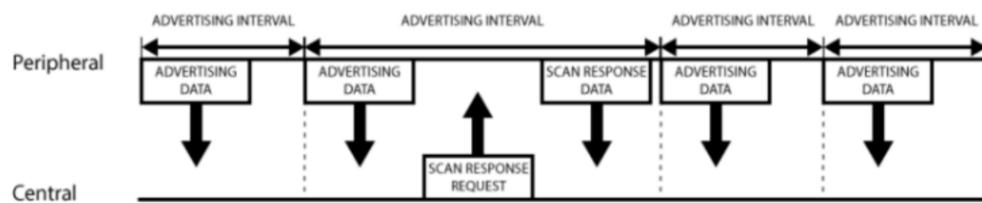


Fig. 8.1: Proceso de anuncio de un dispositivo.

Un dispositivo periférico puede definir el intervalo de anuncio, y en cada tiempo en que se alcance este intervalo de anuncio, se enviará un paquete de datos llamado

Advertising data. A intervalos largos de anuncios, el dispositivo reduce considerablemente el consumo, pero el equipo es menos sensible a las respuestas si el intervalo de anuncio es de 2 segundos en lugar de 20ms. Si algunos de los dispositivos que se encuentran escuchando estos mensajes de anuncio por parte del periférico está interesado en recibir los datos que agrega el mensaje Scan Payload (y a su vez está disponible en el dispositivo periférico), entonces es posible enviar un mensaje que indica que se requiere el Scan Payload del dispositivo, a lo cual este le responderá con los datos solicitados.

-Transacciones GATT: Un concepto importante para entender GATT es la relación servidor / cliente. El periférico se conoce como el servidor GATT, ya que contiene los datos de búsqueda ATT y las definiciones de servicio y características, y el cliente GATT (el teléfono / tableta), que envía solicitudes a este servidor.

Al establecer una conexión, el periférico sugerirá un Intervalo de conexión.^a El dispositivo central, y el dispositivo central intentará volver a conectar en cada intervalo de conexión para ver si hay nuevos datos disponibles. Es importante tener en cuenta que este Intervalo de conexión.^{es} sólo una sugerencia, puesto que es posible que el dispositivo central no pueda cumplir con la solicitud porque está ocupado hablando con otro periférico o los recursos del sistema necesarios no están disponibles.

El siguiente diagrama ilustra el proceso de intercambio de datos entre un periférico (el servidor GATT) y un dispositivo central (el cliente GATT), con el dispositivo maestro iniciando cada transacción GATT.

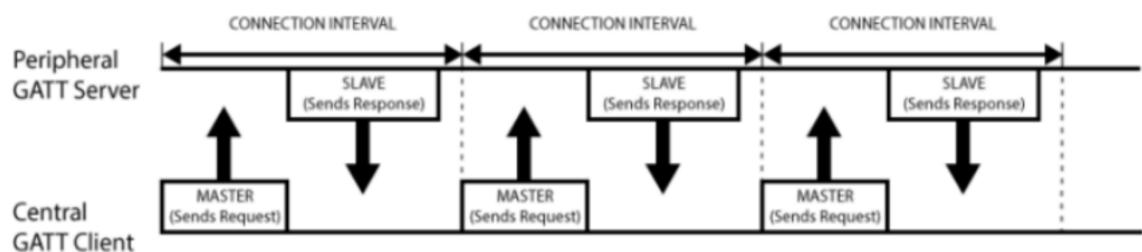


Fig. 8.2: Transacciones GATT.

-Perfiles, Servicios, Características y Descriptores: Las transacciones GATT en los dispositivos BLE están basadas en objetos anidados de alto nivel que se denominan Perfiles, Servicios y Características. El orden de estos elementos se muestra a continuación.

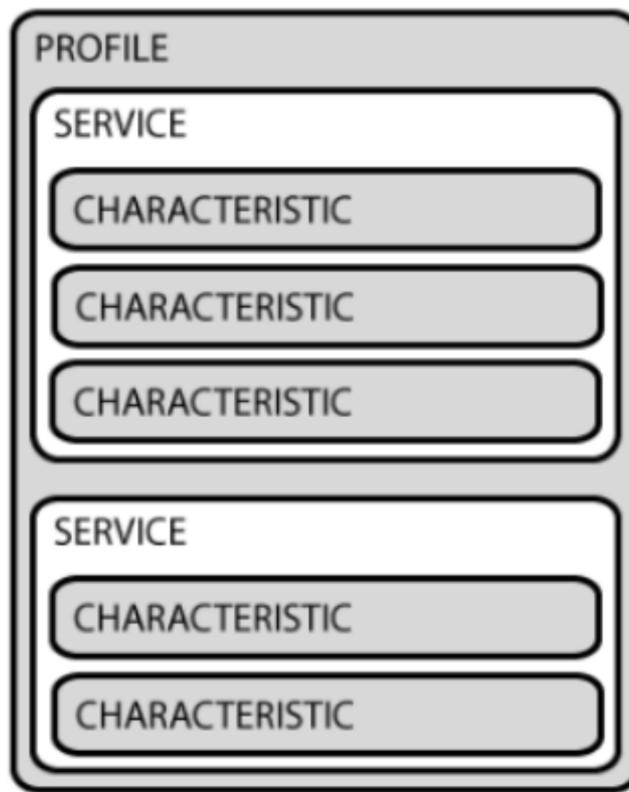


Fig. 8.3: Orden de objetos en transacciones GATT.

Un perfil no existe en el propio periférico BLE, sino que es una colección predefinida de servicios que ha sido compilada por el SIG de Bluetooth o por los diseñadores periféricos. El perfil de frecuencia cardiaca, por ejemplo, combina el servicio de frecuencia cardiaca y el servicio de información de dispositivos.

Los servicios se utilizan para dividir datos en entidades lógicas y contienen porciones específicas de datos llamados Características. Un servicio puede tener una o más características y cada servicio se distingue de otros servicios por medio de un ID numérico único denominado UUID, que puede ser de 16 bits (para servicios BLE adoptados oficialmente) o de 128 bits (para servicios personalizados). Una lista completa de los servicios BLE adoptados oficialmente se puede ver en la página Servicios del Portal de Desarrolladores de Bluetooth.

El concepto de nivel más bajo en las transacciones GATT son las características, que encapsula un único punto de datos (aunque puede contener una matriz de datos relacionados, como valores X / Y / Z de un acelerómetro de 3 ejes, etc.). De forma similar a los Servicios, cada Característica se distingue a través de un UUID predefinido de 16 bits o 128 bits. También se utilizan para enviar datos al periférico BLE, ya que también se puede escribir en la característica. Se puede implementar una interfaz UART simple con un 'Servicio UART' personalizado y dos características, una para el canal TX y otra para el canal RX, donde una característica puede configurarse como de sólo lectura y la otra tiene privilegios de escritura. Una Característica contiene un único valor y entre 0 y n descriptores que describen el valor de la característica, vale decir, algún rango aceptable o la unidad en caso de ser un valor cuantitativo, hasta una breve descripción de texto.

-Topología de red a utilizar: Un periférico sólo puede conectarse a un solo dispositivo central (Como un teléfono móvil) a la vez, pero un dispositivo central puede conectarse a varios periféricos. Una vez ya se estableció la conexión entre un dispositivo periférico y un central, la comunicación puede ser bidireccional. Para los fines del proyecto se espera solo una conexión 1 a 1, pero se destaca la comunicación bidireccional posible con este estándar.

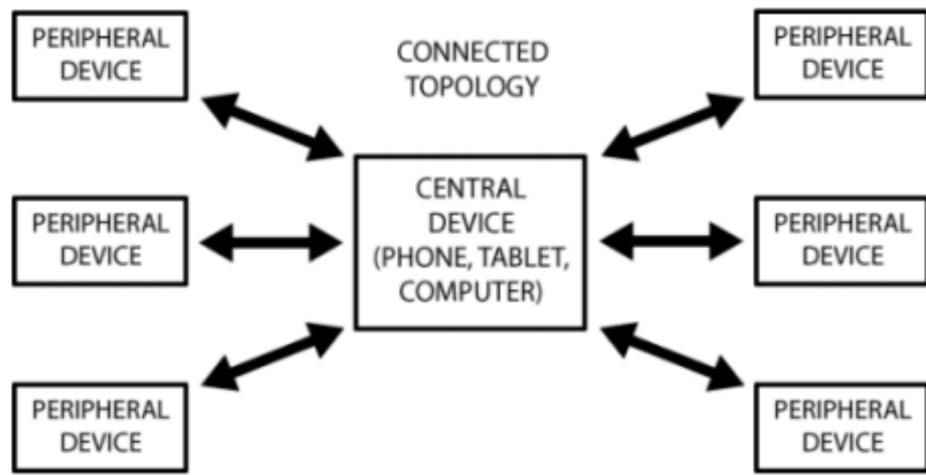


Fig. 8.4: Topología de red de dispositivos conectados.

Algo importante además de todo lo ya mencionado son los distintos estados en los que se pueden encontrar los distintos dispositivos Bluetooth, los cuales varían dependiendo de las versiones Bluetooth y los dispositivos mismos. A grandes rasgos se pueden distinguir 3:

Emparejado: Reconocimiento inicial, en donde se realiza un intercambio de llaves criptográficas (validación por medio de pin) para una comunicación cifrada inmediata.

Enlazado/ligado: Intercambio y almacenamiento de llaves criptográficas (validación por medio de pin), no necesariamente requiere una conexión inmediata, puede realizarse para posteriormente conectar los dispositivos.

Conectado: A partir de algún estado ya mencionado, se continúa con la conexión misma, con la que se espera intercambiar la información objetivo.

8.2. Características disponibles y selección, autoconexión Bluetooth (desde Android)

Para definir la comunicación Bluetooth con el chip RN4020 (con perfiles GATT) según lo establecido para el proyecto, se hace necesaria la creación de un perfil, servicios y características privadas. Por esto se definen los parámetros a continuación:

Comando:

PZ: Limpiar servicios

R,1: Reinicio

SF,1 y R,1: Restablecer de fábrica

SR,20402000 y R,1: Características de chip RN4020

SS,00000001 y R,1: Habilitación de Servicios Privados

PS,66ecf52ce19f11e48a001681e6b88ec1: Servicio con UUID (random) privada

PC,UUID,10,01,01 y R,1: Característica asociada a servicio previo

SUW,UUID,01: Escritura en característica asociada

La mayoría de opciones es utilizada bajo codificación hexadecimal, como lo son tamaños y ciertas características. A partir del o anterior se establecen 2 formas de reconexión:

Auto anuncio: El dispositivo BLE al terminar una conexión automáticamente se hace visible.

Anuncio manual: El dispositivo BLE requiere de una señal para hacerse visible (con la capacidad de recibir otra instrucción para dejar de serlo).

Se decide usar la segunda opción dado que se optimiza el recurso energético y dota de seguridad al dispositivo al no permitir conectarse a cualquiera sin previa interacción directa con el mismo. Además, se utiliza lo aprendido en el capítulo anterior (Servicios en Android para reconexión) para bajo la documentación del chip RN4020 comunicar (bajo notificaciones que no requieren peticiones del cliente Android) el dispositivo BLE con la aplicación. Cabe destacar que la reconexión se puede hacer automática al especificar la conexión GATT, pero esta demora alrededor de 10 segundos.

9. INTEGRACIÓN DE LAS COMPONENTES DE LA SOLUCIÓN

9.1. MLDP RN4020

El concepto de MLDP (Microchip Low-energy Data Profile) propuesto por Micro-Chip no se refiere a otra cosa que a un servicio GATT, como cualquier otro, pero con la capacidad de comunicarse directamente entre RF y el UART del dispositivo BLE, sin la necesidad de pasar por peticiones a través del microcontrolador.

Este servicio construido sobre GATT, corresponde a un servicio privado con la finalidad de simular la operación clásica de un perfil de puerto serial Bluetooth (SPP).

El rendimiento depende en gran medida de los distintos parámetros de conexión, como los asociados a la frecuencia de la comunicación entre el dispositivo central y periférico. Un mayor rendimiento requiere una mayor frecuencia y por ende un mayor consumo energético, acortando así la carga de la batería del dispositivo. Por esta razón es que dependiendo de la aplicación, es imperante un balance correcto de estas características en la comunicación.

Para hacer uso de este recurso, se debe habilitar el bit asociado usando el comando SR,10000000. Una vez ya habilitado, se deben especificar los parámetros de conexión y establecer un enlace activo entre un dispositivo central y otro periférico. Luego de esto, todos los datos de entrada provenientes del módulo UART del RN4020 son enviados al otro dispositivo como un flujo de datos.

The first item is an example code for working with the module and MLDP from android, it uses Bluetooth GATT classes which was implemented from Android 4.3 (API 18)

9.2. Implementación de código ejemplo en Android

Se hace uso de código de ejemplo provisto por la página oficial del módulo Bluetooth RN4020 de MicroChip [21] con el fin de estudiar las clases empleadas y su uso. A partir del estudio de este ejemplo se establecen las bases necesarias para comunicarse con el chip RN4020 de la forma deseada, destacando el uso de Android 4.3 (API 18) o superior por poseer la incorporación de perfiles bluetooth para la versión 4.0 (Low Energy).

9.3. Arduino actuando de servidor con RN4020

Para el correcto funcionamiento del chip RN4020 con el perfil privado a emplear y sus parámetros de conexión se hace uso de la librería SoftwareSerial, la cual ofrece la comunicación de transmisión y recepción entre la placa Arduino el chip en cuestión. Además de lo anterior, se hace uso de dos librerías para el manejo del sensor de temperatura (con el cual se realizan pruebas preliminares): OneWire y DallasTemperature, las cuales ofrecen una interfaz limpia para el manejo de los datos otorgados por el sensor.

Se prueban instrucciones base como Baudios, parámetros de conexión, características de seguridad, tipo de perfil, servicios o características, entre otros según la guía de usuario oficial de Microchip [23].

Por último, se analizan distintas combinaciones de configuración según los requerimientos, dentro del área de seguridad, velocidad, potencia de transmisión y servicios entregados según la guía de usuario oficial de Microchip [23].

10. PROTOTIPO FUNCIONAL V1

Para continuar con el desarrollo del proyecto, se deben hacer funcionar las distintas partes (sensores, microcontrolador, módulo Bluetooth, Smartphone y servidor) como un todo y para ello se crea un primer hito de prototipado con el cual se espera alcanzar esta integración. Además de lo anterior, se espera poder hacer pruebas que asienten o desmientan decisiones de diseño y configuración, siempre con miras a los requerimientos generales del proyecto.

10.1. Configuración final RN4020

En el apartado del microcontrolador junto al módulo Bluetooth, se utiliza un archivo .ino con los parámetros necesarios y un ciclo de ejecución con el cual ofrecer la funcionalidad del envío de datos desde los sensores al Smartphone.

En esta primera configuración se hace uso de un Script para la configuración básica del chip RN4020 por parte del microcontrolador, se establece la visibilidad como activa desde el inicio de operación, se utilizan los parámetros por defecto al mínimo de conexión Bluetooth: Latencia (tiempo de respuesta), intervalo de conexión (frecuencia con que una comunicación es establecida), timeout(tiempo de espera antes de cancelar la comunicación) y se habilita la encriptación en la comunicación Bluetooth (AES-128, criptografía simétrica).

Respecto a la comunicación entre el microcontrolador y el módulo se observa que los Baudios máximos se encuentran en los 19.200 [S/s], pasando por 2.400 [S/s] y 9.600 [S/s].

10.2. Análisis de rendimiento por codificación

En esta oportunidad se le da prioridad a la configuración del ECG por sobre los datos de temperatura, pese ya a tener configurada la característica asociada a ambos. Por lo que se realizan distintas pruebas modificando parámetros relevantes como la potencia (de transmisión del chip RN4020), la seguridad, cantidad de datos por envío y Baudios. Cabe mencionar que los datos deben ser convertidos a hexadecimal para ser entregados al módulo Bluetooth, por lo que se deben convertir tanto en el microcontrolador como en la aplicación móvil.

En la figura 10.1 se utilizaron:

Distancia: Fija a 1 metro.

Potencia: Variable 4 y 7 a -2.5 [dBm] y 7.5 [dBm]

Seguridad: Enlazado y no enlazado.

Línea vista: No.

Datos: Hexadecimal de 1 [byte] representando enteros entre [0-255].

Datos por envío [Bytes]	Envíos	Baudios [S/s]	Seguridad	Potencia Tx	Tiempo [s]	Tasa [Bps]
1	1000	9600	off	4	46.653	21.43
10	1000	9600	off	4	65.807	151.96
20	1000	9600	off	4	86.873	230.22
20	1000	9600	off	4	86.873	230.22
20	1000	19200	off	4	43.241	462.52
20	1000	19200	on	4	51.431	388.87
20	1000	19200	on	7	51.479	388.51

Fig. 10.1: Resultados, pruebas de configuración ECG

Con lo cual se justifica el uso de paquetes de 20 datos (máximo permitido) por envío. Logrando así una frecuencia de al menos 200 [Bps] en el envío de datos si fuere necesario. Además se destaca que la seguridad no afecta significativamente y dada la distancia, la potencia de transmisión tampoco. Pese a lo anterior se baraja la posibilidad de usar una resolución de 10 bits (resolución máxima del sensor) a través del uso de 2 bytes hexadecimales, alcanzando un máximo de 10 datos por envío.

10.3. Elementos en Android

Luego de haber obtenido la base de aplicación en función del ejemplo en la página oficial de Microchip, la cual entrega las herramientas para comunicarse con el chip RN4020, se busca dar forma a una aplicación que cumpla con las características del proyecto y que por el momento sea funcional en torno a lo obtenido hasta el momento.

Android es un SO ampliamente utilizado y por ende posee una gran comunidad a sus espaldas, las cuales frecuentemente otorgan librerías y mejoras constantes además de su propio responsable (Google). Con lo anterior en mente, se hace la búsqueda de distintas herramientas capaces de ofrecer funcionalidad y flexibilidad al proyecto, las cuales se describirán brevemente en conjunto con una explicación de por qué su uso frente a otras alternativas (no necesariamente mencionadas). Cabe destacar que el presente documento se hace al año 2018 y por ende con las capacidades actuales presentes en el desarrollo de una aplicación Android.

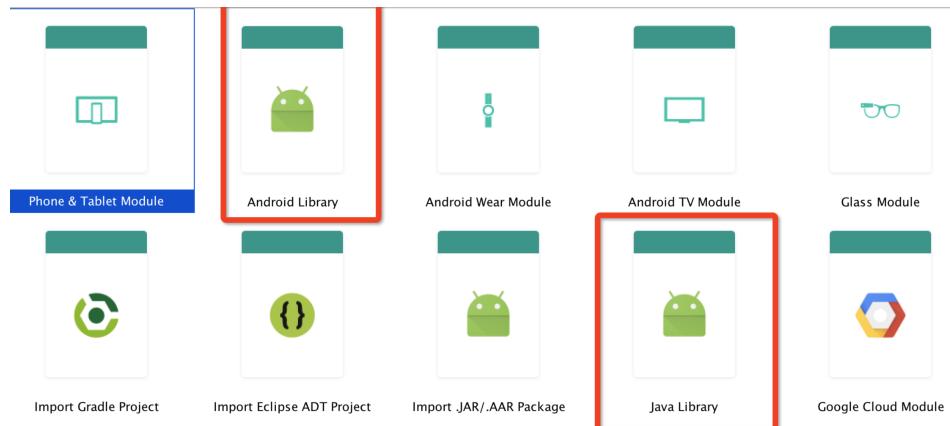


Fig. 10.2: Uso de librerías en Android

10.3.1. RecyclerView

En Android, existen diversas formas de generar visualmente listas para los usuarios, pero actualmente gracias a Google se cuenta con algo llamado RecyclerView. La cual funciona como contenedor de una lista como cualquier otra, pero pensada en el alto rendimiento (gran cantidad de elementos) y una estética cuidada en la fluidez de la misma.

Para una aplicación como la presente no se hace necesario el uso de esta librería, dado que no se tiene una gran cantidad de elementos que desplegar en pantalla, pero siempre es bienvenida la velocidad de codificación que proporciona al presentar una arquitectura de plantillas sobre las cuales generar elementos. Así, es utilizada en el proyecto actual para contener distintos tipos de conexión con su respectivo estado: Al servidor, con el módulo Bluetooth, y con el servicio propio de la aplicación.

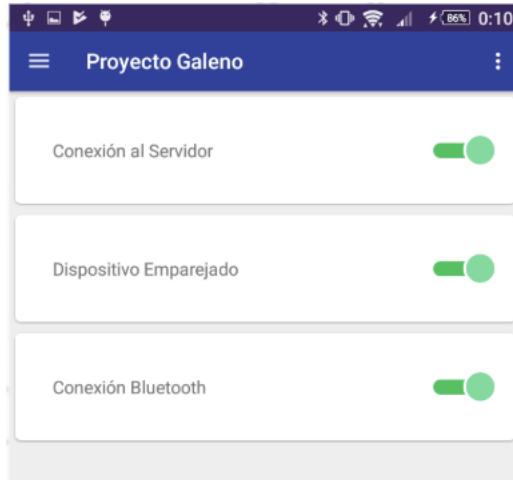


Fig. 10.3: Uso de RecyclerView

10.3.2. Códigos QR

Dado que el proyecto utiliza un emparejamiento con un dispositivo Bluetooth, es importante recordar que para lograr esto existen distintas técnicas, entre las que destacan el escaneo (alto consumo de potencia, lento y engorroso si existen diversos dispositivos), ingreso manual de la dirección física MAC del dispositivo al cual conectarse (lento y engorroso) y por ingreso automático (utilizando la cámara como medio físico y alguna codificación ya establecida). Es en este último punto en donde los códigos de barra surgen como una opción rápida y cómoda, más aún cuando se utiliza a los códigos QR, los cuales estandarizan la codificación básica de cualquier cadena de texto. Otorgando así comúnmente direcciones URL, de las cuales se obtienen diversos recursos (imágenes, video, texto, entre otros).

Es por lo anterior que se decide emplear a los códigos QR como la forma más efectiva de entregar la dirección física de los módulos Bluetooth, con la cual la aplicación podrá efectuar el emparejamiento respectivo.

Para esta acción existen diversas librerías disponibles, pero de entre ellas se destaca a ZXING, la cual ofrece tanto la creación como la lectura de distintos tipos de códigos. Hace unos años esta librería solo podía usarse en conjunto con una aplicación, pero actualmente es posible encontrarla por separado: QRCodeReaderView es una versión modificada de ZXING que permite leer códigos QR de forma simple y rápida (aunque con pocas opciones de personalización de interfaz).



Fig. 10.4: QR con MAC de dispositivo Bluetooth

10.3.3. Fragment

Este es un elemento básico en Android y es utilizado como una versión menor de las Activitys, es utilizado comúnmente como contenedor de vistas menores y controlados en conjunto por una Activity. Su uso en el proyecto fue decidido por el menú lateral escogido para albergar las distintas vistas de la aplicación.

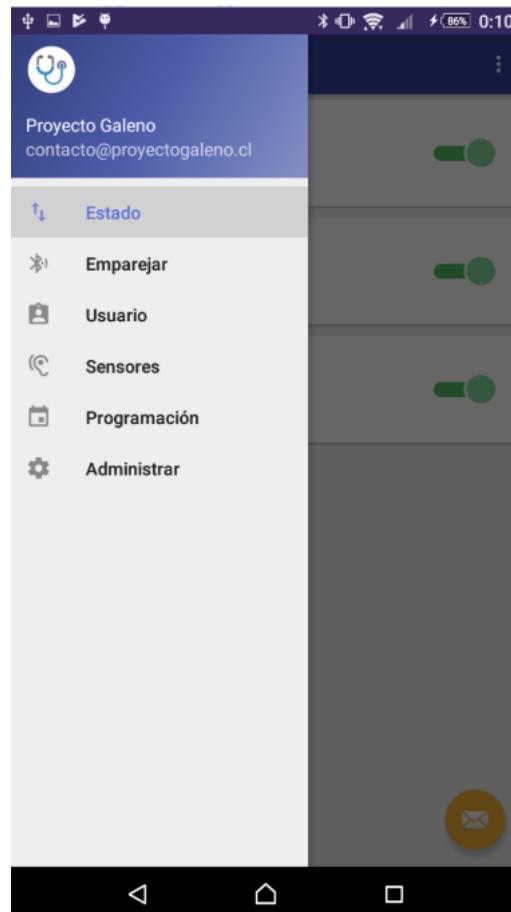


Fig. 10.5: Menú lateral comprendido por distintos Fragment

Si bien no todas esas vistas fueron implementadas (Usuario y Programación) en este punto, se tuvo claridad de las mismas desde un inicio y se tuvo consideración en la facilidad de uso, agrupando las opciones según función.

10.3.4. MPAndroidChart

La visualización es una funcionalidad necesaria con el fin de validar pruebas y otorgar al mismo tiempo un prototipo con mayor solidez hacia la contraparte. Es por esta razón que se analizaron distintas opciones de librerías, dentro de las cuales destacó MPAndroidChart, librería nativa de Android, con gran rendimiento y opciones de personalización. Es sencilla de utilizar y ofrece la posibilidad de gráficos en tiempo real, dependiendo del dispositivo se pudieron obtener sobre 1000 puntos al mismo tiempo con gran fluidez, algo realmente importante si se considera que el ECG posee un gran flujo de datos (típicamente de 50 [Hz] - 150 [Hz]).

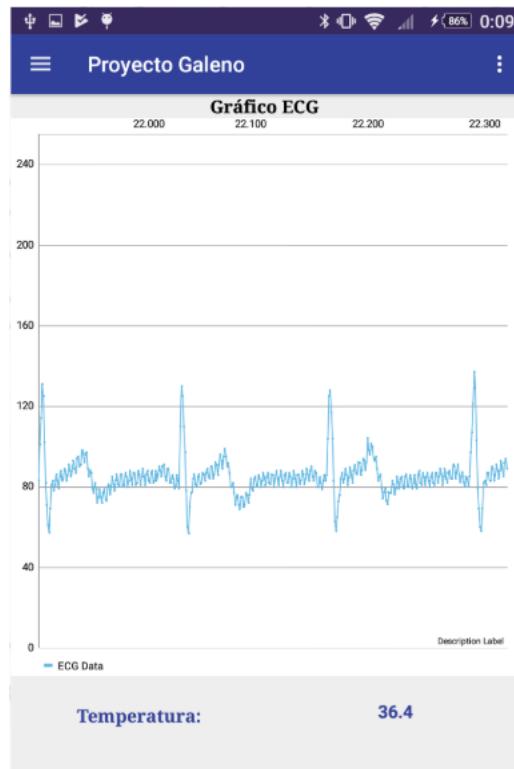


Fig. 10.6: Gráfico en tiempo real con datos de ECG por Bluetooth

Como se puede observar en la figura 10.6, se observa gran cantidad de ruido, aspecto corregido hasta cierto punto en posteriores iteraciones.

10.3.5. Shared Preference

El proyecto requiere que la dirección MAC sea almacenada en alguna memoria persistente con el fin de poder reconectar los dispositivos de ser necesario: Al apagar el Smartphone o el Arduino, al aumentar demasiado la distancia, entre otros. La dirección MAC no es otra cosa que una cadena de texto, por lo que un almacenamiento básico es suficiente para los objetivos del proyecto.

Por lo anterior se decanta por la utilización de Shared Preference (Preferencias compartidas) como el método de almacenamiento, esta es una característica que ofrece Android de forma base, la cual permite por aplicación un almacenamiento tipo llave-valor, estilo diccionario. La cual es accesible solo desde la misma aplicación o compartida entre distintas aplicaciones, pero con persistencia al cierre de las mismas (lo que realmente es fundamental para el proyecto).

10.3.6. Inicio automático de Servicio y otros

En esta sección se detallará un poco más sobre los elementos del menú y el comportamiento del primer prototipo del proyecto:

Estado: Recycler View con distinto estados de conexión (Servidor, BT y Servicio).

Emparejar: Lector de códigos QR para asociar MAC de un dispositivo.

Sensores: Presentación visual de los datos recibidos en tiempo real.

Usuario: Datos almacenados en Android del paciente (aún sin implementar).

Programación: Futura mejora para mostrar próximas tomas de datos.

Administración: Manejo del estado del servicio principal.

Respecto a su comportamiento, se establece por medio del Manifest (archivo de cada proyecto en Android que controla permisos, articula las distintas Activity y ciertos parámetros generales) la apertura del servicio al arranque del SO, mientras el servicio se encuentre activo se encarga de mantener encendido el adaptador Bluetooth del Smartphone y se utiliza la opción de enlazado provisto por el SO Android (por fuera de la aplicación y mencionado al final del capítulo 8).

11. PROTOTIPO FUNCIONAL V2

Luego de un primer prototipo, se analiza el funcionamiento general para planificar cambios y mejoras necesarias. Entre las que destacan un control sobre el envío de datos desde los sensores y una optimización de código en Arduino (relevante por la baja frecuencia de operación que posee frente a las exigencias del proyecto).

Los detalles se presentan a continuación, aunque adelantando que los cambios principales de esta nueva versión fueron realizados a la programación del microcontrolador y su sincronización con la aplicación.

11.1. Control de sensores por Android

Un aspecto fundamental para el proyecto es la posibilidad de controlar el inicio y el fin de las mediciones desde la aplicación, permitiendo en una iteración posterior su control desde la web.

Para este efecto, se desarrolla un protocolo de comunicación con el cual se espera sincronizar las acciones entre los dos principales entes (microcontrolador con módulo Bluetooth y aplicación Android).

Se comienza estableciendo un conjunto de instrucciones homogéneas con las cuales los dispositivos puedan ejecutar órdenes:

00: Detener medición

10: Iniciar solo medición de ECG

01: Iniciar solo medición de T

11: Iniciar ambas mediciones

Como se puede observar, se utiliza un sistema binario en el cual con 2 bit se logran controlar 4 estados fundamentales. Se escoge esta arquitectura dada su fácil escalabilidad y sencilla comprensión al asignar 0 o 1 a cierta posición asociada (y preacordada) a algún sensor.

Esto en conjunto con un Handler (ejecución de una tarea en otro hilo) en Android, permite el control temporal de estas instrucciones. Logrando así que al momento que la aplicación recibe una petición de medicón, ésta sea comandada por el Handler que a su vez tiene un Timer asociado, con el cual se podrá controlar la ejecución de la medición por una cantidad de tiempo preestablecida.

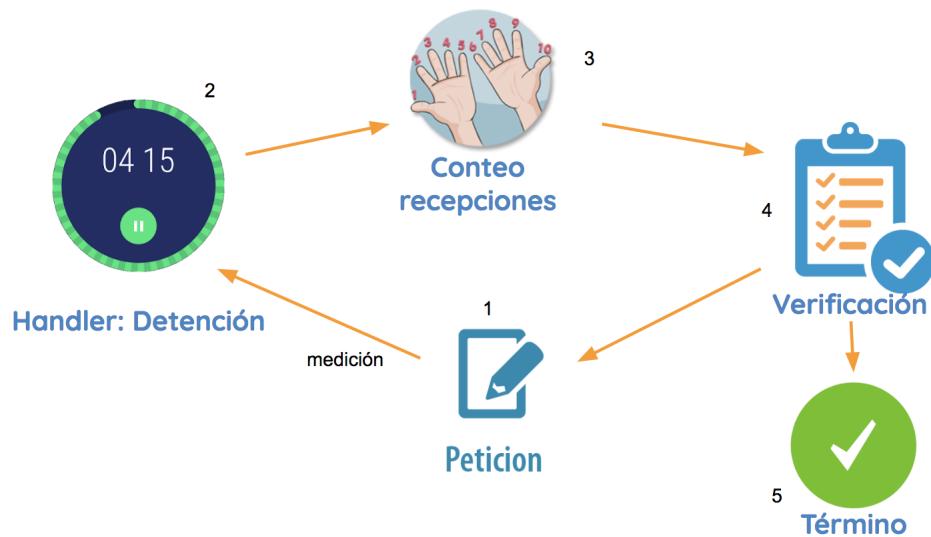


Fig. 11.1: Control de peticiones por Handler y repetición

En la figura 11.1 se pueden observar los distintos estados asociados a una medición: Petición, medición, detención, conteo, verificación y término.

Al acabar con la medición, el microcontrolador comunica la cantidad de paquetes enviados asociados a cada sensor, cantidad que es verificada con los recibidos en la aplicación y dependiendo de esto último se repite la petición de forma automática o se da por terminada la medición.

Las secuencias de control corresponden a una cadena de caracteres compuesto por AA y FF al inicio y término respectivamente de una medición, seguidos por 4 caracteres contenedores de 2 byte hexadecimales con el número de paquetes enviados (0 a 65536 paquetes).

Esto último supone una restricción en la cantidad máxima de paquetes enviados por medición:

ECG: Con una tasa de 150 [Hz] — Máximo 6.8 minutos por medición.

T: Con una tasa de 1 [Hz] — Máximo 18.2 horas por medición.

11.2. *Renovación de servidor Arduino*

Considerando la opción de microcontrolador escogido (Arduino UNO), se tiene una limitante importante en cuanto a la potencia de procesamiento presente, la cual se limita a la frecuencia de operación con la que cuenta el chip ATmega328 que es de 16 [MHz].

Por esta razón se hace indispensable una programación prolífica y con miras en la optimización en la ejecución de instrucciones. En este sentido, se realizan diversos cambios al código fuente del microcontrolador (se pueden ver con detalle en los Anexos):

Control del ciclo principal de operación bajo milisegundos a microsegundos.

Uso de punteros en vez de copia de memoria para el manejo de arreglos de char.

Uso de condiciones según precedencia y uso de else if según casos más frecuentes.

Control de estados por variables, minimizando verificaciones innecesarias.

11.3. Análisis de resolución y frecuencia para ECG

Pasando directamente al manejo del ECG por parte del controlador, se realizan pruebas para establecer frecuencias necesarias y verificar nivel de resolución mínimas, entre las que se destacan los siguientes resultados.

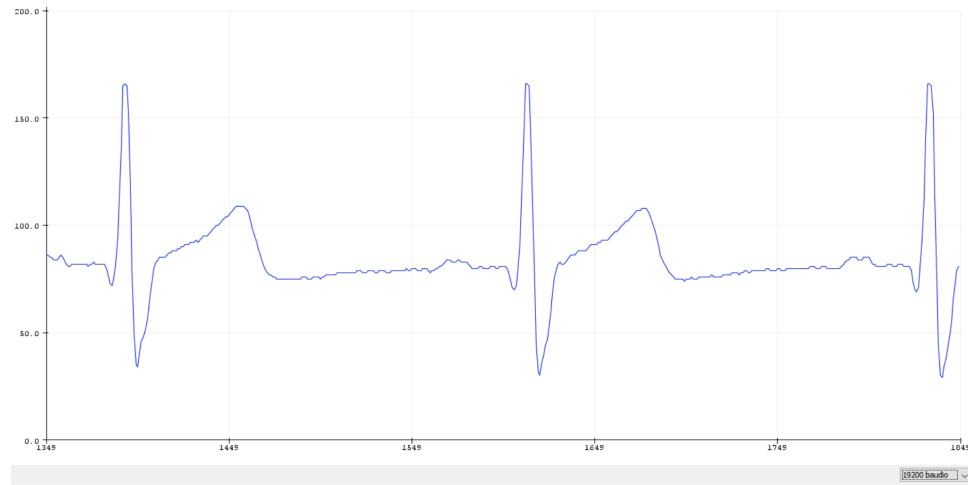


Fig. 11.2: ECG a 300 [Hz] y 8 bit de resolución

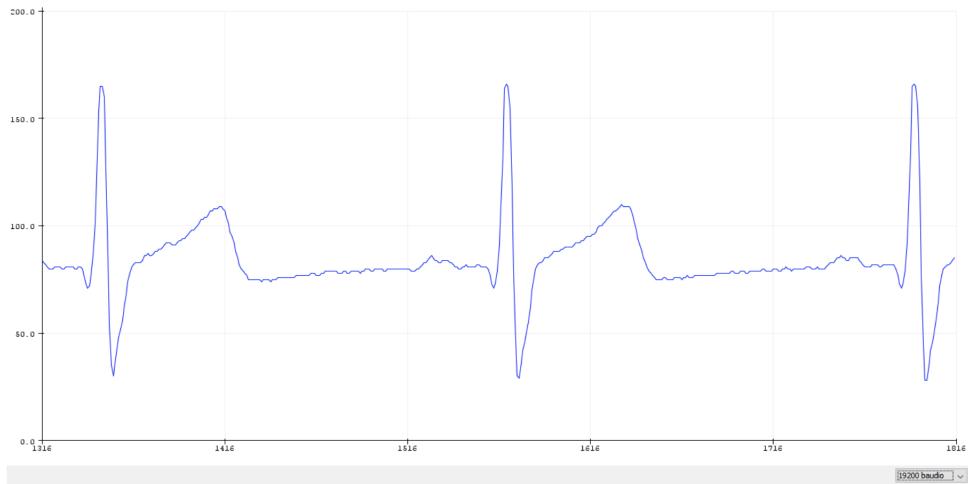


Fig. 11.3: ECG a 300 [Hz] y 10 bit de resolución

Como se puede observar en las figuras anteriores, la resolución obtenida utilizando 8 bit o 10 bit no representa un cambio significativo a nivel visual, mientras que a nivel de procesamiento y almacenamiento sí posee gran injerencia (valor máximo: 256 y 1024 respectivamente).

Esto es especialmente importante cuando se tiene en cuenta que la comunicación hexadecimal funciona de forma óptima con información condensada en múltiplos de 8 bit, puesto que la representación de 1 byte (8 bit) es posible por medio de solo 2 caracteres hexadecimales que son exactamente el byte de información.

Así, se decide emplear esta resolución para la captura y comunicación de los datos de ECG, sin miedo a perder información o tener sobrecarga de caracteres, en una comunicación con paquetes contenedores más grandes que la información mínima.

Se hicieron pruebas de frecuencia para el ECG, probando valores entre 50 [Hz] y 1.000 [Hz]. Como resultado de esto se estableció un mínimo de 50 [Hz] de operación, una base esperable de 150 [Hz] y un máximo de 300 [Hz], los anteriores valores por condensación visual de los datos (se observa una compresión horizontal de las gráficas a menores frecuencias y lo contrario para frecuencias mayores).

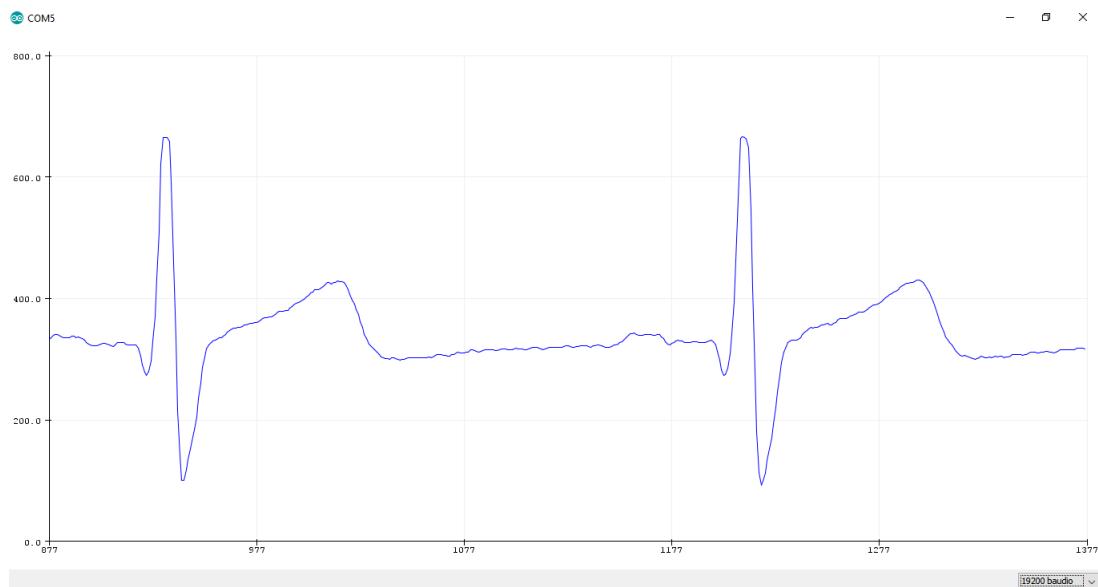


Fig. 11.4: ECG a 1.000 [Hz] y 10 bit de resolución

11.4. Base de datos local (comparativa Realm, SQLite)

De nada sirve obtener y comunicar datos si estos no son almacenados y estudiados posteriormente, es por ello que se contempló desde el inicio del proyecto el uso de una base de datos local para la aplicación (útil especialmente para mediciones sin cobertura) y una base de datos propia del servidor web.

En Android existen principalmente dos grandes alternativas respecto a bases de datos: SQLite y Realm (ambas relacionales y con esquema llave-valor). La primera de ellas, SQLite es un motor de base datos ampliamente utilizada y disponible desde los inicios del SO Android, sus grandes ventajas son la gran penetración que ya posee en los desarrolladores y los distintos sabores entendidos en las librerías que lo implementan.

La segunda alternativa es relativamente nueva pero con un potencial enorme, dada su facilidad de uso, gran rendimiento, visualización de datos con aplicaciones externas, disponibilidad en múltiples SO y su excelente documentación.

A continuación se presentan gráficas provistas por el equipo de Realm en comparativas de rendimiento [22] frente a distintas librerías en Android (en todas más alto es mejor).

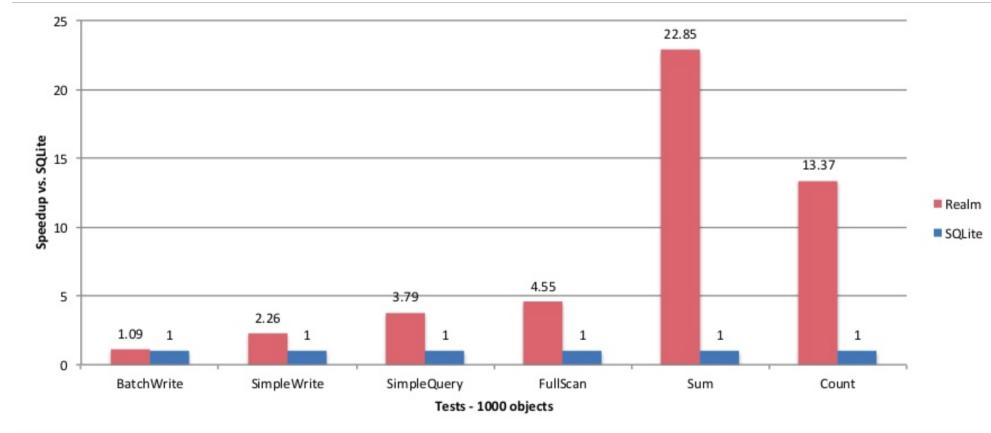


Fig. 11.5: Benchmark frente a SQLite con el uso de distintas librerías

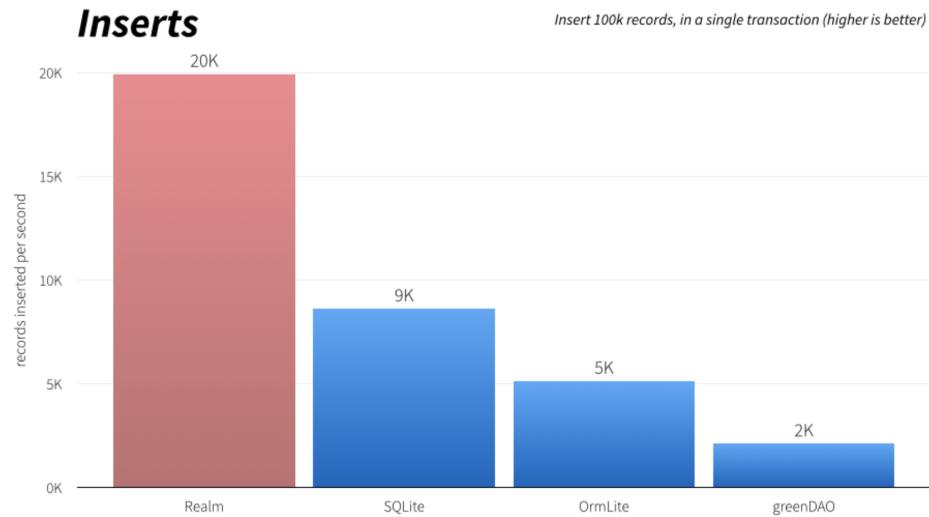


Fig. 11.6: Comparativa en escrituras a la base de datos

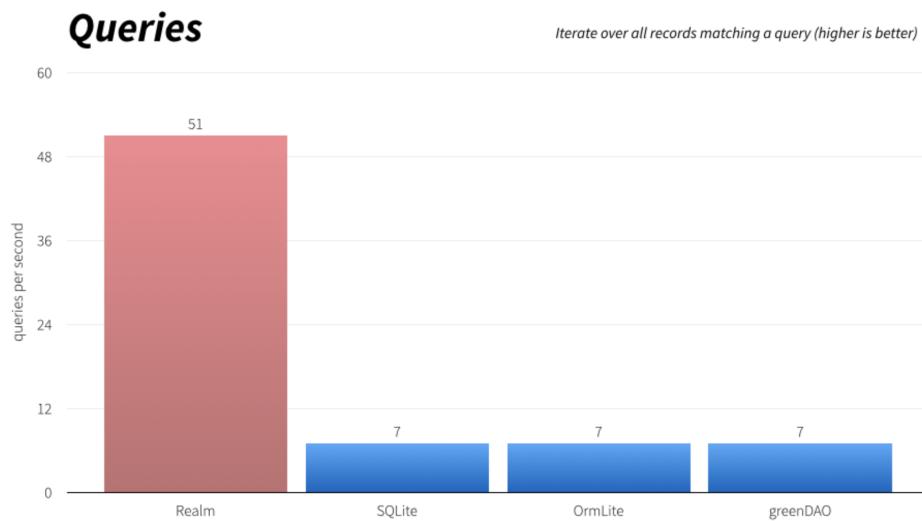


Fig. 11.7: Comparativa en lecturas a la base de datos

Es por las figuras anteriores y su resultado apabullante que se hace uso de Realm y no de otra base de datos local. Cabe mencionar que al ser más reciente, posee características interesantes como un manejo sencillo de hilos, instancias, creación de arreglo de objetos, entre otros.

11.5. Modelo tentativo para Realm

Si bien no se alcanza a implementar la base de datos ya escogida, se articula un modelo tentativo de las tablas que contendrán la información de las mediciones. Estableciendo de esta manera los datos y la relación presente entre ellos, se presenta la llave primaria en negrita para cada tabla.

<i>Dato</i>	<i>Tipo</i>
Rut	String
Nombre	String
Sexo	String
MAC	String
Hospital	String
Código Sistema	int
Edad	int
Teléfono	int
Teléfono emergencia	int
Teléfono emergencia 2	int
Nº Ficha	int

Tab. 11.1: Tabla Paciente: Datos personales del paciente

<i>Dato</i>	<i>Tipo</i>
Rut	String
Fecha_hora_min_seg	String
Duración	int
Sensores	int

Tab. 11.2: Tabla Mediciones: Al iniciar una medición almacena los datos relacionada a esta por posible retransmisión necesaria hacia el servidro web

<i>Dato</i>	<i>Tipo</i>
Rut	String
Fecha_hora_min_seg	String
Duración	int
Enviado	boolean

Tab. 11.3: Tabla ECG: Al iniciar una medición de ECG

<i>Dato</i>	<i>Tipo</i>
Fecha_hora_min_seg	String
Contador	int
Valor	int

Tab. 11.4: Tabla Datos ECG: Se escribe con cada dato, pero asociado a una medición(Fecha_hora_min_seg)

<i>Dato</i>	<i>Tipo</i>
Rut	String
Fecha_hora_min_seg	String
Duración	int
Enviado	boolean

Tab. 11.5: Tabla T: Al iniciar una medición de T

<i>Dato</i>	<i>Tipo</i>
Fecha_hora_min_seg	String
Contador	int
Valor	int

Tab. 11.6: Tabla Datos T: Se escribe con cada dato, pero asociado a una medición(Fecha_hora_min_seg)

12. PROTOTIPO FINAL

Con el prototipo número 2, se logra una base con la cual ya implementar el servidor web al cual conectarse. No sin antes analizar posibles mejoras a lo ya hecho, en base a distintas pruebas de rendimiento y cambios menores principalmente a nivel de comunicación Bluetooth.

En las siguientes secciones se detallan los cambios realizados para el prototipo final, en donde se destacan mejoras necesarias de la versión 2 y la selección e implementación del servidor web.

12.1. Paquetes de control Bluetooth

A partir de la segunda versión del prototipado, se pudo observar cierta tasa de error no menor en la recepción de comandos por parte del microcontrolador. Esto debido a la cantidad de operaciones hechas por segundo y ciertas respuestas de parte del módulo Bluetooth que no son homogéneas, complejizando así su detección.

Por esta razón y para no esperar una medición errónea (realizar una petición de cierto sensor y que el microcontrolador ejecute la medición pero de otro sensor) se implementa un mensaje de confirmación por parte del microcontrolador: Luego de la petición hacia el módulo Bluetooth, esta petición se procesa y confirma de vuelta hacia la aplicación, la cual analiza la respuesta y toma la decisión de reenviar la petición (en caso de que se haya considerado errónea) o ejecutarla (en caso de considerarla correcta).

Lo anterior, por medio de un mensaje con la cadena de texto “AAAA” seguido de 2 caracteres asociados a los sensores: “01”, “11”, “10”.

12.2. Implementación de Realm

Ya hechas las mejoras pertinentes a la programación del microcontrolador, se comienzan a implementar nuevas funciones a la aplicación, como lo es su base de datos local.

Como ya se mencionó en el capítulo anterior, la base de datos seleccionada es Realm. Respecto al modelo presentado se simplifican las tablas 11.6 y 11.4 por la posibilidad de generar un arreglo contenido en una medición de T y ECG respectivamente.

El manejo de las tablas en Realm se hace sencilla por su símil con los objetos en Java, al poder extender clases desde RealmObject y tratarlos como cualquier objeto pero utilizable en conjunto con Realm. Además de lo anterior, existe un tipo específico para listas de estos objetos (RealmList) con la cual se puede omitir las tablas ya mencionadas y agrupar esos datos (en forma de una lista) dentro de un RealmObject asociado a cada medición.

```
public class ECG_Table extends RealmObject {
    @PrimaryKey
    private String fecha_hora_min_seg;
    private String rut;
    private int duracion;
    private boolean enviado;
    private RealmList<Data> datos = null;

    public ECG_Table() { this.enviado = false; }
    public String getFecha_hora_min_seg() { return fecha_hora_min_seg; }

    public void setFecha_hora_min_seg(String fecha_hora_min_seg) {
        this.fecha_hora_min_seg = fecha_hora_min_seg;
    }

    public String getRut() { return rut; }

    public void setRut(String rut) { this.rut = rut; }

    public int getDuracion() { return duracion; }

    public void setDuracion(int duracion) { this.duracion = duracion; }

    public boolean isEnviado() { return enviado; }

    public RealmList<Data> getDatos() { return datos; }
}
```

Fig. 12.1: Clases de Realm utilizados para los datos de ECG

Pensando en la comunicación de los datos y dado que se está trabajando con objetos en Java para las distintas tablas y datos, es importante implementar algún método para serializar los datos, específicamente a JSON por su amplia utilización actual. Esto pensando en la comunicación final con el servidor tanto para los casos de comunicación en vivo como para los casos de sincronización entre bases de datos. Para esto último se utiliza la librería GSon de Google, la cual simplifica la implementación de JSON a partir de objetos.

Además de lo anterior ya expuesto, solo se ha seguido la guía oficial de Realm para su implementación en Android, por lo que se omite esta información por ser de común lectura [22].

12.3. Servicio web

Si bien en capítulos anteriores se implementó un servidor web basado en Python (Tornado), se descarta su uso por el acelerado proceso de desarrollo que se requiere, la pobre documentación que posee y las pocas ayudas frente al desarrollo de interfaces usuarias amigables o sencillas.

Por lo anterior y pensando en un cambio completo del servicio web, se buscan entre distintas alternativas que sean compatibles con los requerimientos del proyecto:

Motor Web: NodeJS (mongoose, socket.io, angularJS), MeteorJS, ExpressJS.

Base de datos: InfluxDB, FireBase, PostgreSQL.

Visualización de datos: Grafana, D3, High Charts, ChartsJS.

Comunicación: WebSocket.

Como se puede observar, la única seguridad de la cual el proyecto no se puede alejar es del uso de WebSocket dadas sus prestaciones (alta velocidad y baja latencia).

Analizando las distintas opciones ya mencionadas se determina configurar un servidor con las siguientes características:

Dominio: Se adquiere por \$10.000 pesos en NIC (nombres de dominio CL) el dominio “proyectogaleno.cl”. Del cual se puede ver el certificado en los anexos.

DNS: De forma gratuita, se consigue con Namecheap (registrador acreditado por la ICANN (Internet Corporation for Assigned Names and Numbers) que brinda servicios de registro de nombres de dominio y ofrece nombres de dominio de venta que están registrados a terceros) bajo el dominio “proyectogaleno.cl”.

Email: Para no montar directamente un servidor de correos electrónicos, se configura una redirección por el mismo servicio de Namecheap hacia el dominio ya mencionado.

Servidor Web: MeteorJS, por su enfoque a las conexiones en tiempo real (con el uso de WebSocket), por ser FullStack basado en JavaScript permitiendo un ágil desarrollo y por poseer el repositorio AtmosphereJS, que cuenta con multitud de paquetes de toda índole y de rápida implementación.

Websocket: DDP, protocolo que define cierto estándar de comunicación entre el servidor y los clientes, con el uso de mensajes JSON sobre websocket. Haciendo realmente rápida la implementación de la comunicación sobre WebSocket.

Librería gráfica JS: D3, librería de JavaScript para producir, a partir de datos, infogramas dinámicos e interactivos en navegadores web, haciendo uso de tecnologías bien sustentadas como SVG, HTML5, y CSS. Su elección radica en el gran control final que otorga y su fácil implementación en conjunto con Meteor con su paquete oficial d3js:d3.

Base de datos: MongoDB, si bien se analizaron otras opciones, se sigue prefiriendo su uso por sus características base y por su buen acople con MeteorJS.

Proxy reverso: NGinx, en este apartado no se realizaron modificaciones pues no existe competencia real dadas las prestaciones que ofrece y los requerimientos del proyecto (pasando por los fondos disponibles).

Seguridad: HTTPS y WSS (CertBot/LetsEncrypt), en este apartado se escoge utilizar certificados provistos gratuitamente por LetsEncrypt para hacer uso de los protocolos ya mencionados. HTTPS ofrece una base segura de intercambio de información entre el servidor y los clientes, sobre la cual además se utiliza WSS que es la versión encriptada de WS (protocolo de websocket). Dotando en conjunto de una segura comunicación sin perder rendimiento, pues ambas técnicas se basan en una sobrecarga principalmente al iniciar la comunicación y no en cada mensaje intercambiado.



Fig. 12.2: Tecnologías empleadas en el lado del servidor

En la figura 12.2 se pueden observar en conjunto y a grandes rasgos, las distintas tecnologías empleadas para dar soporte web al proyecto.

12.4. Selección de usuario, tipo de cuentas y muestras

Dadas las características del proyecto, se articulan dos tipos de cuentas principales en una primera instancia: Normal, con la cual poder acceder solo a la visualización de datos y Total, con la cual se pueden realizar mediciones desde la misma web.

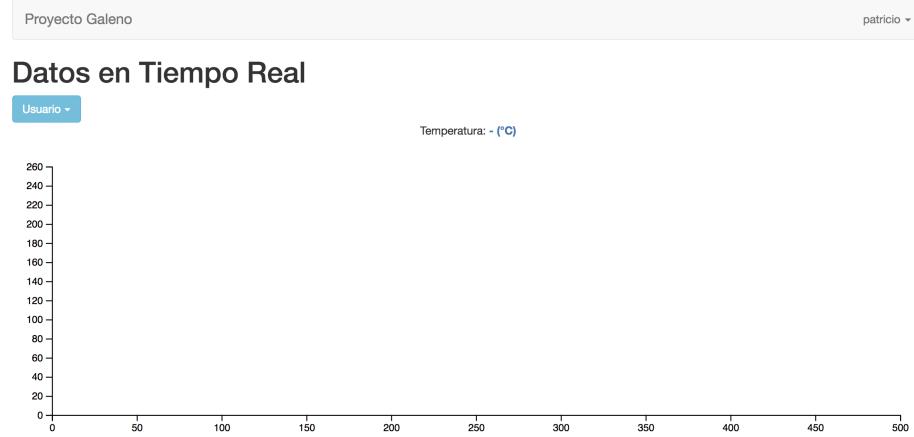


Fig. 12.3: Pantalla principal de un usuario normal

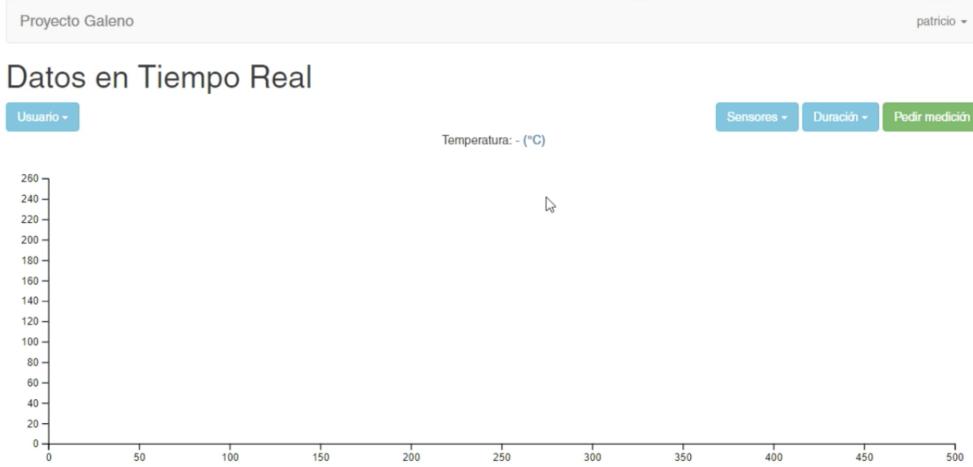


Fig. 12.4: Pantalla principal de un usuario con acceso total

Como se puede observar en las figuras anteriores, la única diferencia entre las distintas cuentas son los botones que especifican una medición (sensores y duración).

De la pantalla principal se pueden destacar el control de ingreso en la parte superior derecha, el botón de la izquierda concerniente a la selección del usuario a visualizar, a la derecha los botones de la medición y el gráfico principal con el ECG junto a la temperatura sobre este último.

Datos en Tiempo Real

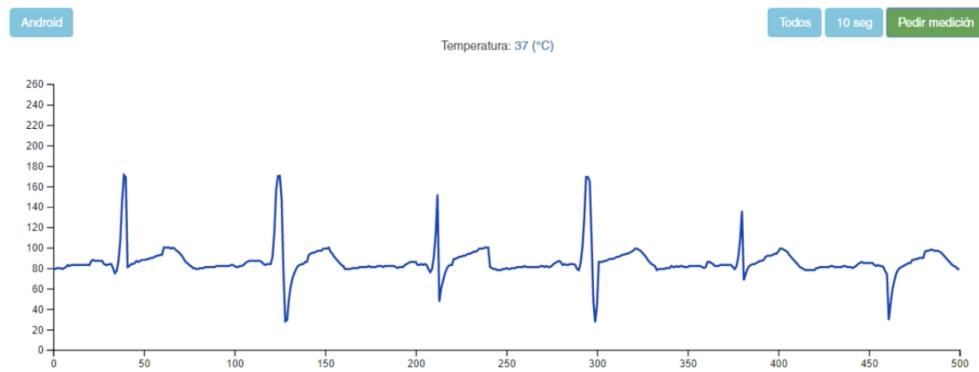


Fig. 12.5: Medición de ambos sensores en tiempo real

Como se puede observar en la figura 12.5, el gráfico resultante se encuentra desfasado en unos 2 segundos respecto al que se muestra en la pantalla de la aplicación. Todo dato que es mostrado, es almacenado tanto de forma local (Android - Realm) como en el servidor web (Meteor - MongoDB).

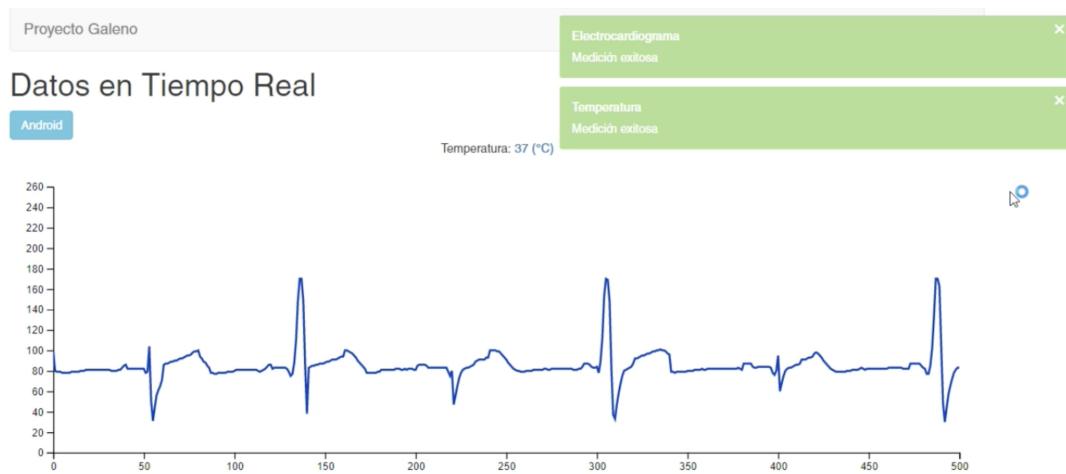


Fig. 12.6: Medición de ambos sensores en tiempo real por página Web



Fig. 12.7: Medición de ambos sensores en tiempo real por aplicación Android

En las figuras 12.6 y 12.7 se puede observar una misma medición siendo finalizada exitosamente, en la que por un lado en la página web se presentan las dos notificaciones asociadas a cada sensor y en la aplicación un Toast indicando su término correcto.

Es importante destacar que de fallar la medición, ya sea por uno o por los dos sensores (conteo total de datos enviados distintos entre microcontrolador y aplicación) ésta se reiniciará automáticamente, sin intervención del usuario y los datos almacenados en las distintas bases de datos serán borrados (como cabe esperar en un caso de error).

Por último, si se es riguroso se puede observar una temperatura bastante exacta en las distintas figuras presentadas, esto es debido a que el sensor de temperatura presentó problemas en su última utilización. A raíz de esto, se reemplazó su valor desde el microcontrolador con un valor fijo, esto último para validar el flujo completo de dicho sensor, aislando así el problema solo al sensor y no a todo el flujo de comunicación.

13. DISCUSIÓN

14. CONCLUSIÓN

15. ANEXOS



UNIVERSIDAD DE CHILE



CERTIFICADO

El nombre de dominio

proyectogaleno.cl

se encuentra inscrito en el registro de nombres de dominio .CL, a nombre de

**Patricio Nicolas Rodriguez Gatica (PATRICIO NICOLAS RODRIGUEZ
GATICA)**

RUT: 18245657-8

hasta el 27 de septiembre de 2018

NIC Chile

Santiago, 9 de julio de 2018

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 project:

```
#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 several variables and a function. It includes comments explaining the purpose of each variable and section.

```
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 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 save, run, upload, and download. The main code editor window displays 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 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 well-commented with multi-line comments explaining the logic and data processing. It includes sections for initialization, data transmission, and button handling.

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

void loop() {
    if (tiempo == 10000) {
        tiempo = 0;
    }
    tiempo = tiempo + 1;
    /*
-----ENVÍO DE DATOS-----
    */
    /* Envío 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 like Open, Save, and Print. The main area displays the following C++ code:

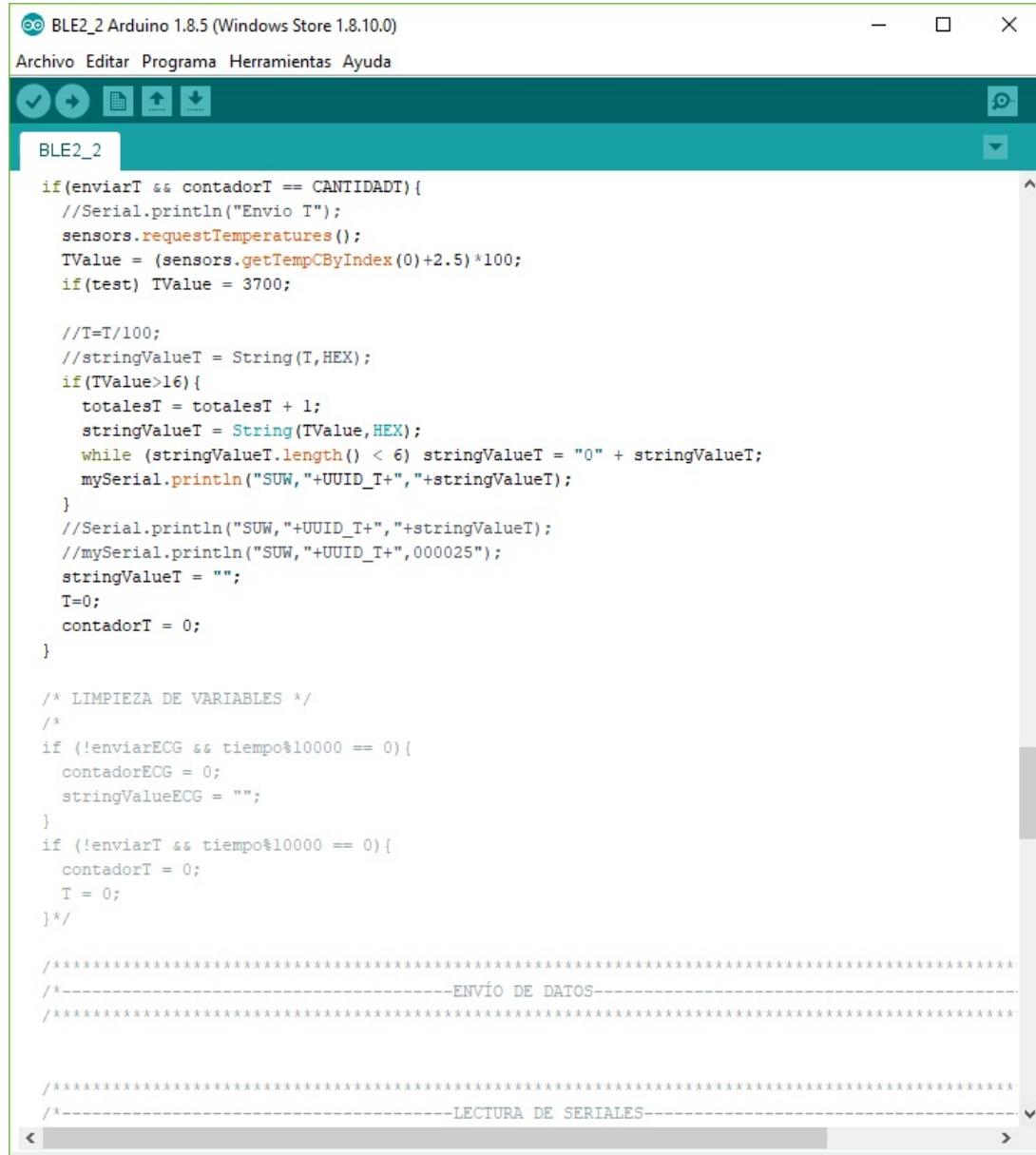
```
else {
    Serial.println("ENCENDER");
    test = 1;
}
}

/*
 * 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 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, Upload, and Download.
- Sketch Name:** BLE2_2
- Code Content:** The code is written in C++ and defines a function named `loop`. It includes logic for sending temperature data via Serial and mySerial, handling variable cleanup, and defining sections for data transmission and serial reading.

```
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 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.
- Sketch Name:** BLE2_2
- Code Area:** Displays the following C++ code for the BLE2_2 sketch:

```
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 "BLE2_2" sketch. The code uses the Serial and mySerial objects to handle communication. It includes logic for turning on a device (T), reading bytes from the serial port, and sending specific commands like "SUR" at regular intervals.

```
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] Sotera Wireless, ViSi Mobile®System, rev. 15 Mayo 2018,
<https://newatlas.com/visi-mobile-wireless-health-monitoring/25583/>
- [3] Qardio Inc., QardioCore, rev. 05 marzo 2018,
<https://www.getqardio.com/es/qardiocore-wearable-ecg-ekg-monitor-iphone/>
- [4] Qardio Inc., QardioCore, rev. 15 Mayo 2018,
<https://store.getqardio.com/products/qardiocore>
- [5] Nuubo, Nuubo wearable ECG, rev. 05 marzo 2018,
<https://www.nuubo.com/producto>
- [6] Nuubo, Nuubo wearable ECG, rev. 15 Mayo 2018,
<http://pdf.medicalexpo.com/pdf/nuubo/necg-minder/83949-96239.html>
- [7] 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>
- [8] Microchip, “Bluetooth Low Energy Module RN4020”, Rev. 15 Mayo 2018,
<http://ww1.microchip.com/downloads/en/DeviceDoc/50002279B.pdf>
- [9] Wikipedia, “Radio Frecuencia”, Rev. 15 Mayo 2018,
<https://es.wikipedia.org/wiki/Radiofrecuencia>

- [10] Wikipedia, “Comunicaciones por satélite”, Rev. 15 Mayo 2018, <https://goo.gl/ykf7tv>
- [11] Wikipedia, “IEEE 802.11”, Rev. 15 Mayo 2018, <https://goo.gl/1hmiWW>
- [12] Wikipedia, “Red de celdas”, Rev. 15 Mayo 2018, <https://goo.gl/K2sKoQs>
- [13] Maxim Integrated, “Programable Resolution 1-Wire Digital Thermometer”, rev Enero 2015, <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>
- [14] XBee, “¿Qué es XBee?”, rev Mayo 2018, <http://xbee.cl/que-es-xbee/>
- [15] Wikipedia, “Conjunto de comandos Hayes”, rev Mayo 2018, <https://goo.gl/xpCfKC>
- [16] Fire EMS, “Understanding ECG Filtering”, rev Mayo 2018, <http://www.ems12lead.com/2014/03/10/understanding-ecg-filtering/>
- [17] Wikipedia, “Bluetooth”, rev Mayo 2018, <https://es.wikipedia.org/wiki/Bluetooth>
- [18] StatCounter, “GlobalStats”, rev Mayo 2018, <https://goo.gl/62GngF>
- [19] Wikipedia, “Alojamiento Web”, rev Mayo 2018, https://es.wikipedia.org/wiki/Alojamiento_web
- [20] Google, “Servicios”, rev Julio 2018, <https://developer.android.com/guide/components/services?hl=419>
- [21] Microchip, “RN4020”, rev Julio 2018, <https://www.microchip.com/wwwproducts/en/RN4020>
- [22] Realm, “Realm Android”, rev Julio 2018, <https://realm.io/blog/realm-for-android>
- [23] Microchip, “RN4020 User Guide”, rev Julio 2018, <http://ww1.microchip.com/downloads/en/DeviceDoc/70005191B.pdf>