Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Impulse: Estandarización y entorno completo del protocolo Pulse para móviles Android

---

Alumno: Sergio Tabarés Medina

Tutor/a/es: Pilar Grande González

Nicolás Sacristán Machín

Impulse: Estandarización y entorno completo del protocolo Pulse para móviles Android

Sergio Tabarés Medina

Índice general

Índice de figuras

Índice de tablas

Sin conocimiento no hay libertad de expresión,
sin libertad de expresión no hay conocimiento

Agradecimientos

A mis padres, que siempre supieron apoyarme cuando más lo necesitaba, sin ellos jamás hubiera llegado tan lejos. A mi familia, por tener tantísimo temple y acompañarme en los buenos y, especialmente, en los malos momentos. A mis tutores, por guiarme. A Fernando Peral, profesor de electróncia, por ser tan alturista como para ayudarme sin ningún tipo de pretensión.

Resumen

Pulse es una biblioteca de transmisión segura (puede recibir a una distancia máxima de $\sim 1cm$) y de bajo bitrate basada en la interpretación inalámbrica de campos magnéticos simplificando y añadiendo compatibilidad prácticamente universal a la mayoría de móviles.

Por otro lado, *Impulse*, se encarga, principalmente, de demostrar lo que puede hacer la biblioteca *pulse*, permitiendo la interacción de cualquier usuario con la biblioteca.

Palabras claves: Protocolo, Comunicación, Inalámbrico, Campo magnético.

Abstract

Pulse is a secure transmission library (it can receive at a maximum distance of $\sim 1cm$) and low bitrate based on the wireless interpation of magnetic fields that simplifies and adds almost universal compatibility to most mobile phones.

On the other hand, *Impulse* is mainly responsible for demonstrating what the *pulse* library can do, allowing any user to interact with the library.

Keywords: Protocol, Communication, Wireless, Magnetic field.

Parte I

Memoria del Proyecto

Capítulo 1

Descripción general del proyecto

Este documento está dividido en 4 partes: Memoria, Documentación, Conclusiones y Manual. La memoria recoge esta introducción y otros elementos como:

- Los motivos y objetivos que se pretenden alcanzar con la realización de este TFG.
- La metodología que se seguirá para desarrollar los objetivos fijados, así como el marco tecnológico en el que se desarrolla.
- La planificación y presupuestación de todo el proyecto, a través del uso de análisis de Puntos de Función, junto con el total de los costes ya registrados.
- Explicación de los distintos contenido incluidos el repositorio adjunto.

La **documentación** contiene información técnica acerca del proyecto. Se ha decidido dividir en tres partes para facilitar la comprensión del trabajo desarrollado:

- Aplicación Impulse: Es la interfaz de usuario que interactúa con la biblioteca Pulse. Se incluyen los requisitos del sistema, las especificaciones, diseño e implementación.
- Hardware: Analiza los distintos componentes y cómo se han ido desarrollando las diferentes interacciones junto al software.
- Biblioteca Pulse: Es un salto disruptivo que conlleva investigación, desarrollo e innovación. Lo que implica un alto nivel de complejidad y requiere una explicación a parte de las distintas tecnologías utilizadas. Una documentación que nada tiene que ver con la estándar de una aplicación.

- Pruebas: Se evaluará si el software y el hardware desarrollado en los puntos previos cumple con los requisitos establecidos y si funciona correctamente en diferentes situaciones y escenarios. Estas pruebas permitirán identificar y corregir los errores y defectos antes de la implementación y entrega.

Las **conclusiones** incorporan, además de las conclusiones personales, las futuras ampliaciones y mejoras tanto a nivel de software como de hardware.

El **manual** incluye la guía de instalación de la aplicación, el método para compilar y el funcionamiento de la aplicación Impulse para un usuario estándar.

1.1. Introducción

Los métodos de comunicación inalámbrica suelen ser engorrosos a la hora de enviar datos. Las alternativas o tienen poca compatibilidad entre los móviles de gama baja o media, cómo NFC, u ofrecen información pública, sin ningún tipo de privacidad posible, como QR.

Pulse es una tecnología para comunicar información a corta distancia ($\sim 1cm$), a través de un campo magnético que funciona en todos los móviles con magnetómetro (brújula digital) que reciban lecturas de campo magnético en un periodo no superior a los 50ms. Pulse permite transmitir información sin complejidad, de forma segura y casi universalmente, ya que hay magnetómetros que no leen datos magnéticos cada 50ms.

Pulse permite transmitir y recibir ficheros y mensajes, automatizar la conexión de redes wifi, enviar e-mails y sms o ingresar automáticamente bitcoins (BTC) a una cartera (Wallet).

Además Pulse permite conectarte a través de bluetooth automáticamente, para las tareas que requieran más ancho de banda, y recibir toda la información, sin contraseñas, sin nombres, pero cifrado.

1.2. Estado del arte

Códigos QR

QR es un código de barras en dos dimensiones. QR, al igual que cualquier código de barras, permite recibir información a través de imágenes.

NFC

NFC es una tecnología para comunicar información a corta distancia que permite realizar pagos o ajustes automatizados (configurar wifi, cambiar a modo silencio el teléfono, compartir un tweet predeterminado,...). Muchos de los teléfonos a día de hoy no suelen tener NFC, a no ser que se trate de un teléfono de gama alta. Sin embargo la inclusión de un magnetómetro sí que es común en los teléfonos, herramienta que no es utilizada aún para comunicaciones a corta distancia.

Comparativa y conclusiones

La diferencia con este tipo de tecnología frente a NFC es que el móvil siempre debe de ser el receptor, en ningún caso el transmisor (puesto que no está capacitado para enviar una señal magnética, sólo para recibirla).

	NFC	QR	Pulse
Compatibilidad	Escasa	Mayoritaria	Mayoritaria
Rango	<20 cm	Visible	$\sim 1cm$
Privacidad	✓	✗	✓
Comunicación	Duplex	Simplex	Simplex
Consumo del transmisor	<15 mA	Nulo	<450 mA
Corrección de errores	-	Aprox. 7 % - 30 %	Aprox. 20 % - 29 %
Cifrado	Sí, excepto con RFID	Ninguno	Sí, si se envía por bluetooth

Tabla 1.1: Tabla de comparación de alternativas

1.3. Objetivos y alcance del trabajo

Se pretende diseñar un software que module el envío de una señal magnética, que sea percibida por el magnetómetro del teléfono. El envío y recepción se abstraerán en una biblioteca que permitirá enviar cada una de las distintas tareas de la que se dispondrá.

El proyecto nace con la orientación de buscar nuevas formas de transmitir información de un punto a otro. Formas que si bien no se han explorado a día de hoy, sí que pueden ser útiles para transmitir información de una manera sencilla.

Este nuevo método de envío, Pulse, pretende simplificar la comunicación y permitir enviar información contactless de forma económica, ya que prácticamente todos los dispositivos móviles son compatibles.

Este proyecto está orientado a la comunicación de información desde un ordenador con un dispositivo hardware hasta un dispositivo móvil a través de variaciones en los campos magnéticos. Los objetivos básicos son:

1. Crear un hardware funcional.
2. Diseñar e implementar una biblioteca que se encargue de transmitir y recibir información (archivos, mensajes, conexión wifi, envío de sms y e-mail, pago de bitcoins) a través de pulsos magnéticos y que permita conectarse por bluetooth automáticamente de manera cifrada.
3. Diseñar e implementar una aplicación que interactúe con la biblioteca, permitiendo demostrar el funcionamiento de la misma.

Capítulo 2

Metodología

El proyecto estará basado en una **metodología incremental iterativa** lo que implica que se trabajará sobre un prototipo inicial, sobre el que se irán construyendo progresivamente versiones del producto que incluirán nuevas características o mejoras hasta alcanzar el producto final.

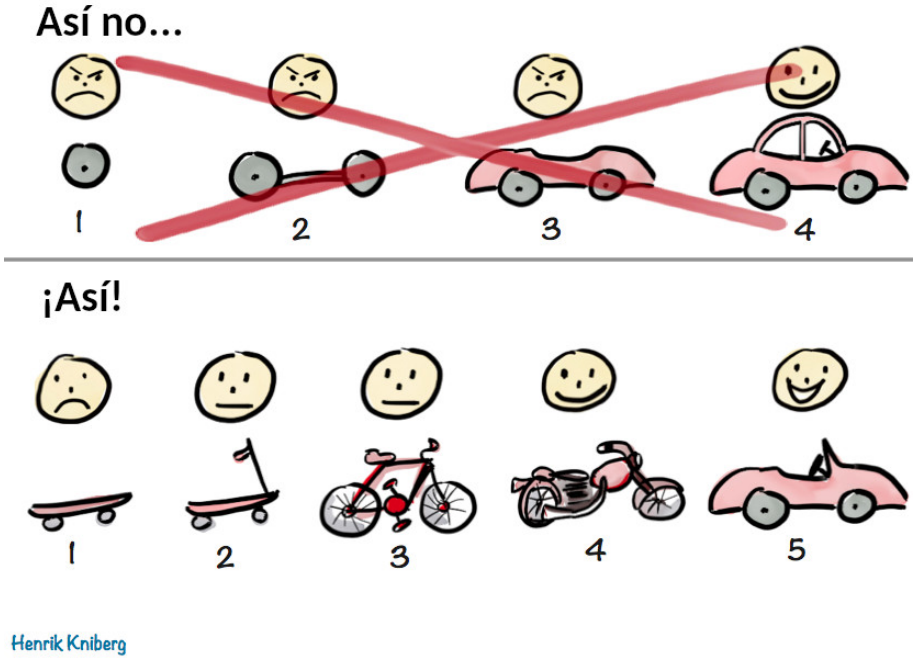


Figura 2.1: Metodología de desarrollo incremental frente a iterativo e incremental [7]

Esta metodología se ha escogido por la facilidad que tiene el usuario de interaccionar con el proceso de desarrollo de los componentes software.

Los incrementos realizados a lo largo del desarrollo software han sido los siguientes:

1- Estudio previo

Recopila información acerca de cómo se debería desarrollar el modelo hardware, ya que dependiendo de las condiciones físicas en las que se desarrolle, el proyecto puede tener fuertes variaciones a nivel de software.

2- Primer incremento

En esta fase se desarrolla un prototipo hardware, simple, contiene las bases esenciales sobre las que se desarrolla el proyecto, además de un software transmisor y receptor rudimentario, a nivel de bits.

3- Segundo incremento

El hardware del circuito principal y sensores, habrán acabado de desarrollarse. Se desarrollará un sistema de redundancia, que permita verificar y recuperar, en caso de error, la información perdida, además de buscar, de manera más óptima, el envío de información, evitando, en la medida de lo posible, que se comentan errores.

4- Tercer incremento

El circuito del hardware se muestra como final. Se desarrolla un sistema que permita enviar y recibir tareas de forma transparente, lo que implica que el desarrollo de la biblioteca Pulse se finaliza.

5- Cuarto incremento

Se miniaturiza el circuito a través de una placa PCB. Además se le añade una carcasa diseñada en 3D. A nivel software se genera una interfaz gráfica, Impulse, que permita interactuar con la biblioteca Pulse.

6- Documentación

Durante todo el desarrollo hardware y software se agregan notas y esquemas acerca de los prototipos. Según se va desarrollando, a nivel de software, se van agregando comentarios, en inglés, que permitan entender y comprender la función de los objetos y sus procedimientos en un idioma universal.

2.1. Tecnologías de desarrollo

Para el desarrollo se han utilizado distintas herramientas de uso público con el objetivo de hacer funcionar el producto final.

Las herramientas utilizadas son:

- IDEs:
  - QT Creator: Desarrollado por QT Group. Especializado en programación en C++ junto al Framework de QT. Se pueden utilizar distintas versiones pero se utiliza la licencia libre (L)GPL.
  - Arduino IDE: Especializado en programación en C y C++ junto a las bibliotecas necesarias para hacer funcionar cualquier dispositivo Arduino, permitiendo escribir el código en la PCB y depurarlo fácilmente.
  - Android Studio: IDE dedicado a los lenguajes Java y Kotlin junto al entorno de bibliotecas desarrolladas por Alphabet para los dispositivos Android.
- Otras aplicaciones
  - Git: Se utiliza junto a Github Desktop como el software que permite gestionar las versiones, sobre un servidor de Gitlab.
  - Texstudio: Entorno para desarrollar toda la documentación pertinente en LaTeX.
  - Draw.io: Editor de diagramas con versión online como de escritorio.
  - EasyEDA: Permite diseñar el circuito y la placa PCB del hardware fácilmente. Además, al ser un editor propiedad de un vendedor, permite automatizar más fácilmente la compra de la placa PCB.
  - Blender: Diseñador 3D, útil para modelar cualquier objeto impreso en 3D. Con él se ha realizado el diseño de la carcasa con las medidas adecuadas.
  - Cura: Convierte un diseño 3D en instrucciones simples para una impresora en específico. Además permite detectar posibles errores a la hora de interpretar el objeto 3D que se puedan dar.
  - GIMP: Editor de imágenes simples (PNG, JPG, WEBP,...) libre e intuitivo. Utilizado para diseñar el logo y adaptarlo a las necesidades del software.
  - Inkscape: Editor de imágenes vectorizadas. Es compatible con cualquier imagen SVG. Además permite importar imágenes simples, vectorizarlas y adaptarlas a las distintas necesidades.

- Bibliotecas:
  - QT Framework: Entorno de trabajo diseñado por QT Framework. Además de permitir el desarrollo gráfico, también incluye herramientas básicas muy comunes útiles para el desarrollo.
  - QCustomPlot: Biblioteca diseñada para Escritorio, dependiente de QT Framework. Permite dibujar gráficas fácilmente.
  - Arduino: Diseñada para crear sistemas embebidos simples en lenguaje C y C++.
  - Mbed: Biblioteca diseñada para controlar el hardware específico de de Arduino 33 BLE.
  - Android Open Source Project: Contiene todo el entorno en el que se debe de desarrollar cualquier aplicación dedicada a teléfonos Android.
  - MPAndroidChart: Permite dibujar fácilmente gráficos, incluso si se tratan de gráficos que se actualizan en el momento.
  - Android ripple pulse animation: Biblioteca que permite dibujar ondas en una interfaz gráfica de Android.
- Bibliotecas orientativas: No se han utilizado directamente, sin embargo han servido como ejemplo a seguir a la hora de programar.
  - Reed-Solomon: Permite codificar y descodificar fácilmente la codificación Reed-Solomon con errores y borrados universales. Escrita en Python.
  - BLESSED for Android: Biblioteca de Bluettoth BLE que nos sirve como orientación a la conexión bluetooth entre el dispositivo y el teléfono.
- Lenguajes de programación:
  - C++: Lenguaje compilado orientado a objetos muy común en la mayoría de dispositivos que nos rodean.
  - Java: Lenguaje precompilado orientado a objetos. Especialmente común entre dispositivos IoT.
- Opciones descartadas:
  - gEDA: Herramienta libre para generar fácilmente diagramas de circuitos PCB. Entró en desuso por falta de mantenimiento y, además, no es tan versátil debido a que no hay una identificación directa con los fabricantes como sí que lo es EasyEDA.
  - GNOME: Biblioteca libre dedicada al diseño de interfaces de usuario. Se desistió en utilizar GNOME debido a la poca documentación que había y a la poca reutilización que tendría.

2.2. Descripción general del producto

El sistema, tal y como se comenta en el capítulo "Biblioteca Pulse" 6, está dividido en dos grandes partes: transmisor y receptor. El transmisor es el resultado de la combinación entre un equipo con sistema operativo Windows o GNU/Linux y un dispositivo hardware diseñado durante el proyecto. Su funcionalidad es, exclusivamente, emitir pulsos magnéticos.

El receptor son todos los dispositivos móviles compatibles, con un sistema operativo Android o derivado (HarmonyOS, LinageOS, /e/,...). Está orientado a recuperar toda la información que el transmisor envió a través de pulsos magnéticos.

Otra gran subdivisión se produce a nivel de programación: La biblioteca Pulse y la interfaz gráfica Impulse. Pulse está orientado a realizar la conexión y hacer funcional el protocolo, permitiendo que los distintos dispositivos se comuniquen exitosamente entre ellos. Impulse es una interfaz gráfica sencilla que dota a la biblioteca Pulse de capacidad para poder interaccionar con cualquier usuario.

Capítulo 3

Planificación

La idea de este proyecto surgió antes, pero debido a problemas de carácter personal no se adoptó el inicio hasta un periodo posterior.

Por esa razón el proyecto se iniciará en Agosto de 2022 y se estima que finalice a finales de Febrero de 2023. Se trata de un proyecto muy amplio con desarrollo hardware y software, además de combinar múltiples lenguajes y 3 piezas de software (biblioteca Pulse en escritorio, dispositivo y móvil), junto a 2 interfaces de usuario totalmente independientes (programa de escritorio y aplicación móvil Impulse).

El desarrollo del proyecto será llevado a cabo por una sola persona que trabajará 2 horas de Lunes a Vienes. Si se produce algún atraso se recuperará a lo largo del fin de semana.

3.1. Estimación del esfuerzo

En este proyecto se ha realizado una estimación por Puntos de Función, para poder calcular adecuadamente el tamaño del proyecto.

3.1.1. Estimación por Puntos de Función

Es una de las técnicas más fiables. Hay dos métodos para calcular los Puntos de función: Albertch y Mark II. Se utilizará el método Albertch, una métrica más precisa que cuantifica la funcionalidad que el usuario debe de entregar al ejecutar la aplicación.

Los pasos para cuantificar a través de este método son los siguientes:

1. Se identifican los componentes por los siguientes tipos:
   - Entradas externas: Información que comunica el usuario a la aplicación.
   - Salidas externas: Información que comunica la aplicación al usuario.
   - Consulta: Representan entradas que producen inmediatamente una salida.
   - Ficheros lógicos internos: Persistencia de la información de la aplicación.
   - Ficheros lógicos externos: Representa todas las interfaces externas usables para transmitir (recibiendo u emitiendo) información entre sistemas.
2. Se calcula el nivel de complejidad de los componentes identificados en baja, media o alta.
3. Se suman todos los componentes multiplicados por el peso, obteniendo así los Puntos de Función Sin Ajustar (PFSA).
4. Se evalúa una tabla de Factores de Ajustes (FA) y pesos con valores entre 0 y 5. Donde cada valor representa:
   0. Sin influencia
   1. Influencia mínima
   2. Influencia baja
   3. Influencia media
   4. Influencia alta
   5. Influencia máxima
5. Se aplica la siguiente fórmula para obtener los Puntos de Función Ajustados (PFA):

$$PFA = PFSA \cdot (0,65 + (0,01 \cdot \sum FA))$$

6. Para poder obtener una estimación correcta de líneas de código (LDC) se utiliza una tabla de equivalencias de PFA a LDC de Capers Jones [5] que dependerá del lenguaje de programación que se tenga pensado utilizar.

Estimación del tamaño

Explicado el proceso paso a paso, se aplicará el cálculo de PF para este proyecto. Se ha decidido dividir entre las 2 grandes partes del proyecto, biblioteca (Pulse), interfaz (Impulse) y los distintos dispositivos hardware, puesto que son todos elementos independientes entre sí a la hora de programar.

Interfaz de escritorio (Impulse)

- Entradas:
  - Modificar la repetición de cualquier tarea (baja).
  - Eliminar una tarea (media).
  - Modificar la configuración del sistema, cambiando el método de envío, el modo noche y las pestañas que se visualizarán (alta).
- Salidas:
  - Mensajes de error que se muestran debido a que el móvil se ha retirado, el dispositivo se ha movido, se ha producido un error de conexión bluetooth o ha habido un error con el dispositivo a nivel de hardware (alta).
- Consulta:
  - Comprobar si existen nuevas actualizaciones (media).
  - Mostrar gráfica, datos e información del móvil Android para el modo desarrollador (alta).
  - Verificar tarea y enviarla a Pulse, si se ha introducido incorrectamente, una tarea fichero (media).
  - Verificar tarea y enviarla a Pulse, si se ha introducido incorrectamente, una tarea mensaje (baja).
  - Verificar tarea y enviarla a Pulse, si se ha introducido incorrectamente, una tarea wifi (alta).
  - Verificar tarea y enviarla a Pulse, si se ha introducido incorrectamente, una tarea sms (media).
  - Verificar tarea y enviarla a Pulser, si se ha introducido incorrectamente, una tarea e-mail (alta).
  - Verificar tarea y enviarla a Pulse, si se ha introducido incorrectamente, una tarea bitcoin (media).
- Ficheros lógicos internos:
  - Almacenamiento de la configuración en un fichero .ini (media).
- Ficheros lógicos externos:
  - Conexión HTTPS al repositorio y lectura en formato JSON (alta).
  - Conexión a la interfaz Pulse (alta).

Biblioteca de escritorio (Pulse)

- Entradas:
  - Petición de conexión y desconexión a un dispositivo (media).
  - Añadir una nueva tarea a la lista (alta).
  - Cambio de la repetición de una tarea (media).
  - Eliminación de una tarea a la lista (alta).
  - Generar una tarea de tipo fichero (media).
  - Generar una tarea de tipo mensaje (media).
  - Generar una tarea de tipo wifi (media).
  - Generar una tarea de tipo sms (media).
  - Generar una tarea de tipo e-mail (media).
  - Generar una tarea de tipo bitcoin (media).
- Salida:
  - Envío de un error producido (media).
  - Mostrar progreso de la tarea (media).
- Consulta:
  - Petición de lista de dispositivos disponibles (media).
- Ficheros lógicos externos:
  - Comunicación a través de un puerto COM (alta).

Biblioteca del dispositivo (Pulse)

- Entradas:
  - Recepción de bytes, tanto por bluetooth como por pulsos magnéticos (alta).
  - Petición de encendido/apagado del bluetooth (alta).
- Salidas:
  - Se notifica que el dispositivo está preparado para continuar con la siguiente tarea (media).
  - Notificación de que el dispositivo se ha conectado por bluetooth (media).
  - Envío de errores (alta).
  - Envío de bytes a través del solenoide (alta).

Biblioteca de aplicación móvil (Pulse)

- Entradas:
  - Recepción de bytes de datos a través del magnetómetro (alta).
  - Iniciar/Detener la biblioteca Pulse (media).
- Salida:
  - Detecta errores producidos debido al movimiento o por un error a la hora de verificar la información recibida (alta).
  - Estado del dispositivo, si está en espera, recibiendo datos Pulse, buscando dispositivo bluetooth, conectado o con un error grave que le impida funcionar (media).
  - Progreso de la tarea (media).
  - Todas las tareas recibidas con éxito a través del magnetómetro (alta).
  - Recepción byte a byte de los ficheros recibidos (alta).
  - Devuelve los campos magnéticos y una marca tiempo en los que se recibió para el modo desarrollador (alta).
- Consulta:
  - Se pide información estadística (media, máximo, mínimo,...) y de compatibilidad a la biblioteca de todos los datos recibidos (alta).
- Ficheros lógicos externos:
  - Magnetómetro y acelerómetro de la API de Android (media).

Interfaz de aplicación (Impulse)

- Entradas:
  - Modificar la configuración del sistema, cambiando el modo noche y la carpeta seleccionada para almacenar los ficheros (media).
- Salidas:
  - Mostrar el estado del móvil (media).
  - Mostrar el progreso de recepción de la tarea (media).
  - Mensajes de error porque el dispositivo se ha movido, se ha producido un error de conexión bluetooth o ha habido un error con el dispositivo a nivel de hardware (media).

- Mostrar gráfica, datos e información estadística en el modo desarrollador (alta).
- Guardar fichero y avisar mediante un mensaje de que se ha recibido un fichero de la tarea recibida (alta).
- Mostrar un mensaje de la tarea recibida (baja).
- Informa del guardado de una nueva red wifi de la tarea recibida (alta).
- Abrir la aplicación encargada de enviar sms con los datos de la tarea recibida (media).
- Abrir la aplicación encargada de enviar e-mail con los datos de la tarea recibida (media).
- Abrir la aplicación con la transacción de bitcoins con los datos de la tarea recibida (media).

- Consulta:
  - Tutorial explicativo que detectará si la aplicación es compatible (o no) y que pedirá todos los permisos necesarios al usuario (alta).
  - El desarrollador pide información estadística del dispositivo (baja).

- Ficheros lógicos internos:
  - Almacenamiento de la configuración en el sistema persistencia de preferencias de Android (baja).

- Ficheros lógicos externos:
  - Conexión a la interfaz Impulse, recibiendo tareas y notificando los cambios que se produzcan (alta).

Una vez identificado todos los componentes y calculado el nivel de complejidad se multiplican el número de componentes por su complejidad. Posteriormente se suman todos los totales para calcular los Puntos de Función Sin Ajustar (PFSA):

Dominio	Complejidad	Nº	Total
Entradas	Alta (×6)	6	36
	Media (×4)	11	44
	Baja (×3)	1	3
Salidas	Alta (×7)	10	70
	Media (×5)	12	60
	Baja (×4)	1	4
Consultas	Alta (×7)	5	35
	Media (×5)	5	25
	Baja (×4)	2	8

Dominio	Complejidad	Nº	Total
Ficheros Log. Internos	Alta (×15)	0	0
	Media (×10)	1	10
	Baja (×7)	1	7
Ficheros Log. Externos	Alta (×10)	4	40
	Media (×7)	1	7
	Baja (×5)	0	0
Total PFSA			349

Tabla 3.1: Puntos de función sin ajustar (PFSA)

Se evalúan la tabla de factores de Ajuste para poder calculas los Puntos de Función Ajustados.

Factor	Valor
1. Respaldo y recuperación	0
2. Comunicación de datos	5
3. Procesamiento distribuido	4
4. Desempeño crítico	4
5. Entorno operativo existente	5
6. Entrada de datos en línea	0
7. Transacción de entrada sobre pantallas múltiples	0
8. Actualización en línea	0
9. Complejo de valores de dominio de información	5
10. Complejo de procesamiento	5
11. Código diseñado para reutilización	5
12. Conversión/instalación en diseño	3
13. Instalaciones múltiples	3
14. Aplicación diseñada para cambios	5
Total FA	44

Tabla 3.2: Factores de Ajuste (FA)

Se procede a calcular PFA según la fórmula ya enunciada:

$$PFA = PFSA \cdot (0,65 + (0,01 \cdot \sum FA)) =$$

$$= 349 \cdot (0,65 + (0,01 \cdot 44)) = 380,41\ PFA$$

Se prevee que, aproximadamente, el 65 % del proyecto esté escrito en C++ y el 35 % restante en Java, así que, se extraen los datos de la tabla de Carpers Jones de ambos lenguajes.

Lenguaje	LDC/PF
C++	53
Java	53

Tabla 3.3: Equivalencia ente PF y LDC. Fuente: Capers Jones [5]

Y por último se calculan las LDC a través de los PFA obtenidos.

Estimaciones finales	Total
Líneas de código en C++ $= PFA \times 53 \times 65\,\%$	13.105
Líneas de código en Java $= PFA \times 53 \times 35\,\%$	7.057
Total de LDC	20.162

Tabla 3.4: Líneas de código

3.2. Planificación temporal

En base a estimaciones anteriores, se ha elaborado una planificación por etapas para el proyecto que se llevará a cabo desde el 01/08/2022 hasta el 24/02/2023. Esta planificación proporciona una hoja de ruta detallada que guiará el progreso del proyecto a lo largo de diferentes etapas, desde el estudio previo hasta el último incremento.

Etapa del proyecto	Días	Inicio	Fin
1. Estudio Previo	12	01/08/2022	16/08/2022
1.1. Aprendizaje hardware	6	01/08/2022	08/08/2022
1.2. Estudio del proyecto	6	09/08/2022	16/08/2022
2. Primer incremento	48	17/08/2022	21/10/2022
2.1. Prototipo hardware	14	17/08/2022	05/09/2022
2.2. Software transmisor	17	06/09/2022	28/09/2022
2.3. Software receptor	17	29/09/2022	21/10/2022
3. Segundo incremento	30	24/10/2022	02/12/2022
3.1. Prototipo hardware	30	24/10/2022	02/12/2022
3.2. Software transmisor	15	24/10/2022	11/11/2022
3.3. Software receptor	15	14/11/2022	02/12/2022
4. Tercer incremento	26	05/12/2022	09/01/2023
4.1. Prototipo hardware	26	05/12/2022	09/01/2023
4.2. Software transmisor	13	05/12/2022	21/12/2022
4.3. Software receptor	13	22/12/2022	09/01/2023
5. Cuarto incremento	34	10/01/2023	24/02/2023
5.1. Placa PCB	34	10/01/2023	24/02/2023
5.2. Interfaz de transmisor	17	10/01/2023	01/02/2023
5.2. Interfaz de receptor	17	02/02/2023	24/02/2023
6. Documentación	150	01/08/2022	24/02/2023
6.1. Notas y Elaboración de la documentación	150	01/08/2022	24/02/2023

Tabla 3.5: Planificación de etapas

Es importante tener en cuenta que las fechas indicadas son estimaciones y podrían estar sujetas a cambios a lo largo del desarrollo, debido a posibles reajustes en los plazos. La planificación por etapas proporciona una guía general para llevar a cabo las actividades necesarias en cada fase, asegurando un enfoque estructurado y ordenado para el desarrollo exitoso del proyecto.

Como resultado, se genera un diagrama de Gantt, representado en la siguiente figura:

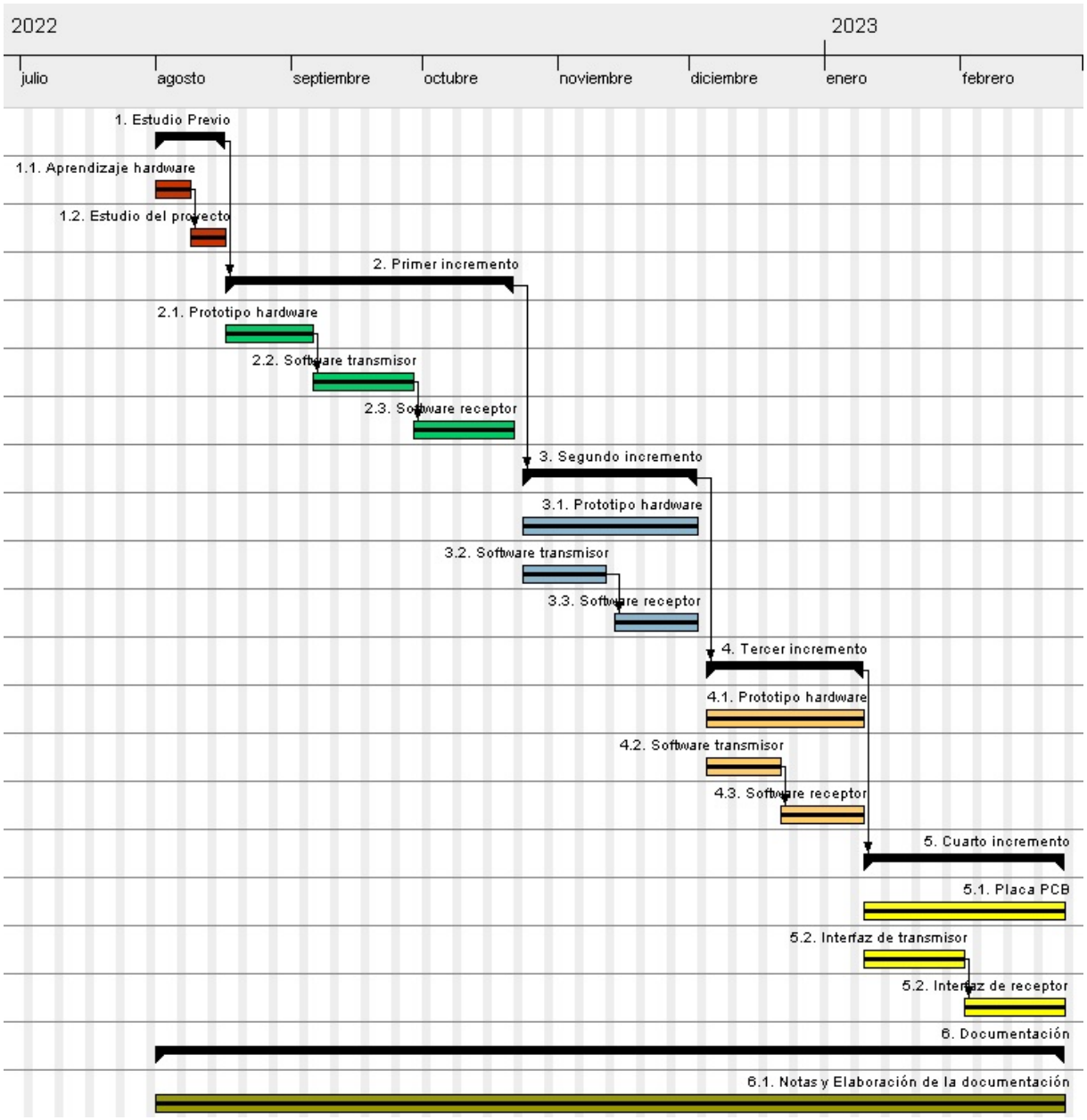


Figura 3.1: Diagrama de Gantt

3.3. Presupuesto económico

Una vez realizada la planificación, se hará una estimación del coste total del proyecto. Es importante comprender que en este desarrollo hay una porción de desarrollo en hardware, lo que da pie a que el presupuesto económico sea una estimación imprecisa, debido a que no se conoce el coste real del hardware.

Este apartado detalla todos los costes previstos para el proyecto dividiéndolos en prototipo, hardware, software y costes personales. Es necesario destacar que, para calcular el coste real de herramientas compartidas, es preciso aplicar las siguientes fórmulas específicas que toman en consideración factores como la frecuencia de uso, el tiempo utilizado y el coste mensual de las herramientas utilizadas.

$$\begin{aligned} Coste\ por\ mes\ \left(\tfrac{€}{mes}\right) &= \tfrac{Coste\ total\ (€)}{Vida\ útil\ (meses)} \\ Coste\ real\ por\ mes\ \left(\tfrac{€}{mes}\right) &= Coste\ por\ mes \times Porcentaje\ de\ uso \\ Coste\ real\ (€) &= Coste\ real\ por\ mes \times Meses\ de\ uso\ (meses) \end{aligned} \tag{3.1}$$

Prototipo

Representa el hardware que se pretende diseñar a lo largo del proyecto. Se incluyen todo los componentes electrónicos que se prevé que se pueden utilizar a lo largo del proyecto. Si bien, al ser hardware, es previsible que se cometan grandes errores de estimación. El coste del prototipo será de **249,72€** y está desglosado en los siguientes componentes:

Componentes	Coste (€)	Cantidad	Total (€)
Protoboard (5 uds.) A	13,99	1	13,99
Cable Jumper (120 uds.) A	9,98	1	9,98
Cobre esmaltado	4,46	2	8,92
Resistencias (2.600 uds.)	19,99	1	19,99
DAC	1,62	20	32,4
Arduino	35,10	3	105,30
Estaño de soldadura	5,99	1	5,99
PCB (5 uds.)	49,48	1	49,48
Carcasa 3D	3,67	1	3,67
Total			249,72

Tabla 3.6: Presupuesto del prototipo

Hardware

El proyecto se realizará sobre un ordenador de gama media con las siguientes especificaciones: procesador i5-7260U a 2,20GHz, 8 GB de memoria RAM, SSD 250GB PCIe NVMe M.2. Lo que será igual a un coste de **109,11€** desglosado en:

Herramienta	Coste total	Vida útil (meses)	Uso (%)	Uso (meses)	Coste (€)
Ordenador personal	650,00	96	80	7	37,92
Conexión a Internet	33,90	-	30	7	71,19
Total					109,11

Tabla 3.7: Presupuesto hardware

Software

Mayoritariamente el software utilizado es libre o tiene una versión gratuita lo suficientemente potente para llevar a cabo el desarrollo de este proyecto, por lo que no requiere un coste extra. El coste total del software será de **2,91€** desglosado en:

Herramienta	Coste total	Vida útil (meses)	Uso (%)	Uso (meses)	Coste (€)
QtCreator	0,00	-	50	7	0,00
Arduino IDE	0,00	-	30	7	0,00
Android Studio	0,00	-	50	7	0,00
Git	0,00	-	100	7	0,00
Texstudio	0,00	-	100	7	0,00
Draw.io	0,00	-	50	7	0,00
EasyEda	0,00	-	30	7	0,00
Blender	0,00	-	30	7	0,00
Cura	0,00	-	30	7	0,00
Gimp	0,00	-	10	7	0,00
Inkscape	0,00	-	20	7	0,00

(Continúa)

Herramienta	Coste total	Vida útil (meses)	Uso (%)	Uso (meses)	Coste (€)
Debian (GNU/Linux)	0,00	-	80	7	0,00
Microsoft Windows 10 OEM	39,90	96	100	7	2,91
Total					2,91

Tabla 3.8: Presupuesto software

Personal

Aunque solo una persona va a trabajar en todo el proyecto, esa persona desempeñará varios roles diferentes durante el mismo, incluyendo analista de sistemas, ingeniero hardware y desarrollador de Android y C++. El coste del personal será **4.943,10€** desglosado en:

Rol	Sueldo (€/hora)	Tiempo (horas)	Coste (€)
Analista de sistemas (30 %)	14,72	90	1.324,80
Ingeniero hardware (10 %)	19,42	30	582,60
Desarrollador Android (30 %)	15,44	90	1.389,60
Desarrollador C++ (30 %)	18,29	90	1.646,10
Total			4.943,10

Tabla 3.9: Presupuesto del personal. Fuente: Jooble.org [6]

Estimación de costes totales

A partir de estos cálculos se estima que el proyecto coste un total de **5.304,84€** desglosado en:

	Prototipo	Hardware	Software	Personal	Total
Porcentaje (%)	4,71	2,06	0,05	93,18	100,00
Coste (€)	249,72	109,11	2,91	4.943,10	5.304,84

Tabla 3.10: Estimación de costes totales

3.4. Balance final del proyecto

El propósito de esta sección es evaluar la precisión de las estimaciones, proporcionando retroalimentación al proceso de planificación y evaluando su acierto. En esta sección se presentan los costes del proyecto ya finalizado y se comparan con las estimaciones, con el fin de evaluar la diferencia de coste.

3.4.1. Planificación

Debido a problemas de planificación externos se ha comprobado que el tiempo resultó muy ajustado a la hora de realizar el proyecto, en especial todo lo relacionado con la biblioteca Pulse, es decir, con los tiempos de entrega de desarrollo hardware y software de la biblioteca.

El **primer incremento** duró más de lo esperado, alcanzando, justamente, los 63 días, ya que:

- No se lograba hacer un hardware funcional. El circuito del dispositivo no funcionaba correctamente y se tuvo que probar de distintas formas.
- Se barajaron varios sistemas de redundancia de información con distintos tamaños. Se probó con código convolucional pero se precisaba de perforaciones sobre el código para tener una tasa de "redundancia/datos enviados" de entrono el 20 %. Al usar un código convolucional perforado se suprimen bits específicos de información que impiden recuperar los datos. Esto solamente permitía detectar errores pero no repararlos.

  Al final se decidió utilizar una codificación en bloque, Reed-Solomon A, con los tamaños adaptados a los de una tabla prefijada 6.21, para que se adaptase a ése 20 %.
- Problemas de lecturas de datos del móvil:
  - Inicialmente se producían muchos errores de calibrado. Debido a eso se decidió que debería de haber una etapa de precalibrado, a parte de una de calibrado, para verificar que el móvil se encuentra sobre un dispositivo y se hagan las lecturas correctas.
  - Se experimentó con otra opción de calibrado: recorrer todos los niveles que emite un solenoide. Pero se vio que podría ser lento para solenoides que emitieran muchos bits por Pulso A.

- Problemas con la API de Android: Inicialmente se iban a leer datos a mayor velocidad (30 ms), pero se dio por imposible porque, aunque los sensores sí que suelen admitir una mayor velocidad, la API Android no recolecta de forma tan continua datos y se realiza de forma aleatoria.

El **cuarto incremento** también duró más de lo esperado, hasta los 40 días. Resultaron muy pocos días para realizar toda la interfaz gráfica del programa y de la aplicación. Además la placa PCB, junto a la carcasa en 3D, tardó más en llegar de lo esperado, lo que retrasó procesos de verificación y testeo.

Esto implica que todo el proyecto se ha visto retrasado 21 días. Trabajando 2 horas diarias de Lunes a Viernes, son 42 horas adicionales sumadas a las 300 horas proyectadas, lo que hace un proyecto con un total de 342 horas con finalización el día 27/03/2023.

3.4.2. Presupuesto económico

Debido al apartado anterior, se produjeron variaciones en el presupuesto del personal, dando, como resultado, un coste de **5.697,42€** desglosado por:

Rol	Sueldo (€/hora)	Tiempo (horas)	Variación (%)	Coste (€)
Analista de sistemas (26,32 %)	14,72	90	0 %	1.324,80
Ingeniero hardware (14,04 %)	19,42	48	60 %	932,16
Desarrollador Android (29,82 %)	15,44	102	13,33 %	1.574,88
Desarrollador C++ (29,82 %)	18,29	102	13,33 %	1.865,58
Total			15,26 %	5.697,42

Tabla 3.11: Coste del personal. Fuente: Jooble.org [6]

Además, tal y como era previsible, la estimación del prototipo hardware fue muy imprecisa. Se incluyen todos los componentes con el envío ya incluido. Se incluye un impuesto adicional imprevisto generado por las aduanas Europeas. El coste total de los componentes del prototipo es de **480,41€**, con un desvío del **92,38 %** del presupuesto original, desglosado en:

Componentes	Coste (€)	Cantidad	Total (€)
Protoboard A	1,06	3	3,18
Protoboard A (3 uds.) y cables Jumpers A	12,99	1	12,99
Cobre esmaltado 0,6ø(25m)	5,33	2	10,66
Cobre esmaltado 0,6ø(100g)	4,47	2	8,94
DAC A (DAC0800LCN)	1,82	19	34,58
Amplificador operacional (LM324)	0,553	10	5,53
Condensador de disco cerámico 0.01nF	0,472	10	4,72
Resistencia $1,2\ k\Omega$	1,13	10	11,30
Resistencia 10 Ω	0,06	25	1,50
Resistencia 10 $k\Omega$	0,035	100	3,50
Conversor de voltaje -5V (ICL 7760M)	0,764	5	3,82
Sensor de proximidad (APDS-9960)	1,36	10	13,60
Arduino micro	20,00	4	80,00
Arduino 33 BLE	35,00	4	140,00
PCB (5 uds.)	58,71	1	58,71
Conectores mini micro SH 1.0mm (20 pares)	10,99	1	10,99
Conectores mini micro JST 1.0mm (100 pares)	11,99	1	11,99
Caracasa 3D	5,98	3	17,94
Tapa Carcasa 3D	2,66	3	7,98
Tapones cerramiento 3D (50 uds.)	6,59	1	6,59
Estaño de soldadura	13,99	1	13,99
Impuestos extraordinarios	17,90	-	17,90
Total			480,41

Tabla 3.12: Coste del prototipo

Así que el coste final del proyecto ha sido de **6.289,85€**, con un desvío del **18,57 %** sobre la estimación realizada, desglosados en la siguiente tabla:

	Prototipo	**Hardware**	**Software**	**Personal**	**Total**
Porcentaje (%)	7,64	1,73	0,05	90,58	100,00
Coste (€)	480,41	109,11	2,91	5.697,42	6.289,85
Variación (%)	92,38	-	-	15,26	18,57

Tabla 3.13: Costes totales

Parte II

Documentación técnica

Capítulo 4

Aplicación Impulse

En este apartado se desarrollará la documentación técnica de Impulse. Para ordenar adecuadamente los distintos elementos se ha decidido dotar a cada uno con un id exclusivo para el.

Se ha decidido utilizar IDs con números específicos para identificar cada registro dentro del sistema. Esta elección se basa en la necesidad de tener un control más riguroso sobre los datos y garantizar que cada registro sea fácilmente identificable por el usuario.

Al asignar un número específico a cada ID, se puede asegurar que no haya duplicados ni errores en la asignación, facilitando el seguimiento de las actividades y cambios realizados en el sistema, lo que resulta útil de cara al análisis de sistema y a futuras ampliaciones. Los IDs han sido asignados de la siguiente forma:

- 1: Características del sistema
- 50: Actores
- 100: Requisitos de usuario, divididos por sistemas:
  - 100: Transmisor
  - 150: Receptor
- 200: Casos de Uso, divididos por sistemas:
  - 200: Transmisor
  - 250: Receptor
- 300: Atributos de calidad

4.1. Requisitos funcionales

4.1.1. Características del sistema

Las características del sistema representan las funcionalidades fundamentales de las que éste debe de contar. Se ha decidido dividir las características en dos árboles debido a que el sistema está dividido en dos: transmisor y receptor.

Transmisor:

- Car-01 Administración de Pulse: Incluye administrar todas las funciones de la librería Pulse.
- Sub-01.1 Eliminar una tarea ya agregada: Permite eliminar una tarea, independientemente de que esté en curso o no.
- Sub-01.2 Reporte de estado: Aporta información acerca del dispositivo, del progreso de las distintas tareas y de los errores que se puedan producir.
- Sub-01.3 Modificar tarea: Se podrá modificar la repetición de las tareas.
- Sub-01.4 Agregar las distintas tareas: Añadirá las tareas (ficheros, mensajes, wifi, sms, e-mail y bitcoin) con los campos necesarios, verificando que sean correctos.
- Sub-01.5 Conexión al dispositivo transmisor: Se mostrará una lista de dispositivos disponibles y se podrá administrar la conexión y desconexión con el dispositivo.
- Car-02 Ajustes adicionales: Incluye toda la configuración necesaria para poder utilizar correctamente el programa.
- Sub-02.1 Configuraciones de envío: Representa cómo se enviarán y cómo se mostrarán las distintas tareas.
- Sub-02.2 Comprobar actualizaciones online: Busca a través de la API del repositorio Git si hay nuevas actualizaciones.
- Sub-02.3 Información software: Muestra información de la versión software, las licencias y librerías utilizadas y sus autores.
- Sub-02.4 Modo noche: Cambia el aspecto del programa a un modo oscuro.
- Car-03 Modo desarrollador: Permite saber información precisa de todo lo que ha sucedido en el móvil.

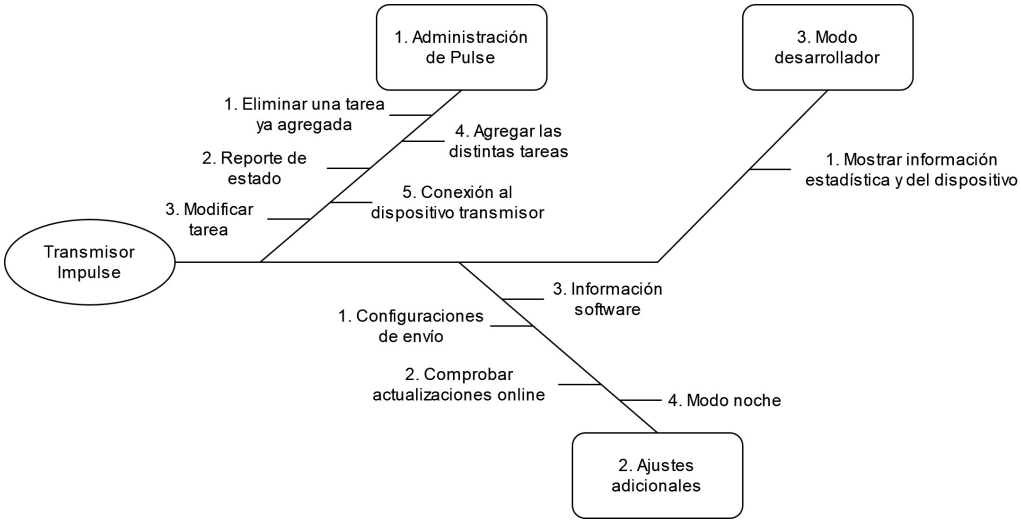


Figura 4.1: Árbol de características Transmisor

- Sub-03.1 Mostrar información estadística y del móvil : Muestra al usuario toda la información necesaria.

Receptor:

- Car-04 Tutorial: Ayuda al usuario a entender la app.
- Sub-04.1 Comprobar compatibilidad Pulse: Verifica si la librería es compatible con el móvil.
- Sub-04.2 Habilitar permisos: Pide los permisos necesarios para buscar el dispositivo, conectarse a él y, si el usuario lo desea, recibir información.
- Car-05 Administración de Pulse: Incluye administrar todas las funciones de la librería Pulse.
- Sub-05.1 Resetea la biblioteca: Detiene la ejecución de la biblioteca Pulse para volver a iniciarla, eliminando todos los datos.
- Sub-05.2 Reporte de Estado: Muestra información del modo de recepción y el progreso de una tarea.
- Sub-05.3 Recepción de tareas: Una vez recibida la tarea, se deben de ejecutar las correspondientes acciones.
- Sub-05.4 Información estadística: Incluye datos simplificados que permitan comprender rápidamente la cantidad de datos que se han recibido.

- Car-06 Ajustes adicionales: Incluye toda la configuración necesaria para poder utilizar correctamente la APP.
- Sub-06.1 Modo noche: Cambia la interfaz y la muestra en modo oscuro.
- Sub-06.2 Cambiar almacenamiento: Cambia la carpeta dónde se almacenará los ficheros.
- Sub-06.3 Información software: Aporta datos acerca de la versión del software, autor y otras librerías utilizadas.
- Car-07 Modo desarrollador: Permite mostrar y almacenar información precisa de todo lo que está sucediendo en el móvil.
- Sub-07.1 Mostrar información procesada: Se muestra información de los distintos pulsos magnéticos que se están recibiendo.
- Sub-07.2 Almacenar fichero de desarrollador: Se almacenan datos que se podrán analizar en el programa transmisor.

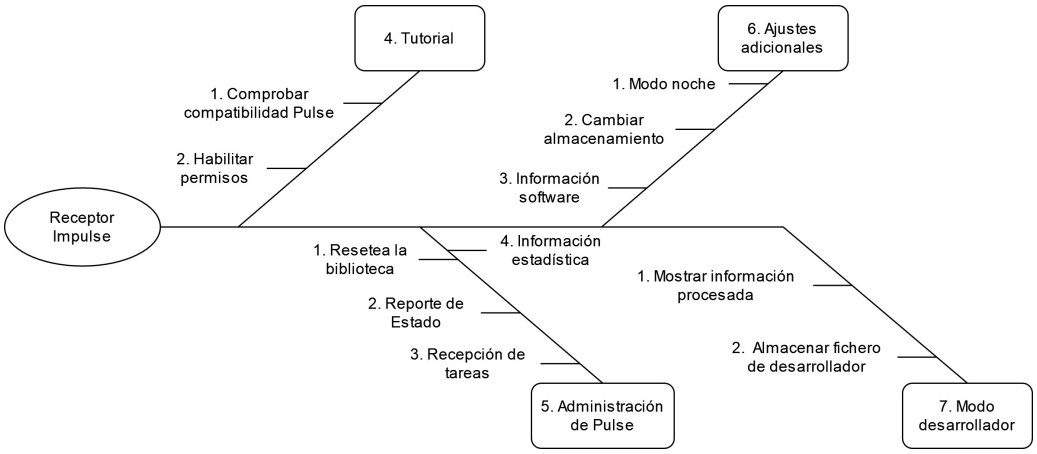


Figura 4.2: Árbol de características Receptor

4.1.2. Actores

Entidad externa al sistema, incluyendo tanto personas como sistemas externos ajenos que interactúan directamente con el software. Los actores de esta aplicación serán:

- Ac-50 Biblioteca Pulse: Recibe información del entorno, a través del dispositivo o del hardware, mediante la API interna de Android, generando nuevos eventos.

- Ac-51 Usuario transmisor: Interactúa con el ordenador. Genera las distintas tareas de la biblioteca Pulse.
- Ac-52 Usuario receptor: Interacciona con el dispositivo móvil mostrando la información recibidas por las tareas de la biblioteca.

No hay una jerarquía entre todos los usuarios. Son todos independientes. Ninguno depende directamente de otro.

4.1.3. Requisitos de usuario

Id	Método	Dependencia
RU-100	Comprueba si hay actualizaciones	Sub-2.2
RU-101	Envío tareas bluetooth	Sub-2.1
RU-102	Lee y almacena la configuración	Car-2
RU-103	Oculta y muestra las tareas	Sub-2.1
RU-104	Modo noche	Sub-2.4
RU-105	Información acerca de licencias	Sub-2.3
RU-106	Expone información estadística de una muestra	Sub-3.1
RU-107	Muestra gráfica	Sub-3.1
RU-108	Mensaje de error	Sub-1.2
RU-109	Se conecta a un dispositivo	Sub-1.5
RU-110	Crear e inicializar la tarea	Sub-1.4
RU-111	Panel de tareas	Sub-1.1 Sub-1.2
RU-150	Acceder al tutorial	Car-4
RU-151	Habilitar permisos	Sub-4.2
RU-152	Comprobar compatibilidad	Sub-4.1
RU-153	Seleccionar almacenamiento	Sub-6.2
RU-154	Ajustes adicionales	Car-6
RU-155	Modo noche	Sub-6.1
RU-156	Información acerca de las licencias	Sub-6.3

(Continúa)

Id	Método	Dependencia
RU-157	Modo desarrollador	Car-7 Sub-7.2
RU-158	Muestra gráfica	Sub-7.1
RU-159	Mostrar información Pulse	Car-5
RU-160	Información estadística	Sub-5.4
RU-161	Mensaje de error	Sub-5.2
RU-162	Progreso de una tarea	Sub-5.2
RU-163	Estado de recepción	Sub-5.2
RU-164	Campos magnéticos	Sub-5.4
RU-165	Recepción de tareas	Sub-5.3
RU-166	Reiniciar la biblioteca	Sub-5.1

Tabla 4.1: Requisitos de usuario

4.1.4. Casos de Uso

En la figura 4.3 se puede ver cómo el "Usuario Transmisor" interacciona cambiando los ajustes y cómo acceder al desarrollador. Además también se puede ver cómo la Biblioteca Pulse interacciona con el usuario transmisor, conectado el dispositivo y administrando las distintas tareas. La "Biblioteca Pulse" se encargará de interpertar la información que se recibirá del dispositivo.

En la figura 4.4 se puede apreciar cómo el "Usuario Receptor" inicia el tutorial, modifica los ajustes, entra en modo desarrollador y recibe las tareas que se emitan. La "Biblioteca Pulse" se encargará de adquirir información del acelerómetro y magnetómetro para, posteriormente, interpretar la información, verificar que se ha recibido correctamente y mostrar información estadística.

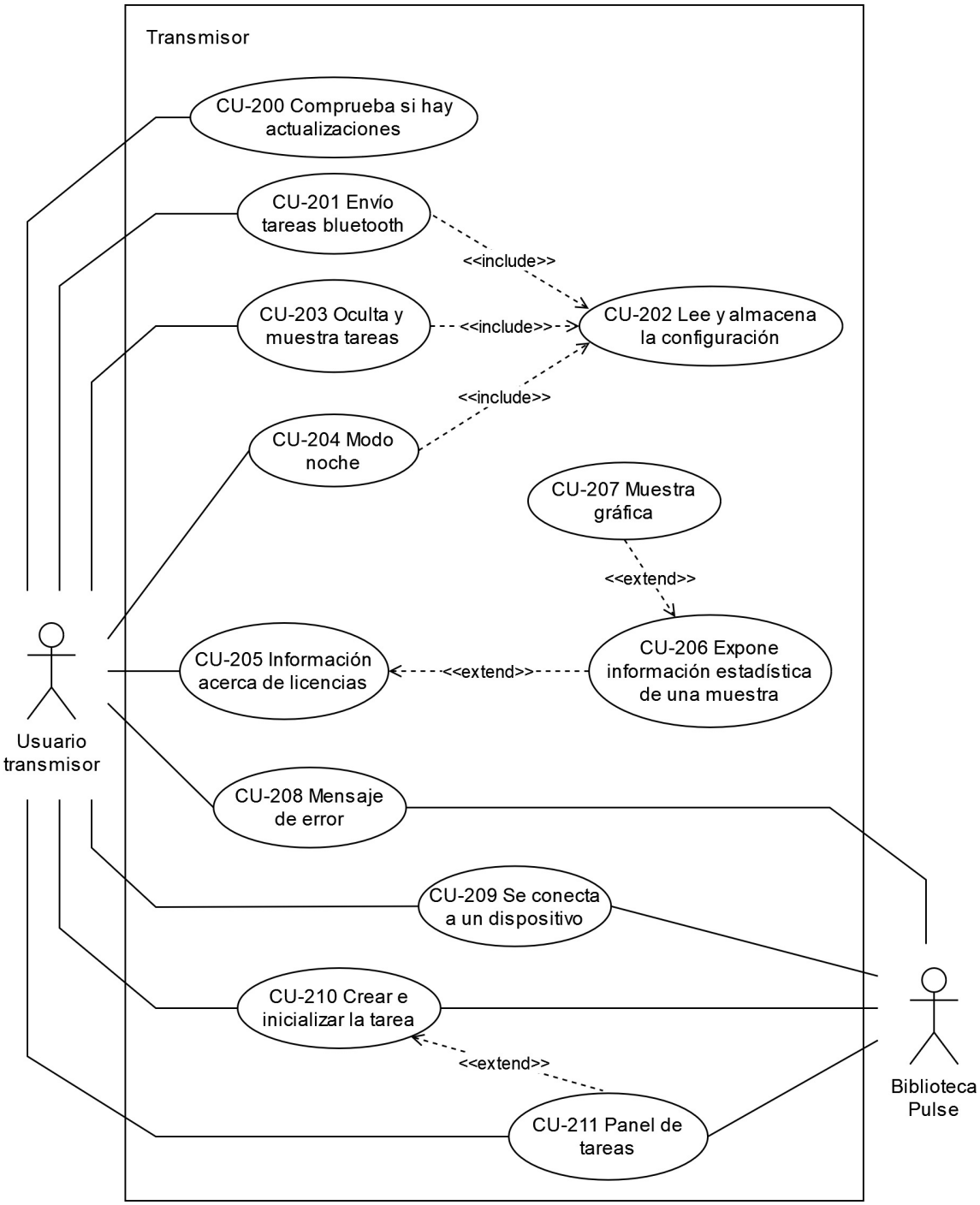


Figura 4.3: Caso de uso del sistema “Transmisor”

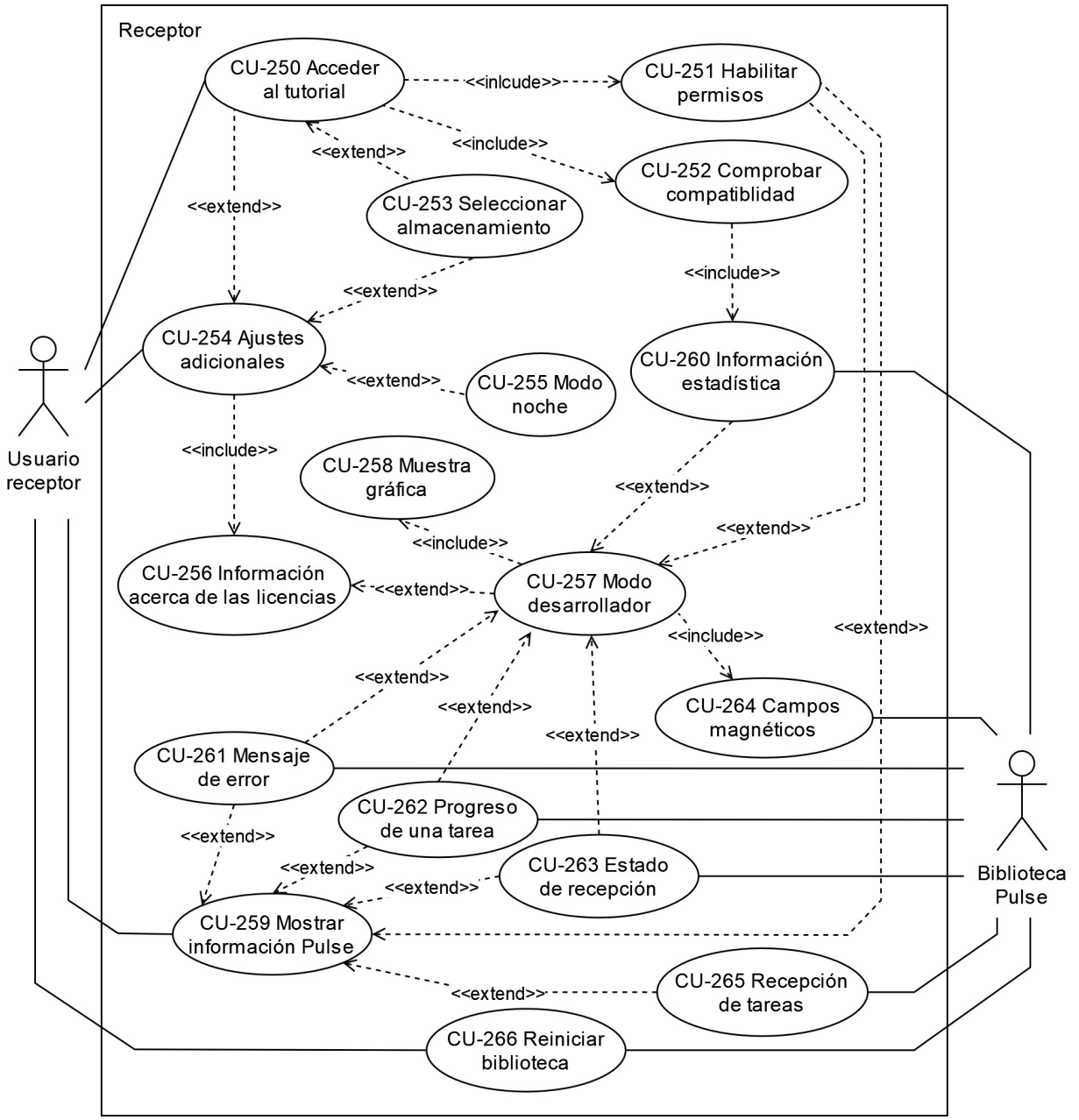


Figura 4.4: Caso de uso con del sistema “Receptor”

4.1.5. Especificaciones de Casos de Uso

[CU-200]: Comprueba si hay actualizaciones

Versión: 1.0

Requisito Asociado: [RU-100]

Descripción: Muestra al usuario si hay pendiente una actualización de software.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Flujo Principal

1. El usuario transmisor solicita comprobar actualizaciones.
2. El sistema se conecta a la API y descarga un JSON con las versiones disponibles.
3. El sistema parsea y comprueba que la última versión no es equivalente a la actual.
4. El sistema requiere al usuario transmisor si desea actualizar.
5. El usuario transmisor solicita no actualizar.
6. El caso de uso finaliza correctamente.

Flujo Secundario

2.a. El sistema no se puede conectar.

2.a.1. Se muestra un mensaje de error.

2.a.2. El caso de uso finaliza de manera errónea.

3.a. El sistema detecta que la versión es equivalente a la actual.

3.a.1. El sistema muestra un mensaje de que no hay nuevas actualizaciones.

3.a.2. El caso de uso finaliza correctamente.

5.a. El usuario transmisor solicita actualizar.

5.a.1. Se abre la página del repositorio para que el usuario descargue el programa.

5.a.2. El caso de uso finaliza correctamente.

[CU-201]: Envío tareas bluetooth

Versión: 1.0

Requisito Asociado: [RU-101]

Descripción: Cambia los ajustes del método de envío de las distintas tareas.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Include (Clase principal)

[CU-202] - Lee y almacena la configuración

Flujo Principal

1. El usuario transmisor solicita modificar el método de envío.

2. Si está marcado el envío, el sistema deshabilita el envío bluetooth de tareas y modifica la configuración.

3. El caso de uso finaliza correctamente.

Flujo Secundario

2.a. Si está desmarcado el envío.

2.a.1. El sistema habilita el envío bluetooth de tareas y modifica la configuración.

[CU-202]: Lee y almacena la configuración

Versión: 1.0

Requisito Asociado: [RU-102]

Descripción: Lee y almacena la configuración del sistema en un fichero.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Flujo Principal

1. El sistema intenta leer el fichero de configuración.

2. El sistema parsea los valores leídos y los almacena en variables.

3. El usuario transmisor solicitará leer o cambiar las variables correspondientes.

4. El sistema serializará la configuración cuando ya no sea preciso leer o alterar los valores, guardándolos en un fichero.

5. El caso de uso finaliza correctamente.

Flujo Secundario

1.a. El sistema no encuentra el fichero de configuración o no es legible.

1.a.1. El sistema inicializa las variables con los valores por defecto.

Postcondición

Se guarda el fichero de configuración satisfactoriamente.

[CU-203]: Oculta y muestra tareas

Versión: 1.0

Requisito Asociado: [RU-103]

Descripción: Permite ocultar o mostrar gráficamente las tareas que el usuario transmisor agregará.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Include (Clase principal)

[CU-202] - Lee y almacena la configuración

Flujo Principal

1. El sistema muestra una lista de las distintas tareas.

2. El usuario transmisor solicita modificar ocultar o mostrar una de las distintas tareas.

3. Si está marcado que se muestre, el sistema deshabilita la tarea seleccionada, modifica la configuración y oculta la tarea de la interfaz.

4. El caso de uso finaliza correctamente.

Flujo Secundario

3.a. Si está desmarcado que se muestre.

3.a.1. El sistema habilita la tarea seleccionada, modifica la configuración y muestra la tarea de la interfaz.

[CU-204]: Modo noche

Versión: 1.0

Requisito Asociado: [RU-104]

Descripción: Se cambia el aspecto de la aplicación a un modo oscuro o claro.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Include (Clase principal)

[CU-202] - Lee y almacena la configuración

Flujo Principal

1. El usuario transmisor solicita modificar el modo noche.

2. Si está marcado el modo noche, el sistema deshabilita el modo noche, modifica la configuración y modifica el aspecto del programa a una tonalidad más clara.

3. El caso de uso finaliza correctamente.

Flujo Secundario

2.a. Si está desmarcado el modo noche, el sistema habilita el modo noche, modifica la configuración y modifica el aspecto del programa a una tonalidad más oscura.

Postcondición

El sistema habrá cambiado de aspecto de la aplicación.

[CU-205]: Información acerca de licencias y otros datos

Versión: 1.0

Requisito Asociado: [RU-105]

Descripción: Muestra información acerca del software, las versión y las distintas bibliotecas. Permite traducir la aplicación o contactar con el creador. Además posibilita acceder al modo desarrollador.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Extend (Clase principal)

[CU-206] - Expone información estadística de una muestra

Flujo Principal

1. El usuario transmisor solicita mostrar licencias.

2. El sistema muestra un listado con las distintas licencias, además de permitir traducir la aplicación o contactar con el creador.

3. El caso de uso finaliza correctamente.

[CU-206]: Expone información estadística de una muestra

Versión: 1.0

Requisito Asociado: [RU-106]

Descripción: Muestra información del dispositivo y de los distintos campos magnéticos en un tiempo especificado.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Extend (Clase principal)

[CU-207] - Muestra gráfica

Precondición

1. El usuario transmisor debe de haber ejecutado el caso de uso [CU-205] - Información acerca de licencias.

2. El usuario transmisor debe de haber solicitado acceder al modo desarrollador.

Flujo Principal

1. El usuario transmisor solicitará abrir un fichero.

2. El sistema leerá el fichero y lo parseará.

3. El sistema mostrará información del dispositivo del que se ha tomado la muestra.

4. El sistema mostrará un listado de los distintos campos magnéticos que se han tomado en un periodo de tiempo, además de una gráfica.

5. El caso de uso finaliza correctamente.

Flujo Secundario

1.a. El usuario solicita regresar a la pantalla principal.

1.a.1. El caso de uso finaliza correctamente.

[CU-207]: Muestra gráfica

Versión: 1.0

Requisito Asociado: [RU-107]

Descripción: Muestra una gráfica bidimensional que puede ser expandida para mostrar con mayor claridad los datos. Además permite exportar la gráfica a una imagen.

Actor(es) relacionado(s)

[Ac-51] - Usuario transmisor

Precondición

1. El usuario transmisor deberá de haber ejecutado el [CU-206] - Expone información estadística de una muestra.

2. El usuario transmisor deberá de haber solicitado abrir un fichero.

Flujo Principal

1. El sistema muestra una gráfica.

2. El usuario solicita mostrar u ocultar un conjunto de campos magnéticos X, Y ó Z.

3. El sistema muestra u oculta el conjunto de datos.

4. El caso de uso finaliza correctamente.

Flujo Secundario

2.a. El usuario solicita exportar la gráfica a una imagen.

2.a.1. El sistema solicita una ruta para almacenar la imagen.

2.a.2.a. El usuario cancela la operación.

2.a.2.a.1. El caso de uso finaliza erróneamente.

2.a.2.b. El usuario responde con una ruta.

2.a.2.b.1. El sistema guarda el fichero en la ruta deseada.

2.a.2.b.2. El caso de uso finaliza correctamente.

[CU-208]: Mensaje de error

Versión: 1.0

Requisito Asociado: [RU-108]

Descripción: Recibe un mensaje de error de la Biblioteca Pulse que se mostrará al usuario transmisor.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-51] - Usuario transmisor

Precondición

1. El usuario transmisor debe de haber ejecutado el caso de uso [CU-209] - Se conecta a un dispositivo con un resultado positivo.

Flujo Principal

1. La biblioteca Pulse emitirá un error.

2. Sólo si es un error irreversible el software se desconectará del dispositivo.

3. El sistema mostrará al usuario transmisor un mensaje informativo acerca del error.

4. El caso de uso se finaliza correctamente.

[CU-209]: Se conecta a un dispositivo

Versión: 1.0

Requisito Asociado: [RU-109]

Descripción: El software se conecta a un dispositivo hardware específico.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-51] - Usuario transmisor

Flujo Principal

1. El sistema pide una lista de dispositivos a Biblioteca Pulse.

2. El sistema muestra la lista de dispositivos a usuario transmisor.

3. El usuario transmisor selecciona el dispositivo al que se desee conectar.

4. El sistema realiza una petición a la Biblioteca Pulse de conexión con el dispositivo.

5. La Biblioteca Pulse responde que la conexión se ha realizado correctamente.

6. El sistema muestra al usuario transmisor que se ha conectado al dispositivo correctamente.

7. El caso de uso se finaliza correctamente.

Flujo Secundario

5.a. La Biblioteca Pulse responde que la conexión no se ha realizado correctamente.

5.a.1. El sistema muestra un mensaje de error.

5.a.2. El caso de uso de uso finaliza incorrectamente.

[CU-210]: Crear e inicializar la tarea
Versión: 1.0

Requisito Asociado: [RU-110]

Descripción: Permite crear una nueva tarea introduciendo los distintos datos.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-51] - Usuario transmisor

Extend (Clase principal)

[CU-211] - Panel de tareas

Precondición

El usuario transmisor debe de haber ejecutado el caso de uso [CU-209] - Se conecta a un dispositivo con un resultado positivo y el dispositivo debe de permanecer conectado

Flujo Principal

1. El sistema muestra un listado de posibles tareas.
2. El usuario transmisor escoge una tarea.
3. El sistema muestra los campos necesarios a rellenar para cumplimentar la tarea.
4. El usuario transmisor rellena los campos.
5. El sistema verifica los datos.
6. El sistema envía la tarea a la Biblioteca Pulse.
7. La Biblioteca Pulse se compromete a enviar la tarea cuando sea posible.
8. El sistema muestra el [CU-211] - Panel de tareas.
9. El caso de uso finaliza correctamente.

Flujo Secundario

5.a. El sistema encuentra errores en los datos.

5.a.1. El sistema muestra los errores cometidos.

5.a.2. Se regresa al paso 4.

[CU-211]: Panel de Tareas

Versión: 1.0

Requisito Asociado: [RU-111]

Descripción: Muestra un listado de las tareas.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-51] - Usuario transmisor

Precondición

1. El usuario transmisor debe de haber ejecutado el caso de uso [CU-209] - Se conecta a un dispositivo con un resultado positivo.

2. El dispositivo debe de permanecer conectado.

Flujo Principal

1. El sistema muestra una lista de las tareas con el progreso, según vaya informando la Biblioteca Pulse.

2. El usuario puede hacer una petición para modificar la repetición de una la tarea y el sistema se la notificará a la Biblioteca Pulse.

3. El usuario puede hacer una petición para eliminar una tarea y el sistema notificará a la Biblioteca Pulse de que debe de eliminarla.

4. El caso de uso se finaliza correctamente.

[CU-250]: Acceder al tutorial
Versión: 1.0

Requisito Asociado: [RU-150]

Descripción: Muestra un tutorial de utilización del sistema.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Precondición

Se ejecutará automáticamente al inicio si no se encuentra un valor que indique lo contrario o, en su defecto, cuando el usuario lo requiera.

Include (Clase principal)

[CU-251] - Habilitar permisos

[CU-252] - Comprobar compatibilidad

Extend (Clase principal)

[CU-253] - Seleccionar almacenamiento

Flujo Principal

1. El sistema muestra información acerca de cómo utilizar la aplicación.

2. El usuario deberá de leer la información necesaria.

3. El sistema verifica positivamente que posee los requisitos mínimos para ejecutarse.

4. El caso de uso acaba correctamente.

Flujo Secundario

3.a. El sistema no puede alcanzar los requisitos mínimos para ejecutarse.

3.a.1. El caso de uso acaba incorrectamente.

Postcondición

Si se ha acabado correctamente el caso de uso, se almacenará que no debe de volver a ejecutarse al inicio.

[CU-251]: Habilitar permisos

Versión: 1.0

Requisito Asociado: [RU-151]

Descripción: Habilita los permisos necesarios para que el sistema tenga capacidad para ejecutarse correctamente.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Flujo Principal

1. El sistema verifica que están habilitados los permisos necesarios para que el programa se ejecute correctamente. El sistema encuentra que están deshabilitados los permisos necesarios.

2. El sistema muestra una petición para que el usuario acepte los permisos

3. El usuario declina los permisos necesarios.

4. El sistema muestra un mensaje informativo aclarando que los permisos son necesarios para ejecutar la aplicación.

5. El usuario acepta que se vuelvan a pedir de nuevo los permisos necesarios. Se regresa al paso 2.

Flujo Secundario

1.a. El sistema encuentra que están habilitados los permisos necesarios.

1.a.1. El caso de uso acaba correctamente.

3.a. El usuario acepta los permisos necesarios.

3.a.1. El caso de uso acaba correctamente.

5.a. El usuario declina volver a pedir de nuevo los permisos necesarios.

5.a.1. El caso de uso acaba incorrectamente

[CU-252]: Comprobar compatibilidad

Versión: 1.0

Requisito Asociado: [RU-152]

Descripción: Se verifica si el dispositivo es compatible con la biblioteca.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario receptor

Include (Clase principal)

[CU-260] - Información estadística

Flujo Principal

1. El sistema inicializa la biblioteca Pulse en modo desarrollador.

2. El sistema espera durante un periodo de tiempo a que la biblioteca recolecte datos.

3. El sistema pide a la biblioteca Pulse información estadística relativa a la compatibilidad.

4. La biblioteca responde afirmativamente.

5. El sistema detiene la biblioteca Pulse.

6. El caso de uso termina correctamente.

Flujo Secundario

4.a. La biblioteca responde negativamente.

4.a.1. El sistema detiene la biblioteca Pulse.

4.a.2. El caso de uso acaba incorrectamente.

[CU-253]: Seleccionar almacenamiento

Versión: 1.0

Requisito Asociado: [RU-153]

Descripción: Se selecciona el lugar dónde se almacenarán los ficheros recibidos.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Flujo Principal

1. El sistema muestra un listado de las carpetas disponibles.
2. El usuario deberá de seleccionar la carpeta deseada.
3. El sistema almacena la ruta de la carpeta seleccionada.
4. El caso de uso acaba correctamente.

Flujo Secundario

2.a. El usuario no selecciona una carpeta.

2.a.1. El caso de uso acaba incorrectamente.

[CU-254]: Ajustes adicionales

Versión: 1.0

Requisito Asociado: [RU-154]

Descripción: Muestra un listado con los distintos ajustes y licencias de la aplicación.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Include (Clase principal)

[CU-256] - Información acerca de las licencias

Extend (Clase principal)

[CU-250] - Acceder al tutorial

[CU-253] - Seleccionar almacenamiento

[CU-255] - Modo noche

Flujo Principal

1. El sistema muestra un listado de ajustes disponibles de la aplicación.

2. El usuario solicita modificar un ajuste o acceder a información acerca de licencias.

3. El sistema ejecuta la modificación del ajuste o muestra la información acerca de licencias.

4. El caso de uso acaba correctamente.

[CU-255]: Modo noche

Versión: 1.0

Requisito Asociado: [RU-155]

Descripción: Modifica el tema de la aplicación a uno más claro u oscuro.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Flujo Principal

1. El sistema muestra el estado del ajuste.

2. El usuario solicita cambiar el estado del ajuste.

3. El sistema cambia el aspecto de la aplicación al opuesto (claro u oscuro) y almacena el ajuste.

4. El caso de uso acaba correctamente.

[CU-256]: Información acerca de las licencias

Versión: 1.0

Requisito Asociado: [RU-156]

Descripción: Muestra información acerca de licencias del desarrollador y de las bibliotecas utilizadas.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Extend (Clase principal)

[CU-257] - Modo desarrollador

Flujo Principal

1. El sistema muestra información acerca de todos las bibliotecas, del desarrollador, de Impulse y Pulse.

2. El sistema habilita el acceso al modo desarrollador de manera oculta.

3. El usuario solicita acceder al modo desarrollador.

4. El sistema permite acceder al modo desarrollador.

5. El caso de uso acaba correctamente.

Flujo Secundario

3.a. El usuario no accede al modo desarrollador.

3.a.1. El caso de uso acaba correctamente.

[CU-257]: Modo desarrollador

Versión: 1.0

Requisito Asociado: [RU-157]

Descripción: Muestra información necesaria para poder comprender los posibles errores que se den.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario receptor

Include (Clase principal)

[CU-258] - Muestra gráfica

[CU-264] - Campos magnéticos

Extend (Clase principal)

[CU-251] - Habilitar permisos

[CU-260] - Información estadística

[CU-261] - Mensaje de error

[CU-262] - Progreso de una tarea

[CU-263] - Estado de recepción

Flujo Principal

1. El sistema informa al usuario de la recepción de datos.

2. El caso de uso acaba correctamente.

Flujo Secundario

1.a. Si el usuario solicita borrar muestras.

1.a.1. El sistema borra todas las muestras guardadas hasta ahora y solicita al [CU-258] resetear la gráfica.

1.b. Si el usuario solicita almacenar muestras y tiene permiso de escritura.

1.b.1. El sistema almacena todas las muestras recogidas en un fichero y solicita al [CU-258] resetear la gráfica.

[CU-258]: Muestra gráfica

Versión: 1.0

Requisito Asociado: [RU-158]

Descripción: Muestra una gráfica limitada y ordenada en el tiempo, con la intensidad de los campos magnéticos que el móvil recibe.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Flujo Principal

1. El sistema muestra al usuario una gráfica con la distinta intensidad a lo largo del tiempo.

2. El usuario solicita mostrar u ocultar un conjunto de campos magnéticos X, Y ó Z.

3. El sistema muestra u oculta el conjunto de datos.

4. Se solicita borrar la gráfica actual.

5. El sistema borra todos los datos actuales.

6. El caso de uso finaliza correctamente.

[CU-259]: Mostrar información Pulse

Versión: 1.0

Requisito Asociado: [RU-159]

Descripción: Aglutina toda la información acerca de la recepción de las distintas tareas.

Actor(es) relacionado(s)

[Ac-52] - Usuario receptor

Extend (Clase principal)

[CU-251] - Habilitar permisos

[CU-261] - Mensaje de error

[CU-262] - Progreso de una tarea

[CU-263] - Estado de recepción

[CU-265] - Recepción de tareas

Flujo Principal

1. El sistema informa al usuario si se está recibiendo información, a través de qué medio, el progreso de recepción de la información, si se ha recibido alguna tarea y si se ha cometido algún tipo de error.

[CU-260]: Información estadística

Versión: 1.0

Requisito Asociado: [RU-160]

Descripción: Devuelve información estadística calculada.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

Flujo Principal

1. El sistema hace una petición a la biblioteca Pulse pidiendo información estadística.

2. La biblioteca Pulse arroja datos (media, máximo y mínimo de tiempo, número de muestras, hora de inicio y fin), además de si es compatible con el dispositivo actual.

3. El caso de uso acaba correctamente.

[CU-261]: Mensaje de error

Versión: 1.0

Requisito Asociado: [RU-161]

Descripción: Informa al sistema de que ha ocurrido un error.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario Receptor

Flujo Principal

1. La Biblioteca Pulse informa al sistema de que ha habido un error y del tipo de error que ha sucedido.

2. El sistema muestra al usuario un mensaje que informa del error.

[CU-262]: Progreso de una tarea

Versión: 1.0

Requisito Asociado: [RU-162]

Descripción: Informa al sistema del progreso de las distintas tareas.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario Receptor

Flujo Principal

1. La Biblioteca Pulse informa del progreso de la tarea al sistema.
2. El sistema muestra al usuario el porcentaje de progreso de la tarea.

[CU-263]: Estado de recepción

Versión: 1.0

Requisito Asociado: [RU-163]

Descripción: Informa al sistema del estado del dispositivo.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario Receptor

Flujo Principal

1. La Biblioteca Pulse informa del estado del dispositivo (preparado para recibir pulsos, escaneando bluetooth, error, recibir tareas por pulsos o bluetooth) al sistema.

2. El sistema informa al usuario del estado del envío.

[CU-264]: Campos magnéticos

Versión: 1.0

Requisito Asociado: [RU-164]

Descripción: Informa al sistema de que se ha recibido una nueva lectura magnético.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario Receptor

Flujo Principal

1. La Biblioteca Pulse envía una lectura magnética en X, Y, Z, y una marca de tiempo al sistema.

2. El sistema procesa las lecturas magnéticas dadas.

[CU-265]: Recepción de tareas

Versión: 1.0

Requisito Asociado: [RU-165]

Descripción: Informa un tipo de tarea con los distintos atributos de la tarea.

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario Receptor

Flujo Principal

1. La Biblioteca Pulse envía el tipo de tarea con los distintos atributos.
2. El sistema informa al usuario de la recepción de la tarea.

[CU-266]: Reiniciar la biblioteca

Versión: 1.0

Requisito Asociado: [RU-166]

Descripción: Reinicia la Biblioteca Pulse

Actor(es) relacionado(s)

[Ac-50] - Biblioteca Pulse

[Ac-52] - Usuario Receptor

Flujo Principal

1. El usuario solicita al sistema reiniciar la Biblioteca Pulse.
2. El sistema detiene la Biblioteca Pulse y la vuelve a iniciar.
3. El sistema informa al usuario que la Biblioteca se ha reiniciado correctamente.

4.2. Requisitos no funcionales

4.2.1. Atributos de Calidad

Se desarrollarán los atributos de calidad (AC) que representan las restricciones que el sistema tendrá y que permitirán verificar si se cumplen los requisitos mínimos necesarios.

Seguridad

- **AC-300**: El sistema verificará todas las entradas, comprobando que no se ejecute código no autorizado.

Usabilidad

- **AC-301**: El sistema deberá de ser fácil de utilizar, buscando el minimalismo.

Accesibilidad

- **AC-302**: El sistema tendrá un breve tutorial para explicar qué debe de hacer el usuario.
- **AC-303**: El sistema deberá de estar en inglés estándar (predeterminado) y en Español.

Robustez

- **AC-304**: El sistema podrá mantenerse activo en situaciones inusuales (errores de hardware, de emisión/recepción,...) permitiendo al usuario comprender el error.

Escalabilidad

- **AC-305**: Será compatible con la mayoría de los móviles (Android) y plataformas de Escritorio (Windows y GNU/Linux).

Capítulo 5

Hardware

El dispositivo, para poder recibir información a través de pulsos magnéticos, necesita emitir un campo magnético uniforme. Para ello se ha decidido utilizar un solenoide, ya que emite un campo magnético uniforme, facilitando las tareas de emisión.

En este apartado se expondrá el diseño del hardware y las distintas decisiones que se tomaron a la hora de crear el dispositivo.

5.1. Fundamentos Teóricos

5.1.1. Fundamentos del electromagnetismo: Solenoides

Un solenoide es una bobina enrollada en forma de hélice con un material por el que circula la corriente eléctrica. Es un elemento con capacidad para crear un campo magnético uniforme, fuerte en su interior y débil en el exterior. El campo magnético de un solenoide en la mitad del solenoide sería:

$$B = \mu_0 \cdot \frac{N \cdot i}{L} \tag{5.1}$$

Sin embargo en los extremos el campo magnético, aproximadamente, serían:

$$B = \mu_0 \cdot \frac{N \cdot i}{2 \cdot L} \tag{5.2}$$

siendo:

- B: Permeabilidad magnética (Teslas)
- μ_0: Permeabilidad magnética del vacío ($4\pi \cdot 10^{-7}$ $\frac{Henrios}{Metros}$)
- N: Número de espiras del solenoide
- i: La corriente que circula (Amperios)
- L: Longitud del solenoide (Metros)

Histéresis

Si el solenoide tiene un núcleo ferromagnético, a pesar de que tendrá un campo magnético más fuerte, el material tenderá a memorizar las propiedades magnéticas inducidas. El núcleo ferromagnético tendrá un imanación B (Densidad del flujo eléctrico) que en función de la excitación H (amp/cm) que se comportaría de la siguiente forma:

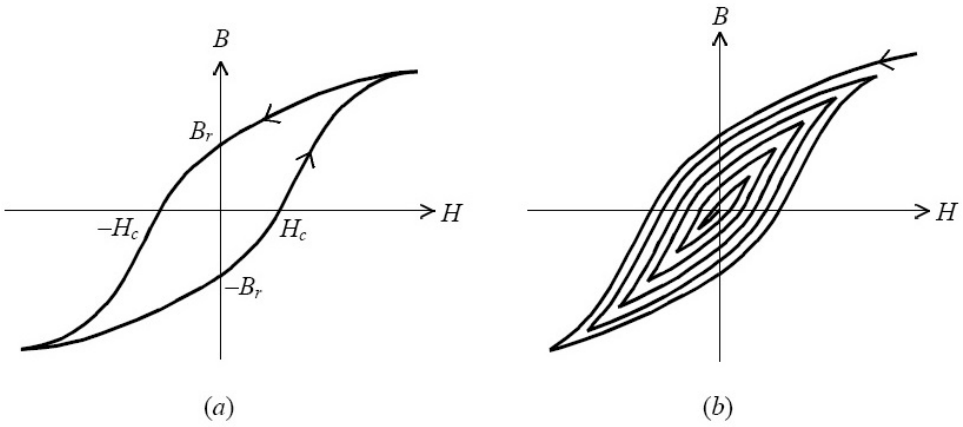


Figura 5.1: Fuerza coercitivas [8]. (a) Oscilación magnética del núcleo ferromagnético (b) Proceso de desimanación del material ferromagnético por una sucesión de oscilaciones magnéticas incompletas y decrecientes.

Para restaurar el campo magnético, habría que repetir distintos ciclos, reduciendo la imanación. Por eso se decidió no añadir un núcleo ferromagnético que permita tener un campo magnético más rápido y preciso, aunque fuera menos intenso.

Decisiones de diseño

Con esta simple introducción a la física se ha llegado a la conclusión de que el dispositivo deberá de contener un solenoide, sin ningún tipo de núcleo que produzca un ciclo de histéresis.

Además la Tierra emite un flujo magnético entre 22 μT − 67 μT, por lo que el solenoide debe de superar esa medida.

Por otro lado el dispositivo trabajará a 5V porque es el voltaje que puede aportar USB en cualquiera de sus versiones, pero no superar los 500 mA, que es la intensidad mínima de USB 2.0. Se recomienda no utilizar todo el amperaje y reservar entorno a unos 50mA, ya que el resto del circuito también consumirá.

A partir de ahí se puede decidir con el número de vueltas y diámetro que tendrá el solenoide, para ajustar los parámetros necesarios.

Según los cálculos realizados el solenoide que se utilizará será de 70 espiras, tendrá una intensidad de 428,58 mA (0,42858 A) y 6cm de diámetro, lo que implica una longitud de $d \cdot \pi = 6cm \cdot \pi = 18,85cm$ (0,1885 m). Así que el solenoide emitirá un campo magnético de:

$$B = \mu_0 \cdot \frac{N \cdot i}{2 \cdot L} = 4\pi \cdot 10^{-7} \cdot \frac{70 \cdot 0,42858}{2 \cdot 0{,}1885} = 0,00017500\ T = 175\ \mu T \tag{5.3}$$

5.2. Fundamentos de la electrónica

En el presente apartado se abordarán los componentes esenciales del circuito que se diseñará, con el propósito de comprender adecuadamente su funcionamiento.

5.2.1. Arduino

Arduino es una plataforma de hardware y software de código abierto que permite diseñar circuitos complejos de una manera fácil y sencilla. Consiste en una placa de circuito impreso que contiene un microcontrolador y una serie de puertos de entrada y salida, así como un entorno de programación integrado que permite cargar y ejecutar programas en la placa. Además posee una gran variedad de sensores compatibles, con lo que es fácil agregar una gran variedad de funcionalidades rápidamente.

El objetivo de utilizar una placa Arduino es que haga de interfaz entre el ordenador

y el resto de circuito, entregando los pulsos A justo en el tiempo preciso, además de encargarse de recibir los eventos que ocurran: movimiento del dispositivo que distorsione la transmisión y detectar si se ha colocado o retirado un móvil.

Asimismo, se considera la opción de integrar un módulo inalámbrico Bluetooth, lo cual habilitaría la transmisión de datos a través de esta tecnología, además de la transmisión por pulsos.

5.2.2. Conversor digital a analógico

Un conversor digital a analógico o DAC (Digital-to-Analog Converter) es un componente electrónico encargado de convertir una señal digital en analógica. Un DAC toma una serie de números digitales y lo convierte en una señal eléctrica que se utiliza para representar un valor en un rango continuo de valores.

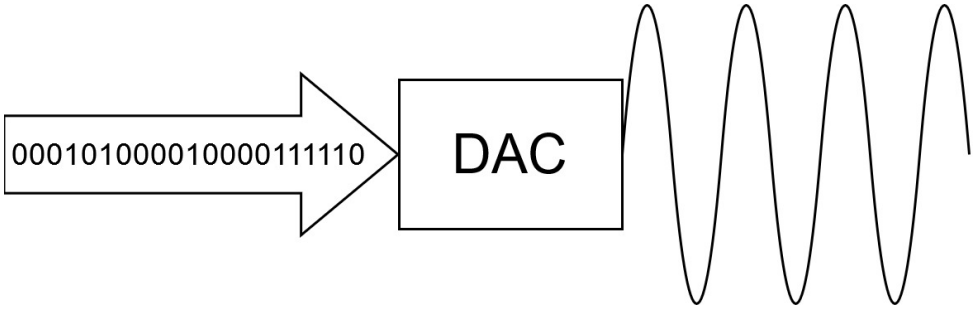


Figura 5.2: DAC convirtiendo una señal digital en analógico

Es habitual la utilización del DAC en sistemas de audio, dónde la información digital se tiene que convertir en una señal analógica reproducida en altavoces o auriculares. Los DAC se clasifican por su resolución, midiéndose en bits. Cuanta mayor sea la resolución más precisa será la señal analógica.

5.2.3. Amplificador de ganancia unitaria

También se pueden llamar seguidores de voltaje. Tal y como se aprecia en la imagen 5.3 la entrada del amplificador operacional forma parte de la propia salida, lo que significa que tanto la entrada como la salida del amplificador tienen el mismo voltaje.

El objetivo de utilizar un seguidor de voltaje es que, al tener una entrada con una impedancia muy alta, no se le exigirá mucha corriente al circuito. En otras palabras, si conectamos dos circuitos (A y B) y A tiene una impedancia muy baja, el circuito B le

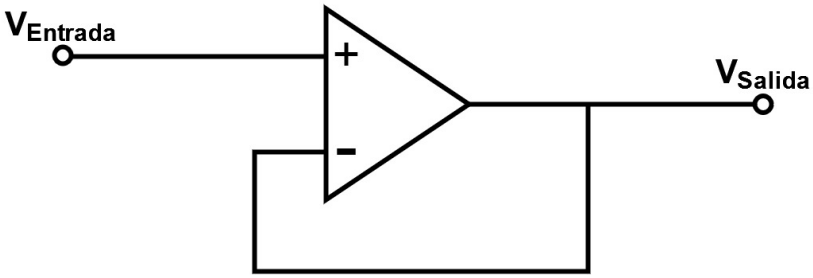


Figura 5.3: Amplificador de ganancia unitaria

exigirá mucha corriente al circuito A, pudiendo alterar los valores de voltaje. Sin embargo, si utilizamos un seguidor de voltaje, la impedancia muy alta de entrada del amplificador operacional hará que la corriente exigida sea mínima y, por lo tanto, que el voltaje no se vea afectado.

De esta forma se pueden conectar dos circuitos con tensiones distintas, aislados, sin distorsionar señales y entregando mayor potencia.

5.3. Desarrollo Hardware

Android acepta la lectura de hasta 3 solenoides, uno por cada eje de coordenadas del que se puede recibir la información.

A pesar de que se pueden colocar 3 solenoides por cada eje de coordenadas se vio que cada móvil tiene colocado en distinto lugar el magnetómetro. Esto haría que los solenoides deban de ser colocados en distintas posiciones del dispositivo para adaptarse al magnetómetro de cada teléfono.

Otro problema habitual es que los campos magnéticos de los distintos solenoides interfieren entre sí. La mejor solución para paliar este problema es crear campos magnéticos de menor tamaño, reduciendo el diámetro del solenoide, generando un campo más localizado.

Es obvio que si el campo magnético es más reducido, los solenoides deberán de estar mejor situados para que el magnetómetro pueda recibir la información correctamente. Entre ambos problemas (situación de los solenoides e interferencias entre ejes), se mostró como inviable tener múltiples solenoides a nivel práctico.

Aún así toda la biblioteca se programó para tener múltiples solenoides, por si a largo plazo, en futuras ampliaciones, se encontraba algún método que subsane ambos problemas a nivel físico.

5.3.1. Primera versión

Inicialmente se deseaba hacer un circuito funcional, que pudiera emitir pulsos. No se buscaba que fuera simple o compacto. Por lo que se hizo un circuito dependiente de una fuente de alimentación externa, muy poco compacto y sin sensores que complicaran el proceso de desarrollo. Como resultado se diseñó este circuito:

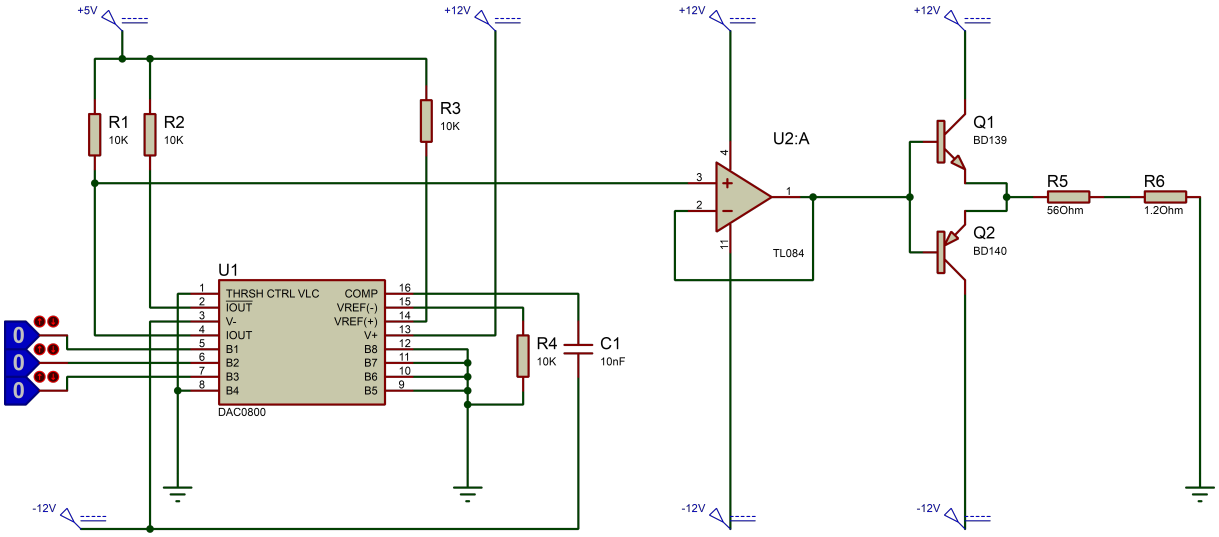


Figura 5.4: Diseño de la primera versión del circuito del solenoide

Listado de componentes:

- R1, R2, R3, R4: Resistencias de 10kΩ
- R5: Resistencia de 56Ω
- R6: Solenoide con resistencia de 1,2Ω
- C1: Condensador de 10nF
- U1: DAC0800
- U2: Amplificador operacional TL084
- Q1: Transistor NPN BD139
- Q2: Transistor PNP BD140

Se comenzó utilizando un Arduino micro, que no tiene ni acelerómetro que permita detectar el movimiento del dispositivo, ni conexión bluetooth. Tampoco incluye un sensor de proximidad, por lo que se añadió un interruptor que simulara que el móvil hubiera sido colocado/retirado del dispositivo.

Arduino recibe la información en formato digital desde el ordenador a través del puerto COM. Posteriormente Arduino envía la información a través de las entradas de bits que hay a la izquierda de la imagen 5.4, en este caso son 3 bits operacionales.

El DAC se encarga de convertir esos 3 bits en la corriente adecuada para cada nivel. El DAC0800 tiene una resolución de 8 bits y, como sólo se pretenden enviar 3 bits, se han

utilizado los inputs del B1 al B3 que son los bits más significativos (MSB). El DAC da su voltaje más bajo cuando todos los bits están a 1 y da su voltaje más alto cuando todos los bits están a 0.

Una vez el DAC devuelve la corriente conveniente, pasa al amplificador operacional que, como se dice en el apartado de "Amplificador de ganancia unitaria" 5.2.3, se encargará de aislar el circuito DAC de las distorsiones que produzcan los siguientes componentes, gracias a su entrada de muy alta impedancia.

Tras el amplificador operacional se encuentran dos transistores. Estos transistores conforman un amplificador de potencia de clase B, push & pull. Juntos se encargan de amplificar todo el rango de la señal. Sólo trabaja un transistor al mismo tiempo. El BD139, que es NPN, se encargará de amplificar las señales positivas. El BD140, que es PNP, se encargará de amplificar las señales negativas. Estos transistores no amplificarán si ninguno entra en activo.

El siguiente componente es una resistencia encargada de reajustar la salida de corriente a la necesitada por el solenoide. El último componente es el propio solenoide de 6cm y 70 vueltas.

Esta es la implementación del circuito descrito:

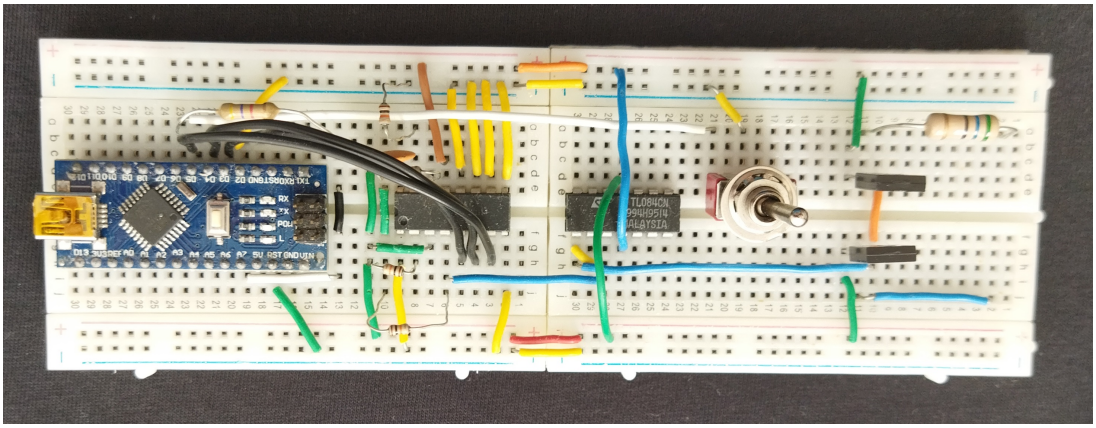


Figura 5.5: Primera implementación

5.3.2. Segunda versión

El objetivo de esta versión ha sido eliminar el interruptor y sustituirlo por un sensor de proximidad para detectar el móvil, simplificar los voltajes lo máximo posible (se eliminan los ±12V, quedando únicamente ±5V) y ajustar el solenoide para que se emitieran los Teslas necesarios para recibir correctamente la información.

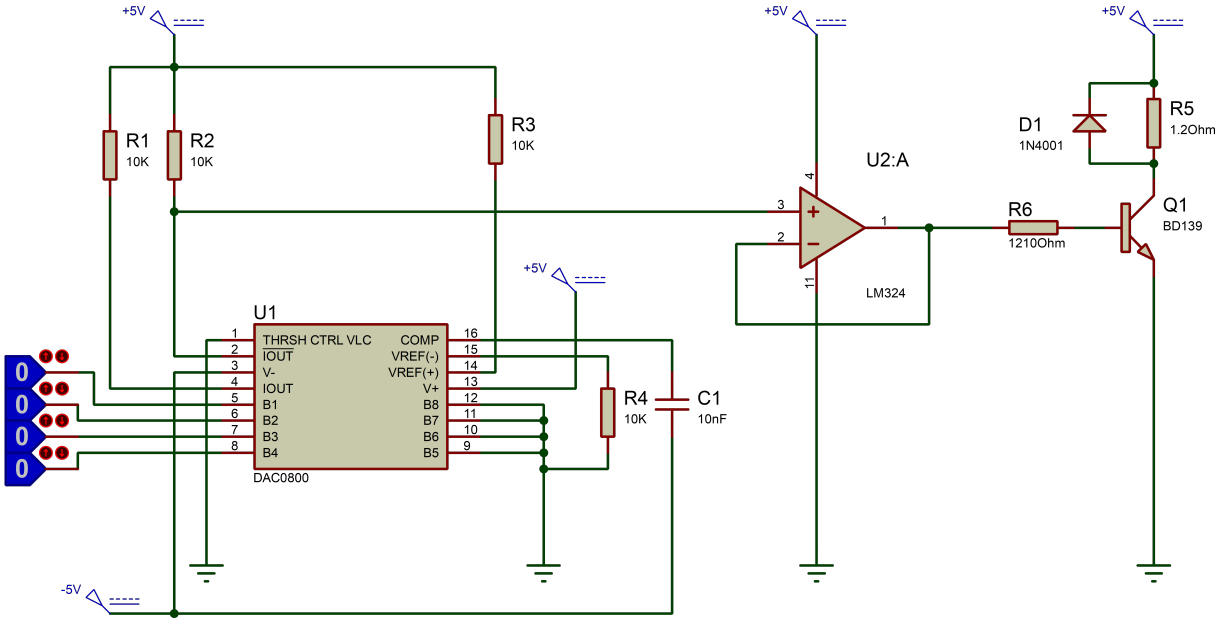


Figura 5.6: Diseño de la segunda versión del circuito del solenoide

Listado de componentes:

- R1, R2, R3, R4: Resistencias de 10kΩ
- R5: Resistencia de 1,21kΩ
- R6: Solenoide con resistencia de 1,2Ω
- C1: Condensador de 10nF
- U1: DAC0800
- U2: Amplificador operacional LM324
- Q1: Transistor NPN BD139
- D1: Diodo 1N4001

Como se puede observar en el DAC0800 se han eliminado todos voltajes ±12V y se han sustituido por ±5V que son valores que sí acepta el DAC0800. También se ha optado por cambiar la salida del DAC por su opuesta. Así el DAC no emitirá nada cuando los bits estén a 0 y sí que lo hará cuando haya más de un bit a 1.

Además el DAC usa 4 bits, aunque sólo se emitan 3. En vez de que el DAC emita con los bits más altos, que dan intensidades muy altas para el circuito, se optó por utilizar niveles más bajos. Al hacerlo así no se sobrecarga el circuito, es más eficiente y no se requiere una resistencia más potente, además de que no requiere ninguna alteración profunda del hardware.

Por otro lado, se ha cambiado el amplificador operacional. El anterior amplificador no soportaba un nivel de voltaje tan bajo de alimentación, además de que este no requiere alimentación negativa para su correcto funcionamiento.

También se ha cambiado el amplificador de potencia push & pull por un amplificador de clase A. Está formado por un único transistor NPN que se encargará de amplificador. Le precede una resistencia debido a que el amplificador LM324 tiene una impedancia más baja que el TL084, lo que implica una mayor cantidad de corriente que dañaría el transistor.

Por último, se incorpora un diodo con el fin de asegurar la integridad del flujo unidireccional de la corriente eléctrica, evitando cualquier posible cambio en su sentido.

Esta es la implementación del circuito descrito:

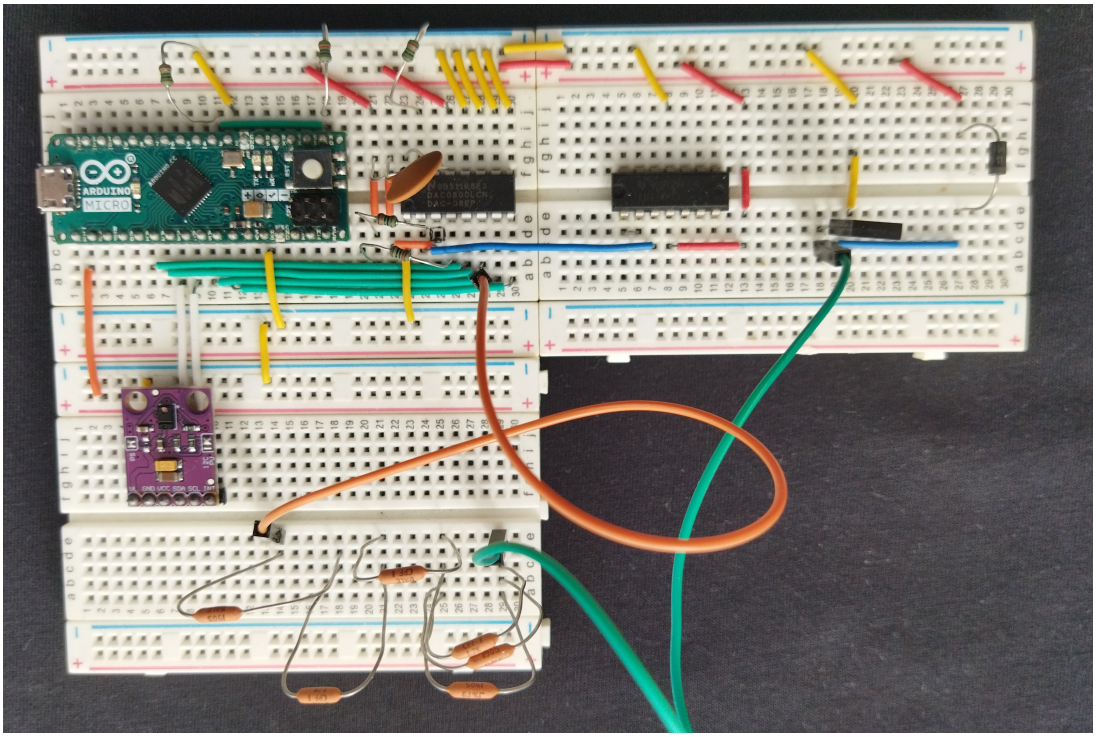


Figura 5.7: Segunda implementación

Como se puede observar, en esta implementación se han utilizado múltiples resistencias. El objetivo de utilizar tantas fue buscar la resistencia que el transistor necesitaba para que funcionar correctamente y para que el solenoide emitiera suficientes μT que evitaran el solapamiento con el campo magnético terrestre.

5.3.3. Tercera versión

El objetivo, en este caso, era buscar un prototipo totalmente independiente de voltajes externos, además de agregarle bluetooth y un acelerómetro para detectar que el dispositivo ha sido movido.

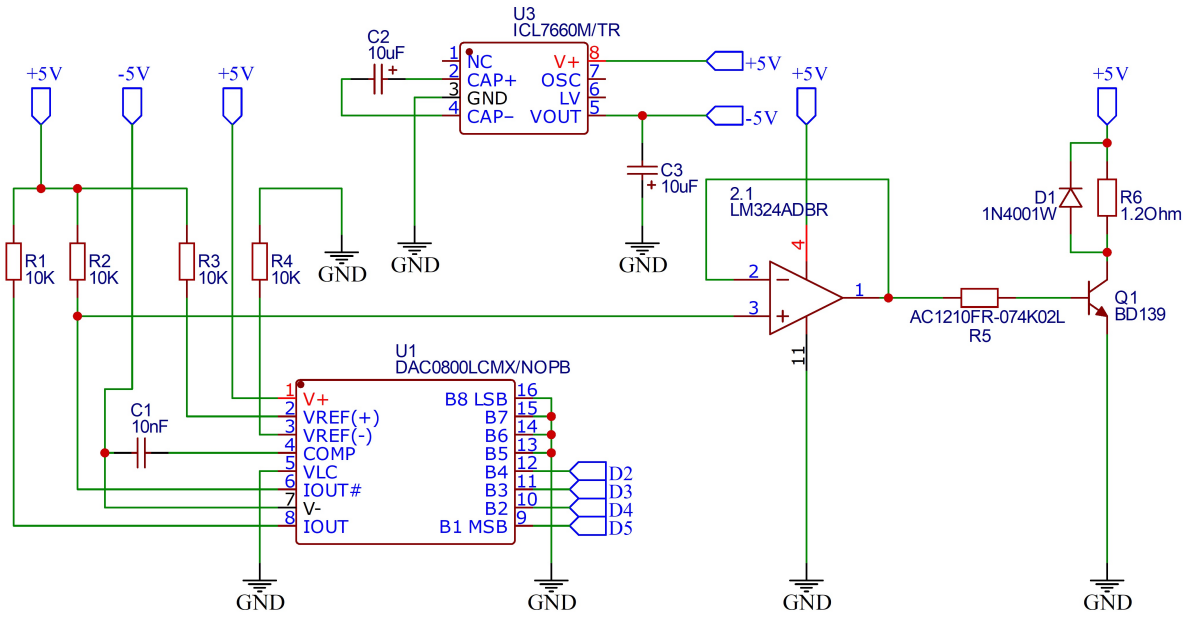


Figura 5.8: Diseño de la tercera versión del circuito del solenoide

Listado de componentes:

- R1, R2, R3, R4: Resistencias de 10kΩ
- R5: Resistencia de 1,21kΩ
- R6: Solenoide con resistencia de 1,2Ω
- C1: Condensador de 10nF
- C2, C3: Condensador de 10μF
- U1: DAC0800
- U2: Amplificador operacional LM324
- U3: Conversor de voltaje ICL7660
- Q1: Transistor NPN BD139
- D1: Diodo 1N4001

Para implementar tanto el bluetooth como el acelerómetro, simplemente se intercambió el Arduino micro por un Arduino 33 BLE.

Por último se ha agregado un conversor de voltaje de +5V a -5V. De esta forma el circuito sólo precisa únicamente de +5V para poder funcionar.

Para conseguir los +5V, sin depender de fuentes adicionales externas, Arduino 33 BLE tiene la capacidad de interconectar un pin de su placa con el voltaje directo del USB que, justamente, es de +5V. Para llevar a cabo esta tarea, se requiere realizar una soldadura manual en un puente de soldadura. Utilizando este pin de fuente de alimentación el prototipo será totalmente independiente de fuentes de alimentación externas.

Esta es la implementación del circuito:

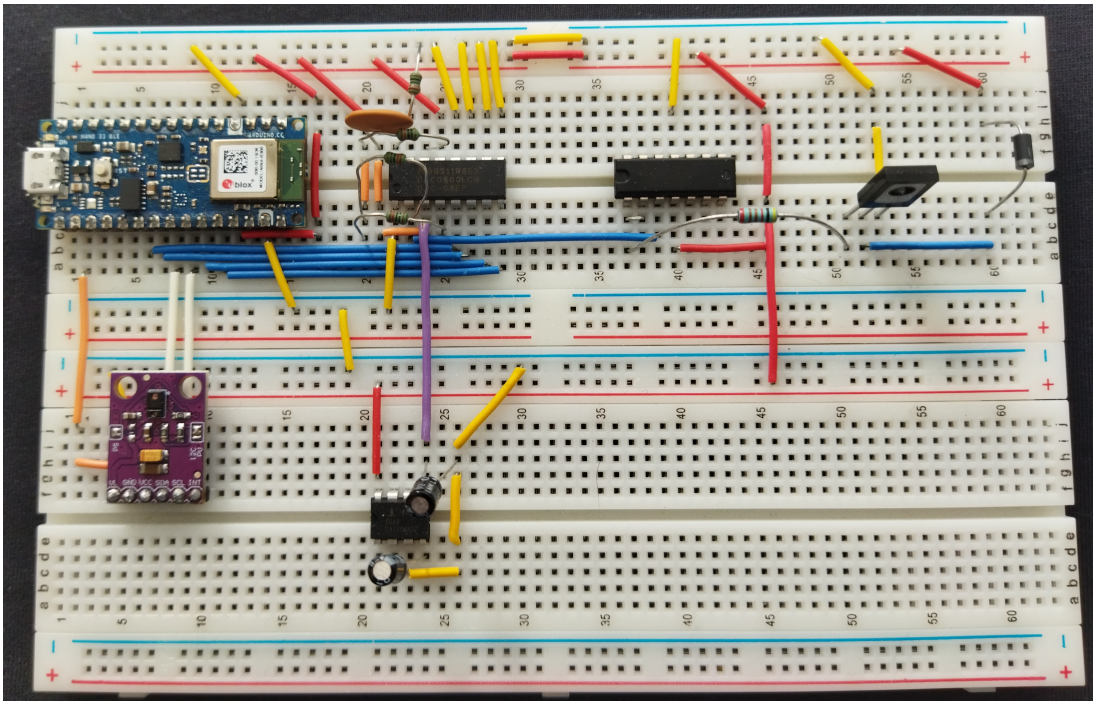


Figura 5.9: Tercera implementación

5.3.4. Cuarta versión

El objetivo de esta cuarta versión es la miniaturización lo máximo posible del circuito, además de permitir la sustitución de algunos componentes a través de conectores.

Listado de componentes:

- R1, R2, R3, R4: Resistencias de 10kΩ
- R5: Resistencia de 1,21kΩ
- C1: Condensador de 10nF
- C2, C3: Condensador de 10μF
- U1: DAC0800
- U2: Amplificador operacional LM321
- U3: Conversor de voltaje ICL7660
- Q1: Transistor NPN BD139
- D1: Diodo 1N4001
- H1: Conector izquierdo de Arduino
- H2: Conector derecho de Arduino
- CN1: Conector Sh de 2 pines
- CN2: Conector Sh de 4 pines

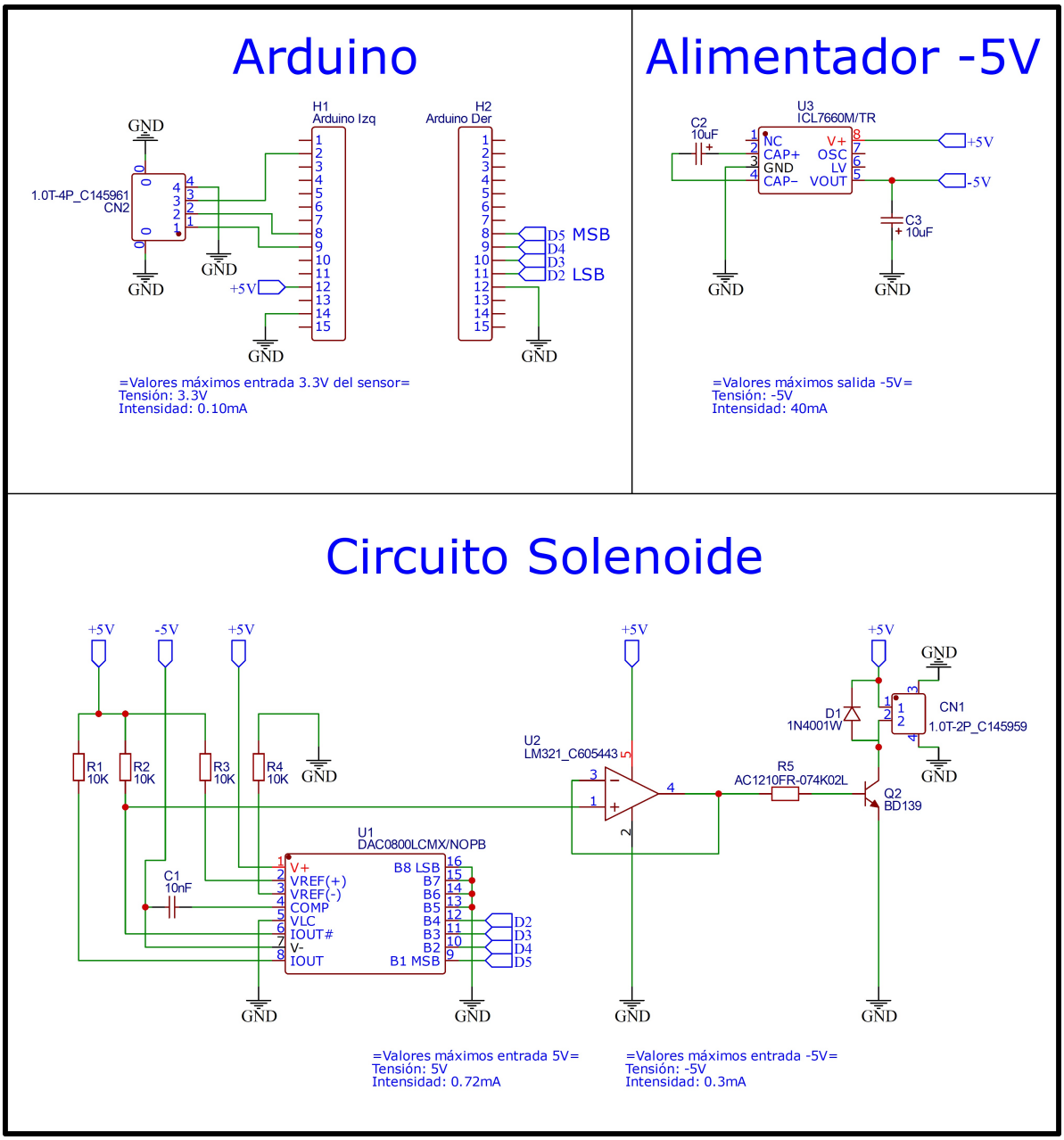


Figura 5.10: Diseño completo de la cuarta versión del circuito

Para miniaturizar se sustituye el amplificador operacional LM324 por otro totalmente equivalente, el LM321. La única diferencia es que el LM321 posee 1 única puerta lógica, mientras que el LM324 tiene 4 puertas. Eso requería más espacio que no se necesitaba.

Además se añaden 4 conectores. 2 conectores hembras de 15 pines, con 2.54 mm entre conector. Se utilizan para conectar un Arduino a la placa. Las protoboard Arduino 33

BLE son demasiado caras y escasas como para que exista el más mínimo error a la hora de soldarlas, por lo que se optó por esta opción. Dos conectores JST Sh, uno de 2 pines para el solenoide y otro de 4 pines para el sensor de proximidad.

De esta forma se podría sustituir fácilmente cualquier parte de los componentes: sensor de proximidad, arduino, solenoide o el propio circuito.

Adicionalmente se diseñó un circuito integrado, es decir, una PCB que compactara aún más el circuito. El diseño de la PCB y su implementación serían los siguientes:

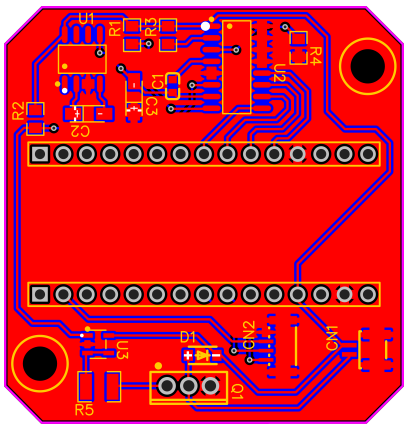


Figura 5.11: Diseño de la PCB

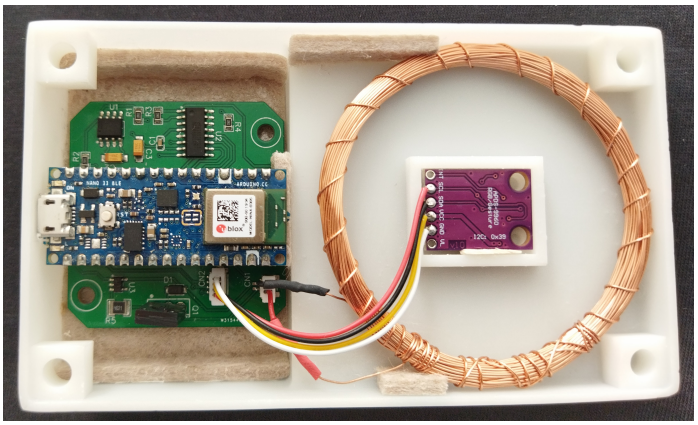


Figura 5.12: Cuarta implementación

5.3.5. Diseños 3D

Se ha diseñado una carcasa en un modelo 3D procurando que el que el dispositivo sea lo menos grueso posible para que colocar el móvil sea fácil y no se caiga.

El diseño se puede llegar a optimizar y aligerar. Se optó por hacerlo levemente más voluminoso para que la circuitería pudiera entrar con cierta facilidad y permitiera desarrollar más fácilmente el dispositivo. Para cerrar el dispositivo se ha optado por unos tapones plásticos, evitando el uso de tornillos y tuercas que compliquen el diseño.

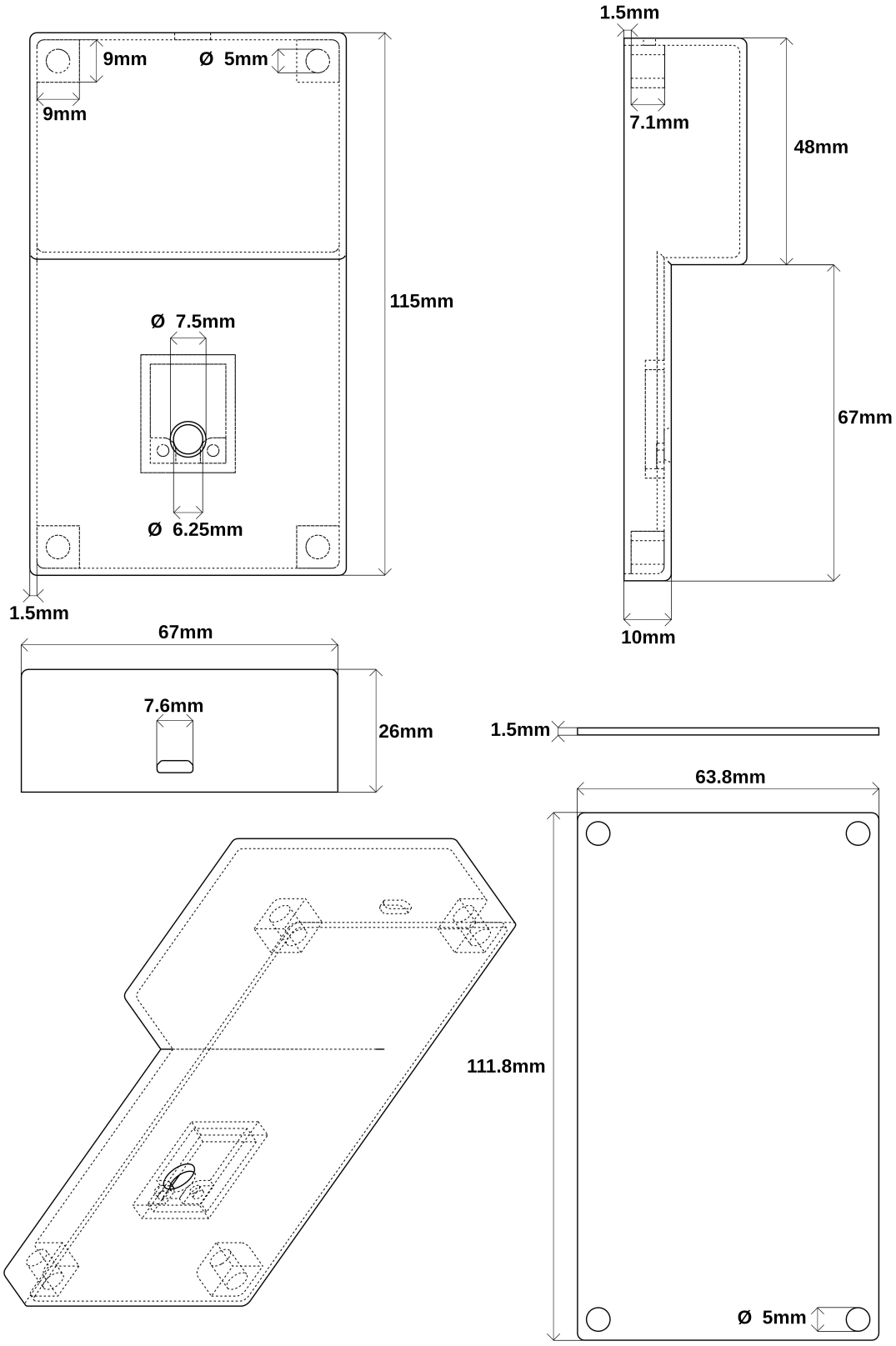


Figura 5.13: Diseño de la tapa del dispositivo (las dos figuras de abajo derecha) y su carcasa (resto)

Capítulo 6

Biblioteca Pulse

El protocolo ha sido programado en 3 capas simplificadas inspiradas en el modelo OSI, además de una cuarta capa, puerto COM, invisible para el receptor que permite comunicarse con el dispositivo:

- Aplicación: Contiene toda la información que se desea enviar.
- Enlace de datos: Esta capa se dedica a la detección y reparación de errores, a la distribución de tramas y al control de flujo. Regula la información que se transmite.
- Puerto COM: Se encarga de controlar los estados del dispositivo y de emitir la información a través del puerto COM para que, posteriormente, el dispositivo la reinterprete y la emita.
- Física: Convierte la información en pulsos magnéticos o viceversa. Está compuesto, por el hardware y por la API de Android y es transparente en el desarrollo de la biblioteca.

La comunicación es simplex, los datos viajan en una única dirección. La información se transmite desde el transmisor al receptor.

Esta biblioteca también permite realizar una comunicación a través de bluetooth. Lo que varía son las capas inferiores de la aplicación 6.2, reestructurando el protocolo de la siguiente manera:

- Aplicación: Contiene toda la información que se desea enviar. Es exactamente igual que la capa de emisión a través de pulsos.

- Bluetooth: Esta capa se dedica a la distribución de tramas y al control de flujo.
- Puerto COM: El tratamiento es distinto que en la capa Pulse. Se encarga de encender/apagar el dispositivo bluetooth y controlar todo tipo de errores de conexión que puedan darse.
- Física: Emisión de ondas electromagnéticas bajo el estándar IEEE 802.15.1

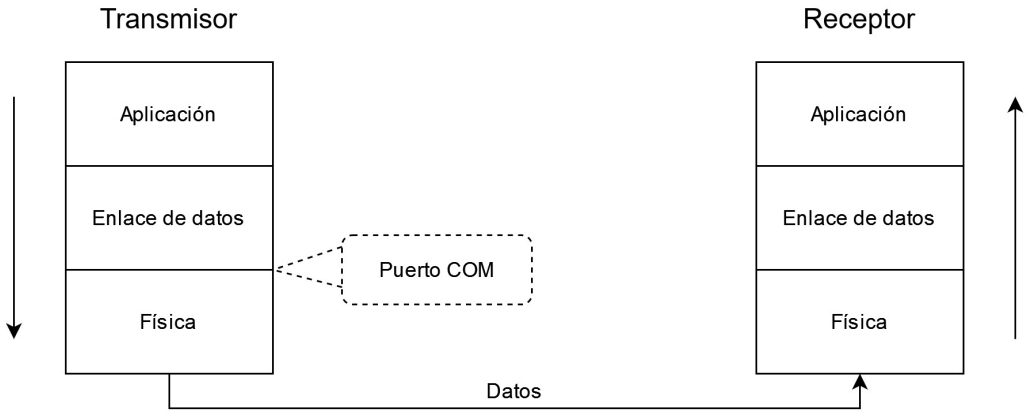


Figura 6.1: Diagrama de capas Pulse

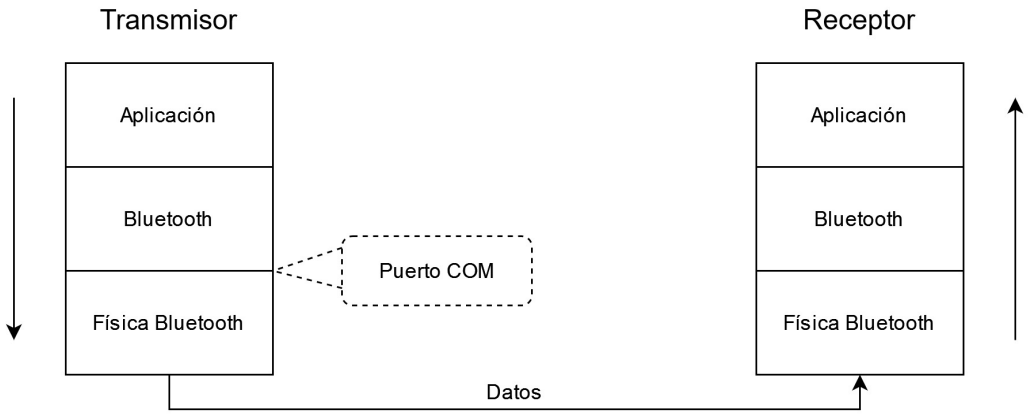


Figura 6.2: Diagrama de capas bluetooth

En los siguientes apartados, tal y como señala el diagrama 6.3, se explicará como se encapsulan y procesan los datos en los distintos dispositivos para los que se ha programado la biblioteca Pulse:

- Biblioteca C++: Envía los datos al dispositivo. Se encarga de encapsular en las distintas capas y enviar la información al dispositivo.
- Dispositivo: Hardware encargado de emitir la información exactamente en cada uno de los pulsos. Este, junto a la biblioteca escrita en C++, conforman el transmisor.

- Biblioteca Java: Recibe los datos en el móvil y los interpreta. El móvil será el receptor.

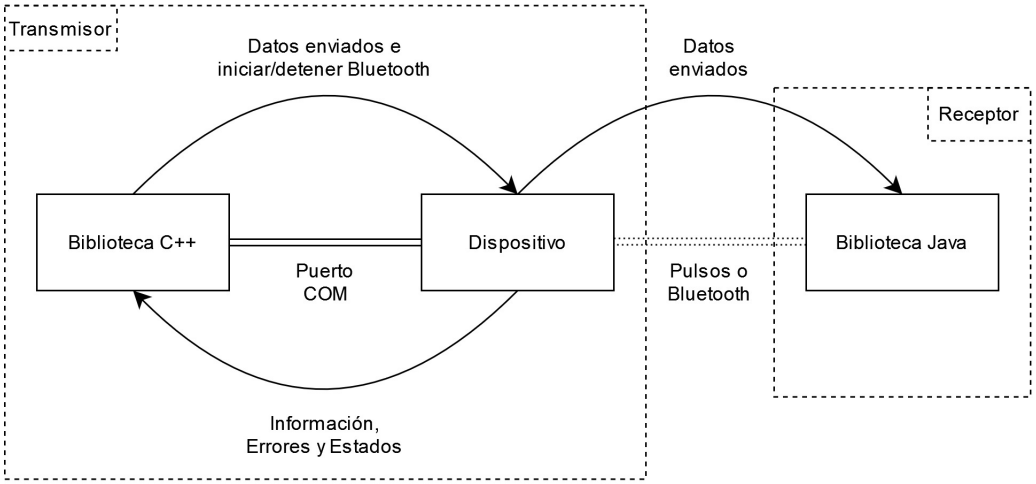


Figura 6.3: Diagrama de comunicación entre dispositivos

6.1. Biblioteca C++

La biblioteca está diseñada para la creación de tareas que se irán encapsulando en las distintas capas que se enviarán. Por otro lado, permite controlar dichos datos, por si se producen errores a la hora de emitir la información. Estos datos se emitirán a través del puerto COM.

La biblioteca permite crear una tarea con la clase Task. Posteriormente la tarea debe de ser añadida con el procedimiento "`addTask`" 6.1.5 de la clase Pulse para ser enviada.

A la hora de seleccionar el tipo de tarea, siempre es necesario indicar si se repetirá (para que se pueda recibir información en distintos teléfonos) y si se transmitirá por bluetooth. Esta biblioteca permite crear distintos tipos de tareas:

- Fichero: Realiza el envío de un fichero binario.
- Mensaje: Envía un mensaje que se mostrará por pantalla.
- Wifi: Permite conectarse a una red wifi.
- Sms: Envía mensajes de texto (sms).
- E-mail: Remite correos electrónicos (e-mail).
- Bitcoin: Realiza pagos bitcoins a un monedero en específico.

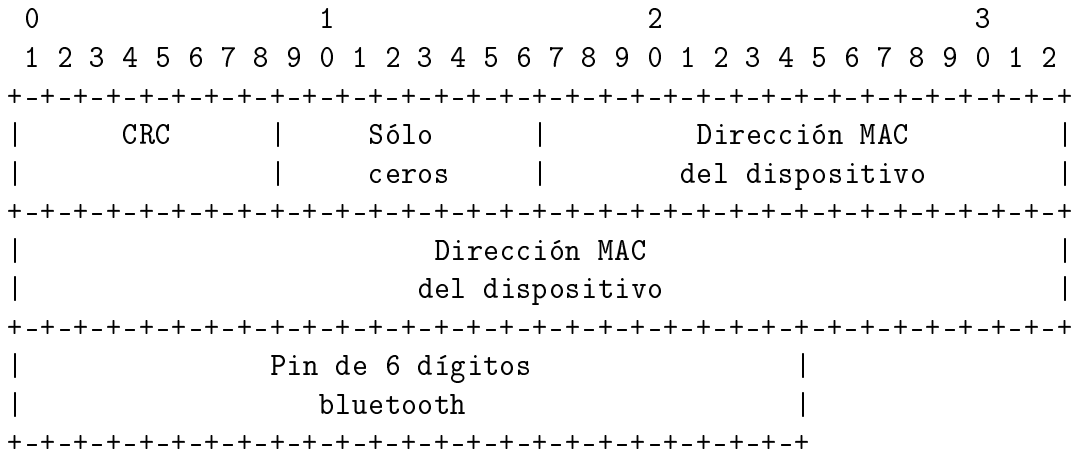
6.1.1. Capa de Aplicación

Esta capa se encarga de estructurar los datos y comprobar la consistencia de los mismos. Se dispone de distintos encabezados por tarea. En un principio se decidió usar encabezados similares a los que puede tener un código QR, pero los encabezados de QR son largos y el protocolo Pulse es un sistema de transmisión con muy bajo bitrate, lo que impedía transmitir la información lo suficientemente rápido y con mayor eficiencia.

Hay 2 tipos de encabezado en la capa Aplicación, dependiendo de si se pretende conectar el dispositivo al móvil por bluetooth o si se desea transmitir la información.

Encabezado bluetooth

Permite enviar toda la información necesaria para realizar una conexión bluetooth con el dispositivo. Posteriormente, cuando el móvil se conecte, se enviará la información a través de bluetooth.

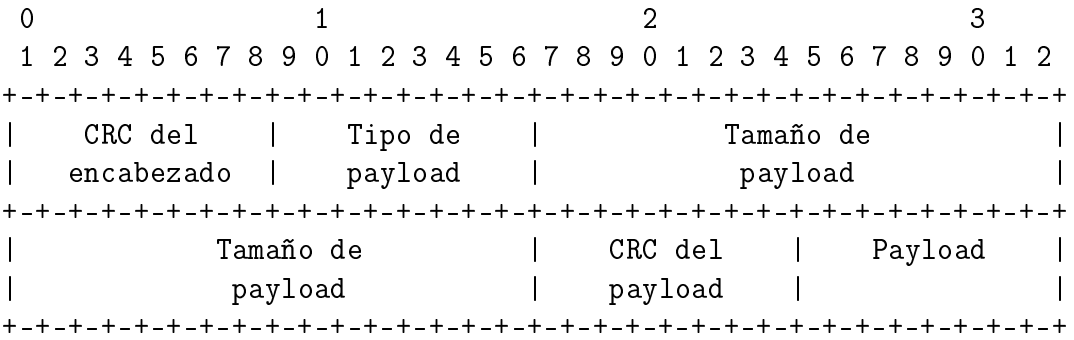


- CRC: Código de Redundancia Cíclica que permitirá comprobar la consistencia de la información enviada en el resto de encabezados. El CRC debe de incluir el resto de los encabezados incluido el byte a ceros. Más adelante se especifican todas las características en el apartado "CRC" 6.4
- Sólo ceros: Permite distinguir el encabezado bluetooth del resto de encabezados.
- Dirección MAC del dispositivo: Debe indicarse la dirección MAC del dispositivo al que se va a conectar el móvil. La dirección MAC es facilitada a la hora de conectarse a través del puerto COM junto a otra información 6.2.1. La dirección MAC se convertirá de hexadecimal a bits.

- Pin de 6 dígitos bluetooth: Secreto numérico de 6 dígitos que será generado aleatoriamente y que permite que la información se transmita cifrada. El número deberá de ser convertido a bits en formato "Integer". No se permite el número 0 ni los números superiores a 999.999, aunque el espacio reservado pueda alcanzar los 16.777.215.

Encabezado genérico

Son los encabezados que se utilizan para enviar los distintos tipos de tareas. Si bien, cada tipo de tarea tiene unos encabezados específicos.



- "CRC" 6.4 del encabezado: Código de Redundancia Cíclica que permitirá comprobar la consistencia de la información enviada en el resto de encabezados.
- Tipo de payload: Byte que permitirá identificar el tipo de información que se enviará en el payload. Más adelante se especifica el número según el payload.
- Tamaño de payload: Es el tamaño total del payload. Permite saber el progreso del envío y descartar los ceros de padding al final del payload.
- "CRC" 6.4 del payload: Código de Redundancia Cíclica que permite comprobar si se ha corrompido algún byte del payload.

  Es importante separar el CRC del encabezado del de payload, porque los bytes del encabezado resultan decisivos a la hora de transmitir la información. Un error en los bytes de encabezado y el protocolo no podría identificar bien la tarea o corromper el payload debido a exceso o escasez a la hora de fijar el tamaño del payload.
- Payload: No tienen un tamaño definido. Es la carga útil. La información útil que se desea enviar.

Como ya se ha dicho, hay distintos payloads dependiendo del tipo de tarea que se quiera enviar.

- Ficheros (Tipo de payload: `0x01`): [Nombre del fichero] + `0x00` + [Fichero]
  - Nombre del fichero: Contiene el nombre del fichero.
  - Fichero: El fichero a enviar.
- Mensajes (Tipo de payload: `0x02`): [Mensaje]
  - Mensaje: Mensaje que se enviará. Sólo se aceptan caracteres legibles que no estén en el rango entre 0x00-0x1F y que no sean 0x7F y 0xA0.
- Wifi (Tipo de payload: `0x03`): [Oculto] + [Tipo de Seguridad] + [BSSID] + `0x00` + [Contraseña]
  - Oculto: Ocupa un único bit. Indica si una red está oculta o no. Dónde:
    - 0: La red está visible.
    - 1: La red está oculta.
  - Tipo de seguridad: Ocupa 7 bits. Indica el tipo de seguridad disponible. Dónde:
    - 0x00: Sin contraseña
    - 0x01: WPA o WPA2
    - 0x02: WPA3
    - 0x03-0x7F: Reservado
  - BSSID: Nombre que identifica la red wifi. Sólo se aceptan caracteres legibles que no estén en el rango entre 0x00-0x1F y que no sean 0x7F y 0xA0.
  - Contraseña: Clave previamente compartida (PSK) que permitirá crear un canal seguro. Sólo se aceptan caracteres ASCII. Si el tipo de seguridad es «Sin contraseña», no se tendrá en cuenta.
- Sms (Tipo de payload: `0x04`): [Teléfono] + `0x00` + [Mensaje]
  - Teléfono: Número de teléfono al que se desea enviar el sms. Sólo se aceptan números, espacios y el símbolo "+". Útil si se desea enviar a números extranjeros.
  - Mensaje: Indica el mensaje que se enviará. Sólo se aceptan símbolos imprimibles de ASCII.
- E-mail (Tipo de payload: `0x05`): [Para] + `0x00` + [CC] + `0x00` + [CCO] + `0x00` + [Asunto] + `0x00` + [Mensaje]
  - Para: Indica para quién se enviará el e-mail. Si se desean incluir varios receptores, es necesario separar los e-mails con comas simples «,». Es obligatorio incluir al menos un e-mail.
  - CC: Copia de Carbón. Es una copia pública, lo que informa al destinatario principal y a todos los que están en la copia. Si se desean incluir varios e-mails, es necesario separarlos con comas simples «,».

  - CCO: Copia de Carbón Oculta. Es una copia privada, lo que no informa ni al destinatario principal, ni a los destinatarios en Copia de Carbón, ni a otros destinatarios en Copia de Carbón Oculta. Si se desean incluir varios e-mails, es necesario separarlos con comas simples «,».
  - Asunto: Indica el asunto que se va a tratar en el mensaje e-mail.
  - Mensaje: Contiene el mensaje que se desea enviar. Sólo se aceptan caracteres legibles que no estén en el rango entre `0x00-0x1F` y que no sean 0x7F y 0xA0. No se aceptan incrustar imágenes, al menos en esta release.

- Bitcoin (Tipo de payload: `0x06`): [Dirección] + `0x00` + [Mensaje] + `0x00` + [Cantidad]
  - Dirección: Wallet a la que se enviará el dinero. Se usa Base58 de entre 14 y 74 caracteres.
  - Mensaje: Mensaje que se desea enviar como concepto. Solo caracteres ASCII imprimibles.
  - Cantidad: Bitcoins que se enviarán. Si se envía números decimales la coma será un «.» (punto).
- Reservado (Tipo de payload: `0x07-0xFF`): Reservado para otras tareas en un futuro.

Al tratarse de cabeceras que se leerán en formato byte, se utilizará el byte 0x00 para separar la información de tamaño dinámico.

6.1.2. Capa de Enlace

Se encarga de añadir los datos necesarios para calibrar el móvil y, posteriormente, enviar las distintas tramas genereadas reconvertidas en código Gray 6.6, fragmentadas por el tamaño de buffer del dispositivo.

Tal y como se ve en el diagrama 6.4, para poder enviar los datos es necesario precalibrar y calibrar los datos, lo que permite detectar al móvil, el número de solenoides que tiene y la cantidad de bits que emitirá cada solenoide del dispositivo. Además se crean tramas que permitan agregar coherencia y orden en los datos.

A la hora de precalibrar y calibrar se usará un pulso doble A. Esto se hace debido a que es información fundamental que, en caso de no ser recibida correctamente, corromperá toda la conexión realizada. Así se consigue reducir el número de errores.

Una vez que se ha precalibrado y calibrado, ya se pueden empezar a emitir los datos. Para enviar se crean tramas que permitirán ordenar los datos de forma coherente con los

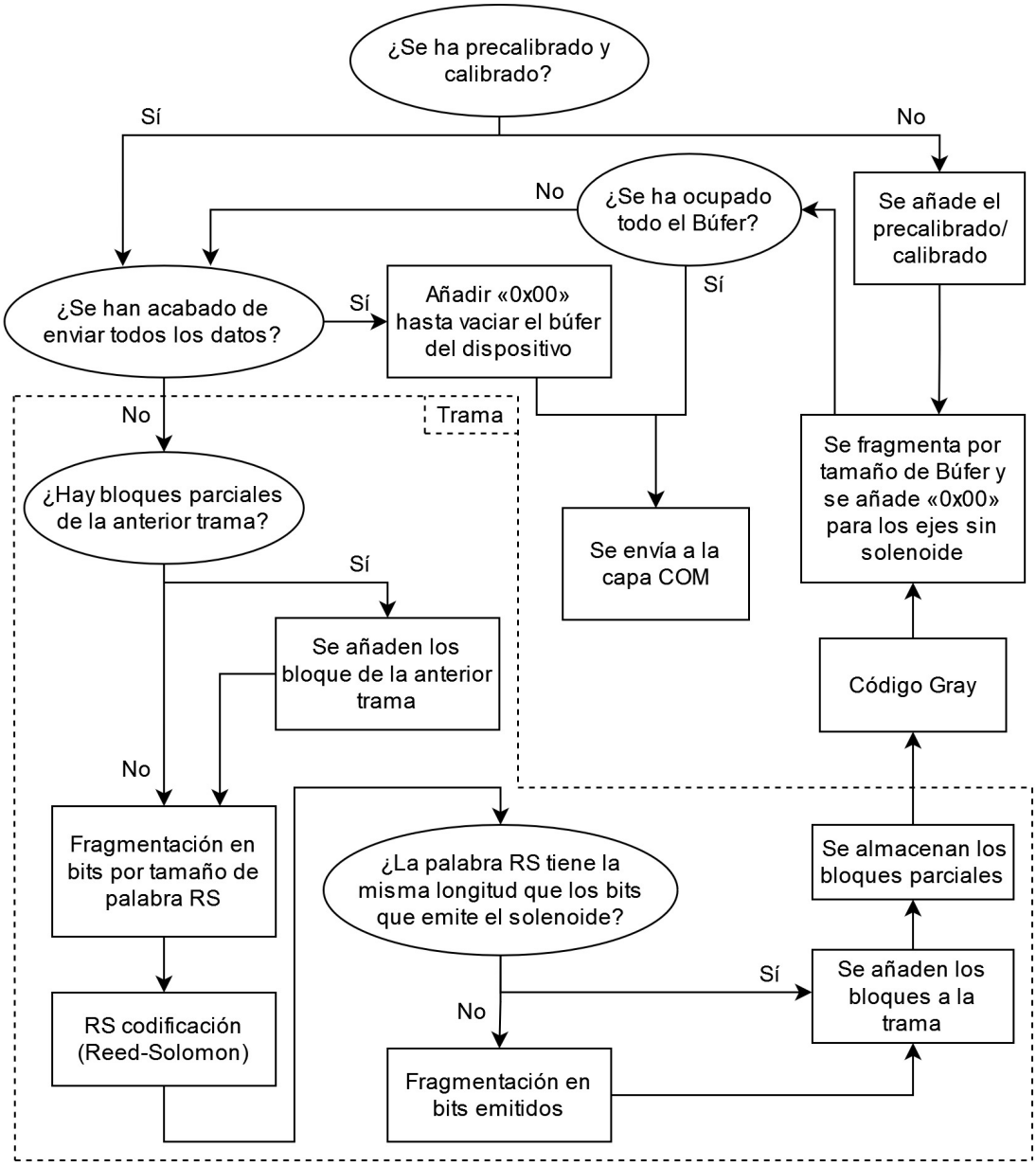


Figura 6.4: Diagrama de decisión de la capa de Enlace en C++

bloques Reed Solomon, ordenar los datos de forma adecuada y, en un futuro, reanudar la tarea en el punto detenido. Posteriormente se convierte los bytes en "Código Gray" 6.6 para reducir el número de errores.

Las tramas se fragmentarán por el tamaño de buffer que ser reciba del puerto COM, como se especifica en "Información emitida por el puerto del dispositivo" 6.2.1, y se añadirá `0x00` la información para aquellos solenoides que no tenga el dispositivo.

Después de acabar con todos los datos se envían `0x00` para vaciar el buffer del dispositivo y asegurarse que todos los datos han sido emitidos correctamente.

Precalibrado

Permite detectar si el móvil está sobre un dispositivo de emisión. Se busca generar una señal que permita detectar fácilmente los dispositivos.

Está formado por 3 pulsos dobles A con el máximo nivel de potencia que emite el solenoide Z y 3 pulsos dobles con la menor emisión posible, intercalando los pulsos altos con los bajos. Se sobrentiende que todos los dispositivos tienen señal magnética en el eje Z, así que, por ello sólo se usa dicho eje.

Para calcular el máximo nivel de potencia se usa $2^n - 1$. Dónde n es el número total de bits que puede emitir un solenoide en un único pulso. El cálculo del mínimo nivel de potencia será siempre 0 para los solenoides que emitan $n = 1$ y para todos los demás será siempre 1. Esto se hace porque el 0 representa no emitir ningún tipo de energía y, tal y como se dijo en el apartado de principios físicos 5.1, se debe de superar el campo magnético terrestre, así que la escala del 0 al $2^n - 1$ no es proporcional, excepto en el único caso de $n = 1$.

Calibrado

Está orientado a indicar el número de solenoides que tiene el dispositivo, la cantidad de bits que emitirá cada solenoide y la intensidad que tienen dichos solenoides. El dispositivo puede tener máximo 3 solenoides (Z,Y,X) con una emisión máxima de 8 bits por solenoide en cada pulso.

Por cada número de bits que puede emitir un solenoide debe de emitir 2 pulsos dobles A, uno con la máxima potencia del solenoide y otro sin emisión. Estos pulsos dobles deben de ser emitidos desde el solenoide que tiene dicho solenoide. El orden a seguir de los distintos solenoides es Z, Y, X.

Para indicar que el proceso de calibrado se ha acabado, se deben de añadir dos pulsos dobles sin emisión al final.

Creación de trama

Las tramas permiten procesar la información de manera organizada. Además de que, en un futuro, podría permitir la reanudación de tareas.

Cada trama estará formada por 50 *pulsos*. Es decir por un periodo de 50 *pulsos* · $50ms = 2500ms = 2,5s$. Las tramas están formadas por bloques Reed-Solomon 6.5, un código cíclico no binario que permite la detección y corrección de errores.

Lo primero es agregar los bloques que quedaban pendientes por terminar de la trama anterior. Posteriormente se irán agregando los nuevos bloques a la trama que entren sin ningún tipo de corte de mayor a menor tamaño. Por último se agregan los bloques que quedarán cortados por la trama de mayor a menor tamaño.

La información, antes de ser enviada, debe de ser convertida a "Código Gray" 6.6.

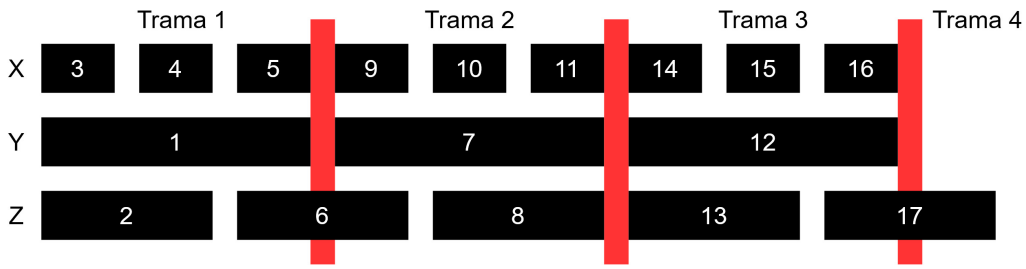


Figura 6.5: Ejemplo de creación de trama. En negro, cada bloque Reed-Solomon.

Detener tareas

Para detener una tarea se debe de enviar una trama especial. Al ser recibida por el dispositivo la tarea se eliminará.

La trama está formada, únicamente, por un bloque "Reed-Solomon" 6.5 en el canal Z, formada por:

- Datos (k): Rellenos de ceros.
- Redundancia ($2t$): Deben de estar rellenos con el máximo número que puede emitir dicho solenoide. Antes de enviar los datos tienen que ser convertidos a "Código Gray" 6.6.

Fragmentación por tamaño de búfer

Una vez creado los distintas tramas la información se debe de fragmentar por el tamaño de búfer de la capaz de enlace. El tamaño viene fijado por la información recibida a través del puerto COM.

6.1.3. Bluetooth

En este caso la capa sólo se encarga de preparar la información con el tamaño de búfer bluetooth fijado por el dispositivo, exactamente igual que la capa de enlace.

6.1.4. Puerto COM

El dispositivo hardware se comunica a través de un puerto COM. Se utilizan distintos códigos que debe de emitir la biblioteca:

- Datos enviados por Solenoide: «S» + [Longitud en bytes] + «,» + [Datos]
  - «S»: Indica que los datos se emitirán a través de los solenoides.
  - Longitud en bytes: Indica la longitud total de los datos en bytes.
  - Datos: Bytes de datos a enviar
- Datos enviados por bluetooth: «B» + [Longitud de bytes] + «,» + [Datos]
  - «B»: Indica que los datos se emitirán a través del bluetooth.
  - Longitud en bytes: Indica la longitud total de los datos en bytes.
  - Datos: Bytes de datos a enviar.
- Inicia el dispositivo bluetooth: «U» + [Contraseña] + «;»
  - «U»: Indica el encendido del dispositivo bluetooth y será visible por otros.
  - Contraseña: Es únicamente numérica sólo permite dígitos del 1 al 999.999. No se permite el 0 (implicaría tener una contraseña vacía). Se recuerda que se interpretará como cadena UTF-8.
- Detiene el dispositivo bluetooth: «D»
  - «D»: Indica el apagado del dispositivo bluetooth. No será visible ni tampoco se podrá conectar.

Al tratarse de información con longitud variable y que se interpretará a través de UTF-8 se utiliza de separador de datos «,» para la información con tamaño dinámico. Excepto si no hay más campos que añadir, entonces se usará «;».

Por otro lado la información con el campo [Datos] se interpretarán como bytes de datos a emitir. Un pulso estará formado por 3 bytes de datos, cada uno de ésos datos se emitirán en X, Y, Z

También se utiliza esta clase para tener cierto control sobre el dispositivo bluetooth y poder coordinar los datos que se envían.

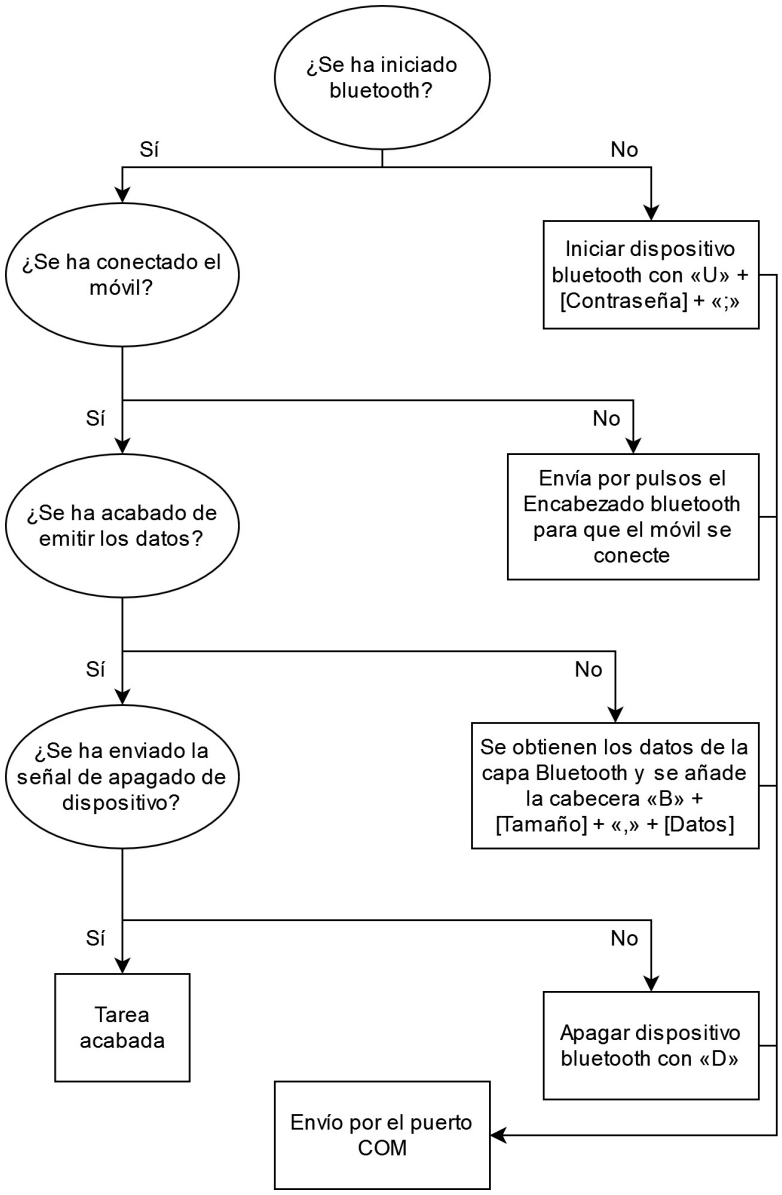


Figura 6.6: Diagrama de decisión de la capa bluetooth

6.1.5. Utilización de la biblioteca

Clase Task

Primero se debe de crear la tarea:

Constructores públicos
Task() Construye el objeto Tarea
Task(QObject *parent) Construye el objeto Tarea con un padre

Tabla 6.1: Constructores de la clase «Task»

Una vez creada la tarea es obligatorio inicializar la tarea con uno los procedimientos set + [Tipo de tarea], sino la tarea no será añadida correctamente.

Métodos públicos	
Retorno	**Método**
void	setFile(QFile *file, bool repeat, bool blue) Convierte la tarea en tipo fichero.
void	setMessage(QString message, bool repeat, bool blue) Convierte la tarea en tipo mensaje.
void	setWifi(QString name, TypeSec sec, bool hidden, QString password, bool repeat, bool blue) Convierte la tarea en tipo wifi.
void	setSms(QString phone, QString message, bool repeat, bool blue) Convierte la tarea en tipo sms.

(Continúa)

Métodos públicos	
Retorno	**Método**
void	setEmail(QString to, QString cc, QString subject, QString bcc, QString Message, bool repeat, bool blue) Convierte la tarea en tipo e-mail.
void	setBitcoin(QString address, double amount, QString message, bool repeat, bool blue) Convierte la tarea en tipo bitcoin.
bool	getRepeat() Devuelve si la tarea se repetirá.
void	setRepeat(bool repeat) Establece si la tarea se repetirá.
QString	getType() Obtiene una cadena del tipo de Tarea en un formato legible por humanos.
QString	static taskToStr(TypeTask type) Convierte cualquier tipo de tarea en un formato legible por humanos.

Tabla 6.2: Procedimientos de la clase «Task»

Permiten dar información acerca del estado en el que se encuentra la tarea actual.

Señales	
Retorno	**Método**
void	progress(const int index, const float progress) Llamado cuando se actualiza el progreso de la tarea

Tabla 6.3: Señales de la clase «Task»

Enumeraciones	
Tipo	**Identificador**
TypeTask	File Tarea que envía un fichero
TypeTask	Message Tarea que envía un mensaje
TypeTask	Wifi Tarea que se conecta a una red wifi
TypeTask	Sms Tarea que envía un mensaje de texto
TypeTask	Email Tarea que envía un e-mail
TypeTask	Bitcoin Tarea que automatiza el envío de bitcoins
StateTask	waitSend La tarea está esperando a ser enviada
StateTask	waitQuitPhone Esperando a que se retire el teléfono y poner otro
StateTask	endTask La tarea ha terminado y ya se ha enviado
TypeSec	OPEN Sin ningún tipo de seguridad wifi
TypeSec	WPA2 Seguridad wifi WPA2

(Continúa)

Enumeraciones	
Tipo	**Identificador**
TypeSec	WPA3 Seguridad wifi WPA3

Tabla 6.4: Enumeraciones de la clase «Task»

Clase Pulse

Se encarga de administrar el dispositivo además de las distintas tareas que hay, permitiendo añadir, borrar o editar tareas.

Métodos públicos	
Retorno	**Método**
Pulse*	getInstance() Obtiene la única instancia de Pulse a través de un puntero.
void	delInstance() Detiene y elimina la única instancia de Pulse
bool	conn(QString con) Se conecta al dispositivo especificado.
void	disconn() Si está conectado se desconecta del dispositivo.
QString	getPort() Obtiene el puerto al que está conectado.
QStringList	listPorts() Obtiene una lista de todos los dispositivos conectados.
void	addTask(Task *task) Añade una tarea al final de la lista.

(Continúa)

Métodos públicos	
Retorno	**Método**
void	removeTask(int index) Elimina la tarea del índice expecificado. Empieza en 0.
void	repeatTask(int index, bool repeat) Cambia el valor de repetición.

Tabla 6.5: Procedimientos de la clase «Pulse» (C++)

Señales	
Retorno	**Método**
void	connectDevice(bool connect) La biblioteca se ha conectado, o desconectado, exitosamente del dispositivo.
void	waiting(bool wait) El dispositivo está esperando a otro teléfono, o si ya se ha puesto el nuevo.
void	error(DeviceException error) El dispositivo está en una situación de error.
void	deleteTask(int task) La tarea ha sido borrada exitosamente de la biblioteca.

Tabla 6.6: Señales de la clase «Pulse» (C++)

Clase de DeviceException

Como se especifica en la tabla de señales 6.3, cuando se de un error se ejecutará el método `error(PulseError error)` que devolverá un objeto `PulseError` definido tal que:

Constructores públicos
DeviceException(PulseError error) Construye el objeto con el tipo de error

Tabla 6.7: Constructores de la clase «DeviceException» (C++)

Métodos Públicos	
Retorno	**Método**
PulseError	getError() Devuelve el tipo de error que se ha producido.
void	raise() Eleva el error a instancias superiores.
DeviceException*	clone() Clona el objeto.

Tabla 6.8: Procedimientos de la clase «DeviceException» (C++)

Enumeraciones	
Tipo	**Identificador**
PulseError	FileNotExist Indica que el fichero al que se apunta para el envío no existe o no se puede leer. Es el único error no relacionado con el dispositivo, sino con el equipo.
PulseError	portCom Error a la hora de emitir o recibir información en el puerto COM. Es probable que el dispositivo se haya desconectado o se hayan eliminado los drivers.

(Continúa)

Enumeraciones	
Tipo	**Identificador**
PulseError	unknown Hay un error en el dispositivo, pero no se sabe el origen.
PulseError	hardware Error en el hardware del dispositivo que le impide funcionar.
PulseError	power No se ha conectado la fuente de alimentación del dispositivo.
PulseError	proximity Se ha retirado el móvil del dispositivo.
PulseError	move Se ha movido el dispositivo.
PulseError	connectionBlue El dispositivo no se puede conectar por bluetooth correctamente.
PulseError	transferBlue El dispositivo ha fallado a la hora de transferir información a través del bluetooth.
PulseError	unknownBlue El dispositivo tiene un error en el bluetooth desconocido.

Tabla 6.9: Enumeraciones de la clase «DeviceException» (C++)

6.2. Dispositivo

Se encarga de recibir/emitir datos COM, procesarlos y emitirlos a través del solenoide o de un dispositivo bluetooth.

6.2.1. Información emitida por el puerto COM del dispositivo

El dispositivo es capaz de enviar información al ordenador para conocer el estado del dispositivo. Permite enviar distintos tipos de información:

- Envía información del dispositivo: «I» + [Bits X] + «,» + [Bits Y] + «,» + [Bits Z] + «,» + [Bytes Pulsos] + «,» + [MAC bluetooth] + «,» + [Bytes bluetooth]
  - «I»: Indica que se va a recibir información acerca del dispositivo.
  - Bits X, Y, Z: Son la cantidad de bits que puede emitir cada solenoide. La biblioteca está diseñada para soportar una emisión de 8 bits por solenoide en cada pulso.
  - Bytes Pulsos, bluetooth: Son la cantidad de bytes que el dispositivo se compromete a almacenar para ser procesados en búfer tras la recepción de un «R».
  - MAC bluetooth: Dirección del dispositivo al que se va a conectar el móvil.
- Preparado para enviar: «R»
  - «R»: Indica que todas las acciones anteriores ya se han realizado y el dispositivo está preparado para empezar a recibir datos.
- Conectado el dispositivo bluetooth: «C»
  - «C»: El dispositivo se ha conectado por bluetooth al teléfono.
- Errores: «E» + [Número de error]
  - «E»: Indica que se ha producido un error.
  - Número de error:
    1. Iniciar el sensor de proximidad: Ha habido un error al inicializar el sensor de proximidad.
    2. Iniciar el sensor de movimiento: Ha habido un error al inicializar el sensor de movimiento.
    3. Iniciar el chipset bluetooth: Ha habido un error al inicializar el chipset bluetooth.

4. Fuente de alimentación: El dispositivo necesita una fuente de alimentación y no se ha conectado al dispositivo, así que no puede funcionar.
5. Proximidad al sensor: El sensor detecta que no hay un teléfono sobre el dispositivo.
6. Mover al dispositivo: El dispositivo se ha movido mientras se estaban emitiendo datos Pulse.
7. Sin conexión: No hay conexión bluetooth.
8. Transferencia bluetooth: La transferencia se ha cortado porque se ha perdido comunicación con el teléfono.
9. Desconocido bluetooth: Se ha producido un error por determinar a la hora de transmitir información por bluetooth.

6.2.2. Información recibida por el puerto COM del dispositivo

Cuando se recibe información, tanto si se emite a través de bluetooth o pulsos, debe de ser almacenada en el dispositivo para poder ser enviada justo a tiempo. Para ello se utiliza un array circular con un puntero indicando la posición donde se debe de continuar leyendo la información y un entero que indica la longitud pendiente en bytes de la información, es decir, el número de bytes que quedan por enviar.

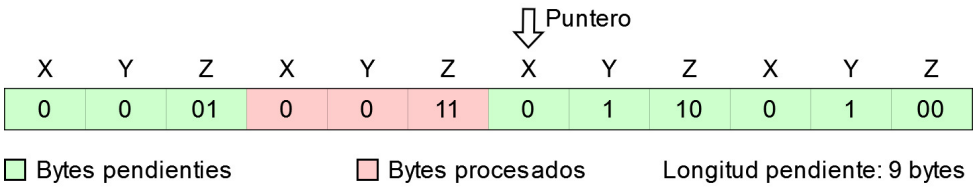


Figura 6.7: Almacenamiento de datos en buffer

Si se produce un error y hay que vaciar el array, el entero que indica la longitud pendiente pasará a 0. El puntero puede continuar en la misma posición debido a que se irá sobrescribiendo la información según se van recibiendo más datos nuevos.

6.2.3. Errores

El dispositivo tiene distintos estados internos de error según los códigos de error transmitidos por el puerto COM:

- 0: El dispositivo no tiene ningún tipo de error.

- 1 al 3: Irreparables. El hardware no funciona correctamente.
- 4: No permite realizar ningún tipo de acción a no ser que el dispositivo tenga una fuente de alimentación conectada.
- 5 al 9: Podrá volver a funcionar correctamente en cuanto se subsanen los errores. Sigue permitiéndose estar conectado al dispositivo. Toda la información almacenada será borrada.

Dependiendo del estado de error que tenga el dispositivo se podrá o no recibir información. El significado de cada error (excepto el 0 que implica la ausencia del error), está explicado en "Información emitida por el puerto COM" 6.2.1.

6.2.4. Emisión de pulsos

La tecnología utilizada para modular la información de forma analógica será ASK (Amplitude-shift keying) A. ASK representa la información digital dependiendo de la amplitud de onda analógica que se emita. En otras palabras, dependiendo de la intensidad del campo magnético se interpretará que se está emitiendo unos bits u otros.

La división proporcional del campo magnético es lo que se llama "nivel". Cada solenoide contará con 2^x niveles, dónde x representa el número de bits que se emitirán por pulso.

El dispositivo se encargará de leer la información del búfer tal y como se explica en "Información recibida por el puerto COM" 6.2.2 y emitirá los bits cada 50 milisegundos por los distintos pines. El DAC 5.2.2 convertirá la información en analógica y, posteriormente, se trasladará al solenoide. Se puede leer una especificación más precisa del funcionamiento en el apartado de "Hardware" 5.

En la figura 6.8 se pueden apreciar dos solenoides emitiendo información. Un solenoide emite 2 bits así que tiene 2 niveles y el otro emite 2 bits, así que tiene 4 niveles.

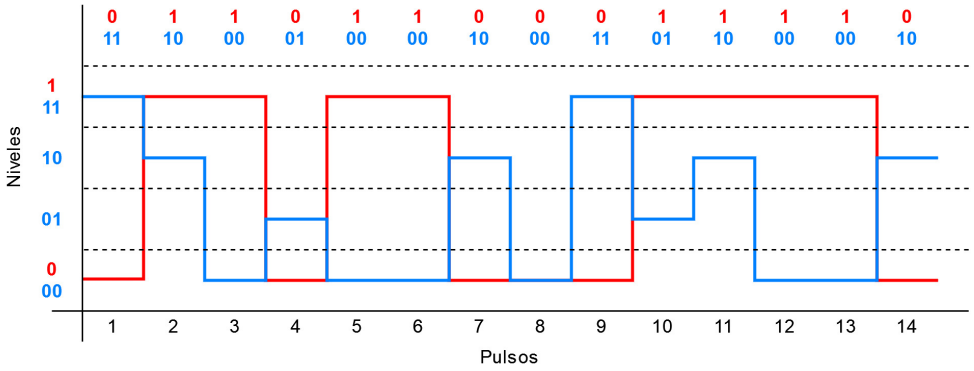


Figura 6.8: Emisión de datos con dos solenoides

6.2.5. Bluetooth BLE

El dispositivo tiene conectividad basada en bluetooth BLE, tal y como se dice en "Arduino" 5.2.1. Bluetooth BLE tiene un alto nivel de complejidad pero el objetivo de este documento no es explicar el trasfondo en el que se desenvuelve BLE. Por ello se explicarán sólo las capas relacionadas con el Host. Son capas de alto nivel, es decir, son las únicas capas a las que se pueden utilizar a la hora de programar y desarrollar.

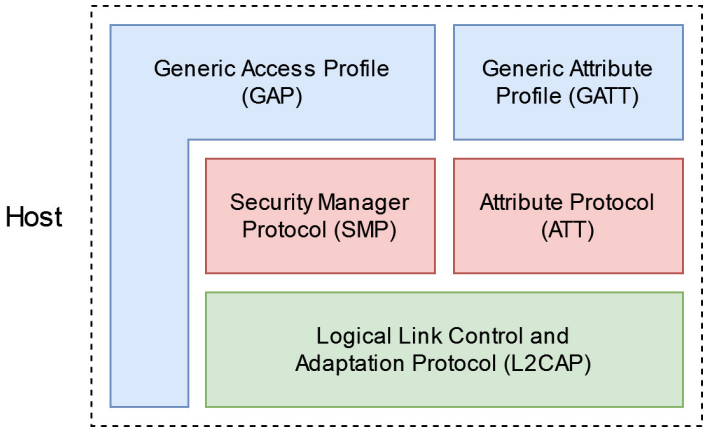


Figura 6.9: Capas Host de bluetooth BLE

Logical Link Control and Adapatation Protocol (L2CAP): Proporciona un canal de datos. Segmenta y reensambla los datos que se envían y además multiplexa 3 canales: 2 dedicados a capas superiores (como ATT y SMP) y 1 dedicado a la propia señalización del L2CAP.

Security Manager Protocol (SMP): Indica el comportamiento para gestionar el emparejamiento, la autenticación y el cifrado entre los distintos dispositivos.

Generic Access Profile (GAP): Explicita los procedimientos relacionados con el descubrimiento y administación de conexión con dispositivos bluetooth.

Attribute Protocol (ATT): Define una arquitectura cliente-servidor por encima de L2CAP. El protocolo permitirá crear un servidor GATT que exponga atributos y valores a su homólogo cliente GATT.

Generic Attribute Profile (GATT): Concreta un marco de servicios que utiliza la capa del protocolo ATT. Define los procedimientos de:

- Servicios: Agrupa las distintas características. Un servicio es utilizado para definir una función en un Perfil. El servicio se identifica con un UUID único universal.

- Características: Contiene un valor y un descriptor que describe el valor de la característica. Todas las características deben de formar parte de un servicio. Al igual que los servicios, tiene un identificador UUID único universal.

- Propiedades: Cada característica tiene un tipo de propiedad que se puede combinar (Lectura, Escritura sin respuesta, Escritura, Notificación (desuso), Indicación), y un nivel de seguridad por cada propiedad (Ninguna, Sin autenticar, Autenticado).

El dispositivo estará configurado como un servidor GATT por lo que las características de los servicios obligatorios podrán ser editables. SMP estará configurado para que el dispositivo posea una clave precompartida que se utilizará para cifrar la información entre dispositivo y móvil. Los servicios GATT implementados serán:

- Servicio de acceso genérico (obligatorio): Proporciona información del dispositivo. El nivel de seguridad para todas las características será «Ninguna». El servicio tiene varias características:

  - Nombre del dispositivo (Propiedad: Lectura): Es el nombre que verán los otros dispositivos. Siempre será «Impulse».
  - Apariencia (Propiedad: Lectura): Indica el aspecto con el que otros verán el dispositivo. Siempre será «Desconocido».
  - Parámetros de conexión periféricos preferidos (Propiedad: Lectura): Incluye información acerca del propio dispositivo. Mínimo y máximo de conexiones en un intervalo, latencia de la conexión esclavo y el tiempo de espera de supervisión de conexión.

- Servicio de atributo genérico (obligatorio): Contiene información acerca de los distintos servicios. El nivel de seguridad será «Ninguna». El servicio tiene una única característica:

  - Servicio cambiado (Propiedad: Indicación): Indica al cliente si algún servicio ha sufrido algún cambio. Nivel de seguridad será «Ninguna».

- Información del dispositivo: Contiene información acerca del dispositivo. El servicio tiene una única característica:

  - Salida Digital (Propiedad: Lectura, Escritura, Indicación): Emite la información que se desea enviar. Cuando se modifique la característica se notificará al dispositivo móvil de que se ha alterado el valor de la característica. Cuando se haya recibido toda la información el móvil escribirá el byte «0x00» que indicará que toda la información se ha recibido correctamente y el dispositivo bluetooth se puede apagar. El nivel de seguridad «Autenticado».

6.3. Biblioteca Java

La biblioteca está diseñada para recepción de tareas que irán pasando por las distintas capas, procesando cada capa sucesivamente es decir, desencapsulando, y, posteriormente, extrayendo la información relevante. Todo con el objetivo de entregar los datos finales al sistema que utilice esta biblioteca.

La app recibirá los datos a través los escuchadores:

- Modo normal: Dirigido a recibir las distintas tareas.

- Modo desarrollador: Orientado a procesar información estadística, información útil para el desarrollo del protocolo y comprobar la compatibilidad del dispositivo.

6.3.1. Capa de Enlace

Esta capa se encargará de convertir la información analógica (campos magnéticos recibidos a través de la API Android) a digital, reconvertir la información a "Código Gray" 6.6 y procesar las distintas tramas.

Tal y como se ve en el diagrama 6.10, para poder recibir datos es necesario que el móvil pase por un proceso de precalibrado y calibrado que permitirá detectar que el móvil está sobre un dispositivo, sincronizar los pulsos, conocer los solenoides que tiene el dispositivo y cuántos bits se emitirán por pulso en cada solenoide.

Una vez que el móvil ya está precalibrado y calibrado, ya se podrá empezar a recibir datos que previamente habrá que reconvertir del "Código Gray" 6.6. Aunque sólo cuando se haya recibido una trama completa, se comenzará a procesar.

La trama, para ser procesada, deberá de descodificar los bloques de Reed-Solomon 6.5 y luego recomponer la información. Si la descodificación es correcta se pasará a la siguiente capa: Aplicación.

Además el proceso se detendrá si se detecta movimiento del móvil, lanzando un mensaje de error.

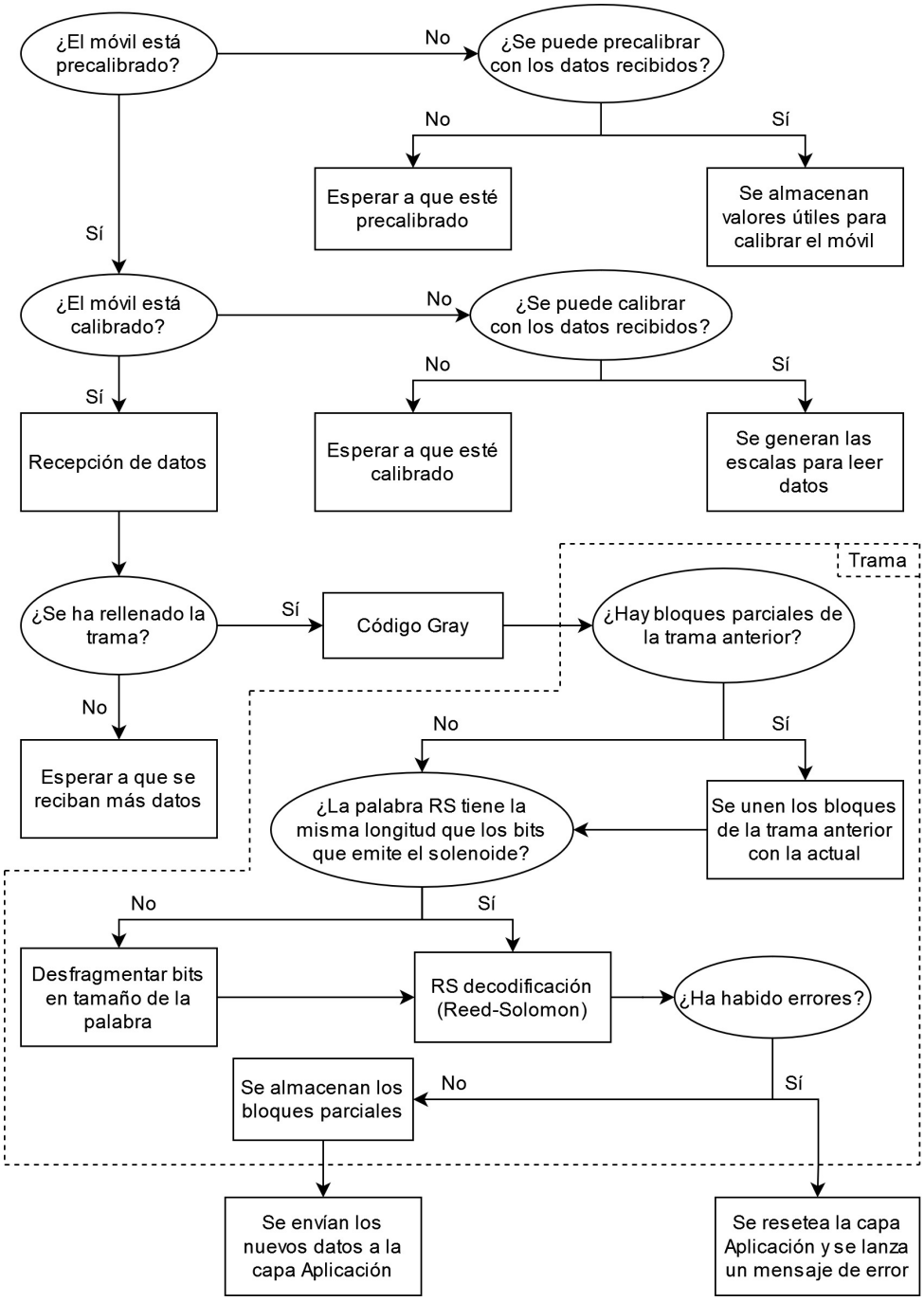


Figura 6.10: Diagrama de decisión de la capa de Enlace en Java

Precalibrado

El proceso de precalibrado permite identificar si el móvil está sobre un dispositivo. Para ello se van leyendo los datos a través de una ventana de 15 pulsos buscando los máximos y

mínimos para el eje de coordenadas Z. Se sobrentiende que todos los dispositivos emitirán en el eje de coordenadas Z ya que representa la parte trasera del móvil.

Se utilizan 15 pulsos de tamaño de ventana debido a que son 3 pulsos dobles a la máxima potencia, con sus 4 valles, es decir 7 pulsos dobles, lo que implica 14 pulsos. Además se añade un pulso extra para tener una mejor muestra de calibrado para descartar el campo magnético terrestre.

Cada vez que se añada un nuevo dato a la ventana, el sistema se encargará de eliminar todas las muestras anteriores que superen los 15 pulsos (750 milisegundos).

Para calcular los máximos y mínimos se dan los siguientes pasos:

1. Se van leyendo los valores y se comparan con la mediana de los valores anteriores, verificando que el valor está dentro de un rango.

2. Si el valor se sale del rango, si es mayor, la mediana será añadida como un mínimo y sino como un máximo. A partir de ése momento, hasta que no deje de crecer o decrecer en los próximos valores, se comparará sin un rango.

Se utiliza una mediana debido a que la distribución de valores es sensible a los sesgos, por eso no se usa la media.

Los rangos son necesarios debido a que no se trata de una función perfecta, pueden darse máximos y mínimos locales que en realidad no representen que el dispositivo haya emitido un pulso, sino que el sensor ha leído ruido del entorno. Sin embargo es útil que cuando un punto se salga de un rango, se elimine el sistema de rangos hasta que el crecimiento/decrecimiento varíe, aumentando así la sensibilidad y permitiendo encontrar máximos y mínimos más precisos.

Una vez que ya se han encontrado los máximos y mínimos se tiene que verificar que o hay 4 máximos y 3 mínimos (negativo) o hay 3 máximos y 4 mínimos (positivo), además de que los máximos y los mínimos son equivalentes entres si, dentro de un mismo rango.

Se debe de detectar si son máximos o mínimos debido a que el dispositivo puede emitir pulsos con valor positivo o negativo, tal y como muestra la figura 6.11

Así se sabe, además de que el móvil está sobre el dispositivo, si se emitirá en positivo o negativo y el tiempo de sincronización.

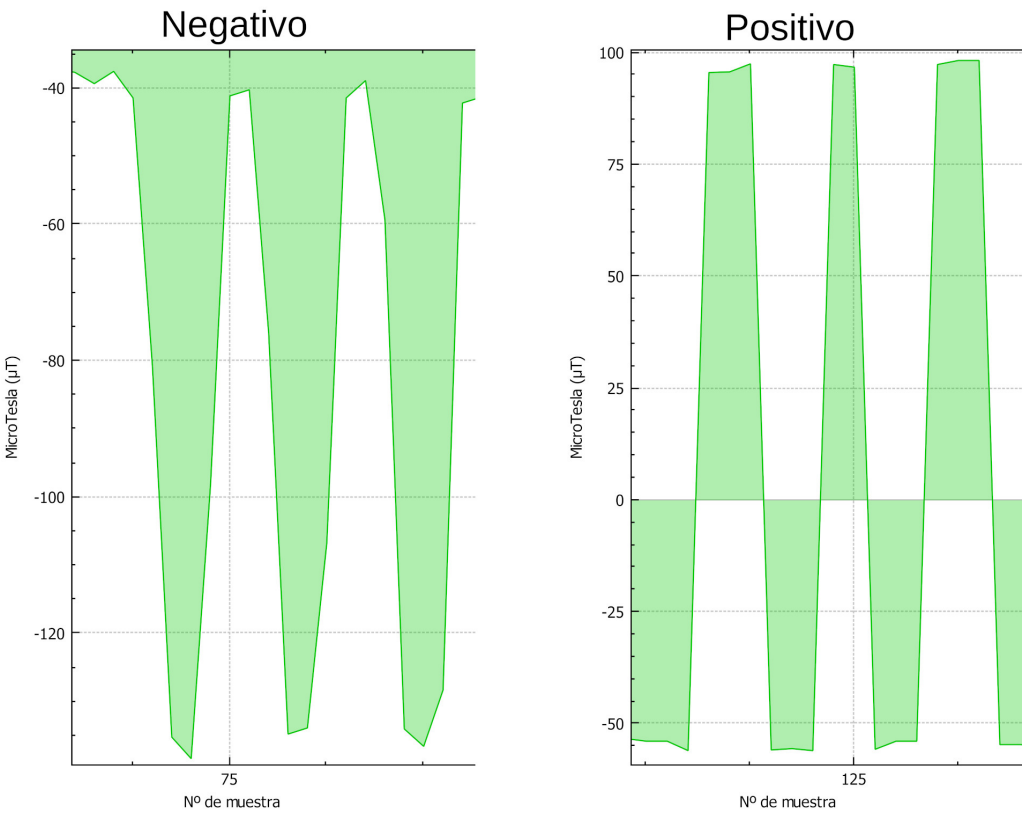


Figura 6.11: Diferencias entre lecturas negativas y positivas

Calibrado

El calibrado permitirá recibir información acerca de la cantidad de solenoides que tiene y sus niveles. Para detectar el tamaño de cada solenoide se buscará que el dispositivo no reciba información durante 2 pulsos dobles.

El móvil almacenará toda la información hasta que se encuentre un periodo de 2 pulsos largos en todos los solenoides sin emisión de información.

En esa información se tendrán que buscar los máximos y mínimos en el orden: Z, Y, X. Cada pico de información indica la potencia en micro-Teslas a la que se emitirá la información y cuantos bits podrá emitir el solenoide.

En la siguiente imagen 6.12 se puede apreciar los campos magnéticos en el solenoide Z con los distintos procesos por los que se pasa hasta que se comienzan a recibir las distintas tramas de datos.

Con los picos de información y su cantidad se puede fragmentar los distintos niveles a

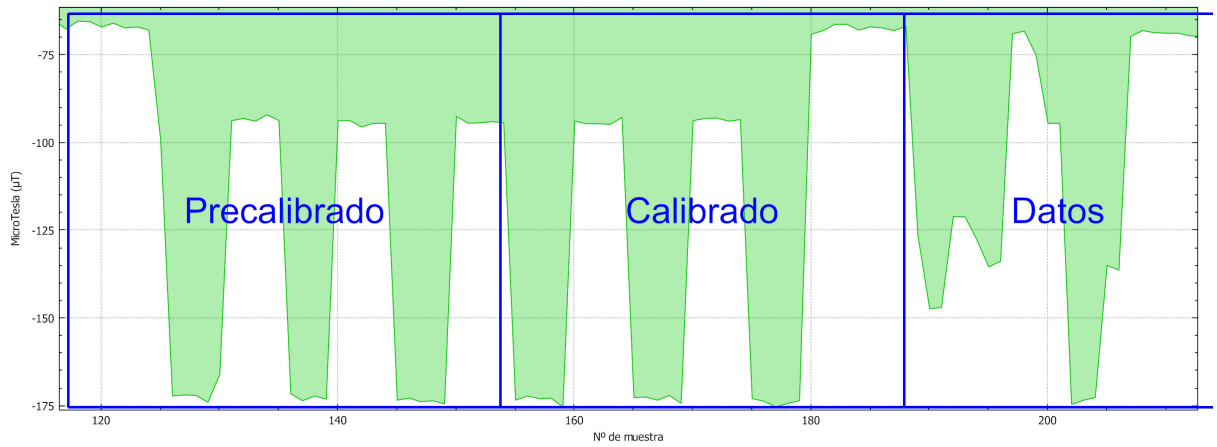


Figura 6.12: Precalibrado, Calibrado y Datos

los que se recibirán los datos de los distintos solenoides. Dependiendo de la cantidad de bits emitidos se contabilizará de distinta manera:

- 1 bit: Los información recibida indicará el mínimo (0) y el máximo (1) por lo que se debe de dividir en 2 la diferencia de Teslas, dónde, si está por debajo de ese valor representa 0 y si está por encima 1.

- Más de 1 bit: La información recibida indicará la potencia mínima (1) y el máximo posible ($2^m - 1$). Esto se hace debido a que la distancia entre 0 (el solenoide está apagado) y 1 no es proporcional, pero la distancia entre 1 y $2^m - 1$, sí. Así que se fragmenta el rango entre 1 y $2^m - 1$ en $2^m - 2$ y se calcula el tamaño proporcional para los valores de 0.

Con estos parámetros ya se podría recibir información, calculando los bits recibidos en cada pulso.

Recepción de datos Pulse

La información es interpretada según los rangos recibidos en la frase de calibrado, siendo el rango superior e inferior abiertos. El campo magnético que se utilizará para leer información será el que esté más cerca del centro del pulso.

Se utiliza el pulso más central, debido a que es el que más probablemente tenga el nivel correcto de Teslas y será el más fiel a lo que se desea transmitir. Además de que los bytes deberán de ser reconvertidos del "Código Gray" 6.6.

Interpretación de tramas

Se esperará hasta recibir toda la longitud de trama (50 pulsos) y, posteriormente, procesar toda la información en el mismo orden que se generó, descifrando con Reed-Solomon 6.5 los bloques más grandes, hasta los más pequeños y almacenando todos los bloques incompletos que serán recompuestos cuando ser reciba la siguiente trama.

Si se detecta una trama especial la interpretación deberá de parar y la tarea debe de ser eliminada. Se notificará que la tarea ha sido borrada al usuario y se reseteará la capa de Aplicación.

Puede darse el caso de que, a la hora de descifrar información con Reed-Solomon 6.5, no se pueda recuperar correctamente la información. En ése caso se tendrá que informar a la capa de Aplicación del error, además del usuario a través del escuchador de que ha habido un error grave. Sucederá lo mismo si el móvil es movido.

6.3.2. Bluetooth BLE

Consta de un proceso de varios pasos:

1. El dispositivo emitirá de forma repetitiva a través de pulsos magnéticos el encabezado de bluetooth 6.1.1 de la capa Aplicación. Dicho encabezado contiene la información necesaria para conectarse al dispositivo a través del bluetooth.
2. El móvil se conectará al bluetooth del dispositivo con las credenciales que se han recibido y se parará de enviar la información a través del protocolo Pulse.
3. El dispositivo enviará al móvil una lista de todos los servicios y de las distintas características de cada servicio.
4. El móvil deberá de habilitar las notificaciones a la característica "Salida Digital" del servicio "Información del dispositivo".
5. El dispositivo comenzará a enviar información al móvil.

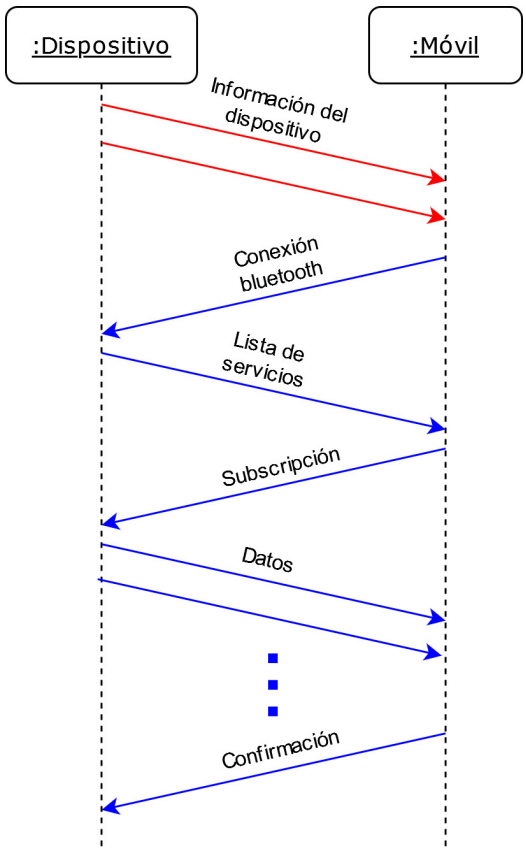


Figura 6.13: Conexión bluetooth

6. Cuando el móvil haya recibido toda la información, el móvil notificará al dispositivo que se ha recibido y se apagará el bluetooth.

En la anterior figura 6.13 se aprecia la secuencia de conexión a través de pulsos (rojo) y la conexión bluetooth (azul) para recibir datos en el móvil. Si se tiene alguna duda sobre el funcionamiento de la API bluetooth BLE, se recomienda consultar el apartado "Bluetooth BLE" 6.2.5.

6.3.3. Capa de aplicación

La información se irá recibiendo de la capa anterior, por lo que no se dispondrá de toda la información desde el principio. Se recuerda al lector que puede consultar la estructura del encabezado en el apartado de la biblioteca C++ en "Capa de Aplicación" 6.1.1.

Procesado de encabezado

Se irá añadiendo la información de los encabezados a la nueva tarea. Cuando se reciban todos los datos del encabezado se comprobará CRC-8 6.4. En caso de que no sea correcto la tarea se detendrá y se notificará a la capa anterior además de al usuario de que la información recibida es incorrecta.

Procesado del payload

Al igual que el encabezado se irán recibiendo los datos y almacenando en un array de strings. La longitud del array y su contenido dependerá del tipo de tarea que se envíe. El orden y la información que contendrá el array será, el siguiente dependiendo de la tarea:

- Fichero: 1.Nombre del fichero.
- Mensaje: 1.Mensaje.
- Wifi: 1.Red Oculta, 2.Tipo de seguridad, 3.Nombre de la red (BSSID), 4.Contraseña[1].
- Sms: 1.Número de teléfono, 2.Mensaje de texto.

[1]Este campo no es obligatorio lo que implica que el String podrá estar vacío

- E-mail: 1.Lista[2] de remitentes, 2.Lista[2] de remitentes en Copia Carbón[1], 3.Lista[2] de remitentes en Copia Carbón Oculta[1], 4.Asunto[1], 5.Mensaje.
- Bitcoin: 1.Dirección (Wallet), 2.Mensaje[1], 3.Cantidad.

Excepto la tarea "ficheros" que irá emitiendo por el escuchador `fileWrite(byte raw)` 6.3.4 los bytes de información que contendrá el fichero y que, sólo al final de la tarea, se emitirá como el resto, el argumento incluirá el nombre del fichero.

Conexión bluetooth

Cuando la capa de aplicación detecta que es un encabezado bluetooth y se ha verificado correctamente a través de CRC-8 6.4 toda la información, la capa de aplicación deberá de pedir a la clase principal que detenga la recepción de pulsos magnéticos y que se inicialice la capa bluetooth 6.3.2 con la información de encabezados que se recibió para la conexión.

La clase principal detendrá la capa de Enlace e iniciará la capa de bluetooth 6.3.2 que se encargará de conectarse al dispositivo y continuar con la comunicación.

6.3.4. Utilización de la biblioteca

Clases Listeners

Lo único necesario para iniciar la biblioteca Java es el contexto de la aplicación y un escuchador encargado de recibir información. Hay dos tipos de escuchadores: escuchador genérico (PulseListener) y desarrollador (PulseListenerDebug). Ambos comparten los siguientes procedimientos:

Escuchadores compartidos	
Retorno	**Método**
void	error(DeviceException exception) La biblioteca está en estado de error.

(Continúa)

[2]Lista separada por comas (,)

Escuchadores compartidos	
Retorno	**Método**
void	state(Pulse.StateDev state) Indica el estado del móvil.
void	progress(float progress) Indica el progreso de la tarea que se está recibiendo.

Tabla 6.10: Escuchadores compartidos

Sin embargo, ambos tienen escuchadores exclusivos:

Escuchadores genérico	
Retorno	**Método**
void	information(Pulse.TypeTask type, String [ ] args) Comunica toda la información que ha recibido de una tarea.
void	fileWrite(byte raw) Comunica toda la información que se deberá de escribir en un fichero.

Tabla 6.11: Escuchador genérico

Escuchadores desarrollador	
Retorno	**Método**
void	dataMagnetic(float [ ] data, long time) Comunica los campos magnéticos que se han recibido a través del sensor (en orden X, Y, Z) y el milisegundo en el que fue recibido.

Tabla 6.12: Escuchador desarrollador

Clase Pulse

Una vez que se ha generado ese escuchador se puede iniciar la biblioteca Java para que se puedan ir recibiendo las distintas tareas.

Métodos públicos	
Retorno	**Método**
void	start(Context context, Listeners listener) throws DeviceException Se inicializa la biblioteca para que comience a recibir información.
void	stop() Detiene la biblioteca. Si hay algún tipo de tarea pendiente será eliminada.
PulseStats	getStats() Obtiene el Estado en el que se encuentra la comunicación.
void	notStorage() Indica a la biblioteca que no se puede almacenar información en el disco duro.

Tabla 6.13: Procedimientos de la clase «Pulse» (Java)

Enumeraciones	
Tipo	**Identificador**
TypeTask	File Tarea que envía un fichero
TypeTask	Message Tarea que envía un mensaje
TypeTask	Wifi Tarea que se conecta a una red wifi

(Continúa)

Enumeraciones	
Tipo	**Identificador**
TypeTask	Sms Tarea que envía un mensaje de texto
TypeTask	Email Tarea que envía un e-mail
TypeTask	Bitcoin Tarea que automatiza el envío de bitcoins
TypeSec	OPEN Sin ningún tipo de seguridad wifi
TypeSec	WPA2 Seguridad wifi WPA2
TypeSec	WPA3 Seguridad wifi WPA3
StateDev	error El estado del móvil es de error
StateDev	preparePulse Se está esperando a recibir información a través de pulsos magnéticos
StateDev	prepareBlue Se está esperando a recibir información a través de bluetooth
StateDev	receivePulse Se está recibiendo información a través de pulsos magnéticos
StateDev	receiveBlue Se está recibiendo información a través de bluetooth

Tabla 6.14: Enumeraciones de la clase «Pulse» (Java)

Clase DeviceException

Tanto el escuchador a través de error(DeviceException exception) como la clase Pulse con start() emiten errores que devolverán un objeto DeviceException definido tal que:

Constructores públicos
DeviceException(DeviceError error) Construye el objeto con el tipo de error.
DeviceException(DeviceError error, String data) Construye el objeto con el tipo de error y el mensaje que se desee

Tabla 6.15: Constructores de la clase «DeviceException» (Java)

Métodos Públicos	
Retorno	**Método**
DeviceError	getPulseError() Devuelve el tipo de error que se ha producido.
PulseError	toString() Convierte el error en una cadena legible.

Tabla 6.16: Procedimientos de la clase «DeviceException» (Java)

Métodos Públicos	
Retorno	**Método**
PulseError	SensorNotFound El móvil no tiene o no se encuentra un magnetómetro.
PulseError	FailedPulse No se ha recibido la información correctamente.

(Continúa)

Métodos Públicos	
Retorno	**Método**
PulseError	IncorrectCRC La información recibida no es correcta.
PulseError	DeviceOff El bluetooth está apagado.
PulseError	FailedBluetooth La conexión bluetooth se ha interrumpido.
PulseError	MoveDevice El dispositivo se ha movido.
PulseError	DeleteTask La tarea ha sido borrada desde el ordenador.
PulseError	WithoutStorage No hay almacenamiento disponible.

Tabla 6.17: Enumeraciones de la clase «DeviceException» (Java)

Clase PulseStats

Además, si se ha iniciado el escuchador en modo desarrollador, se puede acceder a información estadística del dispositivo a través del método `getStats()`. Es información que resulta valiosa para depurar el sistema. La clase estaría definida tal que:

Métodos Públicos	
Retorno	**Método**
double	getAvg() Media en milisegundos entre muestras.

(Continúa)

Métodos Públicos	
Retorno	**Método**
long	getMax() Máximo tiempo en milisegundos entre dos muestras consecutivas.
long	getMin() Mínimo tiempo en milisegundos entre dos muestras consecutivas.
long	getInit() Número de muestras recogidas.
long	getStartUnix() Tiempo en milisegundos de la primera muestra.
long	getEndUnix() Tiempo en milisegundos de la última muestra.
boolean	isCompatible() throws Exception Calcula si el teléfono es compatible con el protocolo Pulse. Debe de haber pasado al menos 15 segundos para determinar si es compatible o no, sino dará un error.

Tabla 6.18: Procedimientos de la clase «PulseStats»

6.4. CRC-8

Permite detectar si la información recibida se ha corrompido, certificando la integridad de los datos. Sin embargo, CRC no tiene la capacidad de proteger frente a modificaciones intencionadas si se produce un ataque MiTM.

CRC está basado en el residuo de una división polinómica, arrojando un tamaño fijo de bits, en este caso de 8 bits de información. Dónde el dividendo serán el polinomio $Datos(x) \cdot x^8$.

El polinomio divisor, o generador, debe de estar preestablecido antes de comenzar la emisión. El utilizado para el protocolo Pulse es $(x^8) + x^2 + x + 1$.

Para el protocolo Pulse no se realiza ninguna inversión ni en la entrada ni en la salida de datos. Entendiendo inversión como intercambiar los bit menos significativos con los más significativos.

Para verificar si la información es correcta el dividendo debe de ser $Datos(x) \cdot x^8 + CRC(x)$. Si en el resultado de la división el resto es 0, la información no se ha corrompido.

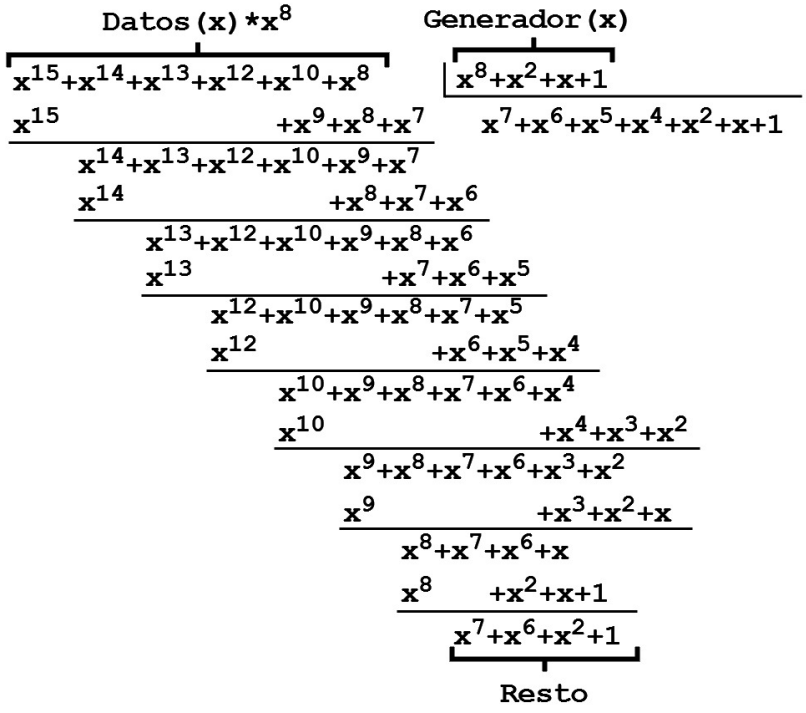


Figura 6.14: Ejemplo de cálculo CRC de $x^7 + x^6 + x^5 + x^4 + x^2 + 1$

Estas operaciones se pueden realizar binariamente sustituyendo los polinomios tal y como se hace en la figura 6.15.

6.5. Reed-Solomon

Reed-Solomon (RS) es un un tipo algebraico de corrección de errores en bloque hacia adelante (FEC). Lo que significa que el mensaje está dividido en bloques separados. Cada bloque contiene el mensaje original y un bloque de paridad. RS pertenece a la familia de códigos Bose-Chaudhuri-Hocquenghem (BCH).

RS, tal y como se ha dicho, opera en bloques de información y lo hace con los siguientes parámetros:

- Símbolo (m): Unidad mínima de información.

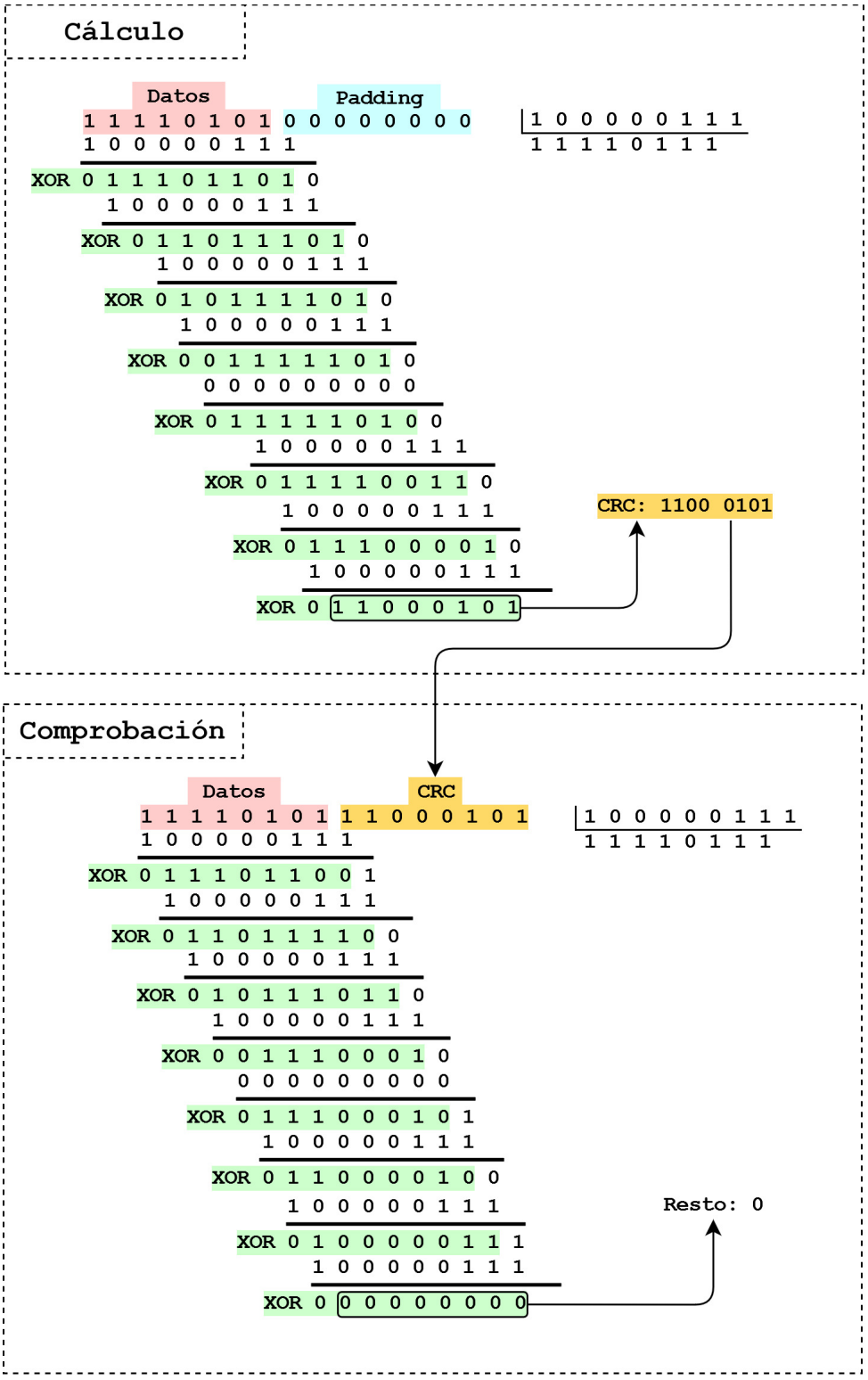


Figura 6.15: Ejemplo de cálculo y comprobación binaria CRC de 0xF5

- Mensaje (k): Información que se desea enviar dividida en bits de m símbolos.
- Paridad ($2t$): Son los símbolos de paridad. t representa los símbolos que pueden ser corregidos en un bloque.
- Bloque (n): Contienen el mensaje y la paridad.

Elegir los parámetros del código fija distintos niveles protección y afecta a la complejidad. Reed-Solomon es descrito como $RS(n, k)$ códigos. Además existe las siguientes relaciones:

$$n \leq 2^m - 1 \tag{6.1}$$

Cuando n no es igual el código hace referencia a un código acortado. En todos los códigos hay $n - k$ símbolos de paridad y se podrán corregir t símbolos en cada bloque, donde:

$$\begin{aligned} t &= (n-k)/2 \quad para\ n-k\ par \\ t &= (n-k-1)/2 \quad para\ n-k\ impar \end{aligned} \tag{6.2}$$

En la siguiente diagrama se muestra la composición de un bloque:

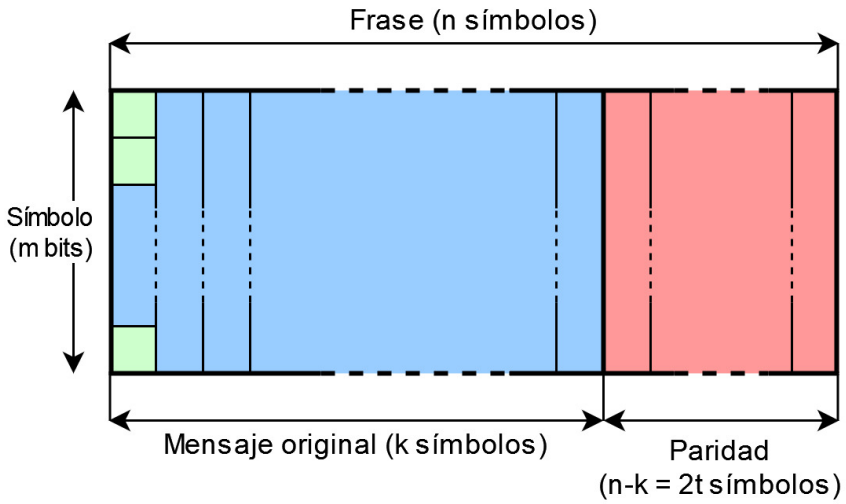


Figura 6.16: Bloque Reed-Solomon

Por desgracia, la anotación usada en Reed-Solomon no es totalmente consistente ni universal, por lo que es muy común encontrar codificadores y descodificaciones inconsistentes. Por ello es particularmente importante ser lo suficientemente simple, claro y específico.

6.5.1. Cuerpos finitos

Para poder explicar Reed-Solomon se requieren unos conocimientos básicos sobre los cuerpos finitos.

Se recuerda al lector que la sección de Reed-Solomon puede rebasar en algunos puntos los conocimientos de desarrollo software en conocimiento matemático avanzado. Se será extremadamente preciso desde un punto de vista de desarrollo software pero no desde el punto de vista matemático.

Se recomienda, si se desea profundizar en la materia, la lectura de libros sobre teoría de Galois y/o cuerpos finitos que teorizan y demuestren los principios matemáticos que aquí se expondrán y que se darán por ciertos.

Elementos en un cuerpo finito

Un cuerpo finito tiene elementos que están basados en un elemento primitivo que se le denotan como α y que pueden tomar los siguientes valores:

$$0, \alpha^0, \alpha^1, \ldots, \alpha^{n-1}$$

Dónde $n = 2^m - 1$. Entonces los cuerpos finitos son definidos como $\mathbb{F}_{2^m}[\alpha]$.

Suma y resta en cuerpos finitos

Para sumar dos elementos de un mismo cuerpo, se suman como si fueran dos polinomios:

$$\begin{gathered}(a_{m-1}x^{m-1} + \ldots + a_1x^1 + a_0x^0) + (b_{m-1}x^{m-1} + \ldots + b_1x^1 + b_0x^0) = \\ = (c_{m-1}x^{m-1} + \ldots + c_1x^1 + c_0x^0) \\ D\acute{o}nde: \quad c_i = a_i + b_i \qquad para\ 0 \leq 1 \leq m-1\end{gathered} \tag{6.3}$$

Como se trata de un cuerpo 2^m sólo pueden darse valores 0 y 1, resultando esta tabla:

Se suman se hace el módulo elemento a elemento. Una operación equivalente sería hacer una operación XOR. Un ejemplo en un cuerpo $\mathbb{F}_{2^3}$ de la suma de los polinomios $x^2 + x + 1$ y $x^2 + 1$ daría como resultado x y podría ser mostrado así:

$$\begin{array}{ll} 101 & (5) \\ \underline{111} & (\underline{7}) \\ 010 & (2) \end{array}$$

+	0	1
0	0	1
1	1	0

Tabla 6.19: Suma en $\mathbb{F}_{2^m}[\alpha]$

La resta es equivalente a la suma debido a que la resta tiene esta forma:

$$c_i = a_i - b_i \qquad para\ 0 \leq i \leq m-1 \tag{6.4}$$

Resultando lo mismo que en el caso de la suma. Usando el anterior ejemplo, sumar o restar, daría el mismo resultado: $5 - 7 = 2$.

El polinomio primitivo

Es una de las partes más relevante a la hora de crear un cuerpo finito. El polinomio debe de de ser de grado m y debe de ser irreducible. Por ejemplo, para un cuerpo finito $\mathbb{F}_{2^3}$, el polinomio puede ser:

$$p(x) = x^3 + x + 1 \tag{6.5}$$

Puede haber varios polinomios irreductibles para un mismo cuerpo. Una alternativa para $\mathbb{F}_{2^3}$ podría ser:

$$p(x) = x^3 + x^2 + 1 \tag{6.6}$$

Construcción de cuerpos finitos

Todos los los elementos, excepto el 0, pueden ser construidos usando el elemento primitivo α como raíz del polinomio generador.

Un polinomio primtivo en un $\mathbb{F}_{2^3}$, puede ser escrito de ambas maneras:

$$\begin{gathered} \alpha^3 + \alpha + 1 = 0 \\ \alpha^3 = \alpha + 1 \end{gathered} \tag{6.7}$$

Multiplicando por α y sustituyendo α^3 por α+1. Y si el proceso mostrado es continuado en α^6, se encontraría $\alpha^7 = \alpha^0,\ \alpha^8 = \alpha^1$ repitiéndose la secuencia ilimitadamente.

Índice	Cálculo	Polinomio	Binario	Decimal
0	0	0	000	0
α^0	1	1	001	1
α^1	α	α	010	2
α^2	α^2	α^2	100	4
α^3	$\alpha + 1$	$\alpha + 1$	011	3
α^4	$\alpha^3 \cdot \alpha = (\alpha + 1) \cdot \alpha$	$\alpha^2 + \alpha$	110	6
α^5	$\alpha^4 \cdot \alpha = (\alpha^2 + \alpha) \cdot \alpha = \alpha^3 + \alpha^2 = \alpha + 1 + \alpha^2$	$\alpha^2 + \alpha + 1$	111	7
α^6	$\alpha^5 \cdot \alpha = (\alpha^2 + \alpha + 1) \cdot \alpha = \alpha^3 + \alpha^2 + \alpha = \alpha + 1 + \alpha^2 + \alpha$	$\alpha^2 + 1$	101	5

Tabla 6.20: Cuerpo finito $\mathbb{F}_8$ con $p(x) = x^3 + x + 1$

Multiplicación y división

La multiplicación de dos polinomios de grado $m-1$ resulta en un polinomio de grado $2m-2$, pero eso no sería un elemento válido para $\mathbb{F}_{2^m}$. Una vez se ha multiplicado, se calcula el resto con el polinomio generador.

Por ejemplo, si se que quiere multiplicar 7 y 5 en $\mathbb{F}_{2^3}$ obtenemos esta representación polinómica:

$$(x^2 + x + 1) \cdot (x^2 + 1) = x^4 + x^3 + x^2 + x^2 + x + 1 = x^4 + x^3 + x + 1 \tag{6.8}$$

Para acabar la multiplicación se debe de dividir por el polinomio primitivo. El resto que de la división será el resultado de la multiplicación.

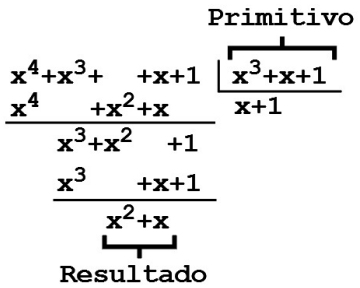


Figura 6.17: Multiplicación en cuerpo finito

Otra forma de calcular multiplicaciones es usando un sistema de logaritmos y poten-

cias. Se a de recordar al lector esta simple relación:

$$\alpha^b = c \mid log_\alpha(c) = b \tag{6.9}$$

Lo que implica que la tabla 6.20, si se lee de izquierda a derecha, es una tabla de potencias y, si se lee de derecha a izquierda, una tabla de logaritmos. Por ejemplo: $log_\alpha(7) = 5$, lo que implica que $\alpha^5 = 7$.

Llegados a este punto, la multiplicación es una operación simple:

$$x \cdot y = \alpha^{log_\alpha(x \cdot y) \ mod \ 2^m - 1} = \alpha^{(log_\alpha(x) + log_\alpha(y)) \ mod \ 2^m - 1} \tag{6.10}$$

Se usa el módulo $2^m - 1$ porque es una secuencia que se repite indefinidamente, tal y como se dijo a la hora de construir la tabla 6.20. Así no hace falta calcular más que los $2^m - 1$ números.

Un ejemplo de multiplicación de 7 y 5 en $\mathbb{F}_{2^3}$ sería:

$$7 \cdot 5 = \alpha^{(log_\alpha 7 + log_\alpha 5) \ mod \ 7} = \alpha^{5+6 \ mod \ 7} = \alpha^{11 \ mod \ 7} = \alpha^4 = 6 \tag{6.11}$$

El inconveniente, tal y como se ve en la tabla 6.20, es que no se puede representar el logaritmo de cero. No hay un valor α posible para el polinomio 0. Se debe de forzar a que, si uno de los valores es 0, el resultado sea 0.

Por otro lado, el sistema de logaritmos y potencias también permite calcular fácilmente inversos multiplicativos:

$$y^{-1} = \alpha^{(-log_\alpha y) \ mod \ 2^m - 1} \tag{6.12}$$

Un ejemplo de inverso multiplicativo de 5 en $\mathbb{F}_{2^3}$ es:

$$5^{-1} = \alpha^{(-log_\alpha 5) \ mod \ 7} = \alpha^{-6 \ mod \ 7} = \alpha^1 = 2 \tag{6.13}$$

Si se desea dividir, simplemente se debe de multiplicar el numerador por el inverso multiplicativo del denominador, lo que generaría la siguiente fórmula:

$$\begin{gathered} x \div y = x \cdot y^{-1} = \alpha^{(log_\alpha x \cdot y^{-1}) \ mod \ 2^m - 1} = \\ = \alpha^{(log_\alpha x + (-log_\alpha y)) \ mod \ 2^m - 1} = \alpha^{(log_\alpha x - log_\alpha y) \ mod \ 2^m - 1} \end{gathered} \tag{6.14}$$

Un ejemplo de división de 7 y 5 en $\mathbb{F}_{2^3}$ sería:

$$7 \div 5 = \alpha^{(log_\alpha 7 - log_\alpha 5) \ mod \ 7} = \alpha^{(5-6) \ mod \ 7} = \alpha^{-1 \ mod \ 7} = \alpha^6 = 5 \tag{6.15}$$

Sin embargo, hay algunos lenguajes de programación que no aceptan módulos negativos o que los calculan mal. Por ello se recomienda utilizar la siguiente fórmula:

$$\alpha^{(log_\alpha x - log_\alpha y) \ mod \ 2^m-1} \approx \alpha^{(log_\alpha x + (2^m-1) - log_\alpha y) \ mod \ 2^m-1} \tag{6.16}$$

Así los valores que se arrojen antes de hacer el módulo serán siempre positivos, permitiendo hacer la operación correctamente, independientemente del lenguaje de programación que se utilice.

6.5.2. Ajuste a la emisión del solenoide

Dependiendo del tamaño de información que se desee emitir a través del solenoide se utilizarán distintos tamaños a la hora de calcular RS. Aquí se muestra una tabla según los bits emitidos.

Bits emitidos	m	n	k	t	Polinomios primitivos	Tamaño en Pulse
1	3	7	5	1	11	21
2	4	15	11	2	19	30
3	3	7	5	1	11	7
4	4	15	11	2	19	15
5	5	31	25	3	37	31
6	6	50	40	10	67	50
7	7	50	40	10	137	50
8	8	50	40	10	285	50

Tabla 6.21: Reed-Solomon por bits emitidos

Como se puede ver, para los solenoides que emiten igual o menos de 2 bits se debe de fragmentar la información. Es decir, el tamaño de un símbolo (m) es superior a la capacidad que puede emitir el solenoide. Esto se hace, debido a que el símbolo (m) debe de ser igual o mayor a 3. Así que, por ejemplo, la información que emite el solenoide de 1 bit se debe de codificar como $\mathbb{F}_{2^3}$ y, una vez el proceso está hecho, dividir cada símbolo en 3 bits. Para el transmisor tiene que hacerse el proceso inverso, unir los 3 bits en símbolos y, posteriormente, decodificar como $\mathbb{F}_{2^3}$.

Además se aprecia que a partir de los solenoides que emiten igual o más de 6 bits no se llenarán nunca los $2^m - 1 - 2t$ símbolos. Para ello los símbolos de mayor peso estarán

rellenos de ceros hasta cubrir los $2^m - 1 - 2t$ que habría que tener para poder codificar y descodificar.

6.5.3. Codificación

Genera una bloque de n símbolos, añadiéndole paridad. El bloque no puede ser superior a los $2^m - 1$ símbolos.

Polinomio generador

Se debe de hacer antes de codificar un mensaje es generar el polinomio generador $g(x)$. Es en un polinomio de $2t$ factores:

$$g(x) = (x + \alpha^0)(x + \alpha^1) \dots (x + \alpha^{2t-1}) \tag{6.17}$$

Hay que tener en cuenta que cada campo del polinomio es en realidad un polinomio en un cuerpo finito. Así que para hacer cualquier operación se tendrá que tener en cuenta todo el apartado de cuerpos finitos 6.5.1.

Obtención del bloque a transmitir

Se debe de formar el polinomio $M(x)$ que contendrá k símbolos de mensaje original, tal y como se enuncia en la figura 6.16. Deberá de ser multiplicado por x^{2t}.

Posteriormente se dividirá el polinomio $(M(x)x^{2t})/g(x)$ para poder conseguir el resto. El resto siempre tendrá un grado $2t - 1$.

El resto deberá ser añadido al final del mensaje. Así que la información transmitida será:

$$T(x) = M(x)x^{2t} + (M(x)x^{2t} \ mod \ g(x)) \tag{6.18}$$

Ejemplo de codificación

Si se quiere enviar $3x^4+4x^3+3x^2+2x+1$ en un cuerpo $\mathbb{F}_{2^3}$ con un polinomio primitivo $\alpha^3 + \alpha + 1 = 0$, cuando $RS(7, 5)$, lo primero que habría que hacer es obtener el polinomio

generador:

$$\begin{aligned} g(x) &= (x+\alpha^0)(x+\alpha^1) = (x+1)(x+2) \\ &= x^2+x+2x+2 = x^2+3x+2 \end{aligned} \tag{6.19}$$

Una vez que tenemos el polinomio generador, hay que preparar el cociente, multiplicando $M(x) \cdot x^{2t}$, lo que daría este resultado:

$$\begin{aligned} M(x) \cdot x^{2t} &= (3x^4+4x^3+3x^2+2x+1)x^2 \\ &= 3x^6+4x^5+3x^4+2x^3+x^2 \end{aligned} \tag{6.20}$$

La división sería un proceso similar al de la multiplicación. Hay que tener en cuenta que tanto cuando se suma como cuando se multiplica se está haciendo sobre elementos de un cuerpo finito, exactamente de la misma forma que se ha hecho al obtener $g(x)$. Sería tal que así:

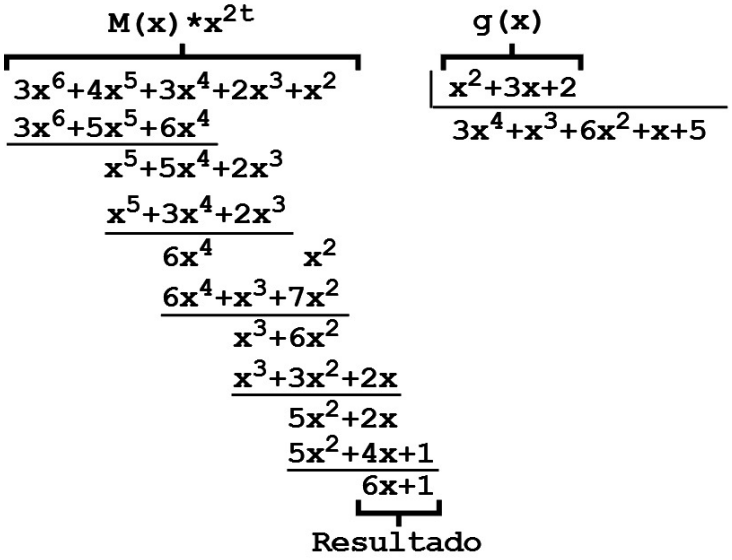


Figura 6.18: Obtención de resto

Una vez se ha obtenido el resto, se suma a $M(x)x^{2t}$. Ésa será la información a transmitir:

$$\begin{aligned} T(x) &= M(x) \cdot x^2 t + (M(x)x^{2t} \ mod \ g(x)) \\ &= 3x^6+4x^5+3x^4+2x^3+x^2+6x+1 \end{aligned} \tag{6.21}$$

Resumen de codificación

Debido a que es un proceso con muchos pasos se ha hecho un breve resumen:

1. Fija, según los bits emitidos, los distintos valores de Reed-Solomon de la tabla 6.21
2. Crear el "Polinomio generador" 6.5.3
3. La información se fragmenta según el tamaño de símbolo (m) y de mensaje por bloque (k)
4. "Obtención del bloque a transmitir" 6.5.3
5. Si, según la tabla 6.21, no coinciden "Tamaño en Pulsos" con n, se fragmenta la información.

6.5.4. Descodificación

Los errores serán añadidos al mensaje polinómico, T(x) en el formato E(x). Eso generará el polinomio R(x) que será el recibido por el receptor:

$$R(x) = T(x) + E(x) \; dónde \; E(x) = E_{n-1}x^{n-1} + \ldots + E_1x + E_0 \tag{6.22}$$

Cada coeficiente $E_{n-1}x^{n-1} + \ldots + E_0$ representan un error de m-bits.

Síndrome

El primer paso para descodificar la información es dividir el polinomio recibido por cada factor $(x+\alpha^i)$ del polinomio generador 6.5.3 de la ecuación . Esta operación produce un cociente y un resto tal que:

$$\frac{R(x)}{x+\alpha^i} = C_i(x) + \frac{S_i}{x+\alpha^i} \quad para \; b \leq i \leq b+2t-1 \tag{6.23}$$

Así que el síndrome sería equivalente a:

$$S_i = R(x) + C_i(x) \cdot (x+\alpha^i) \tag{6.24}$$

Y cuando $x = a^i$ el síndrome es igual a:

$$S_i = R(\alpha^i) = R_{n-1}(\alpha^i)^{n-1} + R_{n-2}(\alpha^i)^{n-2} + \ldots + R_1(\alpha^i) + R_0 \tag{6.25}$$

Dónde los coeficientes $R_{n-1} \ldots R_0$ son los datos recibidos. Lo que significa que el polinomio se evalúa sustituyendo $x = \alpha^i$. Así se calcula el resto de la división de $R(x)$ y $(x+\alpha^i)$.

Estas operaciones pueden reescribirse con el método Horner:

$$S_i = (\ldots(R_{n-1}\alpha^i + R_{n-2})\alpha^i + \ldots + R_i)\alpha^i + R_0 \tag{6.26}$$

El método Horner, empieza por la multiplicación del primer coeficiente R_{n-1} por α^i. La ventaja es que siempre se multiplica por el mismo valor en cada vez, lo que facilita el cálculo.

Sustituyendo en la ecuación 6.22, se obtendría:

$$R(\alpha^i) = T(\alpha^i) + E(\alpha^i) \tag{6.27}$$

En la fórmula $T(\alpha^i) = 0$ debido a que $x + \alpha^j$ es un factor de $G(x)$, implicando que también es un factor de $T(x)$. Así que:

$$R(\alpha^i) = E(\alpha^i) = S_i \tag{6.28}$$

Lo que significa que el error depende totalmente del síndrome y que ningún otro valor le afecta. Así que si todos los valores del síndrome son cero, podemos decir que no se ha dado ningún error.

Ecuaciones del síndrome

La ecuación 6.25 establece una relación entre la información recibida y el síndrome mientras que la ecuación 6.28 indica una relación entre el síndrome y el Error. Ambas relaciones permiten reescribir el polinomio de error $E(x)$, asumiendo que v errores han ocurrido, dónde $v \leq t$:

$$E(x) = Y_1 x^{e_1} + Y_2 x^{e_2} + \ldots + Y_v x^{e_v} \tag{6.29}$$

Dónde $e_1, \ldots, e_v$ indica la localización de los errores en exponentes de x, mientras que $Y_1, \ldots, Y_v$ representan los valores de error en esas localizaciones. Si se sustituye en la ecuación 6.28 daría:

$$\begin{aligned} S_i &= E(\alpha^i) \\ &= Y_1\alpha^{ie_1} + Y_2\alpha^{ie_2} + \ldots + Y_v\alpha^{ie_v} \\ &= Y_1{X_1}^i + Y_2{X_2}^i + \ldots + Y_v{X_v}^i \end{aligned} \tag{6.30}$$

Dónde $X_i = \alpha^{ie_1}, \ldots, X_v = \alpha^{ie_v}$ representa al localizador de polinomios. Entonces las

ecuaciones del síndrome 2t pueden ser simplificadas como:

$$\begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ \vdots \\ S_{2t-1} \end{bmatrix} = \begin{bmatrix} {X_1}^0 & {X_2}^0 & \dots & {X_v}^0 \\ {X_1}^1 & {X_2}^1 & \dots & {X_v}^1 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ {X_1}^{2t-1} & {X_2}^{2t-1} & \dots & {X_v}^{2t-1} \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_v \end{bmatrix} \tag{6.31}$$

Es importante fijarse que los sindromes son escritos como $S_0 \dots S_{2t-1}$ que corresponden con las raíces de $\alpha^0 \dots \alpha^{2t-1}$ y las potencias de X dependen del polinomio generador.

Polinomio localizador de errores

Es denotado como $\Lambda(x)$. Contiene los v factores con la forma de $1 + X_j x$ y por lo tanto contiene el inverso ${X_1}^{-1}, \dots, {X_1}^{-1}$ de las v raíces localizadoras de errores. Una vez se expanden los valores, se producirá un polinomio de grado v con coeficiente $\Lambda_1 \dots \Lambda_v$:

$$\begin{aligned} \Lambda(x) &= (1 + X_1)(x + X_2) \dots (x + X_v) \\ &= 1 + \Lambda_1 x + \dots + \Lambda_{v-1} x^{v-1} + \Lambda_v x^v \end{aligned} \tag{6.32}$$

Búsqueda de coeficientes en polinomio localizador de errores

Para cada uno de los errores corresponde la raiz ${X_j}^{-1}$ que hará $\Lambda(x)$ igual a cero. Tal que:

$$1 + \Lambda_1 {X_j}^{-1} + \dots + \Lambda_{v-1} {X_j}^{-v+1} + \Lambda_v {X_j}^i = 0 \tag{6.33}$$

o también se podría multiplicar por ${Y_j X_j}^{i+v}$:

$${Y_j X_j}^{i+v} + \Lambda_1 {Y_j X_j}^{i+v-1} + \dots + \Lambda_{v-1} {Y_j X_j}^{i+1} + \Lambda_v {Y_j X_j}^i = 0 \tag{6.34}$$

Lo que permitiría sustituir ${Y_j X_j}^{i+v}$ por S_{i+v}, que sí que lo conocemos, dejando la ecuación finalmente así:

$$S_{i+v} + \Lambda_1 S_{i+v-1} + \dots + \Lambda_{v-1} S_{i+1} + \Lambda_v S_i = 0 \textit{ dónde } i = 0, \dots, 2t - v - 1 \tag{6.35}$$

Resolviendo la ecuación de $\Lambda_1 \dots \Lambda_v$ podemos llegar a representar la función en la siguiente matriz. Excepto por que no conocemos el valor de v:

$$\begin{bmatrix} S_v \\ S_{v+1} \\ S_{v+2} \\ \vdots \\ S_{2v-1} \end{bmatrix} = \begin{bmatrix} S_{v-1} & S_{v-2} & S_{v-3} & \dots & S_0 \\ S_v & S_{v-1} & S_{v-2} & \dots & S_1 \\ S_{v+1} & S_v & S_{v-1} & \dots & S_2 \\ \vdots & \vdots & \vdots & & \vdots \\ S_{2v-2} & S_{2v-3} & S_{2v-4} & \dots & S_{v-1} \end{bmatrix} \cdot \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \vdots \\ \Lambda_v \end{bmatrix} \tag{6.36}$$

Así que $v = t$ y se seguirá procesando la información hasta que el determinante encontrado sea 0. Estas ecuaciones siempre serán independientes y se podrán resolver. El coeficiente de polinomio de error $\Lambda_1 \ldots \Lambda_v$ puede ser encontrado invirtiendo la matriz para resolver las ecuaciones.

Algoritmo de Berlkamp-Massey

Permite calcular el polinomio localizador de errores $\Lambda(x)$ de una forma más eficiente que la anteriormente citada. Es un algoritmo con t interacciones y se resuelve a través de los siguientes pasos:

1. Inicialización: $k = 0$, $\Lambda_0(x) = 1$, $T_0(c) = 1$, $L_0 = 0$
2. $k+ = 1$ y $Z_k = S_k - \sum_L^{i=1} \Lambda_i^{k-1} S_{k-1}$
3. Si $Z_k = 0$ ir al paso 7
4. $\Lambda_k(x) = \Lambda^{k-1}(x) + Z_k \cdot [T_k(x)]$
5. Si $2L \geq k$, ir al paso 7
6. $L = k - L$ y $T_k(x) = \Lambda_{k-1}(x)/Z_k$
7. $T_k(x) = x \cdot T_k(x)$
8. Si $k < t$ regresar al paso 2

Resolver el polinomio localizador de errores: Búsqueda Chien

Una vez tenemos calculado el coeficiente de los valores $\Lambda_1 \ldots \Lambda_v$ del polinomio localizador de errores, se puede encontrar las distintas raíces. Si el polinomio es escrito en este formato:

$$\Lambda(X) = X_1(x + X_1^{-1})\ X_2(x + X_2^{-1}) \ldots \tag{6.37}$$

entonces está claro que el valor de la función será cero si $x = X_1^{-1}, X_2^{-1}, \ldots$ es decir si $x = \alpha^{-e_1}, \alpha^{-e_1}, \ldots$

Las raíces y, por consiguiente, los valores $X_1 \ldots X_v$ son encontrados por prueba y error. Esto es conocido como "Búsqueda Chien", dónde cada uno de los posibles valores de los cuerpos (los valores $\alpha^i,\ 0 \leq i \leq n-1$) son sustituidos en las ecuaciones del polinomio localizador de errores 6.32 y los resultados son evaluados. Si la ecuación se reduce a cero, entonces el valor x es la raíz e identifica la posición del error.

Cálculo de errores

Una vez ya tenemos los errores localizadores $X_1 \dots X_v$, los sustituimos en las ecuaciones del síndrome 6.36, las primeras v ecuaciones pueden ser resueltas invirtiendo la matriz, dando como resultado los valores de error $Y_1 \dots Y_v$.

Algoritmo de Forney

Es una forma más eficiente de calcular el valor de error de Y_j establecido en el polinomio $\Lambda(x)$. El algoritmo usa la derivada del polinomio localizador.

Para empezar, en un polinomio $f(x)$ formado tal que $f(x) = 1 + f_1 x + f_2 x^2 + \ldots + f_v x^v$, se tendría la siguiente derivada:

$$f'(x) = f_1 + 2f_2 x + \ldots + v f_v x^{v-1} \tag{6.38}$$

Sin embargo, para el polinomio localizador de errores $\Lambda(x)$, para $x = X_j^{-1}$, la derivada se reduciría a:

$$\Lambda'(X_j^{-1}) = \Lambda_1 + \Lambda_3 X_j^{-2} + \Lambda_5 X_j^{-4} \dots \tag{6.39}$$

Por otro lado, para poder calcular el error de magnitud con Forney es necesario definir el polinomio del síndrome:

$$\Omega(x) = \Omega_{v-1} x^{v-1} + \ldots + \Omega_1 x + \Omega_0 \tag{6.40}$$

Esta ecuación puede ser reescrita como:

$$\Omega(x) = [S(x)\Lambda(x)] \; mod \; x^{2t} \tag{6.41}$$

Dónde $S(x)$ representa el polinomio del síndrome y $\Lambda(x)$ el polinomio localizador. Cualquier término superior al grado x^{2t} o al producto es ignorado tal que:

$$\begin{aligned} \Omega_0 &= S_b \\ \Omega_1 &= S_{b+1} + S_b \Lambda_1 \\ &\vdots \\ \Omega_{v-1} &= S_{b+v-1} + S_{b+v-2}\Lambda_1 + \ldots + S_b \Lambda_{v-1} \end{aligned} \tag{6.42}$$

Con este polinomio, el algoritmo de Forney sería escrito con la siguiente ecuación:

$$Y_j = X_j^{1-b} \frac{\Omega(X_j^{-1})}{\Lambda'(X_j^{-1})} \tag{6.43}$$

Dónde $\Lambda'(X_j^{-1})$ es la derivada de $\Lambda(x)$ para $x = X_j^{-1}$. La ecuación sólo dará resultados válidos para posiciones que contengan un error. Si se calcula en otras posiciones el resultado, muy probablemente, sea inválido. Por eso sigue siendo preciso realizar la "Búsqueda Chien" 6.5.4.

Corrección de errores

Teniendo ya generado el polinomio $E(x)$ lo único que hay que hacer es sumarlo a $R(x)$ y eliminar los bits de 2t símbolos de paridad.

Si se quiere dar mayor garantía, se puede volver a calcular el síndrome. Si el resultado es cero se puede decir que, muy probablemente, se hayan calculado correctamente los errores. Sin embargo, puede darse casos extremos en que el error no haya sido detectado correctamente y el síndrome no sea cero, lo que implicaría que la información es irrecuperable.

Resumen de descodificación

Debido a que es un proceso muy complejo se ha hecho un breve resumen:

1. Fija, según los bits emitidos, los distintos valores de Reed-Solomon de la tabla 6.21
2. Si, según la tabla 6.21 no coinciden "Tamaño en Pulsos" con n, se deberá de reensamblar toda la información que se reciba, según el tamaño de símbolo (m) que se haya fijado en el paso anterior.
3. Calcula el síndrome. Si el síndrome tiene grado 0 ir al paso 1.
4. Busca el polinomio localizador de errores a través del algoritmo más óptimo "Berlkamp-Massey" 6.5.4.
5. Resuelve el polinomio localizador de errores mediante "Búsqueda Chien" 6.5.4.
6. Calcula la magnitud del error mediante el algoritmo de "Forney" 6.5.4.
7. Suma la información recibida con el $E(x)$ calculado en todos los pasos anteriores.
8. Vuelve a calcular el síndrome. Si el síndrome no tiene grado 0, no se puede recuperar la información correctamente.
9. Extrae los k símbolos del bloque. La información ha sido descodificada correctamente.

6.6. Código Gray

Es una escala binaria en la que dos números consecutivos difiere, únicamente, un único bit. Permite eliminar los saltos mayores de 1 bit en una escala numérica. De esta forma, si se comete un error a la hora de leer algún nivel, se evita cometer errores mayores.

Para construir la escala de Gray se han de reflejar los bits de un número a otro.

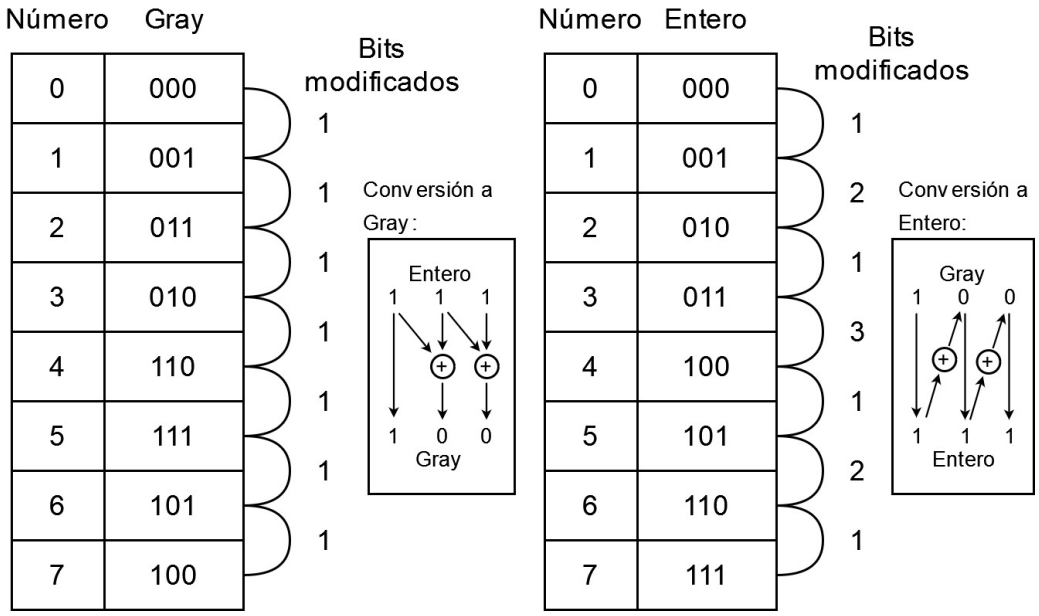


Figura 6.19: Conversión a código Gray

Capítulo 7

Pruebas

Las pruebas son fundamentales para garantizar el correcto funcionamiento del software y hardware desarrollado. Permiten identificar y corregir errores, verificar el funcionamiento correcto de los componentes del sistema y asegurar que se cumpla con los requisitos y expectativas del usuario.

En este proyecto todas las pruebas deben de realizarse manualmente, ya que todo el software depende directa o indirectamente de un hardware muy específico, lo que impide su automatización. Por este motivo, se ha creado una interfaz de depuración en Impulse, que permite agilizar todo el proceso.

Se realizarán test unitarios de caja blanca y negra para evaluar la calidad y confiabilidad del software en cada etapa del proyecto. Los test de caja blanca analizarán y probarán el código fuente individualmente, evaluando su funcionamiento interno y verificando que cumpla con los requisitos y comportamiento esperado.

Por otro lado, los test de caja negra evaluarán el software sin conocer su estructura interna, probando su capacidad para manejar entradas y salidas de datos de acuerdo con los requisitos establecidos. Estos enfoques permitirán identificar posibles errores en la lógica interna de las unidades de código y en la funcionalidad general del software, corrigiéndolos de manera temprana en el proceso de desarrollo y garantizando la calidad y confiabilidad del software.

Los test unitarios de caja blanca y caja negra mejoran la calidad y confiabilidad del software, facilitando su mantenimiento a lo largo del tiempo al identificar posibles problemas temprano y ayudar a comprender el comportamiento esperado del código.

7.1. Pruebas de funcionalidad

7.1.1. TU-01 Conexión del puerto COM

Versión: 1.0

Objetivo

El programa debe de conectarse al dispositivo a través del puerto COM.

Precondición

Ninguna.

Datos de entrada

El usuario solicita conectarse al puerto COM del dispositivo.

Secuencia

1. El sistema busca el puerto COM.

2. El sistema se conecta al puerto COM.

Datos de salida

Se muestra como conectado al puerto COM.

7.1.2. TU-02 Verificar envío de una tarea a través de Pulse

Versión: 1.0

Objetivo

El programa debe de enviar un mensaje a través de Pulse.

Precondición

Estar conectado al dispositivo.

Datos de entrada

Se solicita el envío a través de Pulse con el mensaje: `Hola mundo`.

Secuencia

1. El sistema envía el mensaje a través del programa.

2. El sistema recibe el mensaje a través de la aplicación.

Datos de salida

Se muestra el mensaje: `Hola mundo`.

7.1.3. TU-03 Verificar envío de una tarea a través de bluetooth

Versión: 1.0

Objetivo

El programa debe de enviar un mensaje a través de bluetooth.

Precondición

Estar conectado al dispositivo.

Datos de entrada

Se solicita el envío a través de bluetooth con el mensaje: `Hola mundo`.

Secuencia

1. El sistema envía el mensaje a través del programa.

2. El sistema recibe el mensaje a través de la aplicación.

Datos de salida

Se muestra el mensaje: `Hola mundo`.

7.1.4. TU-04 Comprobar sensor de proximidad

Versión: 1.0

Objetivo

Se verifica si el sensor es capaz de detectar si hay un objeto cercano a él.

Precondición

Haber ordenado el envío de una tarea.

Datos de entrada

Se aproxima el teléfono al dispositivo.

Secuencia

1. El sistema detecta el móvil.

Datos de salida

Se comienza a realizar el envío de la tarea.

7.1.5. TU-05 Comprobar sensor de movimiento

Versión: 1.0

Objetivo

Se verifica si el sensor es capaz de detectar si el dispositivo se ha movido.

Precondición

Estar realizando el envío de una tarea.

Datos de entrada

Se mueve el dispositivo.

Secuencia

1. El sistema detecta el movimiento del dispositivo.

Datos de salida

Se muestra un mensaje de que el dispositivo se ha movido.

7.1.6. TU-06 Verificar que la interfaz bluetooth se enciende

Versión: 1.0

Objetivo

Comprobar que la interfaz bluetooth está encendida y visible por otros dispositivos.

Precondición

Estar conectado al dispositivo.

Datos de entrada

Se solicita el envío a través de bluetooth con el mensaje: `Hola mundo`.

Secuencia

1. El sistema enciende la interfaz bluetooth.

Datos de salida

Se detecta la existencia de una interfaz bluetooth con el nombre "`Impulse`".

7.1.7. TU-07 Verificar que la interfaz bluetooth se apaga

Versión: 1.0

Objetivo

Comprobar que la interfaz bluetooth está apagada.

Precondición

Estar conectado al dispositivo.

Datos de entrada

Se solicita detener el envío de todas las tareas.

Secuencia

1. El sistema detecta que han sido eliminadas todas las tareas.

2. El sistema apaga la interfaz bluetooth.

Datos de salida

No se puede encontrar la interfaz bluetooth con el nombre "`Impulse`".

7.1.8. TU-08 El dispositivo supera el magnetismo terrestres

Versión: 1.0

Objetivo

Verificar que se superan los 67μT para los bits para el mínimo valor que emite el dispositivo, es decir el byte `0x01`.

Precondición

Recoger datos de varias tareas a través del modo desarrollador.

Datos de entrada

Abrir los distintos ficheros recogidos en modo desarrollador.

Secuencia

1. El sistema parsea los datos.

2. El sistema muestra una gráfica con todos los datos recogidos.

Datos de salida

Se comprueba que los bytes `0x01` superan los 67μT.

7.1.9. TU-09 El dispositivo debe de cambiar su campo magnético en periodos de 50ms

Versión: 1.0

Objetivo

Verificar que el hardware cambia su campo magnético cada 50ms.

Precondición

Recoger datos de varias tareas a través del modo desarrollador.

Datos de entrada

Abrir los distintos ficheros recogidos en modo desarrollador.

Secuencia

1. El sistema parsea los datos.

2. El sistema muestra una gráfica con todos los datos recogidos.

Datos de salida

Se comprueba que todos los datos obtenidos cambian cada 50 ms.

7.1.10. TU-10 Guardado de configuración del programa/ aplicación

Versión: 1.0

Objetivo

Verificar que se ha guardado la configuración correctamente.

Precondición

Ninguna.

Datos de entrada

El usuario solicita al programa/aplicación modificar el modo noche, cierra la aplicación y la vuelve a abrir.

Secuencia

1. El sistema detecta que se ha cambiado la configuración.

2. El sistema guarda la configuración.

3. El sistema se cierra.

4. El sistema se vuelve a abrir y se recupera la configuración.

Datos de salida

Se muestra la misma configuración del modo noche que la modificación que se realizó.

7.1.11. TU-11 Correcto funcionamiento del modo noche en el programa/ aplicación

Versión: 1.0

Objetivo

Verificar que se ha cambiado el modo noche adecuadamente.

Precondición

Ninguna.

Datos de entrada

Se solicita al sistema cambiar a modo noche.

Secuencia

1. El sistema modifica el aspecto a un modo más oscuro.

Datos de salida

Se comprueba que todos los componentes han cambiado al color adecuado.

Parte III

Manual de la aplicación

Capítulo 8

Manual

8.1. Manual de instalación

8.1.1. Aplicación Escritorio

Windows

Para ejecutar el programa, simplemente es preciso hacer doble clic en el archivo ejecutable con extensión .exe que descargó del repositorio. Al hacerlo, se abrirá la ventana principal del programa y estará listo para empezar a utilizarlo. No se requiere la instalación de software adicional.

GNU/Linux

Es necesario tener instaladas las bibliotecas "libqt5serialport5" y "libqt5printsupport5", ya que es un programa con librerías dinámicas. Aquí tiene algunas formas de instalación dependiendo del gestor de paquetes que utilice su distribución:

```
sudo apt install libqt5serialport5 libqt5printsupport5 # Debian/Ubuntu

sudo dnf install qt5-serialport qt5-qtprintsupport # Fedora

sudo pacman -S qt5-serialport qt5-printsupport # Arch Linux
```

Listado 8.1: Instalación de bibliotecas

Además debe de pertenecer al grupo "dialout" que otorga permisos para poder leer y escribir puertos serie. Puede hacerlo simplemente con:

```
sudo adduser USUARIO dialout
```

Listado 8.2: Añadir usuario

Deberá sustituir "USUARIO" por el nombre de usuario al que desee permitir usar Impulse.

8.1.2. Aplicación Android

Una vez descargada la aplicación a través de una tienda, se irán mostrando las distintas pantallas que le servirán para informar del funcionamiento de la aplicación además de mostrar la compatibilidad con su teléfono móvil.

La primera pantalla es una breve presentación de la utilidad de la aplicación. Es una pantalla exclusivamente de carácter informativo.

La segunda informa de que se realizará una comprobación de cara a verificar que el móvil es compatible con la biblioteca Pulse. Sólo si resulta compatible podrá seguir avanzando en la instalación. En caso de que no lo sea la aplicación dará la oportunidad de volver a repetir la comprobación.

La tercera pantalla está dedicado a conceder permisos de ubicación para que la aplicación tenga capacidad para buscar todos los dispositivos bluetooth del entorno.

La cuarta y quinta pantalla dará permisos de almacenamiento sobre el móvil y la carpeta dónde se almacenarán los ficheros. No es obligatorio aceptar este permiso, pero si no lo acepta, simplemente, no podrá recibir ficheros. Sólo si se acepta, deberá escoger una carpeta de almacenamiento, sino no será preciso.

La sexta le recomienda la instalación de Electrum, un wallet A de bitcoin (BTC) de código abierto. Sin embargo, se acepta cualquier aplicación que acepte transacciones BTC. Sin una aplicación compatible con BTC no se podrá intercambiar BTC correctamente.

Por último se le informa que ya cumple con todos la información y requisitos para poder ejecutar la aplicación normalmente. Al igual que la primera, es una pantalla meramente informativa.

Figura 8.1: Instalación completa ordenada de izquierda a derecha y de arriba a abajo

8.2. Manual de usuario

8.2.1. Introducción

Este manual explica el funcionamiento de la herramienta Impulse para la comunicación entre dispositivos a través de campos magnéticos.

¿Qué es Impulse?

Una aplicación que le permitirá transmitir información fácilmente a través de pulsos magnéticos desde un transmisor (un equipo junto a un hardware específico) y un receptor (un móvil Android).

8.2.2. Requisitos mínimos

Para garantizar el correcto funcionamiento del sistema es preciso cumplir con unos requisitos precisos a nivel de hardware y software. Se trata de los requisitos mínimos impuestos por las distintas bibliotecas utilizadas.

Requisitos mínimos del equipo:

- 256 MB de RAM
- 500 MHz CPU
- Soporte OpenGL ES 2.0
- Para sistemas Windows: Windows 7 o superior
- Para sistemas GNU/Linux: Cualquier distribución con las bibliotecas "libqt5serialport5" y "libqt5printsupport5".

Requisitos mínimos del móvil:

- El sensor del móvil debe de tener una capacidad de muestreo inferior a los 50 ms para poder recibir correctamente la información.
- Android 5.0 (Lollipop) o superior.

Adicionalmente debe de poseer una placa de circuito impreso (PCB) como la diseñada durante el proyecto, una tarjeta Arduino 33 BLE y un solenoide como el especificado.

8.2.3. Programa de escritorio

La aplicación de escritorio es, a los ojos del usuario, sencilla. No requiere ningún tipo de instalación, simplemente es un ejecutable que funcionará desde el principio, sin bibliotecas complementarias que descargar.

Para poder enviar una tarea lo primero que se debe hacer es conectarse al puerto COM del dispositivo diseñado. Una vez hecho esto ya se puede realizar el envío de cualquier tarea.

Si se desea se pueden cambiar algunos ajustes básicos, que se guardarán automáticamente:

- Envío bluetooth (habilitado por defecto): Impulse usará los campos magnéticos para conectarse al dispositivo bluetooth y la tarea, posteriormente, se enviará por bluetooth.
- Modo noche: La interfaz de usuario cambiará a un modo oscuro.
- Ocultar/Mostrar tareas: Muestra u oculta las distintas pestañas de las tareas. Útil si sólo se va a utilizar unas únicas tareas de forma repetitiva.

Cada tarea tendrá los campos correspondientes que deberá de rellenar. Sin embargo, todas las tareas tendrán una casilla de verificación "Repetir" y un botón de "Enviar". "Repetir" permitirá que, cuando una tarea finalice, se pueda volver a enviar la misma información rápidamente a otro nuevo móvil cuando se retire. "Enviar", verificará que la tarea se puede realizar correctamente y añadirá la tarea al listado de tareas.

Una vez se ha pulsado el botón "Enviar", tal y como se ha dicho, se añade la tarea al final del listado y esperará a ser irán enviando, mostrando el porcentaje de envío de las distintas tareas. Las tareas se pueden eliminar o modificar la repetición. Una vez la tarea ha sido enviada se eliminará automáticamente del listado.

Además se incluyen otras funcionalidades:

- Actualizar: Verifica la existencia de nuevas actualizaciones para el proyecto a través de un canal seguro (HTTPS). Si hay actualizaciones abrirá la página en el navegador por defecto para descargarla. Es preciso que, si se ejecuta esta funcionalidad, haya conexión a Internet.
- Contactar: Abre el administrador de correo electrónico por defecto del sistema para enviar al desarrollador un e-mail con datos relevantes.
- Traducir: Accede al proyecto para realizar una petición de traducción al desarrollador.
- Acerca de: Muestra todas las bibliotecas y su correspondiente licencia de proyecto.

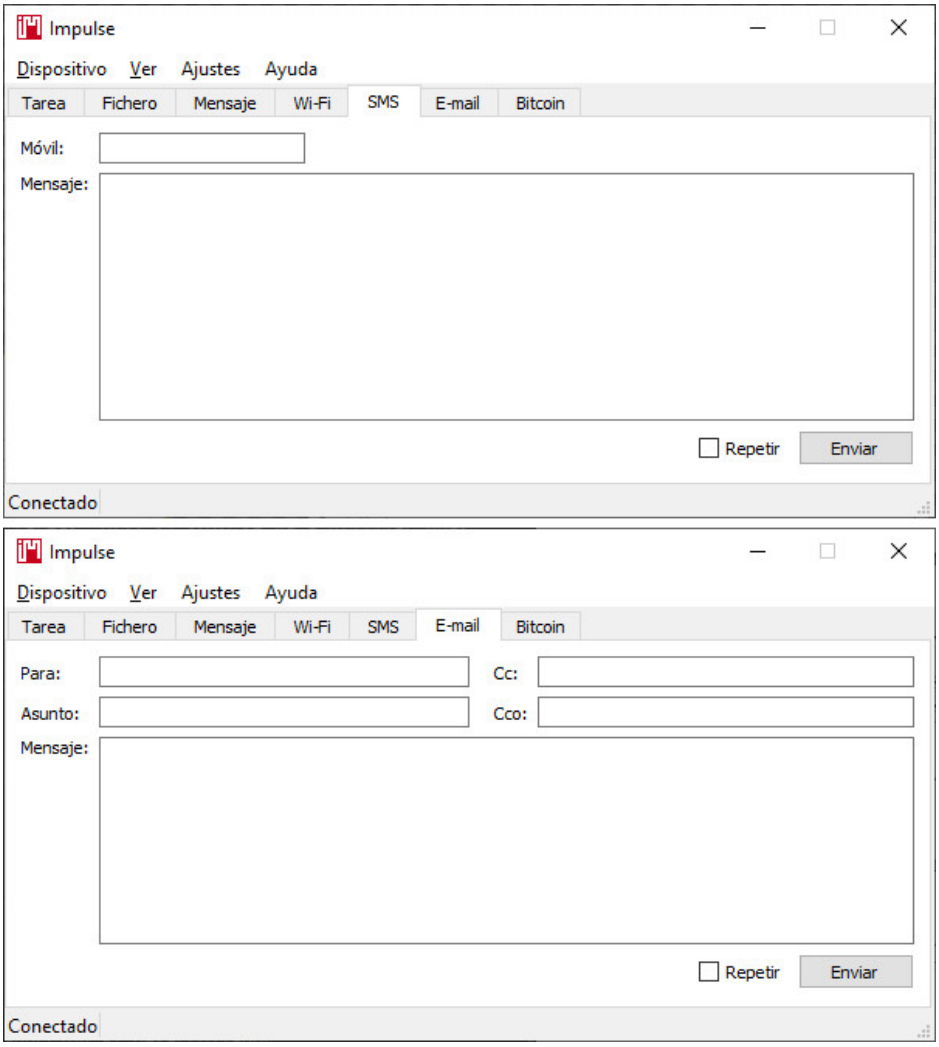


Figura 8.2: Envío de un sms (Arriba). Envío de un e-mail (Abajo).

8.2.4. Aplicación móvil

La aplicación móvil estará disponible en tiendas de aplicaciones (como Play Store o Aptoide), que permitirá descargar y actualizar la aplicación fácilmente.

Una vez se ha descargado la aplicación se debe de comenzar con la instalación, que se encargará de informar al usuario y verificar la compatibilidad y permisos del móvil.

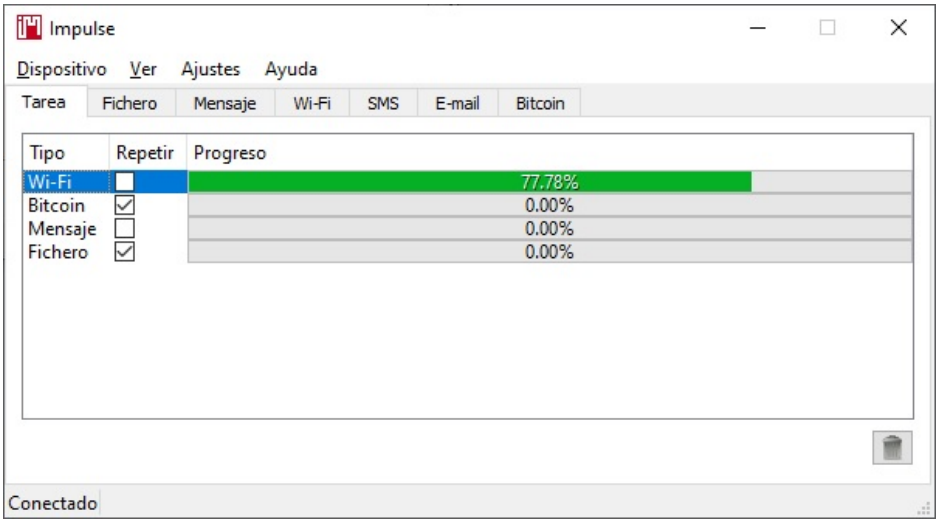


Figura 8.3: Listado de tareas pendientes

Utilización de la aplicación

La utilización de la aplicación es simple, únicamente se debe de colocar el móvil sobre el dispositivo una vez que se muestra el mensaje "Preparado para recibir datos". La barra de progreso, cuando se empiece a recibir una tarea, indicará el porcentaje de tarea entregado.

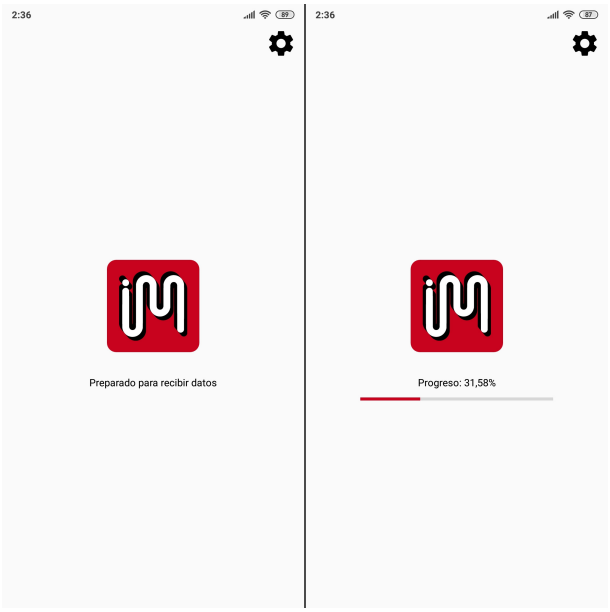


Figura 8.4: Recepción de tareas

Si se desea, se pueden realizar algunos ajustes de forma persistente en la aplicación como:

- Modo noche: Cambia el tema de la aplicación a unos tonos más oscuros.
- Almacenamiento de la aplicación: Cambia la carpeta por defecto en la que se almacenarán los ficheros a la carpeta indicada por el usuario.
- Ayuda: Vuelve a mostrar el tutorial desde el inicio.

Adicionalmente se incluyen otras funcionalidades:

- Información de la aplicación: Se muestra información acerca de la versión, las bibliotecas utilizadas junto a su licencia e información acerca del autor.
- Traducir: Abre el repositorio de la aplicación para ponerse en contacto con el desarrollador y realizar la traducción de la aplicación.
- Comentario: Abre el administrador de correo electrónico por defecto del sistema con información autocomplementada acerca del hardware y software del móvil.

8.3. Manual desarrollador

8.3.1. Introducción

Este manual está orientado a explicar el funcionamiento de la biblioteca Pulse por si se desea ampliar las funcionalidades de Impulse haciendo una subdivisión del proyecto o si se desea conocer más acerca del funcionamiento del sistema.

8.3.2. Compilación estática en Windows

Si se precisa una compilación estática, además de las librerías OpenSSL (para la conexión SSL que se realiza para verificar el canal de actualizaciones), se debe de recompilar toda la biblioteca QT. Esto es debido a que las únicas librerías precompiladas libres que el proyecto QT posee sólo pueden compilar ficheros con librerías dinámicas.

Para ello lo primero que se debe de compilar es OpenSSL con MSYS2 y el fichero "OpenSSL_1.1.1l.sh" que está en el proyecto.

Posteriormente se deberá instalar "ActiveState" con Perl, Python y Ruby (lenguajes necesarios para interpretar las librerías QT), además de 7zip, que se encargará de descomprimir los ficheros descargados. Una vez hecho todo esto, se deberá ejecutar "QT_Static_5.15.2.ps1"

y, tras un largo periodo de compilación, ya se tendrían las librerías estáticas necesarias. Ahora sólo queda agregarlas, como cualquier otra biblioteca, al IDE de QT para que funcione correctamente.

No se recomienda generar un binario estático en QT para distribuciones GNU/Linux, son más pesados y muy inestables, debido a que las bibliotecas GNU/Linux se adaptan al kernel interno de la distribución y, si se entrega un binario estático, pueden darse incompatibilidades y cierres inesperados del programa.

Además de que GNU/Linux tiene capacidad de utilizar una misma biblioteca para múltiples aplicaciones, cosa que Windows no integra. Por lo que se estaría sobreutilizando disco duro para algo que no es funcional para el sistema operativo.

8.3.3. Acerca de la biblioteca Pulse

La biblioteca Pulse, de cara al control de otras aplicaciones, está orientada a la administración de una serie de tareas. Se pueden integrar tareas aplicando la interfaz de usuario ya descritas en los puntos "Utilización de la biblioteca" tanto en Escritorio 6.1.5 como en móvil 6.3.4.

8.3.4. Depuración de la aplicación

Impulse también incluye una interfaz gráfica dedica a la depuración pero de señales. Permite mostrar la intensidad de un campo magnético y analizarla detenidamente.

Por un lado la aplicación móvil se encarga de borrar datos, mostrar una gráfica junto a una estadística y almacenar datos para ser compartidos y analizados a través de la herramienta de escritorio. Para acceder a el modo desarrollador se debe de pulsar 10 veces sobre el indicador de versión de ajustes. Una vez se ha entrado, con pulsar únicamente 1 vez sobre la versión se abrirá.

Si se pulsa sobre el símbolo de guardado se almacenarán todos los datos que se han recibido desde el inicio (o desde que se pulsó el icono de borrar) en un fichero con nombre de número que indica la marca de tiempo UNIX y con extensión *.csv. Estará guardado en la subcarpeta "Debug" del lugar de almacenamiento seleccionado en el tutorial.

Por otro lado el programa de escritorio analiza los datos conseguidos a través del modo desarrollador del móvil, detenidamente. Además exporta gráficas en formato *.png, lo que permite hacer referencias más cómodamente a las lecturas que se produjeron. Incluye

información del móvil, como la versión de Android, estadísticas, versión del sensor,...

Para acceder al modo desarrollador se debe de entrar en "Acerca de" del programa de escritorio y pulsar 10 veces sobre el icono de Impulse de la ventana.

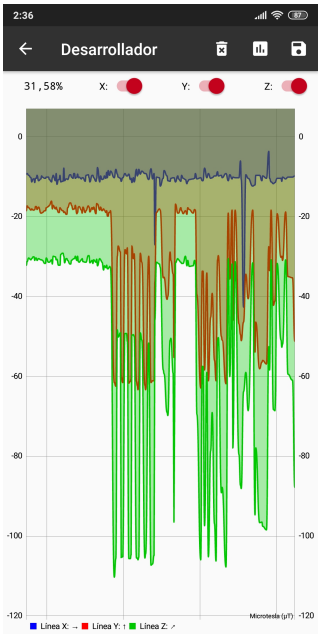


Figura 8.5: Modo desarrollador de aplicación

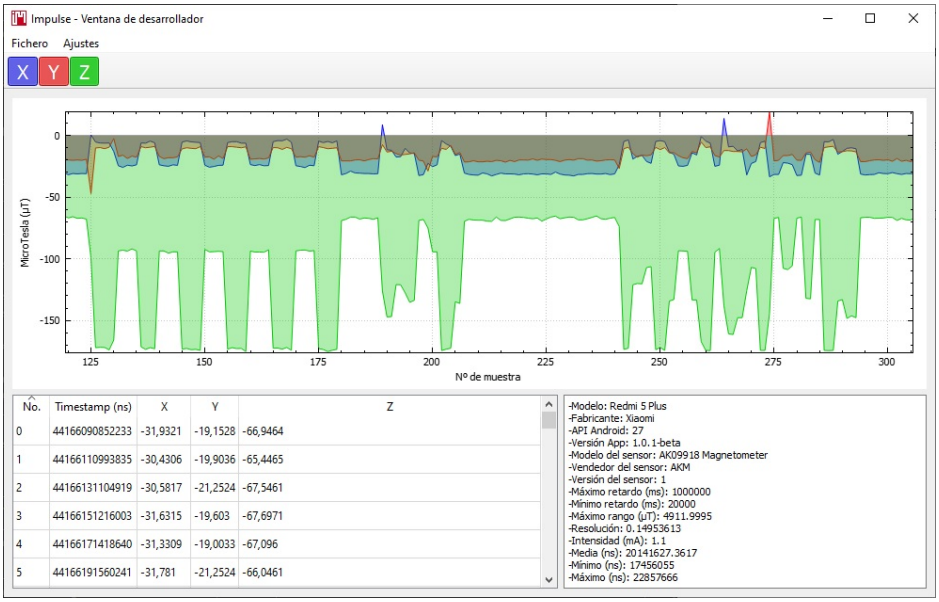


Figura 8.6: Modo desarrollador del Escritorio

Parte IV

Conclusiones

Capítulo 9

Conclusiones

9.1. Trabajo futuro

Con el fin de optimizar tanto el hardware como el software, se podrían considerar las siguientes mejoras:

- Encontrar una mejor manera de obtener un mayor apantallamiento entre los solenoides o, en su defecto, algún método de comunicación multicanal que sí que lo permita: Esto conllevaría que se puede enviar más información en un mismo periodo de tiempo.
- La placa PCB y el diseño 3D podría mejorarse notablemente: La placa PCB y el diseño 3D podría mejorarse notablemente.
- Comunicación full-duplex: Es interesante, desde el punto del móvil, poder enviar información, en vez de sólo recibirla. La comunicación full-duplex abriría la posibilidad de implementar nuevas tareas.
- Mejorar la interfaz bluetooth: La interfaz bluetooth depende de APIs de la seguridad de APIs de terceros. Se tiene poco control sobre lo que se puede hacer a bajo nivel y pueden darse problemas de seguridad, especialmente a nivel de conexión o de cifrado.
- Buscar una mayor independencia de cara a los frameworks en las biblioteca Pulse: La biblioteca debería poder funcionar correctamente sin frameworks de terceros.
- Lecturas de los campos magnéticos controladas: Android no permite realizar lecturas de campos magnéticos en los periodos que se desee. Es la API Android la que se

encarga entregar las lecturas magnéticas cuando considere correcto. Para hacer esto se requiere que la aplicación sea una aplicación de sistema, con permisos especiales.

- Poder realizar lecturas con la pantalla apagada. Android pone cualquier aplicación que se esté ejecutando de fondo en ahorro de energía. Además da por hecho que, cuando se está ejecutando de fondo la aplicación, no es preciso recibir tantas lecturas de campos magnéticos. Al igual que el punto anterior, se requiere que la aplicación sea de sistema para poder tener permisos especiales.

9.2. Conclusiones personales

El desarrollo de este proyecto ha sido un proceso arduo que ha demandado dedicación y compromiso para mejorar tanto el hardware como el software. Ha implicado la adquisición de conocimientos en áreas técnicas especializadas, como la electrónica y los estudios sobre cuerpos finitos, con el objetivo de fortalecer la capacidad de diseño y optimización de nuestro sistema.

La inversión de tiempo ha sido uno de los aspectos más notables de este proyecto, ya que se han dedicado horas de trabajo exhaustivas para investigar, diseñar, probar y ajustar el hardware y software. Se ha llevado a cabo una formación continua y rigurosa para adquirir habilidades técnicas y conocimientos actualizados que permitan aprovechar al máximo el potencial del sistema.

Además, la parte económica del proyecto ha sido un factor relevante. Se han realizado inversiones significativas en la adquisición de equipos, herramientas y tecnologías necesarias para el desarrollo del hardware y software mejorado. También se han destinado recursos económicos para la formación y capacitación del equipo de trabajo, así como para la realización de pruebas y análisis exhaustivos para asegurar la calidad y el rendimiento del sistema se han adquirido componentes en abundancia dado que su escasez en España se han tenido en otros países, lo que conlleva un sobrecoste por envío y por demora en la entrega.

Observando el prototipo final, experimento una gratificante sensación de satisfacción en relación a los esfuerzos invertidos en el ámbito de estudio. Debido a mi jornada laboral, no he podido disponer del tiempo adecuado que hubiera deseado para llevar a cabo el trabajo con la plena disponibilidad temporal que habría sido deseable.

Parte V

Apéndices

Apéndice A

Glosario de términos

- Arduino: Plataforma de prototipado electrónico de código abierto, conformada por una serie de placas de circuitos integrados programables y un entorno software fácil de usar.
- ASK (Amplitude-shift keying): Representación analógica de la información digital a través de la amplitud de onda que se emita.
- Cable Jumper: Cable con conectores utilizados para conectar componentes en una Protoboard A.
- Código Gray: Escala binaria dónde los números consecutivos sólo les separa 1 bit de distancia.
- CRC (Código de Redundancia Cíclica): Identifica si la información se ha alterado a la hora de ser transmitida.
- DAC (Digital to Analogic Conversor): Dispositivo que se encarga de convertir la información de digital (datos binarios) a analógico (corriente o tensión).
- Dispositivo: Hace referencia al hardware diseñado, debido a que no tiene nombre, se usará unicamente para hacer referencia a él.
- Impulse: Aplicación que permite interactuar al usuario con la biblioteca Pulse, dotándole de una interfaz gráfica.
- MAC: Dirección física de una tarjeta de red. Incluye tanto tarjetas wifi o tarjetas bluetooth.
- MiTM (Man in The Middle): Ataque que modifica los mensajes entre emisor y receptor a voluntad.

- PCB (Printed Circuit Board): Placa de circuito impreso. Formada por pistas laminadas conductoras de, generalmente, cobre que permiten interconectar componentes electrónicos y separadas por material aislante.
- Protoboard (Placa de pruebas): Placa de plástico con agujeros interconectados por conductores metálicos internos que permiten insertar los componentes electrónicos
- Pulse: Librería del protocolo
- Pulso: Campo magnético en un periodo de tiempo de 50 milisegundos.
- Pulso doble: Campo magnético en un periodo de tiempo el doble de largo que un pulso normal. Es decir, de 100 milisegundos.
- RS (Reed-Solomon): Determina si se han producido errores en la transmisión e intenta repararlos. Si el bloque tiene demasiados errores no podrá ser reparada. Se puede dar que la reparación del bloque no sea la correcta.
- Wallet o Monedero electrónico: Dirección electrónica dónde se almacenan las distintas criptomonedas.

Apéndice B

Contenido adjunto

Junto a la memoria del proyecto desarrollado, se anexarán los siguientes directorios:

- Hardware: Incluye todo lo relacionado con el dispositivo. Contiene los siguientes subdirectorios:
  - Board: Incluye todo lo necesario para elaborar tu propia PCB.
  - Case: Carcasa del dispositivo.
  - mbedOS: Software que se deberá de cargar en el dispositivo basando en mbedOS.
- Icons: Iconos Impulse en los distintos formatos (SVG, PNG e ICO), además de la separación del icono entre el primer plano y el fondo.
- Receiver: Código del receptor.
- Transmitter: Código del transmisor.
- Releases: Compilaciones receptor y transmisor. Incluye una carpeta con las versiones completas compiladas.

Estos directorios se compartirán a través del repositorio proporcionado por la Escuela de Ingeniería Informática de Segovia.

Apéndice C

Licencias utilizadas

- Proyecto Impulse (este proyecto):
  - Pulse: Licencia GPLv3.
  - Impulse: Licencia GPLv3.
- Bibliotecas Android: Utilizadas para realizar la aplicación Android.
  - MPAndroidChart: Licencia Apache 2.0, disponible en `https://github.com/PhilJay/MPAndroidChart`
  - Android ripple pulse animation: Licencia Apache 2.0, disponible en `https://github.com/gaurav414u/android-ripple-pulse-animation`
  - Smashicons: Licencia Flaticon License, disponible en `https://smashicons.com/`
  - Android Open Source Project: Licencia, mayoritariamente (ya que es un compendio de múltiples licencias para cada artefacto) Apache 2.0, disponible en `https://source.android.com/`
- Bibliotecas QT:
  - QCustomPlot: Licencia GPLv3, disponible en `https://www.qcustomplot.com/`
  - QT Framework: Licencia LGPLv3, disponible en `https://www.qt.io/`
- Otras bibliotecas:
  - Reed-Solomon: Sin licencia, libre, de dominio público, disponible en `https://github.com/tomerfiliba/reedsolomon`
  - BLESSED for Android: Licencia MIT License, disponible en `https://github.com/weliem/blessed-android`

Bibliografía

[1] Ángel Antonio Bayod Rújula. *Principios de electromagentismo*. ed. II. Colección textos docentes. PUZ (Prensas Universitarias de Zaragoza), 2002. ISBN: 94-7733-582-6.

[2] Richard E. Blahut. *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003. DOI: 10.1017/CBO9780511800467.

[3] C. K. P. Clarke. *Reed-Solomon error correction*. R&D White Paper. British Broadcasting Corporation (BBC), ene. de 2023. URL: https://www.bbc.co.uk/rd/publications/whitepaper031 (visitado 03-08-2022).

[4] Weiwei Jiang et al. «Pulse: Low Bitrate Wireless Magnetic Communication for Smartphones». En: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. New York, NY, USA: Association for Computing Machinery, 2014, 261–265. ISBN: 9781450329682. DOI: 10.1145/2632048.2632094. URL: https://doi.org/10.1145/2632048.2632094.

[5] Capers Jones. *Programming Languages Table*. Ed. por Ball State University. URL: http://www.cs.bsu.edu/homepages/dmz/cs697/langtbl.htm (visitado 16-03-2023).

[6] Jooble.org. *Buscador de salarios medio de puestos de trabajo y ubicaciones*. URL: https://es.jooble.org/salary/ (visitado 04-04-2023).

[7] Henrik Kniberg. *Dar sentido a PMV (Producto Mínimo Viable) y por qué prefier un temprana Testabilidad / Usabilidad / Deseabilidad*. 25 de ene. de 2016. URL: https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp (visitado 07-08-2022).

[8] Héctor Mauricio Molina Semblantes Luis Eduardo ; Velasteguí Masapuncho. *Diseño y construcción de un conversor AC/DC no controlado de 3, 6, 9, 12 y 24 pulsos, y analizador de armónicos a través de una PC*. Lataguna - Ecuador: Escuela Politécnica del Ejército, 2007. URL: https://repositorio.espe.edu.ec/bitstream/21000/4208/1/T-ESPEL-0433.pdf (visitado 05-01-2023).

[9] Julio Palacios. *Electricidad y magnetismo*. 2ª ed. corr. y aum. Madrid: Espasa-Calpe, 1959.

[10] Fernández Perez Cruz. *Técnicas de protección frente a errores*. Inf. téc. MMC (UC3M). URL: https://www.tsc.uc3m.es/~fernando/Codificacion_canal.pdf (visitado 08-08-2022).

[11] Daniel Pardo Collantes; Luis A. Bailón Vega. *Fundamentos de electrónica digital*. ed. I. Ediciones Universidad de Salamanca, 2006. ISBN: 84-7800-401-7.