industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Dispositivo inalámbrico para señalizar sucesos a personas con discapacidad

TRABAJO FIN DE GRADO

GRADO EN INGENIERIA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Mª Teresa Torres Fernández**
Director: José Carlos Fernández Andrés

Cartagena, septiembre 2021

Universidad Politécnica de Cartagena

AGRADECIMIENTOS

A mi familia, por su apoyo durante mis años de carrera.

A mis compañeros de IEEE, y en especial a Mariano, por su inestimable ayuda y años de proyectos en común.

A Arturo, por aportar su visión cuando más lo necesitaba.

A mis compañeros de la carrera, por todos esos recuerdos compartidos.

RESUMEN

En los últimos años, la tecnología ha sido un factor crucial para mejorar las condiciones de vida de las personas con diversidad funcional, permitiéndoles su integración sin barreras en el resto de la sociedad. Sin embargo, la incorporación de diversos colectivos al mundo laboral aún sigue siendo un terreno con amplio margen de mejora, dado que en muchas ocasiones no tienen acceso igualitario a una gran mayoría de puestos de trabajo.

El presente proyecto busca investigar y desarrollar una solución que ayude a la correcta contribución de personas con discapacidad auditiva en los entornos industriales, con un énfasis en la salud y seguridad de dichos individuos. En particular, se implementará un sistema de alarma inalámbrico adaptado a las necesidades de este colectivo.

ABSTRACT

In recent years, technology has been a crucial factor in improving the living conditions of people with functional diversity, enabling their barrier-free integration into the rest of society. However, the incorporation of various groups into the labor market is still a field with a wide margin for improvement, since in many cases they do not have equal access to a large majority of jobs.

This project aims to research and develop a solution that helps the correct contribution of hearing-impaired people in industrial environments, with an emphasis on the health and safety of such individuals. In particular, a wireless alarm system adapted to the needs of this group will be implemented.

1. ÍNDICE

1 INTRODUCCIÓN

Hoy en día, la tecnología ofrece infinitas posibilidades a los seres humanos, aumentando enormemente nuestras capacidades. Ciertas tecnologías han permitido alcanzar avances que antes parecían meramente ficción, tales como los semiconductores, las comunicaciones inalámbricas o Internet. De esta forma, ya no solo las grandes empresas pueden colaborar en esta bonanza tecnológica, sino también los individuos tienen a su alcance una gran capacidad de llevar a cabo sus ideas y aspiraciones.

Así mismo, existe una gran variedad de dispositivos genéricos que permiten un fácil prototipado de los más diversos diseños. Plataformas como Arduino o Raspberry Pi facilitan enormemente el ciclo de desarrollo, lo cual facilita un mayor grado de experimentación y desarrollo y validación de ideas. De esta forma, la innovación resulta más sencilla que nunca.

Por otro lado, la inclusión de los dispositivos IoT en nuestra vida diaria es cada vez más patente, abriendo un amplio abanico de posibilidades. En concreto, los *wearables* permiten una simbiosis nunca antes vista entre tecnología e individuo.

Sin embargo, este auge no siempre está guiado por las necesidades sociales, y pueden existir colectivos que no se benefician de igual manera de estas mejoras en la calidad de vida obtenidas gracias a la tecnología.

En concreto, existe un amplio margen de mejora en cuanto a la utilización de métodos tecnológicos para facilitar la vida de las personas con diversidad funcional, las cuales se ven comúnmente avocadas a situaciones laborales inciertas o poco flexibles, desarrollando tareas que pueden estar por debajo de su potencial real.

La tecnología disponible puede ayudar a solventar esta situación, y conseguir una igualdad de posibilidades que nunca antes hubiera sido posible.

Es por ello que el presente proyecto se centra en aportar en ese sentido, materializándose en el prototipado de un dispositivo de alarma que permita aumentar la seguridad y correcta integración de personas con discapacidad auditiva en un entorno laboral adverso, como lo es el industrial.

1.1 MOTIVACIÓN Y OBJETIVOS

La motivación de este proyecto nace en parte de la experiencia personal de la autora. Debido a diversos factores genéticos, diversos miembros de su familia sufren de una pérdida de audición, que en algunos casos ha supuesto un obstáculo para el normal desarrollo de su actividad social y profesional. Si bien el uso de audífonos palia enormemente este fenómeno, aún existen dificultades que superar y que a día de hoy no cuentan con una solución completa.

En particular, las observaciones realizadas durante la visita a un entorno industrial, la cual se describe en secciones posteriores, fueron el germen de la idea a desarrollar en el presente proyecto.

En base a ello, los objetivos principales son:

1. Investigar las soluciones tecnológicas actuales referentes a la correcta integración de personas con discapacidad auditiva en entornos industriales.
2. Diseñar, programar y prototipar un dispositivo que notifique a su usuario de potenciales riesgos presentes en el entorno industrial, de una manera adaptada a las capacidades sensoriales de dicho usuario.

1.2 ESTADO DEL ARTE

Actualmente, existen numerosas soluciones de seguridad doméstica para personas con discapacidad auditiva o sensorial. Algunos ejemplos comunes de estos dispositivos pueden ser timbres o alarmas de incendios que emitan luz intensa o vibración, despertadores adaptados, alarmas con diferentes frecuencias o sistemas inteligentes que recogen diferentes sonidos de la casa y los retrasmiten a través de aplicaciones de móvil, etc.

Según la Ley 13/1982 de Integración Social de los Minusválidos, por la que todas las empresas públicas o privadas españolas, cuya plantilla sea superior a los 50 trabajadores, deben tener una "cuota de reserva a favor de las personas con discapacidad" de un 2% de la plantilla, es decir, que este tanto por ciento esté reservado para personas con una discapacidad igual o mayor al 33%.

Es por esto que encontramos un gran número de trabajadores en ámbitos labores con alguna discapacidad, concretamente, auditiva o sensorial. Es

sorprendente que con todos los avances tecnológicos y nuevos dispositivos que ofrece el mercado cada día para estos usuarios a nivel personal/domestico, no se vea reflejado en la mayoría de sus lugares de trabajo.

En el sector industrial no encontramos tantas adaptaciones para los trabajadores con discapacidad, ya que, por ejemplo, casi en la totalidad industrias/hospitales/edificios comerciales las alarmas de seguridad por incendios son las clásicas que emiten un sonido, también llamadas "sirenas" o "campanas" que algunas veces van acompañadas de avisos luminosos. Estas no siempre son captadas por personas con déficit sensorial, ya sea sordera o ceguera.

Es por esto que se encuentra la necesidad de seguir planteando adaptaciones y comodidades para todos los tipos de trabajadores en sus diferentes puestos de trabajo, añadiendo de esta manera seguridad y calidad al entorno laboral.

1.3 ESTUDIO PRÁCTICO

Meses atrás se comprobó de forma práctica esta falta de adaptación que antes se detallaba en el apartado anterior.

Se realizó una visita a una de las sedes de lavanderías industrial de ILUNION en Madrid. Esta empresa tiene un modelo empresarial único puesto que el 40% de personal contratado tiene algún tipo de discapacidad (visual, auditiva, física o psíquica). Esta sede se encarga de limpieza, lavado, esterilización y planchado de textil hospitalario, para ello se utilizan voluminosas máquinas de producción, lavadoras y planchas industriales las cuales se caracterizan por el abundante ruido acústico que generan a la vez que altas temperaturas.

En dicha visita se pudo comprobar que las alarmas de incendios estaban colocadas en diversas columnas de la fábrica y eran de tipo campana acompañadas de algunos avisos luminosos.

Pudiendo concluir con esto que este tipo de medidas de seguridad no encajaban acorde al personal trabajador, encontrando de esta manera la motivación de desarrollar mejores soluciones de seguridad y adaptación a la industria.

2 TECNOLOGÍAS DE RADIOCOMUNICACIÓN

2.1 COMUNICACIONES INALÁMBRICAS

Las comunicaciones inalámbricas, también llamadas comunicaciones sin ningún tipo de cables, son aquellas donde se encuentra un emisor y uno o varios receptores. Estos no se encuentran vinculados a través de difusión física, sino que se realiza el uso de modulación de ondas electromagnéticas por medio de un área determinada.

En sentido general, siempre que haya un emisor y un receptor que transmitan información a través de ondas de radio y con diferentes distancias de transmisión, se denomina comunicación inalámbrica.

En la actualidad, con el desarrollo continuo de la tecnología de la información y las comunicaciones, la aplicación de la tecnología de las comunicaciones inalámbricas de corto alcance se está acelerando y madurando cada vez más.

Las tecnologías inalámbricas que se ven hoy en día tienen sus propias características o requisitos especiales, diferenciados en la velocidad de transmisión, la distancia y el consumo de energías, ancho de banda, entre otros.

A la hora de transmitir datos, se recurre a muchas tecnologías inalámbricas. Cada una de ellas presenta una serie de ventajas e inconvenientes que las hacen tener mayor o menor validez según los requisitos deseados en la aplicación.

Algunas de estas tecnologías son:

2.1.1 Bluetooth

Se corresponde con un estándar de comunicaciones inalámbricas basado en radiofrecuencia, de bajo coste y bajo consumo energético. Originariamente, la tecnología bluetooth se desarrolló como un mecanismo alternativo que permitiese sustituir los enlaces cableados de diversos periféricos. No obstante, las características y versatilidad que presenta dicha tecnología han hecho que se pueda utilizar en gran cantidad de situaciones diferentes, como pueden ser el establecimiento de conexiones entre dos terminales móviles inteligentes, concretamente para la comunicación inalámbrica de datos y voz.

La tecnología bluetooth se caracteriza por:

- Operar en la banda de frecuencia libre de los 2,4 GHz
- Tiene una capacidad máxima de transmisión de hasta 3 Mbps.
- Implementa diversos mecanismos de ahorro energético de forma que el dispositivo no siempre va a consumir la misma potencia con el consiguiente ahorro energético en la batería del dispositivo.
- Alcance de hasta 100 metros en función de la potencia de emisión que posea el transmisor Bluetooth.

Aunque la tecnología Bluetooth es globalmente utilizada, generalmente los usuarios no eligen este tipo de comunicación porque la distancia de transmisión no es muy alta y existen dificultades de seguridad con respecto a la información que se transmite.

2.1.2 Wi-Fi

La tecnología Wi-Fi es una extensión inalámbrica de Ethernet. En teoría, siempre que el usuario se encuentre en un área determinada alrededor de un punto de acceso. Pero, de hecho, también puede darse que varios usuarios accedan a través de un punto al mismo tiempo y el ancho de banda será compartido por estos.

Se basa en estándares 802.11 que utiliza frecuencias de radio (RF) para extender las redes de área local (LAN) cableadas basadas en Ethernet a dispositivos habilitados para Wi-Fi, lo que permite a los dispositivos recibir y enviar información desde Internet. El Wi-Fi se puede transmitir en las frecuencias 2,4GHz y 5Ghz.

2.1.3 ZigBee

Se puede decir que ZigBee es el hermano de Bluetooth. Utiliza la banda 2,4GHz y la tecnología de salto de frecuencia. ZigBee se caracteriza por las siguientes características:

- Baja capacidad de transmisión, en torno a 250 Kbps, que nos permitirá desarrollar sistemas de muy bajo coste.
- Protocolo sencillo, pudiendo ser implementado sin ningún tipo de limitación en sistemas microcontroladores de 8 bits.

- Muy bajo consumo energético permitiendo que la fuente de alimentación del sistema pueda durar años.

  Como gran desventaja, podemos mencionar la baja capacidad de transmisión adoptada lo que restringe el uso de esta especificación para usos muy concretos y actividades que requieran poco intercambio de datos, como accionar un interruptor de la luz o monitorizar un sensor de temperatura o luminosidad.

Las comunicaciones inalámbricas, también llamadas comunicaciones sin ningún tipo de cables, son aquellas donde se encuentra un emisor y uno o varios receptores. Estos no se encuentran vinculados a través de difusión física, sino que se realiza el uso de modulación de ondas electromagnéticas por medio de un área determinada.

En sentido general, siempre que haya un emisor y un receptor que transmitan información a través de ondas de radio y con diferentes distancias de transmisión, se denomina comunicación inalámbrica.

En particular las comunicaciones inalámbricas usan ondas de radiofrecuencia de potencia y banda determinada para emitir señal entre aparatos.

2.1.4 Radiofrecuencia

Las comunicaciones inalámbricas usan ondas de radiofrecuencia de potencia y banda determinada para emitir señal entre aparatos.

La Radiofrecuencia, en adelante RF, se aplica para nombrar a ciertas frecuencias del espectro electromagnético. En concreto, este tipo de frecuencias se sitúan entre los 3 Hz y 300 GHz del espectro electromagnético.

Dicho espectro se divide en regiones las cuales están clasificadas por frecuencia y longitud de onda. En ellas encontramos diferentes señales electromagnéticas como los rayos infrarrojos y microondas entre otros.

La radiofrecuencia es la tecnología más común en lo que respecta al control remoto en la industria.

Las bandas de radio industriales pertenecen a la familia de las denominadas ISM, bandas de radio reservadas internacionalmente para el uso de energía de RF para fines industriales, científicos y médicos, las cuales pueden ser de libre licencia.

Generalmente la RF ofrece un alcance mayor que el resto de tecnologías. Por ejemplo, los infrarrojos comparados con RF tienen aplicaciones muchas más limitadas debido principalmente a la necesidad de acercamiento entre los dos puntos y a la cantidad de imprevistos que pueden surgir y que podrían interrumpir la comunicación.

Las señales RF se utilizan en la comunicación inalámbrica debido a su propiedad para penetrar a través de objetos y viajar largas distancias.

Es por todas estas ventajas descritas anteriormente por lo que la RF ha sido elegida para el desarrollo del proyecto, en concreto para la comunicación entre los dos prototipos de módulos que se han creado (brazalete y alarma).

2.2 BANDAS ISM

Como se ha comentado anteriormente, las bandas ISM (siglas en inglés Industrial, Científico y Medico) son definidas como bandas de frecuencias para usos no comerciales con fines industriales, científicos, médicos, domésticos o similares, las cuales no requieren de una licencia para su uso, denominándose por lo tanto "sin licencia" o "license-free". Esto no significa que se puedan utilizar libremente, ya que siguen estando fuertemente regladas, tanto a nivel técnico, como en sus condiciones de uso.

Los organismos reguladores definen las bandas de frecuencias para usos sin licencia:

- Banda PMR446, (446 MHz) regulada por el ETSI para el uso de intercomunicadores de mano (walkies) en el territorio Europeo (CEPT).
- Banda FSR (462 MHz y 467 MHz) regulada por la FCC para el uso de intercomunicadores de mano (walkies) en EEUU.
- Banda LDP433 (433 Mhz) regulada por el ETSI para el uso de intercomunicadores de baja potencia en el territorio Europeo (CEPT).

- Banda CB (27 MHz) o "Banda Ciudadana", que, aunque regulada nacionalmente, su aceptación es prácticamente global.
- Bandas U-NII (5 GHz) "Unlicensed National Information Infrastructure", regulada por la FCC para su uso en redes digitales de nueva generación en EEUU.

La mayoría de los casos estas bandas se comparten, total o parcialmente, con otros usos, como puede ser: radiolocalización, radiobúsqueda, radiobalizas, telemandos, etc. y de forma muy recurrente, con las bandas atribuidas a los radioaficionados con licencia.

Las bandas de radioaficionados cubren todos los segmentos del espectro radioeléctrico, ya que cada segmento posee unas características de propagación y uso particulares, que dependen de diferentes factores atmosféricos y climatológicos. Se clasifican por su longitud de onda al igual que por rango de frecuencia.

Las bandas de radioaficionados no están reservadas exclusivamente para un determinado uso, ya que, comparten su utilización en muchas ocasiones, por lo que no deben causar interferencias perjudiciales.

Destacamos las bandas de 433 MHz, ya que, será esta banda en la que se trabajará con los módulos RF realizados en el proyecto.

2.2.1 Banda 433 MHz

La banda ISM de 433MHz, cohabita en la banda UHF de 70cm de radioaficionados.

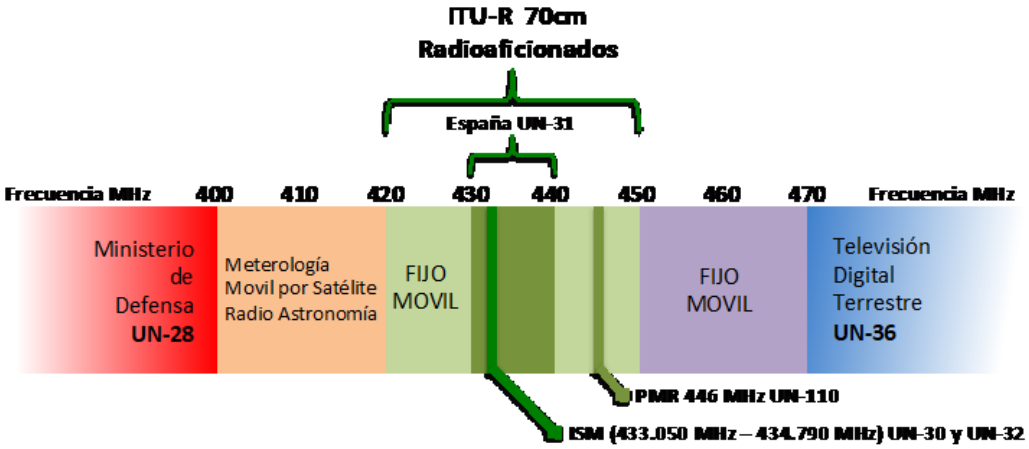

Figura 1: Banda de frecuencias

El ancho de la banda ISM de 433 MHz es, comprende entre 433,050 MHz y 434,790 MHz. Siendo la máxima potencia radiada aparente (PRA) permitida tan solo de 100 mW (milivatios), en comparación con los hasta 1000 W (vatios) de potencia isotrópica radiada equivalente (PIRE), a los que pueden llegar los radioaficionados en algunos casos.

Las diferentes aplicaciones que hacen uso de la banda ISM de 433 MHz pueden dividirse en dos grandes grupos en función del tipo de modulación que utilicen, AM o FM.

2.3 MODULACIÓN

Llamamos modulación al conjunto de técnicas que se usan para transportar información sobre una onda portadora, usualmente una onda senoidal. Estas técnicas permiten un mejor uso del canal de comunicación lo que posibilita enviar más información en forma simultánea además de actuar contra posibles ruidos e interferencias.

Elementos que intervienen en el proceso de modulación:

- Señal portadora: Señal periódica (generalmente senoidal) encargada de "transportar" la información que queremos transmitir, y cuya frecuencia de transmisión es la deseada.
- Señal moduladora: Señal que representa el mensaje que queremos que sea enviado, y cuya frecuencia en general no será la frecuencia de

transmisión deseada. Por eso, esta señal modificará algunos de los parámetros de la onda portadora.

- Señal modulada: Señal resultante generada por la modulación de la señal portadora por la señal moduladora.

Este consiste en la modificación de determinados aspectos de la onda portadora con respecto a otra onda moduladora, la cual reflejará la información que queremos transmitir. Generando, al final, del proceso de modulación una onda modulada.

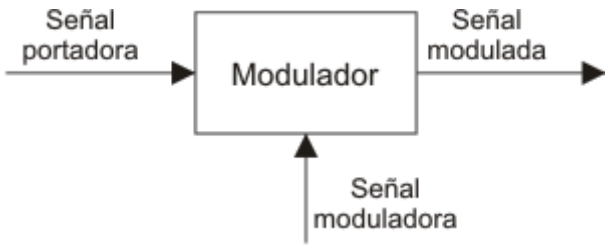

Figura 2: Figura modulación

Con el proceso de modulación intentamos conseguir una mayor facilidad de propagación de la información que deseamos. Las principales ventajas que encontramos al realizar este proceso son:

- Evitan la presencia de interferencias entre canales, ya que si mandásemos la información a la frecuencia de la onda portadora encontraríamos problemas al recibir la señal de varios usuarios.
- También protegemos que nuestro mensaje no se distorsione por presencia de ruido de manera que esto aumenta su calidad.
- Disminuye el tamaño de las antenas que utilizamos para poder enviar estas señales

2.3.1 Tipos de modulaciones

Podemos clasificar los tipos de modulación respecto a dos criterios: Tipo de señal (analógica o digital) y parámetros de la onda portadora que son modificados

(amplitud, frecuencia o fase). Según estos criterios encontramos los siguientes tipos de modulación:

	Moduladora analógica	**Moduladora digital**
P. Analógica	AM (Amplitude Modulation) FM (frecuency Modulation) PM (Phase Modulation)	ASK (Amplitude Shift Key) FSK (Frecuency shit Key) PSK (Phase shift Key)
P. Digital	PAM (Pulse Amplitude Modulation) PDM (Pulse Duration Modulation) PPM (Pulse Position Modulation) PCM (Pulse Codification Modulation)	NRZ (Non Return to Zero) RZ (Return to Zero) Bifase Bipolar

Tabla 1: tipos de modulación

- Señal portadora y moduladora son analógicas:
  - Modulación en Amplitud (**AM**): este tipo de modulación generará una señal la cual su frecuencia se mantiene fija y la amplitud variará respecto a la señal moduladora.
  - Modulación de frecuencia (FM): en este modo la forma de transmitir información será variando la frecuencia de la señal portadora. La propagación de estas dependerá de la amplitud de la señal moduladora.
  - Modulación de fase (PM): esta también es llamada modulación lineal ya que la señal modulada es una combinación lineal de la señal portadora y moduladora.

- Señal portadora analógica y moduladora digital:
  - Modulación por desplazamiento de amplitud (**ASK**): este tipo de modulación es un caso particular de AM. En esta, se modula la amplitud y la señal moduladora es digital. Encontramos una señal portadora senoidal de alta frecuencia y una señal moduladora digital con valores binarios (0 y 1) que representan la información

que queremos enviar. Los valores aparecerán con dos amplitudes diferentes: uno de los dígitos se representará a amplitud constante (la de la portadora) y en el otro la amplitud será cero, lo que podemos denominar como ausencia de señal portadora. La señal modulada resultará:

$$V(t) = \{V_p\, sen(2\pi f_p t) \quad cuando\ "1"\ binario\ 0 \qquad\qquad cuando\ "0"\ binario$$

Este proceso lo podemos visualizar de la siguiente forma:

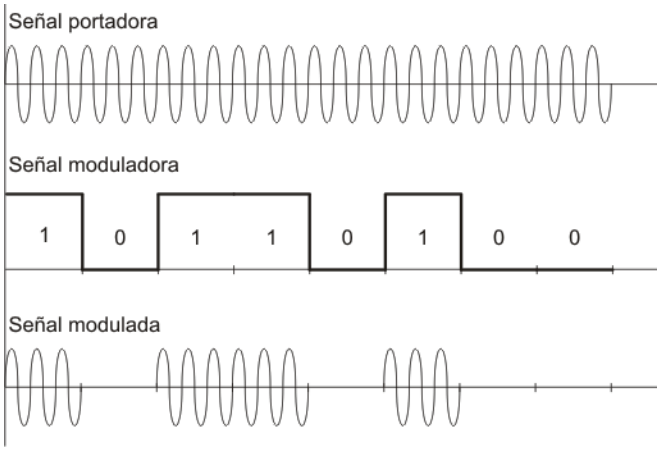

Figura 3: Señal ASK

En esta modulación sólo se puede transmitir un solo bit al mismo tiempo en una determinada frecuencia.

- Modulación por desplazamiento de frecuencia (FSK): esta técnica de modulación digital consiste en la variación de la frecuencia en la señal portadora. Siendo la señal digital moduladora un flujo de pulsos binarios y cuyos valores serán ± la frecuencia portadora, sin variación de amplitud ni de fase.

  Un ejemplo de representación de la señal modulada sería:

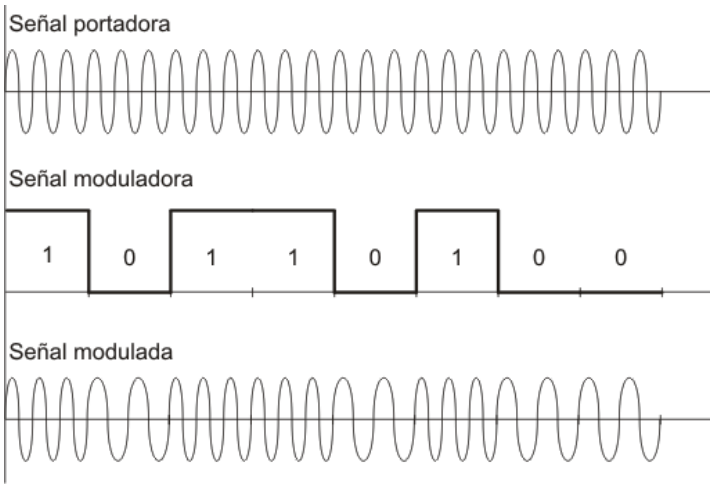

Figura 4: Señal FSK

La modulación FSK se utiliza sobre todo para transmisión de datos full-duplex sobre líneas telefónicas.

- Modulación por desplazamiento de fase (PSK): esta modulación se caracteriza por el cambio de fase de la onda portadora de amplitud y frecuencia constante, ocasionado por una señal moduladora binaria. La
  forma más simple de modulación PSK se denomina BPSK. En este caso, la señal tendrá un valor de desfase de 0 grados cuando el nivel lógico sea "1" y de 180 grados cuando este sea "0". Representándose de la siguiente manera:

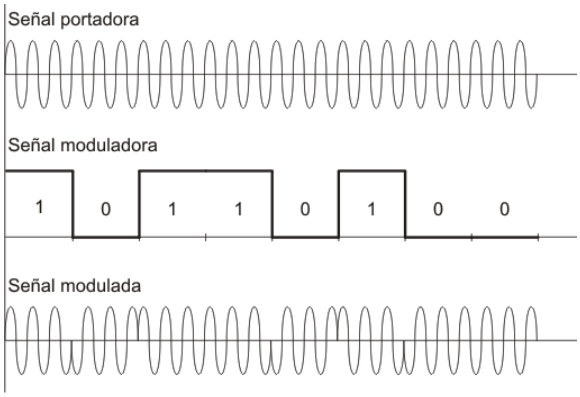

Figura 5: Señal PSK

- Señal portadora digital y moduladora analógica: En estos casos tendremos como señal portadora un tren de pulsos digitales y todas ellas utilizan el proceso de muestreo para conseguir la modulación requerida.
    - Modulación por amplitud de pulsos (PAM): Este tipo de modulación es la consecuencia inmediata del muestreo de una señal analógica. En él, la amplitud de cada pulso es proporcional a la amplitud de la señal moduladora. Se obtendrá una onda modulada con valores discretos a intervalos específicos.
    - Modulación por duración del pulso (PDM o PWM): En este tipo de modulación la amplitud se emplea para variar la anchura o duración de los pulsos.
    - Modulación por posición de impulsos (PPM): En este tipo de modulación, la señal moduladora produce una variación en el instante donde se produce el pulso.
    - Modulación por codificación de impulsos (PCM): Esta modulación consiste en codificar en binario el valor de los pulsos codificados mediante modulación PAM.

- Señal portadora y moduladora digitales: esta modulación también se denomina recodificación y generará una señal digital mediante distintos niveles lógicos.

En conclusión, siguiendo el hilo del presente proyecto y según el uso tradicional de la banda ISM de 433 MHz destacamos la modulación de amplitud de onda (AM), y concretamente en el modo por desplazamiento de amplitud "ASK" (Amplitude Shift Keying).

2.4 MODOS DE TRANSMISIÓN

Los sistemas de transmisión se pueden clasificar según la direccionalidad del flujo de señal entre dos dispositivos conectados. Hay tres modos de transmisión:

- Simplex: en este modo la transmisión es unidireccional, es decir, que solo es posible la transmisión en un sentido. Por lo que el emisor solo puede

enviar datos y el receptor solo podrá recibirlos. Por lo que encontramos el problema de que el emisor no tiene forma de saber si los datos enviados han sido recibidos correctamente. Un ejemplo de transmisión simplex es la señal que se envía de una estación de TV a la TV de su casa.

- Semidúplex (half – duplex): este otro modo permite la transmisión en dos direcciones distintas, pero no al mismo tiempo, es decir, que el emisor y el receptor pueden actuar de receptor y emisor respectivamente pero no a la misma vez. Un ejemplo son las conversaciones de radioaficionados. Cuando presiona el botón del micrófono para hablar no puede escuchar a la persona del otro extremo. De hecho, si estas dos personas intentan hablar a la vez no se produce ninguna de las transmisiones.

- Dúplex (full – duplex): en este caso encontramos al emisor y al receptor emitiendo y recibiendo simultáneamente. Es posible la transmisión en ambas direcciones y de forma simultánea. Podemos decir que esta forma de trabajo es la más eficiente. Un claro ejemplo de full-duplex son las conversaciones telefónicas, ambas personas pueden hablar y escuchar al mismo tiempo.

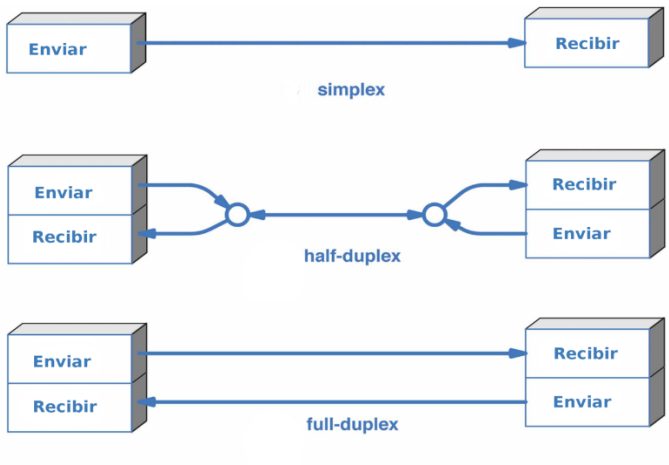

Figura 6: Esquema modos de transmisión

Concretamente para la realización de las conexiones entre los dos módulos del proyecto se usará el modo de transmisión Simplex, es decir, unidireccional ya

que, el módulo que solo podrá enviar datos será la alarma al brazalete del usuario.

3 MICROCONTROLADORES Y MICROORDENADORES

3.1 ARDUINO

Arduino es una plataforma de prototipos electrónica de código abierto (OpenSource) basada en hardware y software libre, flexibles y fáciles de utilizar para programadores y desarrolladores. Esta plataforma permite la creación de varios tipos de microcontroladores en una sola placa. Está pensado para artistas, diseñadores o para cualquiera interesado en crear objetos o entornos interactivos de diferentes usos.

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos y aplicaciones diferentes, desde objetos cotidianos hasta complejos instrumentos científicos. Nació como una herramienta fácil para la creación de prototipos y tuvo una gran acogida en la comunidad mundial de creadores. Esto ha permitido que las placas de Arduino se actualicen para adaptarse a las nuevas necesidades. Existiendo una cantidad de repositorios accesibles que pueden ser de gran ayuda tanto para los principiantes como para los expertos.

Arduino tiene todos los elementos necesarios para que se pueden conectar periféricos a las entradas y salidas del microcontrolador. Vemos reflejado esto en el entorno mediante la recepción de entradas desde una variedad de sensores (sensores, pulsadores, regulación de potenciómetro, etc), que son los que se encargan de convertir las señales de medios físicos a señales eléctricas (pueden ser entradas digitales o analógicas). Estas serán las que lleguen al Arduino y serán procesadas, dando paso a que actúe la programación previamente cargada y, como consecuencia, dará las ordenes pertinentes a los puertos de salida. El objetivo de los periféricos de salida se verá reflejado, por ejemplo, en el control de luces, motores, servomotores entre otros.

Arduino simplifica el proceso de trabajo con microcontroladores, y ofrece algunas ventajas en comparación con otros sistemas:

- Económico: Las placas Arduino son relativamente baratas comparadas con otras plataformas de microcontroladores. La versión menos costosa del módulo Arduino puede ser incluso ensamblada a mano.

- Multiplataforma: El software Arduino (IDE) funciona en los sistemas operativos Windows, Macintosh OSX y Linux, por lo que tiene una gran versatilidad.

- Entorno de programación sencillo y claro: El Software Arduino (IDE) es fácil de usar para los principiantes, pero lo suficientemente flexible para que los usuarios avanzados también puedan aprovecharlo.

- Software de código abierto y extensible: El software de Arduino se publica como herramientas de código abierto, esto quiere decir, que el código es accesible para cualquiera pudiendo modificarlo y utilizarlo para el fin que se quiera. Arduino ofrece la plataforma Arduino IDE (Entorno de Desarrollo Integrado), que es un entorno de programación el cual nos facilita interactuar con la placa, de este modo se puede ampliar el lenguaje a través de las bibliotecas de C++ creando así aplicaciones para dichas placas de Arduino. De esta forma cada creación puede tener muchas utilidades diferentes.

- Código abierto y hardware extensible: Los diseños y especificaciones de las placas Arduino se publican bajo una licencia Creative Commons de acceso libre. Por lo que, Arduino ofrece las bases para que cualquier persona o empresa pueda hacer su propia versión del módulo, ampliándolo y mejorándolo para diferentes usos requeridos.

Figura 7: Logo Arduino

3.2 ESP32

El ESP32 es un SoC (System on Chip) diseñado por la compañía china Espressif y fabricado por TSMC. Integra en un único chip un procesador Tensilica Xtensa de doble núcleo de 32bits a 160Mhz con máximo de hasta 240Mhz, conectividad WiFi y Bluetooth.

Integra 3 tipos de productos:

- Chips: aquí es donde se encuentro el procesador del ESP32 (ESP32-DOWD y ESP32-D2WD)
- Modulos: incluyen dentro del chip anterior (ESP32-WROOM-32D y ESP32-WROVER-IB). Lo más significativo es que cuentan con una antena imprimida en la placa, esta se usa para las conexiones por wifi y por bluetooth. Estos módulos facilitan la implementación del ESP en otras placas ya que incluyen los pines de entrada y salida listos para soldadura ASMD.

- Placas de desarrollo: estas ya llevan un módulo soldado y facilita mucho su integración en prototipos esto es gracias a los pines diseñados para encajar en una protoboard porque la distancia entre ellos es de 2,54 mm. Incluyen regulador de tensión y controlador serie USB para poder dar alimentación y programarlos desde el mismo conector micro USB.

El ESP32 añade muchas funciones y mejoras respecto a microcontroladores anteriores. Uno de sus puntos fuertes es la capacidad de establecer comunicaciones inalámbricas, estas pueden ser tanto por WiFi como por bluetooth. Además del convencional también incluye bluetooth Low Energy. Esto es una gran ventaja respecto a otros microcontroladores que necesitan de módulos extra para poder comunicarse de manera inalámbrica.

El ESP32 aparte de tener bluetooth aporta un aumento significativo de potencia respecto a microcontroladores anteriores, ya que lleva un procesador de doble núcleo con potencias de hasta 240 MHz. Su predecesor en cambio integra un procesador de un único núcleo de hasta 160 Mhz, es decir, un 50% menos.

Además, incluye un coprocesador de ultra bajo consumo que permite un ahorro de energía máximo cuando no se necesita el procesador principal.

El precio de este es ligeramente superior a otros de bajo coste, pero como se aprecia, con unas prestaciones notables.

Otra mejora que presenta es el aumento de número de pines de entrada/ salida o GPIO. También incluye un convertidor analógico digital de mayor resolución, encriptación por hardware, sensor de temperatura, sensor hall, sensor táctil, reloj de tiempo real (RTC) y más funcionalidades como ethernet MAC o controlador CAN.

Respecto a lenguajes de programación y configuración de este, encontramos varias posibilidades. Es posible usar el IDE de Arduino o instalar MicroPython entre otros.

Según lo detallado anteriormente sobre el ESP32, podemos comprobar que este microcontrolador muy versátil y de alta funcionabilidad, ya sea para pequeños proyectos de IoT o de mayor envergadura en productos comerciales.

Es por esto que se ha escogido un ESP32- WROMM para la elaboración del proyecto, aprovechando todas las cualidades de las que puede dotar a los módulos.

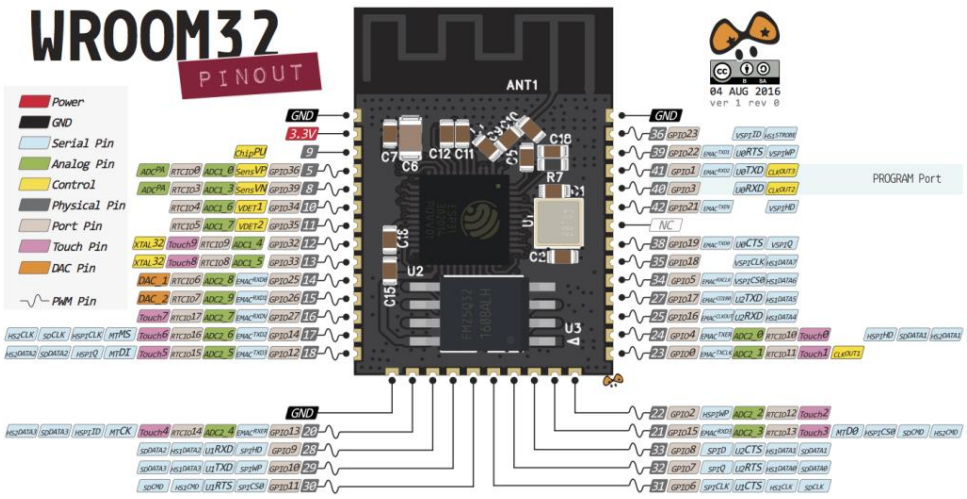

Figura 8: ESP32

4 DESCRIPCIÓN DE LA SOLUCIÓN

4.1 DIARIO DE DESARROLLO

4.1.1 Requisitos del sistema

El desarrollo del sistema viene dado por los requisitos que debe cumplir para que tenga las funcionalidades que se han descrito anteriormente.

Por la propia naturaleza del proyecto se requiere de dos dispositivos: un dispositivo emisor y un dispositivo receptor portátil. El punto principal que había que tratar era el sistema de comunicación. Tras ello cabía definir qué características debía tener cada uno de los dispositivos y, por último, qué diseño sería más eficiente para cumplir con funciones requeridas.

A continuación, se expondrá en detalle qué requisitos corresponden a cada uno de estos puntos.

4.1.1.1 Sistema de comunicación

Los requisitos que debía cumplir el sistema de comunicación entre los dos dispositivos son:

- Protocolo simple. Debe ser sencillo de implementar para que cualquier dispositivo que trabaje en la banda de RF elegida pueda servir de base/receptor sin grandes requisitos de velocidad de procesamiento, memoria, etc.
- Bajo coste. El proyecto nace con un interés social y por tanto el coste de implementación debe ser asumible por cuantas más empresas mejor.
- Escalabilidad. Hablamos de la capacidad de ampliar el sistema indefinidamente sin costes asociados a la escala ni problemas en el protocolo.
- Alcance medio. El sistema debe poder funcionar en espacios cerrados amplios tales como edificios, plantas industriales, almacenes, etc.

4.1.1.2 Dispositivos

Cada uno de los dispositivos debía que ser capaz de cumplir estas funciones.

1. Emisor.

Para cumplir con sus funciones la base debe contar con las siguientes características (a parte de la capacidad de comunicarse con el receptor):

- Sistemas de detección de incendios.
- Conexión con sistemas comerciales de detección de incendios u otras emergencias.
- Emisión de mensajes personalizados de forma remota.
- Modo de configuración de mensajes, comportamiento, etc.

2. Receptor.

Para cumplir con sus funciones el receptor debe contar con las siguientes características (a parte de la capacidad de comunicarse con el emisor):

- Sistema de alerta luminosa.
- Sistema de alerta por vibración.
- Autonomía de al menos una jornada de trabajo.
- Capacidad de recarga de energía sencilla.

4.1.1.3 Diseño

Los requisitos de diseño deben permitir que se cumplan los anteriores, pero más específicamente deben de cumplir los siguientes criterios:

- Minimalismo. Minimizar el número de piezas y componentes para simplificar el sistema y prevenir fuentes de error.
- Robustez. Tanto a nivel de hardware como de software este debe ser lo suficientemente robusto para funcionar bajo cualquier circunstancia.

En cuanto al diseño visual para el dispositivo emisor, este debe ser discreto y adherible a una superficie vertical. El dispositivo receptor por su parte debe ser portátil y cómodo de llevar.

4.2 CARACTERÍSTICAS DEL SISTEMA

Una vez que se definen los requisitos de ambos dispositivos pasamos a tomar decisiones de diseño para cumplir con los mismos.

4.2.1 Emisor

Para cumplir con los requisitos del emisor se integran los siguientes elementos:

Para el sistema de comunicación utilizamos la banda de 433Mhz como hemos descrito anteriormente. A fin de simplificar el prototipo utilizamos un módulo RF emisor comercial.

- El sistema de detección de incendios se cubre con dos subcircuitos:
    - Sensor de temperatura PTC. El sensor de temperatura responde frente a los cambios de temperatura en el ambiente. Es una buena respuesta frente a incendios sin humo o con fuerte ventilación.
    - Sensor de humo. Este transductor cubre la mayoría de incendios con humo. Utilizamos un módulo comercial para simplificar el prototipo.
- La interconexión con sistemas comerciales de alarma se da a través de dos relés que actuarán como entrada y salida respectivamente. La salida como interruptor y la entrada como open drain.
- Para el sistema de emisión de mensajes personalizados se implementará una REST API. Por ello se debe integrar un sistema con conexión a internet. Por ello a la hora de elegir microcontrolador elegimos un SOC con WiFi.
- Para la configuración integramos pulsadores e interruptores DIP.
- El sistema será alojado en una caja que se adhiere a la pared con tornillos.

4.2.2 Receptor

Para cumplir con los requisitos del receptor se integran los siguientes elementos:

- Para el sistema de comunicación utilizamos la banda de 433Mhz como hemos descrito anteriormente. A fin de simplificar el prototipo utilizamos un módulo RF receptor comercial.
- El sistema de alerta al usuario se cubre con dos subcircuitos:

    - Alerta luminosa. Un led RGB de alto brillo puede representar un código de color capaz de mostrar si todo está correcto o por el contrario hay alguna emergencia.
    - Alerta por vibración. Un pequeño motor vibrador ayuda a atraer la atención a la señal luminosa y completar el código antes descrito.
- La autonomía se ve garantizada mediante una batería de litio recargable.
- Se integra además un circuito de carga para la batería.
- La portabilidad del dispositivo se garantiza alojado el circuito receptor en una pulsera o brazalete cómodo de llevar.

4.3 PROTOTIPO EN PROTOBOARD

A continuación, se realizaron prototipos de ambos circuitos en una placa de prototipado para probar las funcionalidades.

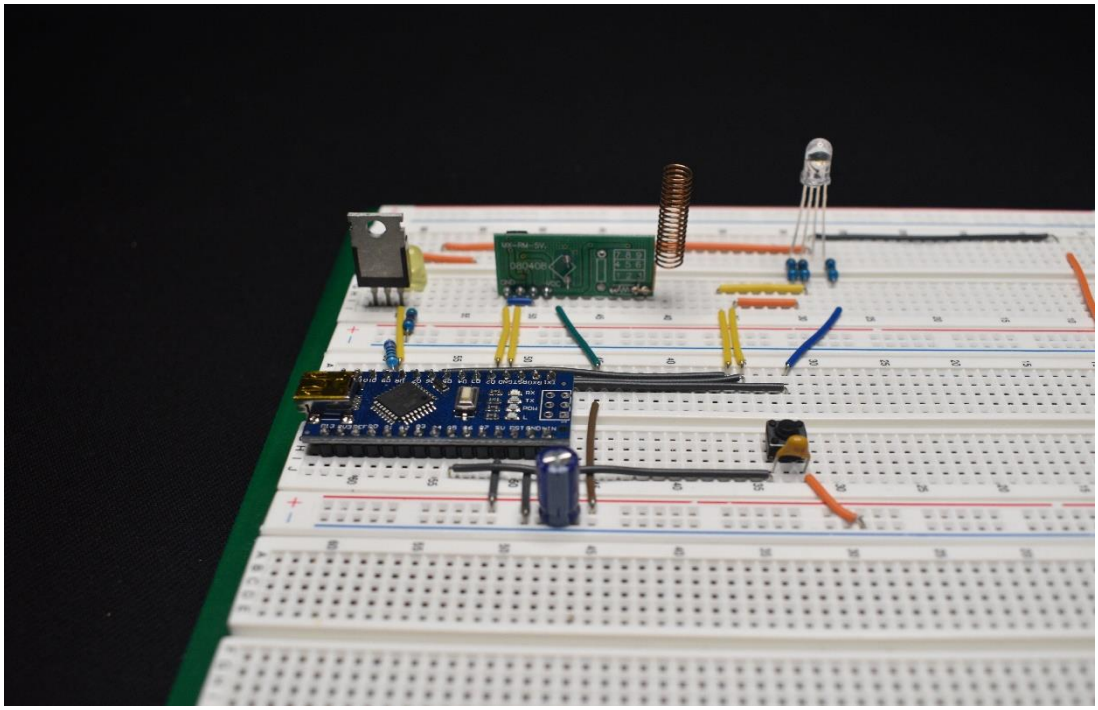

Figura 9: Prueba en Protoboard

Tras estas pruebas se toman las siguientes decisiones:

- Para el circuito emisor se utilizará como microcontrolador un SOC de ESP32 que añade la funcionalidad de WiFi.

- Para el circuito receptor se hace la prueba con un Arduino Nano así que el microcontrolador que se utilizará para el prototipo final será un ATMEGA328p.
- Los módulos de comunicación de RF serán el FS1000A (emisor) y el XY-MK-5V (receptor).
- El entorno de desarrollo será Arduino.

4.4 DISEÑO

En la fase de diseño del prototipo final utilizamos el programa de edición de circuito KiCad para realizar los diseños de las PCB (desde el esquemático hasta el enrutamiento de la PCB).

4.4.1 Emisor

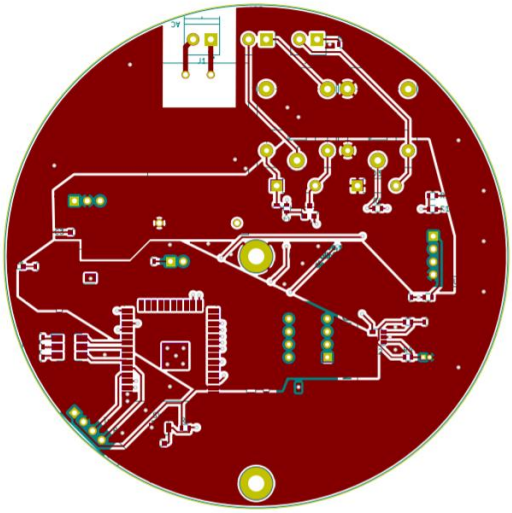

Figura 10: Capa superior PCB Emisor

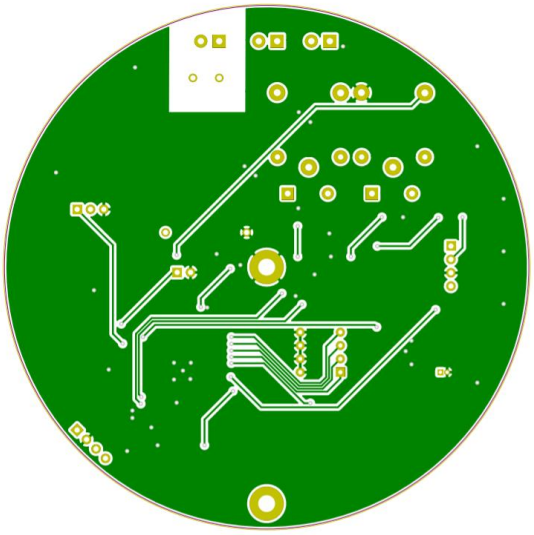

Figura 11: Capa inferior PCB Emisor

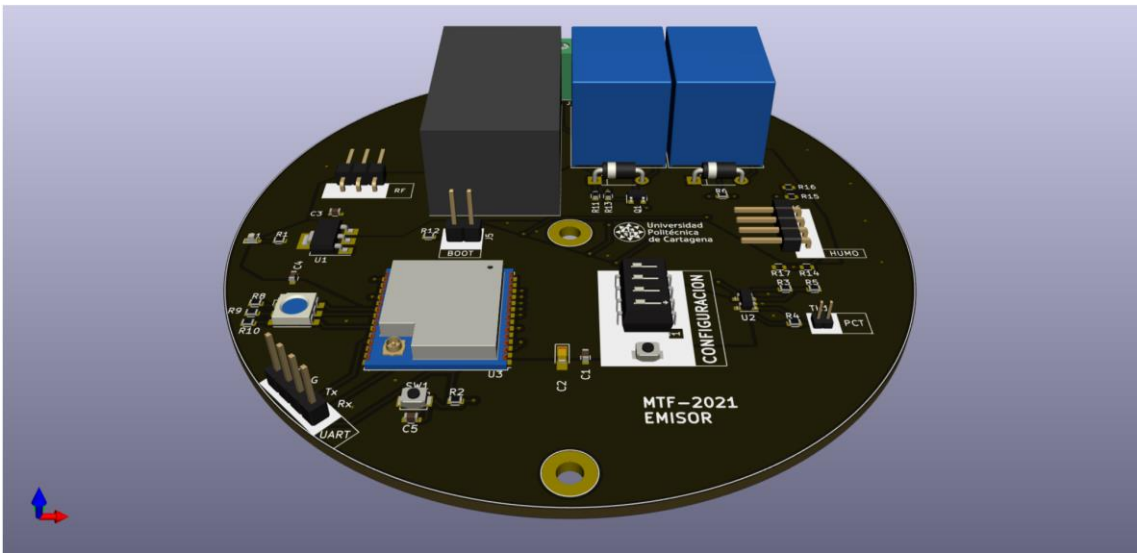

Figura 12: Diseño 3D PCB Emisor

4.4.2 Receptor

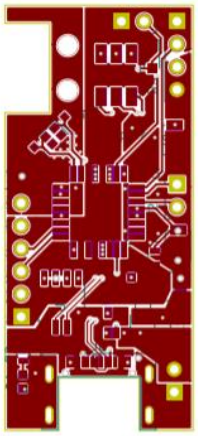

Figura 13: Capa superior PCB Receptor

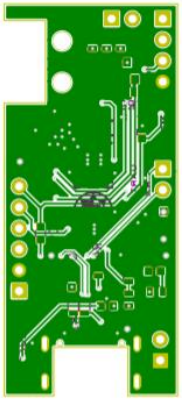

Figura 14: Capa inferior PCB Receptor

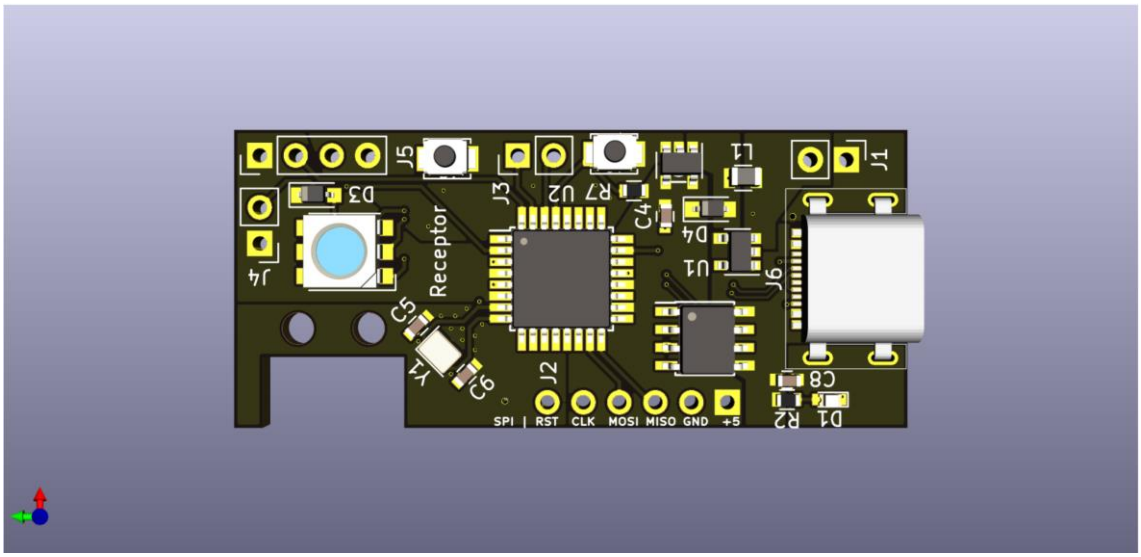

Figura 15: Diseño PCB Receptor

El diseño de las carcasas se realiza con el programa de CAD FreeCAD y posteriormente imprimirse en 3D. El emisor está pensado para ser utilizado como brazalete, de uso diario.

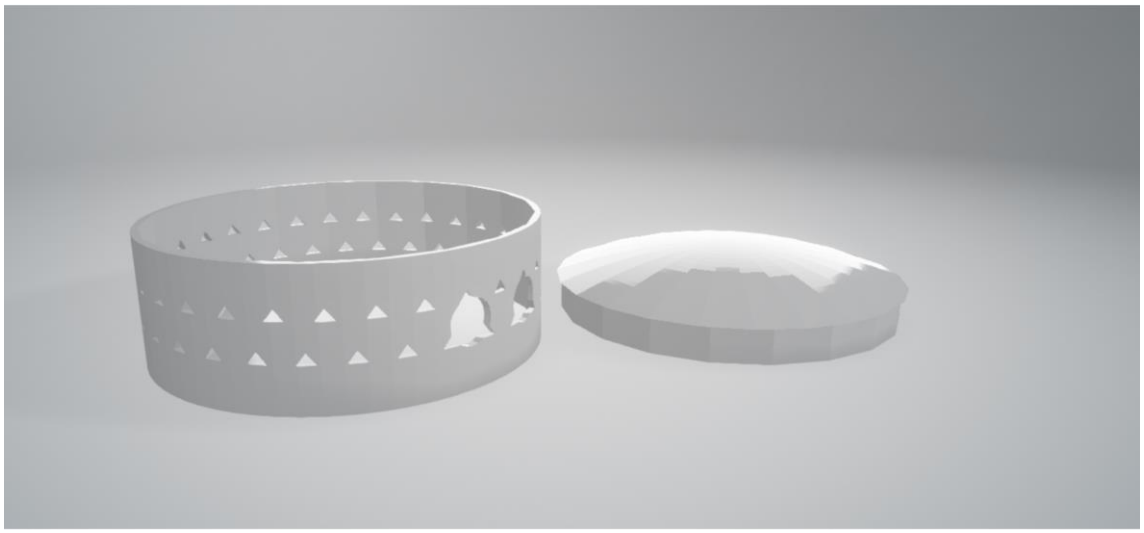

Figura 16: Diseño de carcasa en 3D para Emisor

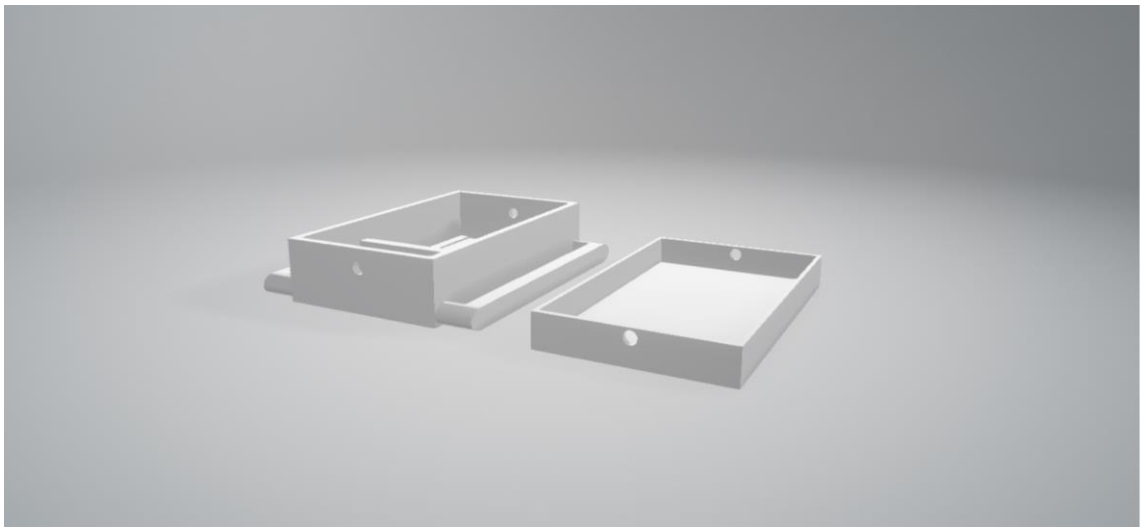

Figura 17: Diseño de carcasa en 3D para Receptor

4.5 ELECTRÓNICA

Esta sección explica en detalle los circuitos del emisor(alarma) y el receptor (brazalete).

4.5.1 Descripción del circuito emisor

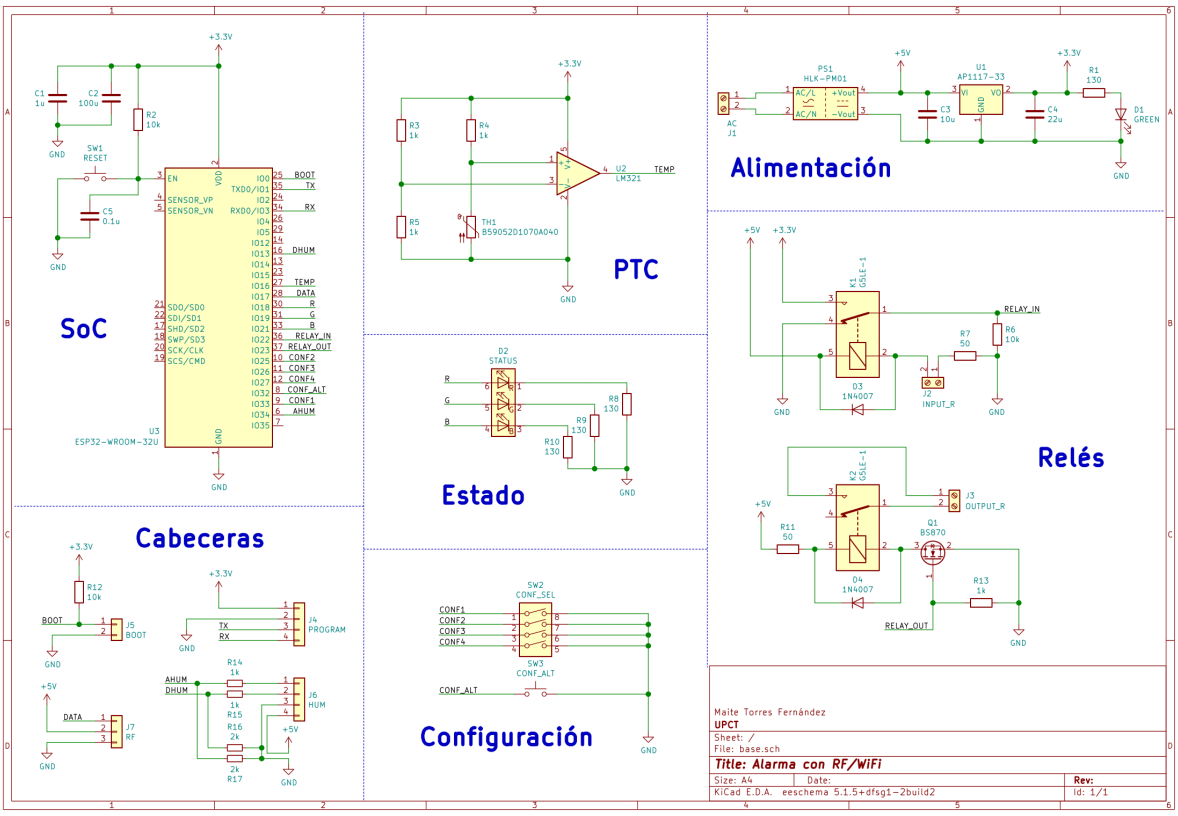

Figura 18: Esquema circuito emisor

4.5.1.1 SoC

Utilizamos un módulo ESP32-WROOM como SoC con funcionalidad de microcontrolador y transceptor WiFi.

4.5.1.2 Cabeceras

Las cabeceras nos permiten la conexión al sensor de humo, a la cabecera de programación y al módulo RF. Son pines de 2.54 mm de espaciado.

- ***Sensor de Humo***

El sensor de humo es un módulo MQ-2. El módulo te permite leer los valores analógicos y cuenta con un circuito disparador ajustable mediante potenciómetro que nos da un valor digital.

- ***Módulo RF***

El módulo RF es un FS1000A, fabricado por varias compañías. Trabaja a 433MHz.

- ***Cabecera de programación***

Nos da acceso al UART, lo que nos permite cargar programas en el SoC.

4.5.1.3 Circuito PTC

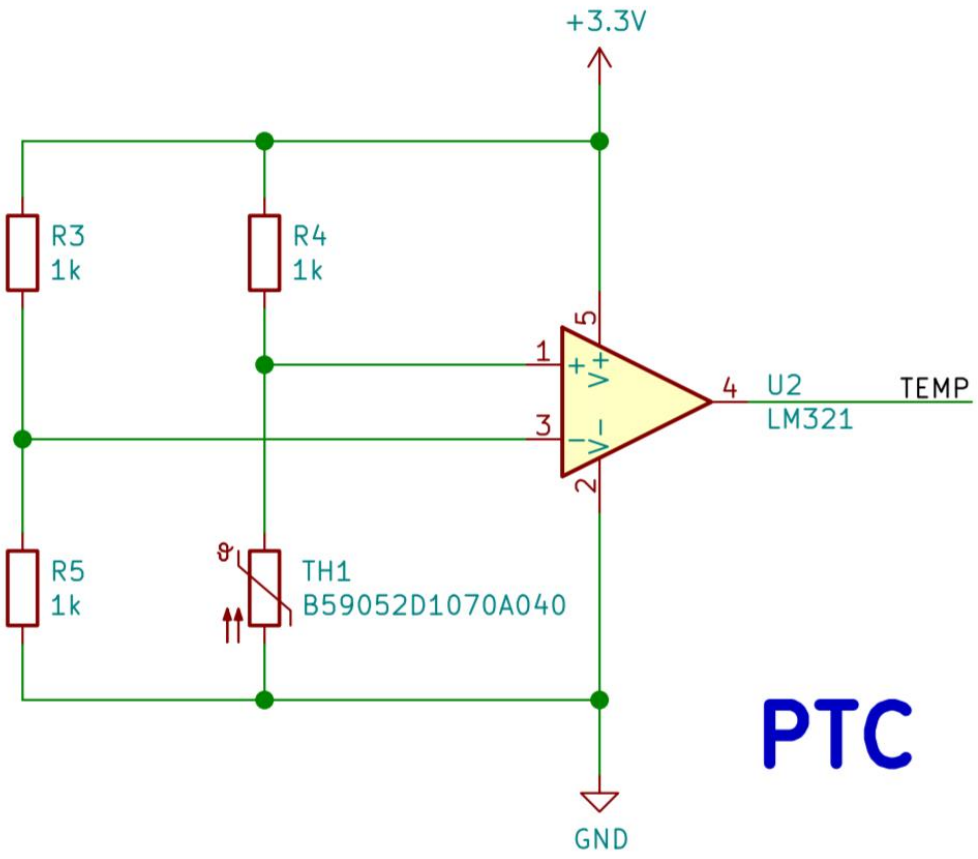

Figura 19: Esquema PTC circuito emisor

La detección de calor se realiza con un circuito de disparo mediante PTC. El disparo lo realiza un circuito comparador con operacional.

$$V_{out} = \begin{cases} 3.3, & si\ V_+ > V_- \\ 0, & si\ V_+ < V_- \end{cases} [V]$$

Tenemos en V_- como referencia un divisor de tensión que nos da un valor fijo de 1.6V. La PTC que usamos (B59052D1070A040) tiene una temperatura de Courie T_c de 70°C y alcanza el orden de 1kΩ sobre los 80°C. Sabiendo esto:

$$V_+ = 3.3 \cdot \frac{R_{TH}}{R_4 + R_{TH}} [V]$$

Y como el valor resistivo del termistor depende de la temperatura, significa que:

$$V_{out} = \begin{cases} 3.3, & si\ T > 80ºC \\ 0, & si\ T < 80ºC \end{cases} [V]$$

Una interrupción del ESP32 activa un flag de alarma durante 5s antes de volver a leer la interrupción. Esto elimina el factor ruido y hace innecesario añadir histéresis al circuito.

4.5.1.4 Relés

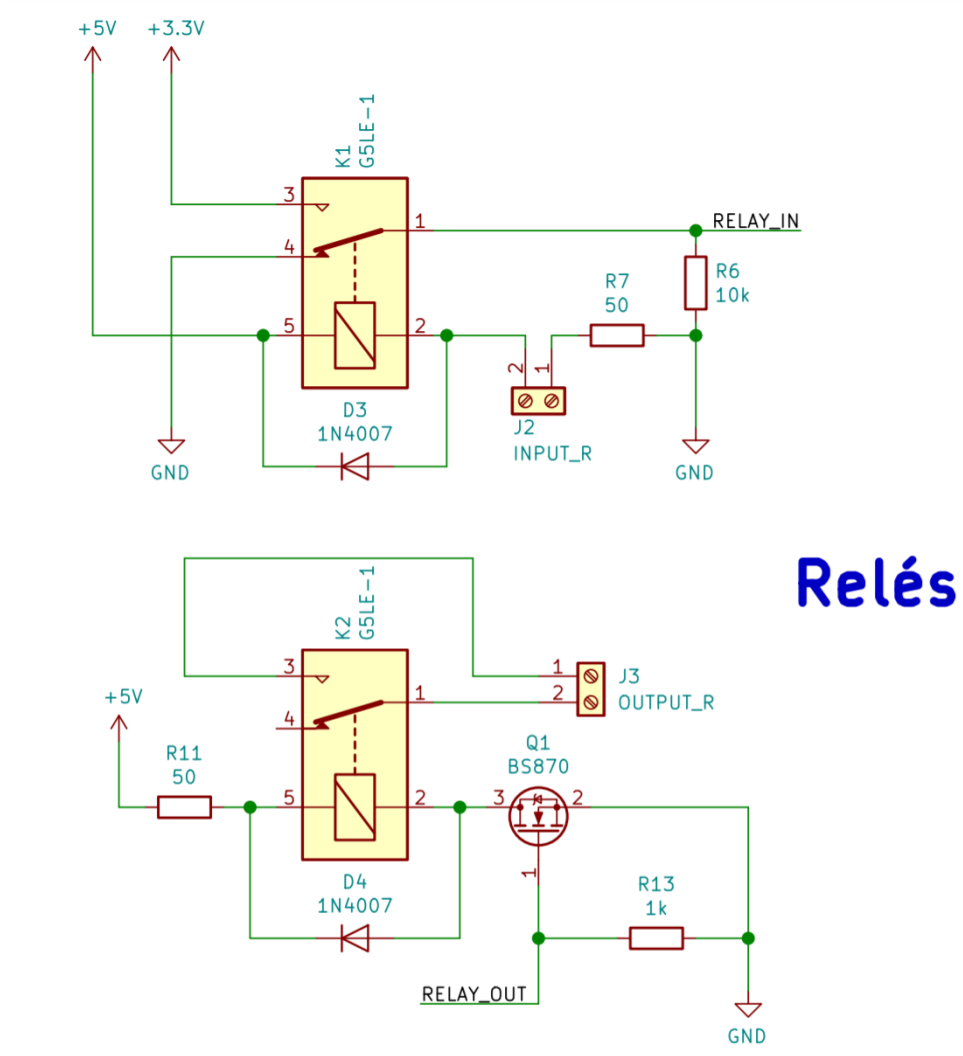

Figura 20: Esquema circuitos de relés

El circuito cuenta con dos relés para participar en circuitos de alarmas comerciales activadas por relés. Puede ser activado por otra alarma y puede activar una alarma externa. En ambos casos se acciona poniendo a masa la

entrada o la salida. La salida además la asiste un MOSFET como driver de conmutación.

4.5.1.5 Configuración y estado

Contamos con un selector DIP de 4 bits y un botón para realizar la configuración y un LED RGB que nos indica en qué estado de trabajo se encuentra el microcontrolador.

4.5.1.6 Alimentación

La alimentación del circuito se realiza mediante una mini fuente encapsulada $230V_{AC}/5V_{DC}$ y un convertidor de tensión step-down a $3.3V_{DC}$.

4.5.2 Descripción del circuito receptor

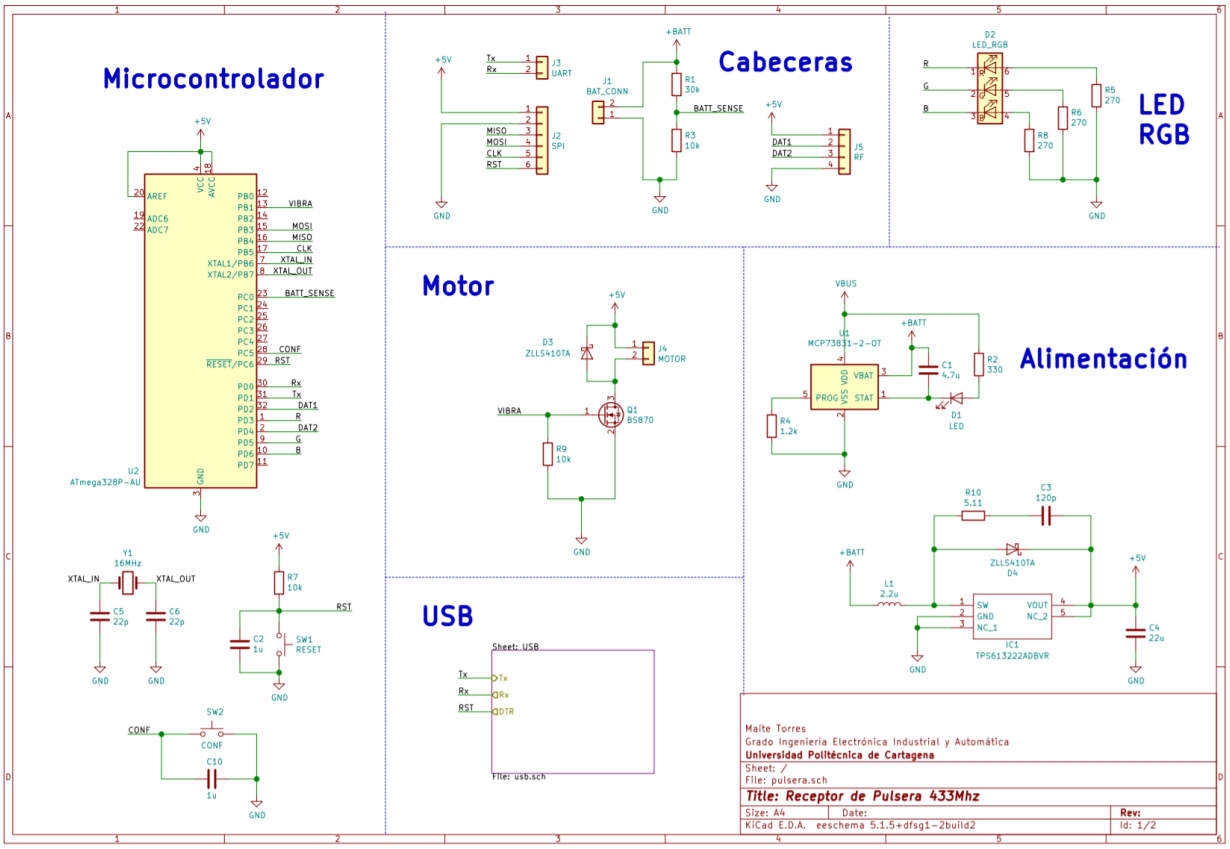

Figura 21: Esquema circuito receptor.

4.5.2.1 Microcontrolador

Utilizamos un microcontrolador ATMEGA328p-AU (como el del Arduino Nano). Es lo suficientemente pequeño y potente para nuestra aplicación y cuenta con los periféricos necesarios.

4.5.2.2 Cabeceras

Las cabeceras nos permiten la conexión de la batería, el módulo RF y las interfaces de comunicación UART e SPI.

- ***Módulo RF***

El módulo RF de recepción es un XY-MK-5V, fabricado por varias compañías. Trabaja a 433MHz y cuenta con un pin de interrupción y otro para datos.

4.5.2.3 Motor Vibrador y LED RGB

El motor vibrador utiliza un MOSFET como driver y es controlado mediante un GPIO con capacidad de PWM.

El LED RGB permite colores de 10 bits mediante 3 GPIOS con PWM. Ambos se utilizan para el mostrar el tipo de alerta que se ha recibido.

4.5.3 Alimentación

La alimentación la provee una batería de litio (3.7VN) que a través de un convertidor de tensión step-up se provee 5V. Contamos además con un circuito cargador por USB.

4.5.4 USB - UART

El circuito cuenta para su rápido desarrollo un puente USB-UART que permite la reprogramación del microcontrolador tras quemar un bootloader por SPI.

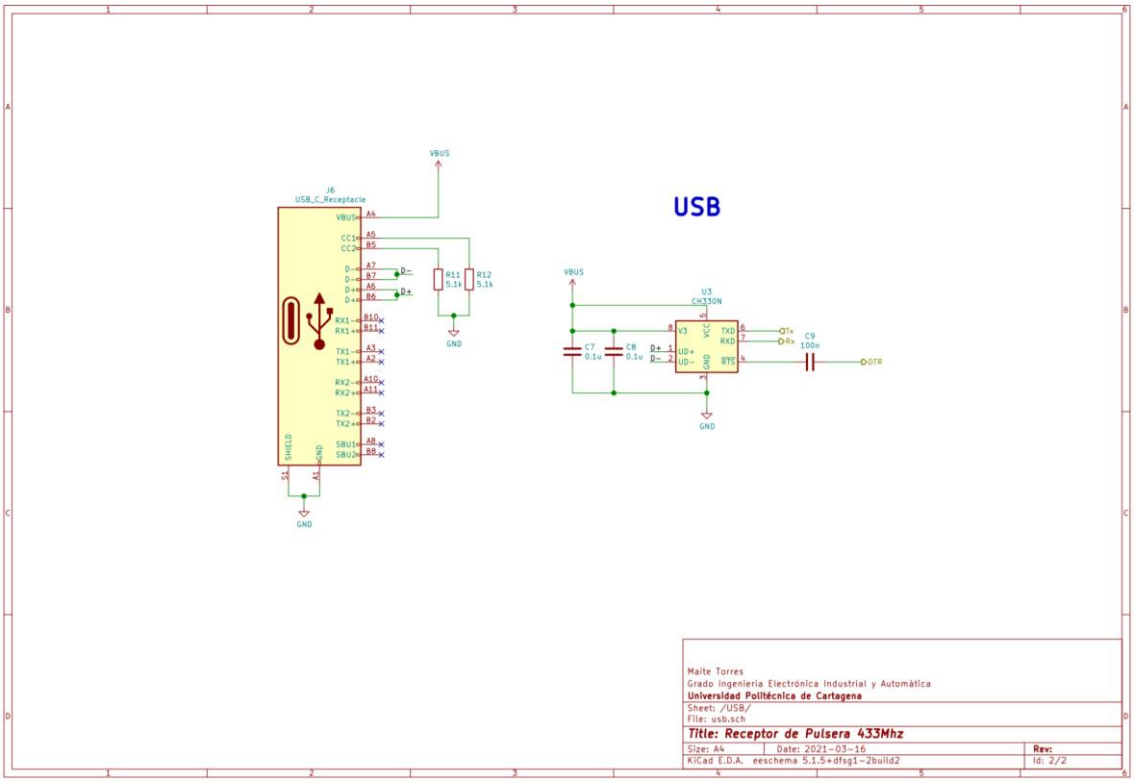

Figura 22: Esquema circuito receptor.

4.6 PROTOCOLO COMUNICACIÓN RF

4.6.1 Descripción del protocolo

Como hemos comentado en anteriores apartados, la comunicación RF se realiza mediante módulos RF de 433 MHz.

Para asegurar la comunicación hemos diseñado un protocolo de comunicación a fin de evitar falsos positivos ya que la banda de 433 MHz está saturada (mandos a distancia de puertas, coches, etc.).

El protocolo utiliza 3 bytes para establecer una comunicación unidireccional con la pulsera de los cuales 12 bits son para identificarse como parte del mismo sistema. El protocolo tendrá la siguiente estructura de bytes:

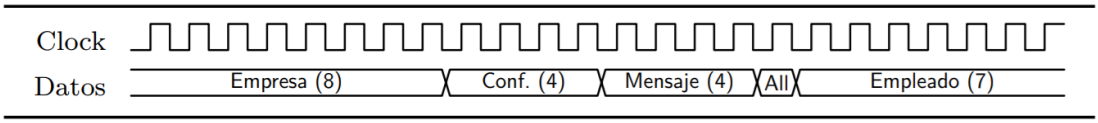

Figura 23: Diagrama de tiempo del protocolo

El significado de cada término es:

- Empresa - Es un identificador propio de nuestra compañía y que se mete en el programa del detector.
- Configuración - Es un identificador secundario que se configura a través de la propia placa. Es para evitar que dos clientes próximos se contaminen con mensajes.
- Mensaje - Aprovechando la infraestructura podemos mandar más tipos de mensajes cada uno con su código de color.
- All - Si este bit es 1 el mensaje es para todos los usuarios.
- Empleado - Identifica al usuario de la pulsera para mandar mensajes individuales. Cada pulsera lleva impreso el número (1-128) que corresponde a cada usuario.

Este protocolo es sencillo y fácil de implementar y permite reutilizar la infraestructura como forma de comunicación con los empleados.

La posible contaminación de la banda de 433 MHz está condicionada a los 12 primeros bits lo que significa que códigos de 3 o más bytes en esa banda pueden interpretarse erróneamente 1 de 4096 veces. Si alguna falsa alarma se diese por la utilización de un dispositivo externo con código similar se pueden cambiar fácilmente los 4 bits de configuración desde el emisor. Si en cualquier caso se diera un entorno de intensa contaminación también podemos añadir otros 8 bits a la palabra de empresa y de ese modo tendremos 20 bits para discriminar (lo que daría >1M de combinaciones).

En cuanto a la seguridad, cualquier sistema basado en una comunicación unidireccional es susceptible de ataques por fuerza bruta. Sin embargo, un ataque a este sistema no supone ningún riesgo para la seguridad de los clientes. Como mucho se podrían generar falsas alarmas y mensajes en zonas concretas.

Para determinar si ha habido o no un ataque el emisor mantiene un registro de toda la actividad (log) que se puede configurar para mandar a un servidor centralizado.

4.6.2 Algoritmo

El algoritmo se describe en el siguiente diagrama:

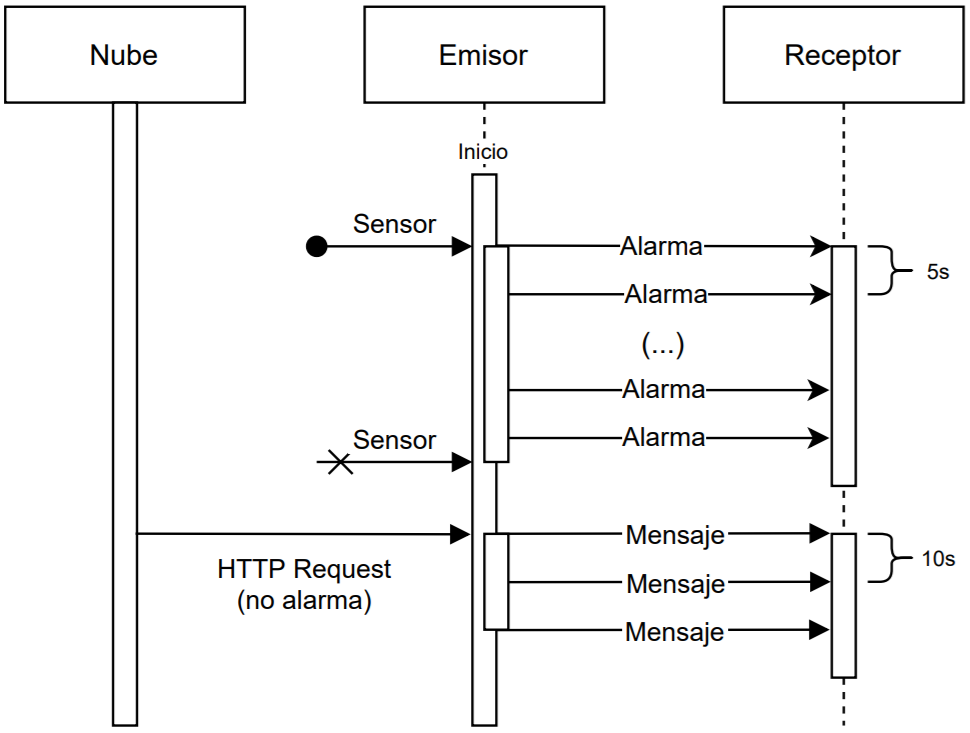

Figura 24: Protocolo de comunicación.

Como podemos observar se dan dos casos en los que el emisor se pone en contacto con el receptor, a saber: disparo por sensores y disparo por petición HTTP.

4.6.2.1 Disparo por sensores

Cuando alguno de los dos sensores o el relé de entrada se dispara, el emisor comienza una rutina de envío de mensajes de alarma. El mensaje seguirá mandándose cada 5 segundos mientras el sensor siga activo. Durante este periodo el emisor no acepta peticiones HTTP. El receptor seguirá en modo alarma hasta que pasen 5 segundos desde el último mensaje.

4.6.2.2 Disparo por petición HTTP

Cuando realizamos una petición a la API REST del dispositivo mediante HTTP este repetirá el mensaje 3 veces cada 10s para asegurar su recepción. Mientras esto ocurre el emisor puede lanzar mensajes de alarma disparados por sensores.

4.6.3 Códigos de mensajes

Code	*RGB*	*Parpadeo*	*VIBRACION*	*INFO*
0x0	Off	No	Off	Ping
0x1	Rojo	Sí	A pulsos	Alarma, evacuar
0x2	Yellow	No	Pulso Largo	Emergencia
0x3	Orange	No	Pulso Largo	Quédate donde estas
0x4…0xb				
0xc	Verde	Sí	Única	Comienzo jornada
0xd	Cian	Sí	Única	Acude a supervisor
0xe	Verde	No	Única	Fin jornada
0xf	Azul	No	Única	Acude a recepción

Tabla 2: Códigos de mensajes

4.7 API REST

4.7.1 Descripción

Una interfaz de programación de aplicaciones por transferencia de estado representacional o por sus siglas en inglés REST API (Application Programming Interface by Representational State Transfer) es una interfaz construida mediante una arquitectura de recursos accesibles mediante el protocolo HTTP.

La decisión de desarrollar el añadido del proyecto sobre una API REST es debido a las siguientes ventajas:

Separación cliente/servidor - La implementación de la API es independiente del sistema. Como objeto de este proyecto no es el desarrollo de una aplicación (web o de escritorio) de control o gestión una REST API nos permite probar la API en cualquier navegador.

Multiplataforma/lenguaje - Una API REST funciona sobre HTTP y este protocolo cuenta con implementaciones en todas las plataformas y en prácticamente todos los lenguajes de desarrollo.

Escalabilidad y flexibilidad - Se puede ampliar los recursos de la API fácilmente si afectar a los recursos anteriores.

Implementación ligera - La implementación de REST es mucho más ligera que otras basadas en SOAP o implementaciones RPC.

4.7.1.1 Protocolo HTTP

Los protocolos basados en el modelo cliente-servidor, como es el caso del HTTP, contemplan tres fases:

1. El cliente establece una conexión TCP.
2. El cliente manda su petición, y espera la respuesta. La petición de datos de un cliente HTTP, consiste en directivas de texto, separadas mediante CRLF (retorno de carro, y cambio de línea), y se divide en tres partes:
    a. La primera parte, consiste en una línea, que contiene un método, seguido de sus parámetros:
        - La dirección del documento pedido: por ejemplo, su URL completa, sin indicar el protocolo o el nombre del dominio.
        - La versión del protocolo HTTP
    b. La siguiente parte, está formada por un bloque de líneas consecutivas, que representan las cabeceras de la petición HTTP, y dan información al servidor, sobre qué tipo de datos es apropiado (como qué lenguaje usar, o el tipo MIME a usar), u otros datos que modifiquen su comportamiento (como que no envié la respuesta si ya está cacheada). Estas cabeceras HTTP forman un bloque que acaba con una línea en blanco.

c. La parte final es un bloque de datos opcional, que puede contener más datos para ser usados por algunos métodos (como POST).

3. El servidor procesa la petición, y responde con un código de estado y los datos correspondientes.

4.7.2 Métodos

HTTP define un conjunto de métodos de peticiones en los que se indican las acciones que se piden realizar al recibir un conjunto de datos. A pesar de que pueden referirse como 'nombres', estos métodos de petición son denominados a veces como 'verbos' de HTTP. Las peticiones más comunes son GET y POST:

- El método GET hace una petición de un recurso específico. Las peticiones con GET únicamente hacen peticiones de datos. Usaremos GET para recursos cuya finalidad sea obtener información.
- El método POST envía datos al servidor de manera que este pueda cambiar su estado. Este es el método usado normalmente para enviar los datos de un formulario HTML. En nuestro caso usamos POST para ejecutar acciones en el dispositivo emisor.

4.7.3 Recursos

En el marco de una API REST llamamos recurso a una funcionalidad asociada a una petición HTTP de un servidor. Cada recurso por tanto es accesible mediante una ruta URL y un método HTTP asociados a la dirección IP del emisor.

Debido a ello el dispositivo receptor debe tener una dirección IP estática configurada en el modo de configuración. En nuestro caso particular cuando accedemos a un recurso de la API lanzamos una rutina en el microcontrolador que o realiza acciones o devuelve información en formato JSON.

Un ejemplo de recurso puede ser, por ejemplo:

```
GET /api/message?type=0&target=all HTTP/1.1
Host: 192.168.1.100:8080
User-Agent: insomnia/2020.3.3
Accept: */*
```

Figura 25: Ejemplo de recurso

Analicemos este caso concreto para entender mejor cómo funcionan los recursos:

- GET - El método HTTP de acceso al recurso. Dependiendo del tipo de funcionalidad que queramos implementar (acceso, configuración, envío de datos, etc.) se utiliza un tipo u otro (GET, POST, PUT, OPTIONS, etc.). En realidad, esta asociación no es obligatoria (podríamos utilizar GET para todos los recursos). Sin embargo, nos ajustaremos a la convención.
- /api/message - Es la ruta URL al recurso.
- type=0&target=all - Las querys. Son parámetros que podemos pasar al recurso como argumentos de una función. Los usaremos para definir el tipo de mensaje, destino, etc.
- 192.168.1.100:8080 - Es el host, el destino de la URL.

4.7.4 Lista de recursos

- GET /api - Devuelve un JSON con info de la configuración.
- POST /api - Devuelve un error 403. Por seguridad no se puede cambiar la configuración desde la API.
- GET /api/read - Devuelve un JSON con info de los sensores y el estado de la salida. Los sensores y el relé muestran un 0 cuando están sin disparar y un 1 cuando están disparados. La salida es simplemente el código que se está repitiendo. Por defecto se envía 'ping'. E.j: {temp=0, hum=0, rele=0, out=0}.
- POST /api/message?type=X&target=Y - El recurso para mandar mensajes tiene dos 'querys'.
    - type es el código de mensaje.
    - target puede ser o el número de identificación del empleado o bien la palabra 'all' para referirse a todos los empleados.
- GET /api/message?type=X&target=Y - Devuelve un JSON con un valor booleano. Será TRUE si el formato de la petición corresponde a una petición válida. FALSE en otro caso. E.j: {op=TRUE}

4.8 PROGRAMACIÓN: C++

4.8.1 Código Pulsera

A continuación, el código utilizado en el prototipo de la pulsera acompañado de una breve explicación:

4.8.1.1 Librerías incluidas

Las librerías que utilizamos, una a una:

```
#include <RCSwitch.h>
#include <Arduino.h>
#include <EEPROM.h>
```

- RCSwitch.h nos da acceso a una interfaz para manejar el módulo de RF.
- Arduino.h es donde se sitúan las funciones principales del framework de Arduino.
- EEPROM.h sirve de API para manejar la EEPROM interna del Atmega328p. Es una librería estándar del framework Arduino.

4.8.1.2 RF

```
RCSwitch RF = RCSwitch();
struct rfdata_t
{
  uint32_t empresa: 8;
  uint32_t conf: 4;
  uint32_t mensaje: 4;
  uint32_t all: 1;
  uint32_t empleado: 7;
} rfdata;
```

En esta sección declaramos de en el ámbito global un objeto RCSwitch para controlar el módulo RF a través de sus métodos (RF).

También definimos y declaramos una estructura para manejar los datos que nos lleguen.

Utilizamos campos de bits para codificar toda la información en 3 bytes (24 bits en total).

4.8.1.3 Datos

```
struct configuracion
{
  byte empresa;
  byte conf;
  byte empleado;
} confw;
void readConf();
void writeConf();
const byte pinSW = 19;
bool enConf = false;
```

En esta sección del código se declaran y definen diferentes variables y funciones para el manejo interno de datos.

- *Struct configuration {…} confw* sirve para manejar la configuración del dispositivo.
- *Const byte pinW=19* constante que indica el número de pin del botón de configuración.
- *Bool enconf=false* variable de la máquina de estados que indica si el dispositivo está en modo configuración.
- La definición de las dos funciones es:

```
void readConf()
{
  EEPROM.get(0x00, confw);
}
void writeConf()
{
  EEPROM.put(0x00, confw);
  delay(50);
}
```

  Simplemente escribe y lee una estructura al inicio de la EEPROM.

4.8.1.4 Timing

```
unsigned long t;
unsigned long t0;
unsigned long count;
bool enT = false;
```

Variables utilizadas para calcular y desencadenar eventos temporales

4.8.1.5 *Actuadores*

```
void launch();
void do_code(byte code);
void restore();
```

En esta sección se encuentran tres funciones relacionadas con el lanzamiento de eventos por RF. Veamos las definiciones:

```
void launch()
{
  if (rfdata.empresa != confw.empresa)
    return;
  if (rfdata.conf != confw.conf)
```

```
    return;
  if (rfdata.all || rfdata.empleado == confw.empleado)
  {
    do_code(rfdata.mensaje);
    enT = true;
#ifdef DEBUG
    Serial.println("Mensaje aceptado");
#endif
  }
}
```

Esta función comprueba si una vez recibido un mensaje el dispositivo comparte configuración con el emisor. De ser así lanza la función *do_code()*.

```
void do_code(byte code)
{
  // Funciona dependiendo del código que recibe.
  switch (rfdata.mensaje)
  {
    case 0x00:
      if (enConf)
      {
        confw.empresa = rfdata.empresa;
        confw.conf = rfdata.conf;
        writeConf();
      }
      break;
    case 0x01:
      color = ROJO;
      writeRGB(color);
      motor(true);
      pulsos = true;
      parpadeo = true;
      break;
    case 0x02:
      color = AMARILLO;
      writeRGB(color);
      motor(true);
      pulsos = true;
      parpadeo = false;
      break;
    case 0x03:
      color = NARANJA;
      writeRGB(color);
      pulsos = true;
      parpadeo = false;
      break;
    case 0x04:
```

```
    case 0x05:
    case 0x06:
    case 0x07:
    case 0x08:
    case 0x09:
    case 0x0a:
    case 0x0b:
      break; // No implementados
```

```
    case 0x0c:
      color = VERDE;
      writeRGB(color);
      motor(true);
      delay(200);
      motor(false);
      parpadeo = true;
      pulsos = false;
      break;
```

```
    case 0x0d:
      color = CIAN;
      writeRGB(color);
      motor(true);
      delay(200);
      motor(false);
      parpadeo = true;
      pulsos = false;
      break;
    case 0x0e:
      color = VERDE;
      writeRGB(color);
      motor(true);
      delay(200);
      motor(false);
      parpadeo = false;
      pulsos = false;
      break;
    case 0x0f:
      color = AZUL;
      writeRGB(color);
      motor(true);
      delay(200);
      motor(false);
      parpadeo = false;
      pulsos = false;
      break;
    default:
      return;
      break;
  }
  count = 5000;
```

```
  tpul = millis();
  tpar = millis();
  t0 = millis();
}
```

En función del código recibido se cambian las variables de la máquina de estados para determinar el comportamiento del dispositivo (parpadeo, vibración...) y reinicia los contadores relacionados con el timing.

```
void restore()
{
    writeRGB(0);
    enConf = false;
    pulsos = false;
    parpadeo = false;
    enT = false;
}
```

Esta función se lanza cuando el dispositivo vuelve a modo reposo.

4.8.1.6 LED RGB

```
const byte R = 3, G = 6, B = 5;
void writeRGB(uint32_t value);
void toggleRGB(uint32_t value, bool reset = false);
enum Colores : uint32_t {
  ROJO = 0xff0000, VERDE = 0xff00, AZUL = 0xff,
  NARANJA = 0x802000, AMARILLO = 0x402010, CIAN = 0x00ffff,
  ROSA = 0x400020
};
uint32_t color;
bool parpadeo;
unsigned long tpar;
```

Funciones y variables utilizadas en la gestión del LED RGB. El LED RGB está conectado a los pines 3, 5 y 6 y hemos definido colores en hexadecimal para manejar todos los colores con variables de 32 bits de forma sencilla (al estilo HTML). Examinando la definición de las funciones se entiende el funcionamiento fácilmente:

```
void writeRGB(uint32_t value)
{
  // value = 0xFFFFFFul - value; // Descomentar si ánodo común.
  analogWrite(B, (value & 0xff));
  analogWrite(G, ((value & 0xff00) >> 8));
  analogWrite(R, ((value & 0xff0000) >> 16));
}
```

La función extrae del parámetro introducido los tres colores de 8 bits codificados. Se le pueden pasar como parámetro directamente códigos de color HTML como los definidos en el *enum*.

```
void toggleRGB(uint32_t value, bool reset = false)
{
  static bool toggle = true;
  toggle = reset ? false : !toggle;
  if (toggle)
  {
    writeRGB(value);
  }
  else
  {
    writeRGB(0);
  }
}
```

Esta función sirve para realizar parpadeos de velocidad dependiente de la variable *tpar* (en el loop veremos cómo).

4.8.1.7 Motor

```
const byte pinM = 9;
void motor(bool marcha);
void motor();
bool pulsos;
unsigned long tpul;
```

En esta sección se definen y declaran variables y funciones relacionadas con la vibración como en qué pin controla el motor vibrador *pinM* = 9; o si este funciona a pulsos o de forma continua.

```
void motor(bool marcha)
{
  if (marcha)
  {
    analogWrite(pinM, 200);
  }
  else
  {
    analogWrite(pinM, 0);
  }
}

void motor()
{
  static bool marcha = false;
  marcha = !marcha;

  motor(marcha);
}
```

La función *motor* tiene dos definiciones. Si se quiere hacer funcionar a pulsos se utiliza *void motor();* que alterna entre marcha y paro cada vez que se llama a la función. Si se quiere determinar un estado concreto se utiliza *void motor(bool marcha);*.

4.8.1.8 Setup

```
void setup() {
  // Debug
#ifdef DEBUG
  Serial.begin(9600);
  Serial.println("Modo Debug");
#endif

  // RF
  RF.enableReceive(0);  // Receiver on interrupt 0 => that is pin #2
  readConf();

  // RGB
  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);
  for (size_t i = 0; i < 3; ++i) // Parpadeo inicio.
  {
    writeRGB(0xff0000ul >> (i * 8));
    delay(200);
    writeRGB(0);
    delay(200);
  }

  // Boton Ping.
  pinMode(pinSW, INPUT_PULLUP);

  // Motor
  pinMode(pinM, OUTPUT);
  motor(true);
  delay(500);
  motor(false);
}
```

El setup es una función que es llamada al inicio del programa y donde se realiza la configuración del dispositivo. Realizamos las siguientes acciones de forma secuencial:

- Relacionado con el módulo RF configuramos el objeto que lo controla asignándole la recepción a la interrupción 0 (pin 2) y leemos la configuración de la memoria EEPROM.

- Configuramos los pines del RGB como salida y realizamos un encendido de los 3 colores secuencialmente para comprobar su funcionamiento.
- Configuramos el pin del botón de configuración como entrada pull-up.
- Configuramos el pin del motor vibrador como salida y realizamos una vibración de prueba.

4.8.1.9 Loop

```
void loop() {
  // RF
  if (RF.available()) {
    uint32_t rawdata = RF.getReceivedValue();
    rfdata = *(reinterpret_cast<rfdata_t*>(&rawdata));
    launch();
    RF.resetAvailable();
#ifdef DEBUG
    {
      Serial.println("Mensaje Recibido:");
      Serial.print("Código empresa:\t");
      Serial.println(rfdata.empresa, HEX);
      Serial.print("Código configuración:\t");
      Serial.println(rfdata.conf);
      Serial.print("Mensaje:\t");
      Serial.println(rfdata.mensaje, HEX);
      Serial.print("Destino:\t");
      if (rfdata.all)
      {
        Serial.println("All");
      }
      else
      {
        Serial.println(rfdata.empleado, HEX);
      }
    }
#endif
  }

  // Timing
  if (enT)
  {
    t = millis();
    if (abs(t - t0) > count)
    {
      restore();
      enT = false;
    }
    if (pulsos)
    {
```

```
      if (abs(t - tpul) > 200)
      {
        motor();
        tpul = millis();
      }
    }
    if (parpadeo)
    {
      if (abs(t - tpar) > 200)
      {
        toggleRGB(color);
        tpar = millis();
      }
    }
  }

  // Ping rec
  if (!digitalRead(pinSW))
  {
    restore();
    enConf = true;
    enT = true;
    t0 = millis();
    count = 10000;
    toggleRGB(0, true);
    color = ROSA;
    parpadeo = true;
#ifdef DEBUG
    {
      Serial.println("-- Modo emparejamiento (10s) --");
      Serial.println("Datos actuales");
      Serial.print("Empresa\t");
      Serial.println(confw.empresa, HEX);
      Serial.print("Conf\t");
      Serial.println(confw.conf, HEX);
      Serial.print("Empleado\t");
      Serial.println(confw.empleado, HEX);
    }
#endif
  }

}
```

La función *void loop();* se llama de forma cíclica después de *setup()* mientras el dispositivo esté encendido. Está definida como una máquina de estados donde el código que ejecuta depende de unas variables de control. La mayor parte del tiempo no se ejecuta nada y el bucle se llama una y otra vez esperando a que se lance el evento. Los eventos posibles son:

- Recepción de mensaje RF. Cuando se recibe un mensaje se ejecutan las siguientes líneas:

```
uint32_t rawdata = RF.getReceivedValue();
rfdata = *(reinterpret_cast<rfdata_t*>(&rawdata));
launch();
RF.resetAvailable();
```

  En resumen:

  1. Se lee el mensaje guardando la información en bruto en una variable de 32 bits.
  2. Se reinterpreta la información sobrescribiéndola directamente sobre la estructura de rfdata. Ahora tenemos los datos codificados en la estructura pudiendo acceder a sus miembros uno a uno.
  3. Se llama a la función *launch()* para ver si procede realizar alguna acción con el código.
  4. Se reinicia el driver para evitar errores al recibir otro código.
- Timing. Si la variable *enT* es igual a *true* significa que se están realizando eventos temporales y dependiendo de los contadores se realizan acciones como parar y encender el motor o parpadear el LED.
- Entrar en modo configuración. Si se pulsa el botón automáticamente se entra en el modo configuración que permite la escritura en la configuración cuando se envía un mensaje con el código de mensaje 0x00. Esto permite emparejar el dispositivo a un sistema concreto.

4.8.2 Código Base

A continuación, el código utilizado en el prototipo de la base/alarma acompañado de una breve explicación:

4.8.2.1 Librerías

```
#include <Arduino.h>
#include <RCSwitch.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include <ESPAsyncWebServer.h>
#include "WifiConf.h"
```

Las librerías que utilizamos, una a una:

- *Arduino.h* es donde se sitúan las funciones principales del framework de Arduino.

- *RCSwitch.h* nos da acceso a una interfaz para manejar el módulo de RF.
- *WiFi.h* sirve de API para controlar la conexión a internet. Es una librería del framework Arduino de Espressif Systems (ESP32).
- *ArduinoJson.h* brinda una colección de funciones para manejar archivos JSON en la memoria de programa (estáticos).
- *ESPAsyncWebServer.h* API para controlar un servidor asíncrono en SOCs ESP8266 y ESP32.
- *WifiConf.h* Contiene de forma estática una web para configurar la red WiFi. Se accede a ella a través durante el modo configuración. Su contenido es texto HTML plano y no es interesante de analizar.

4.8.2.2 RF

```
RCSwitch RF = RCSwitch();
const byte RFPin = 17;
struct rfdata_t
{
    uint32_t empresa:8;
    uint32_t conf:4;
    uint32_t mensaje:4;
    uint32_t all:1;
    uint32_t empleado:7;
} rfdata;
void lanzarRF(); // Envía RFDATA
```

En esta sección se declaran y definen objetos y variables para el funcionamiento del módulo RF.

- El objeto *RCSwitch* RF maneja el módulo.
- La variable *const byte RFPin* indica en qué pin está conectado el módulo.
- La estructura *struct rfdata_t {...} rfdata* contiene la información sobre el mensaje a envíar. Se usa campos de bits para codificar toda la información en 4 bytes (32 bits).
- La función *void lanzarRF()* envía el mensaje con la información codificada en la variable global *rfdata*.

  ```
  void lanzarRF()
  {
      uint32_t *data = reinterpret_cast<uint32_t*>(&rfdata);
      RF.send(*data, 24);
  }
  ```

  Se reinterpreta la información como un entero de 32 bits y se envía por el módulo RF.

4.8.2.3 *LED RGB*

```
const byte pinB = 18, pinG = 19, pinR = 21;
const byte chR = 0, chG = 1, chB = 2;
const unsigned LedFreq = 50000;
const byte LedResolution = 8;
```

```
enum class LEDSTATUS : uint32_t
{
  OFF = 0x0, RUN = 0x004000, RUNCONF = 0xff0500, WIFI = 0x000080, PING =
};
void showStatus(uint32_t);
void showStatus(LEDSTATUS);
```

Funciones y variables utilizadas en la gestión del LED RGB. El LED RGB está conectado a los pines 18, 19 y 21 y hemos definido colores en hexadecimal para manejar todos los colores con variables de 32 bits de forma sencilla (al estilo HTML). El ESP32 contiene un driver para la gestión de LEDs que permite controlarlos de forma más sencilla que en el Atmega328p (frecuencia, resolución y canal del timer). La función tiene dos definiciones:

```
void showStatus(uint32_t hexcode)
{
  ledcWrite(chR, (hexcode & 0xff0000) >> 4);
  ledcWrite(chG, (hexcode & 0xff00) >> 2);
  ledcWrite(chB, hexcode & 0xff);
}
void showStatus(LEDSTATUS ledstatus)
{
  showStatus(static_cast<uint32_t>(ledstatus));
}
```

La segunda definición realiza el casting y llama a la primera.

4.8.2.4 *Disparadores y relés*

```
// Relés
const byte pinReleOut = 23;
const byte pinReleIn = 22;
inline void activarRele(){digitalWrite(pinReleOut,HIGH);}
inline void desactivarRele(){digitalWrite(pinReleOut,LOW);}
// PTC
const byte PTC = 16;
// Humo
const byte AHum = 34;
```

En esta sección se declaran variables relacionadas con los disparadores tales como los pines donde están conectados. Las dos funciones encienden y apaga el relé de salida respectivamente.

4.8.2.5 Configuración/DIP

```
const byte SWPin = 32;
const byte DIP[] = {33,25,26,27};
```

Declaramos variables que indican en qué pines están conectados el array de conmutadores DIP y el pulsador de configuración.

4.8.2.6 REST API

```
AsyncWebServer server(80);
bool APItrigger = false;
StaticJsonDocument<250> jsonDocument; // Para enviar y recibir JSON
char buffer[250];
const char* R_api = "/api"; // Ruta
void get_api(AsyncWebServerRequest *request);
void post_api(AsyncWebServerRequest *request);
const char* R_read = "/api/read"; // Ruta
void get_api_read(AsyncWebServerRequest *request);
const char* R_message = "/api/message"; // Ruta
void get_api_message(AsyncWebServerRequest *request);
void post_api_message(AsyncWebServerRequest *request);
void notFound(AsyncWebServerRequest *request);
```

En esta sección se definen y declaran funciones y variables relacionadas con la API REST. Las funciones indican qué hacer cuando se recibe a través del servidor asíncrono *AsyncWebServer server* una petición HTTP. Las definiciones son:

```
void get_api(AsyncWebServerRequest *request)
{
    jsonDocument.clear();
    jsonDocument["IP"] = WiFi.localIP();
    jsonDocument["SSID"] = WiFi.SSID();
    jsonDocument["ValorEmpresa"] = rfdata.empresa;
    jsonDocument["ValorConfiguracion"] = rfdata.conf;
    jsonDocument["UltimoMensaje"] = rfdata.mensaje;
    serializeJson(jsonDocument, buffer);
    request->send(200, "application/json", buffer);
}

void post_api(AsyncWebServerRequest *request)
{
    request->send(403);
}

void get_api_read(AsyncWebServerRequest *request)
{
    jsonDocument.clear();
    jsonDocument["SensorTemperatura"] = !digitalRead(PTC);
    jsonDocument["EntradaRele"] = digitalRead(pinReleIn);
    jsonDocument["SensorHumo"] = (analogRead(AHum) > 4000);
    serializeJson(jsonDocument, buffer);
    request->send(200, "application/json", buffer);
}

void post_api_message(AsyncWebServerRequest *request)
{
```

```
jsonDocument.clear();
const char *Query1 = "type";
const char *Query2 = "target";
int target, type;
if(request->hasParam(Query1) && request->hasParam(Query2))
{
    String q1 = request->getParam(Query1)->value();
    String q2 = request->getParam(Query2)->value();
    type = q1.toInt();
    if(q2 == "all")
    {
        target = 0x80;
    }
    else
    {
        target = q2.toInt();
    }
    if(type < 0 || type > 15)
    {
        jsonDocument["status"] = "Error";
        serializeJson(jsonDocument, buffer);
        request->send(200, "application/json", buffer);
        return;
    }
    if(target < 0 || target > 0x80)
    {
        jsonDocument["status"] = "Error";
        serializeJson(jsonDocument, buffer);
        request->send(200, "application/json", buffer);
        return;
    }
```

```
    if(target==0x80)
    {
        rfdata.all = 1;
    }
    rfdata.empleado = target;
    rfdata.mensaje = type;
    lanzarRF();
    showStatus(LEDSTATUS::WIFI);

    jsonDocument["status"] = "ok";
    request->send(200, "application/json", buffer);
}
else
{
    jsonDocument["status"] = "Error";
    serializeJson(jsonDocument, buffer);
    request->send(200, "application/json", buffer);
```

```
    }
}

void get_api_message(AsyncWebServerRequest *request)
{
    jsonDocument.clear();
    const char *Query1 = "type";
    const char *Query2 = "target";
    int target, type;
    if(request->hasParam(Query1) && request->hasParam(Query2))
    {
        String q1 = request->getParam(Query1)->value();
        String q2 = request->getParam(Query2)->value();
        type = q1.toInt();
        if(q2 == "all")
        {
            target = 0x80;
        }
        else
        {
            target = q2.toInt();
        }
        if(type < 0 || type > 15)
        {
            jsonDocument["op"] = false;
            serializeJson(jsonDocument, buffer);
            request->send(200, "application/json", buffer);
            return;
        }
        if(target < 0 || target > 0x80)
        {
            jsonDocument["op"] = false;
            serializeJson(jsonDocument, buffer);
            request->send(200, "application/json", buffer);
            return;
        }
        jsonDocument["op"] = true;
        serializeJson(jsonDocument, buffer);
        request->send(200, "application/json", buffer);
    }
    else
    {
        jsonDocument["op"] = false;
        serializeJson(jsonDocument, buffer);
        request->send(200, "application/json", buffer);
    }

}
```

```
void notFound(AsyncWebServerRequest *request) {
    request->send(404, "text/plain", "Not found");
}
```

4.8.2.7 Modos de operación

```
enum class OPMODE : byte
{
    NORMAL_OP, CONF_MODE
};
OPMODE op; // Variable de operación.
void normal(); // Funciona en modo normal.
void conf(); // Funciona en modo configuración
```

Definimos dos modos de operación y una variable que guarda el modo en el que se encuentra la máquina. Las funciones definen ambos estados:

```
void normal()
{
    // Watchdog
    if(!WiFi.isConnected())
    {
        ESP.restart();
    }

    showStatus(LEDSTATUS::RUN); // Verde, todo ok.

    if(!digitalRead(SWPin)) // Ping
    {
        rfdata.all=1;
        rfdata.mensaje=0;
        lanzarRF();
        showStatus(LEDSTATUS::PING);
        delay(2500);
        return;
    }

    // Comprobar sensores
    bool trigger = false;
    trigger |= digitalRead(pinReleIn);
    trigger |= !digitalRead(PTC);
    trigger |= (analogRead(AHum) > 4000);
    if(!trigger)
        return;

    // Alarma a partir de aquí
    showStatus(0xff0000); // Rojo
    activarRele();
```

```
    rfdata.all=1;
    rfdata.mensaje=0x01; // Alarma, evacuar
    lanzarRF();
    delay(2500);
    desactivarRele();
}
```

En el modo de funcionamiento normal se realizan las siguientes acciones de forma secuencial:

1. Comprobación de la conexión de red. A modo de watchdog, si se pierde la conexión a internet reinicia el dispositivo.
2. Restaura el LED RGB al color del funcionamiento normal (verde).
3. Comprueba si se pulsa el botón de configuración. Si es así se envía por RF el mensaje "ping" (a todos los dispositivos con el código 0x00) utilizado para emparejar el dispositivo.
4. Comprueba los 3 posibles disparadores de la alarma (externo, PTC y humo) y si se dispara se lanza el mensaje de alarma (a todos los dispositivos con el código 0x01).

```
// Modo configuración
void conf() ////////////////////////////////
{
    // No hace nada. El servidor de configuración
    // asíncrono lanza la web en el otro núcleo.
    delay(500);
    showStatus(LEDSTATUS::OFF);
    delay(200);
    showStatus(LEDSTATUS::RUNCONF);
}
```

En el modo de configuración tan solo parpadea el LED RGB. El servidor asíncrono corre en el otro núcleo del ESP32.

4.8.2.8 Setup

```
void setup() {
    // Debug
    Serial.begin(115200);

    // LED
    ledcSetup(chR, LedFreq, LedResolution);
    ledcSetup(chG, LedFreq, LedResolution);
    ledcSetup(chB, LedFreq, LedResolution);
    pinMode(pinR, OUTPUT);
    pinMode(pinG, OUTPUT);
    pinMode(pinB, OUTPUT);
    ledcAttachPin(pinR, chR);
```

```
    ledcAttachPin(pinG, chG);
    ledcAttachPin(pinB, chB);

    // Temperatura
    pinMode(PTC, INPUT);

    // Set status
    pinMode(SWPin, INPUT_PULLUP);
    if(digitalRead(SWPin))
    {
        showStatus(LEDSTATUS::WIFI);
        op = OPMODE::NORMAL_OP;
    } else
    {
        showStatus(LEDSTATUS::RUNCONF);
        op = OPMODE::CONF_MODE;
    }
    #ifdef DEBUG
    Serial.print("[MSG] ");
    Serial.print("Modo \t");
    switch (op)
    {
    case OPMODE::NORMAL_OP:
        Serial.println("Normal");
        break;
    case OPMODE::CONF_MODE:
        Serial.println("Configuración");
    default:
        Serial.println("---");
        break;
    }
    #endif
```

```
// Entradas de configuración
for (byte i = 0; i < 4; i++)
{
    pinMode(DIP[i],INPUT_PULLUP);
    rfdata.conf |= digitalRead(DIP[i]) << i;
}
rfdata.conf = ~rfdata.conf;
#ifdef DEBUG
Serial.print("[MSG] ");
Serial.print("Configuración DIP: ");
for(int i=0; i<4; ++i)
{
    if((rfdata.conf >> i) & 0x01)
    {
        Serial.print("1");
    } else {
        Serial.print("0");
    }
}
Serial.println();
#endif

// Relés
pinMode(pinReleOut, OUTPUT);
pinMode(pinReleIn, INPUT_PULLUP);

// Wifi
switch (op)
{
case OPMODE::NORMAL_OP:
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
    }
    showStatus(LEDSTATUS::RUN);
    break;
case OPMODE::CONF_MODE:
    configurarWifi(); // WifiConf.h
    WiFi.softAP(ap_ssid, ap_pass);
    return; // Salimos del setup
    break;
}
```

```
    // Configuración del servidor
    server.on(R_message, get_api_message);
    server.on(R_message, HTTP_POST, post_api_message);
    server.on(R_read, get_api_read);
    server.on(R_api, get_api);
    server.on(R_api, HTTP_POST, post_api);
    server.onNotFound(notFound);
    server.begin();

    // RF
    RF.enableTransmit(RFPin);
    rfdata.empresa = 0x5f;
}
```

En la función *setup()* se realiza la configuración del dispositivo y se efectúa al comienzo del programa. Se realizan las siguientes acciones de forma secuencial:

- Se configura la comunicación serie para la depuración.
- Se configura el driver de los LEDs.
- Se configura la entrada del disparador PTC como entrada con pull-up.
- Se configura las entradas de configuración (pulsador y DIP) como entradas con pullup.
- Comprueba en qué modo de operación nos encontramos (estado del pulsador de configuración) y qué código está registrado en el DIP.
- Configura como entrada y salida los relés.
- Realiza la conexión a la red WiFi configurada o empieza a emitir una red dependiendo del modo de operación.
- Si está en modo de funcionamiento normal, realiza la configuración del servidor de la RESP API.
- Si está en modo de funcionamiento normal, realiza la configuración del driver del módulo RF. Asignamos además un valor de empresa (arbitrario) para el mensaje RF.

4.8.2.9 Loop

```
void loop() {
    switch (op)
    {
    case OPMODE::NORMAL_OP:
        normal();
        break;
    case OPMODE::CONF_MODE:
        conf();
    default:
        normal();
        break;
    }
}
```

En la función que define el bucle principal del programa (se repite indefinidamente) simplemente redirecciona a las funciones definidas en la sección de modos de operación.

4.9 EJEMPLOS DE FUNCIONAMIENTO

En el presente apartado se explicarán con detalle algunos ejemplos prácticos del funcionamiento del prototipo de emisor-receptor (alarma- brazalete) realizado en el proyecto.

4.9.1 Escenario 1: Incendio en una planta Industrial

4.9.1.1 Condiciones

- Planta industrial con uno o más trabajadores con discapacidad auditiva total o parcial.
- Grandes espacios abiertos separados por maquinaria, cintas de producción, etc.
- Ruido ambiental alto (50dB a 80dB).
- Sistema con bases RF instaladas a no menos de 200 metros unas de otras. Todas se encuentran configuradas y en modo de operación normal.
- Sistema de alarma contra incendio industrial conectados al sistema desarrollado (bases RF) mediante las entradas de relé NO.
- Los usuarios que lo requieran cuentan con una pulsera receptora

4.9.1.2 Disparo

Un fallo grave en el proceso de fabricación desemboca en un incendio. El sistema de alarma industrial se activa y dispara los relés de entrada de las diferentes bases con RF repartidas por la planta.

4.9.1.3 Actuación

El disparo del relé de entrada desemboca en las siguientes acciones por parte de las bases RF del sistema:

1. En el bucle principal del programa se detecta mediante la técnica de polling el disparo del relé.
2. El indicador luminoso pasa a color rojo.
3. Se activa el relé de salida. Válido para conexiones en cascada.
4. Se define el mensaje para enviar por RF como 0x01 (ALARMA) y se habilita el flag de 'All' para que el mensaje llegue a todos los usuarios sin filtro.
5. Se envía el mensaje por RF a todos los dispositivos.

Estas acciones se repiten en bucle cada 2.5 segundos mientras el sistema de alarma externo siga accionando el relé de entrada o los sensores del sistema estén disparados por las condiciones del incendio.

En las pulseras de los usuarios por su parte cuando reciben el mensaje por RF se desencadenan las siguientes acciones:

1. Una interrupción lanza la rutina del puerto de entrada de datos simulado por software mediante la librería "RFSwitch". Se habilita un flag que informa de que hay nuevos datos disponibles.
2. En el bucle principal del programa se detecta que hay nuevos datos y se leen del buffer y se decodifican.
3. Como hemos recibido el mensaje 0x01 (ALARMA) con el flag de 'All' activados se dispara la siguiente configuración:
    - LED RGB: Rojo.
    - Vibración: Encendida.
    - Parpadeo: Encendido.

Ese estado se perpetuará mientras se reciba el mismo mensaje y durante 5 segundos después de recibir el último mensaje. Como la base envía cada 2.5 segundos el mensaje de alarma, la pulsera no abandonará ese estado al menos mientras la base funcione.

4.9.1.4 Conclusiones

En un entorno de alta contaminación acústica las alarmas convencionales pueden no alertar con la suficiente contundencia a personas con discapacidad auditiva. Gracias al sistema diseñado en este caso reciben una señal vibratoria que les advierte de que algo sucede. La señal luminosa además les indica que se trata de una emergencia de primer nivel y que deben evacuar la planta.

4.9.2 Escenario 2: Emergencia hotel

4.9.2.1 Condiciones

- Hotel de varias plantas con algunos empleados y clientes con discapacidad auditiva o sensorial.

- Distribución por pasillos con acceso a habitaciones.
- Habitaciones con aislamiento acústico.
- Sistema de alarma contra incendios estándar conectado al sistema desarrollado (bases RF).
- Sistema con bases RF en cada planta del edificio alternando su situación en los pasillos (principio y final).
- Los empleados que lo requieran y los clientes que lo pidan cuentan con una pulsera receptora.

4.9.2.2 Disparo

Un usuario desconocido activa la alarma manualmente. Según el protocolo de seguridad las habitaciones deben ser evacuadas. El sistema de alarma dispara los relés de entrada de las bases RF de cada planta desarrolladas.

4.9.2.3 Actuación

La actuación es la misma que en el caso del escenario 1. El disparo del relé de entrada desemboca en las siguientes acciones por parte de las bases RF del sistema:

1. En el bucle principal del programa se detecta mediante la técnica de polling el disparo del relé.
2. El indicador luminoso pasa a color rojo.
3. Se activa el relé de salida. Válido para conexiones en cascada.
4. Se define el mensaje para enviar por RF como 0x01 (ALARMA) y se habilita el flag de 'All' para que el mensaje llegue a todos los usuarios sin filtro.
5. Se envía el mensaje por RF a todos los dispositivos.

Estas acciones se repiten en bucle cada 2.5 segundos mientras el sistema de alarma externo siga accionando el relé de entrada o los sensores del sistema estén disparados por las condiciones del incendio.

En las pulseras de los usuarios por su parte cuando reciben el mensaje por RF se desencadenan las siguientes acciones:

1. Una interrupción lanza la rutina del puerto de entrada de datos simulado por software mediante la librería "RFSwitch". Se habilita un flag que informa de que hay nuevos datos disponibles.

2. En el bucle principal del programa se detecta que hay nuevos datos y se leen del buffer y se decodifican.
3. Como hemos recibido el mensaje 0x01 (ALARMA) con el flag de 'All' activados se dispara la siguiente configuración:
    - LED RGB: Rojo.
    - Vibración: Encendida.
    - Parpadeo: Encendido.

Ese estado se perpetuará mientras se reciba el mismo mensaje y durante 5 segundos después de recibir el último mensaje. Como la base envía cada 2.5 segundos el mensaje de alarma, la pulsera no abandonará ese estado al menos mientras la base funcione.

4.9.2.4 Conclusiones

El sistema es válido tanto en entornos industriales como en el sector servicios.

4.9.3 Escenario 3: Incendio en el hogar

4.9.3.1 Condiciones

- Piso habitable de 90m2 con uno o más ocupantes con discapacidad auditiva total parcial.
- Distribución estándar con pasillo a diferentes habitaciones.
- El sistema desarrollado hace las veces de alarma contra incendios. Se ubica en la cocina.
- Un timbre se acopla al relé de salida de la base RF. Este hará de alerta sonora.
- Los ocupantes duermen. Aquellos que lo necesitan utilizan una pulsera.

4.9.3.2 Disparo

Uno de los convivientes olvida apagar el fuego de la cocina de gas que calienta una sartén y va adormir. Con el tiempo el aceite de esta pierde propiedades y prende. El humo y/o calor disparan el protocolo de alarma de la base RF de nuestro sistema.

4.9.3.3 Actuación

El disparo del relé de entrada desemboca en las siguientes acciones por parte de las bases RF del sistema:

1. En el bucle principal del programa se detecta mediante la técnica de polling el disparo del detector de humo o del termistor.
2. El indicador luminoso pasa a color rojo.
3. Se activa el relé de salida. En este caso hace sonar un timbre.
4. Se define el mensaje para enviar por RF como 0x01 (ALARMA) y se habilita el flag de 'All' para que el mensaje llegue a todos los usuarios sin filtro.
5. Se envía el mensaje por RF a todos los dispositivos.

Estas acciones se repiten en bucle cada 2.5 segundos mientras los sensores del sistema estén disparados por las condiciones ambientales del incendio.

En las pulseras de los usuarios por su parte cuando reciben el mensaje por RF se desencadenan las siguientes acciones:

1. Una interrupción lanza la rutina del puerto de entrada de datos simulado por software mediante la librería "RFSwitch". Se habilita un flag que informa de que hay nuevos datos disponibles.
2. En el bucle principal del programa se detecta que hay nuevos datos y se leen del buffer y se decodifican.
3. Como hemos recibido el mensaje 0x01 (ALARMA) con el flag de 'All' activados se dispara la siguiente configuración:

- LED RGB: Rojo.
- Vibración: Encendida.
- Parpadeo: Encendido.

Ese estado se perpetuará mientras se reciba el mismo mensaje y durante 5 segundos después de recibir el último mensaje. Como la base envía cada 2.5 segundos el mensaje de alarma, la pulsera no abandonará ese estado al menos mientras la base siga recibiendo señales de incendio y continúe operativa.

4.9.3.4 Conclusiones

El sistema es apto para el hogar. Personas con discapacidad auditiva que vivan solas pueden contar para ser alertadas en caso de emergencia. También es posible utilizar este sistema conectado a un sistema de alarma contra intrusos utilizando la entrada de relé como punto de entrada y configurando diferente la actuación frente a ese estímulo.

4.9.4 Escenario 4: Notificaciones en oficina

4.9.4.1 Condiciones

- Oficina de 200m2 con cubículos. Despachos y salas de reunión al fondo.
- Uno o más empleados con discapacidad auditiva cuentan con una pulsera.
- Lugar tranquilo. No se puede hacer ruido.
- En ocasiones un cliente llega y solicita una reunión con un agente en alguna de las salas de reunión.
- El sistema desarrollado hace las veces de alarma contra incendios y como centro de notificaciones.

4.9.4.2 Disparo

Un cliente llega a la oficina y requiere de una reunión con algún empleado con discapacidad auditiva. A través de una App realiza una petición HTTP a uno de los recursos del sistema para notificar al empleado que acuda a recepción.

4.9.4.3 Actuación

La petición HTTP a la API REST del sistema desemboca en las siguientes acciones por parte del sistema:

1. Una función call back se lanza dentro del ESP32 para gestionar la petición. En este caso se llama al recurso GET /api/message?type=14&target=1.
2. Copia los parámetros que se utilizarán (empleado=1; mensaje=0x0d).
3. El indicador luminoso pasa a color naranja.
4. Se define el mensaje para enviar por RF como 0x0d (Acudir a recepción) y se dirige al empleado con la pulsera #1.
5. Se envía el mensaje por RF.

Esta acción de notificar, a diferencia de las alarmas, se lanza una sola vez.

En las pulseras de los usuarios por su parte cuando reciben el mensaje por RF se desencadenan las siguientes acciones:

1. Una interrupción lanza la rutina del puerto de entrada de datos simulado por software mediante la librería "RFSwitch". Se habilita un flag que informa de que hay nuevos datos disponibles.
2. En el bucle principal del programa se detecta que hay nuevos datos y se leen del buffer y se decodifican.
3. Se discrimina el mensaje si no está dirigida a esa pulsera concreta. Si es la pulsera #1 continúa.
4. Como hemos recibido el mensaje 0x0d (Acudir a recepción) activados se dispara la siguiente configuración:
    - LED RGB: Cian.
    - Vibración: Encendida. Una vez.
    - Parpadeo: Encendido.

    Ese estado se perpetuará durante 5 segundos.

4.9.4.4 Conclusiones

Se puede utilizar la infraestructura del programa para realizar notificaciones sin contaminar el ambiente con sonidos innecesarios. El sistema en este caso puede ser útil incluso en personas sin discapacidad auditiva.

4.10 FABRICACIÓN DEL PROTOTIPO

Como fase final del desarrollo de la solución del presente proyecto se ha realizado la construcción del prototipo, basándose en todos los apartados anteriormente descritos.

El encargo de las PCBs se realizó en la empresa JLCPCB y el tiempo de envío se estimó en 15 días con valor de unos 30€ + gastos de envío. Los componentes principales se compraron a LCSC y Mouser.

Posteriormente, podemos resumir el proceso de soldadura de los componentes sobre la PCB en dos tipos:

- **Soldadura con cautín y estaño** - Es un tipo de soldadura blanda. Utilizamos un cautín a 300ºC para calentar el lugar de la PCB donde se debe soldar y la patilla del componente THT o SMD. A continuación, se introduce un hilo de estaño donde se juntan ambos elementos y el estaño se desliza entre ambos logrando una unión.
- **Soldadura por refusión -** Es otro caso de soldadura blanda, pero en este caso utilizamos una pasta fundente y una fuente de calor. Se coloca un poco de pasta (formada por estaño, plata y cobre y un fundente) sobre la huella del componente que queremos soldar en la PCB. Después colocamos el componente encima y mediante un horno de refusión se realiza un tratamiento térmico que deja tras de sí la unión limpia de ambos elementos. Este tipo de soldadura solo se ha usado para componentes SMD. Otros elementos que también han intervenido en la soldadura ha sido flux, cinta trenzada de cobre y pasta fundente para correcciones y esponja para limpiar el cautín.

Por último, la impresión de las carcasas se ha realizado con una impresora Anet A4 pro y material PLA color blanco. La duración de la impresión duró unas 15 horas entre las dos partes del proyecto.

Las siguientes figuras muestran el resultado final del prototipo.

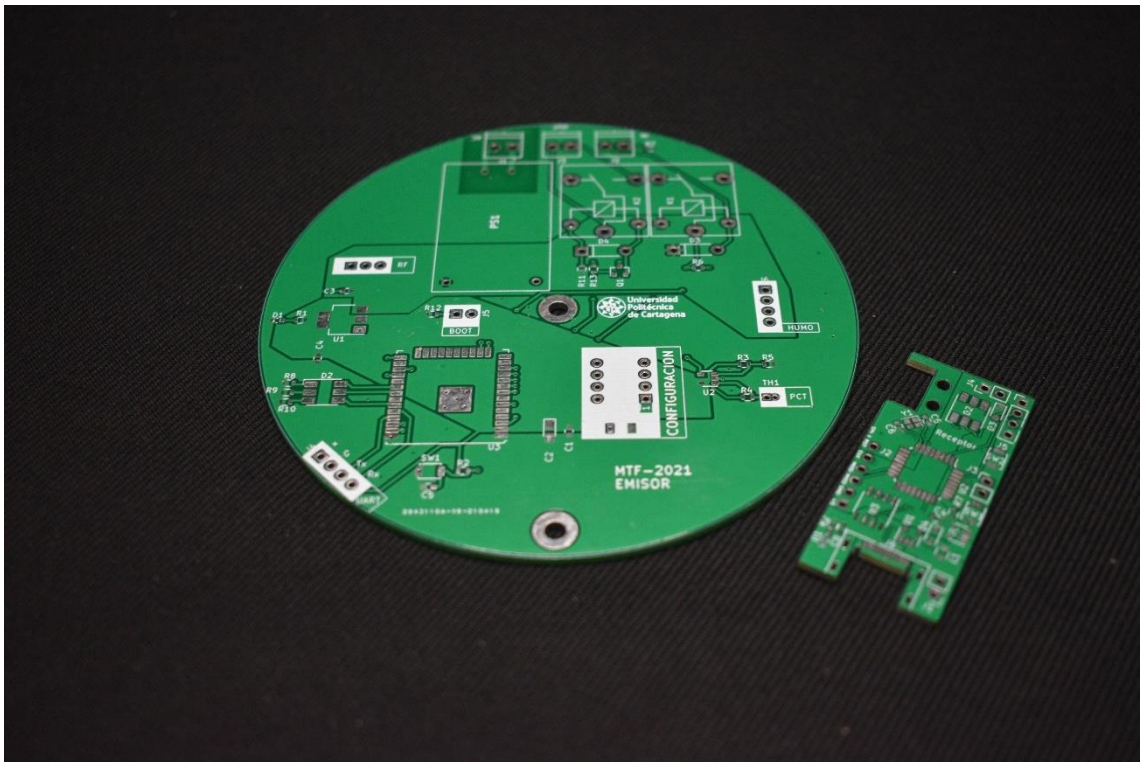

Figura 26: PCBs emisor y receptor

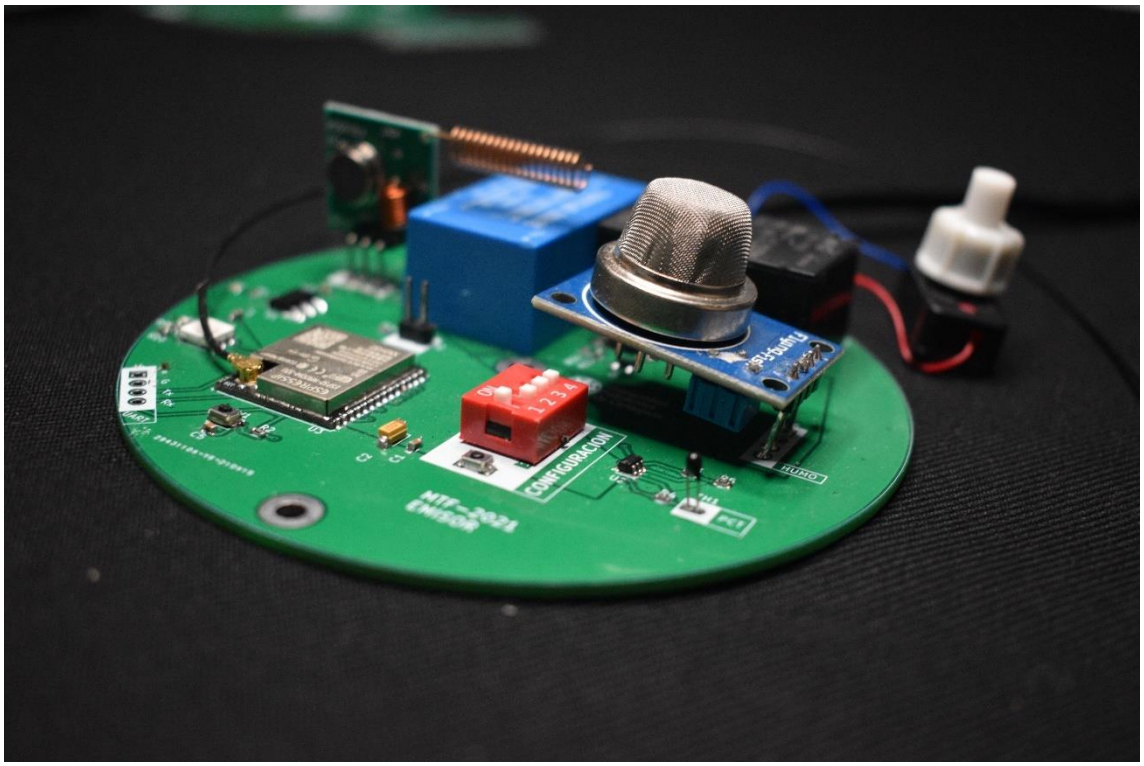

Figura 27: Prototipo emisor/alarma

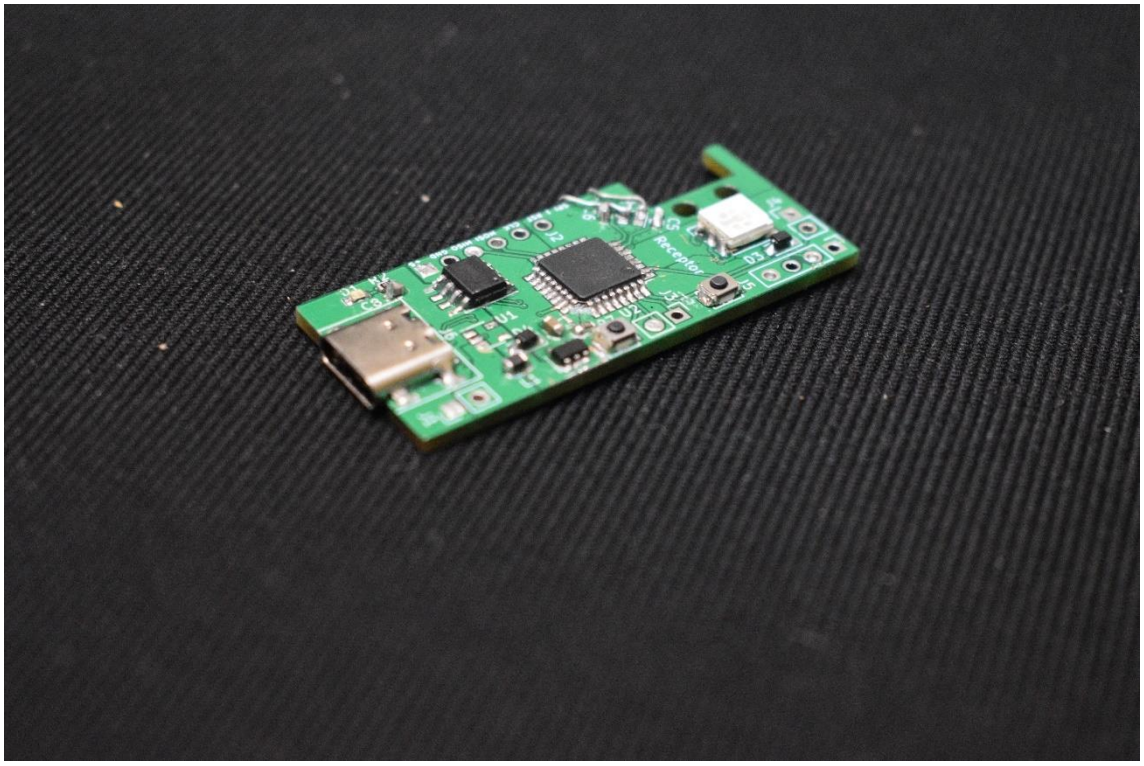

Figura 28: Prototipo receptor/pulsera

Figura 29: Prototipo final con carcasa 3D

5 PROYECTOS FUTUROS

La realización de este TFG ha dado como resultado el prototipo funcional descrito en anteriores secciones. Por tanto, la más lógica continuación del trabajo descrito vendría dada por la mejora y refinamiento del prototipo, la cual podría llevarse a cabo desde distintas perspectivas, en función de las limitaciones del mismo:

- Tamaño: las dimensiones actuales del prototipo lo hacen solo parcialmente útil para determinadas actividades profesionales. Ciertos entornos y tareas requerirían seguramente unas dimensiones más reducidas, que permitieran al empleado desempeñar su función sin ningún tipo de incomodidad causada por el dispositivo de alarma. Las dimensiones finales de la base-alarma son 105mm de diámetro y 45mm de altura y las dimensiones de la pulsera son 69x58x16 mm, el peso de los módulos es de 150gr y 100gr respectivamente. Estas podrían ser reducidas utilizando un diseño electrónico más eficiente, por ejemplo, con el uso de más componentes SMD en lugar de tipo through-hole, los cuales son adecuados para prototipado, pero no tanto para productos finales.
- Ergonomía: el prototipo final está planteado para ser montado como una pulsera-brazalete. Sin embargo, el tamaño es ciertamente voluminoso, sobre todo en comparación con otras opciones comerciales disponibles. La reducción en tamaño y una reorganización de la localización de los componentes en el PCB podrían recurrir a una reducción del volumen y del peso de la pulsera, haciendo que su uso resultase más cómodo.
- Alcance: las pruebas realizadas con el dispositivo han arrojado alcances de al menos 40-50 metros. Cabría estudiar si este alcance es adecuado para una mayoría de edificios industriales, o si por el contrario este resulta insuficiente. Una línea de investigación podría ser estudiar otras tecnologías de comunicación, como LoRaWAN u otras tecnologías de largo alcance típicamente utilizadas en IoT.
- Seguridad: para un uso plenamente comercial, sería necesario implementar mayores niveles de protección que garantizasen que un atacante externo no pudiese perturbar el funcionamiento del dispositivo

de alarma, lo cual podría tener consecuencias económicas graves debido a la paralización de la producción por parte de los trabajadores.

Más allá de las mejoras técnicas del prototipo, otras posibles vías de investigación futuras podrían ahondar en una mayor diversidad de problemas a los que se enfrenten las personas con diversidad funcional en el entorno laboral, y cómo la tecnología podría ayudarles.

Algunos ejemplos incluyen:

- Personas con un cierto porcentaje de ceguera: qué dispositivos podrían mejorar su visión para tareas concretas, como, por ejemplo, para visibilizar piezas pequeñas, distinguir adecuadamente componentes de distintos colores, o la correcta transmisión de información sobre el estado de funcionamiento de una máquina, utilizando señales sonoras en lugar de visuales.
- Personas con movilidad reducida: qué tecnologías, sobre todo desde la perspectiva de la robótica (CITAR The Berkeley lower extremity exoskeleton), podrían permitirles desempeñar las mismas tareas que una persona sin movilidad reducida, o incluso tareas más demandantes gracias a la ayuda de dispositivos externos.

En resumen, el prototipo presentado podría mejorarse sustancialmente, hasta alcanzar un estado más próximo a un producto comercial. Por otro lado, otras vías de investigación relacionadas, aunque no estrictamente similares podrías llevarse a cabo, dado que se trata de una temática donde existen amplias posibilidades de estudio y desarrollo de dispositivos para mejorar la integración social.

6 CONCLUSIONES

La presente memoria ha mostrado el desarrollo de un prototipo de sistema de alarma para la integración de personas con discapacidad auditiva y sensorial en el entorno laboral. Se ha realizado una revisión de las soluciones comerciales actualmente disponibles, así como la metodología utilizada actualmente por una empresa real para este propósito. Como parte del análisis realizado para el diseño del prototipo, diversos aspectos claves del mismo han sido analizados.

Las diferentes tecnologías de comunicación inalámbrica han sido comparadas, indicando sus distintas ventajas e inconvenientes. Así mismo, se han observado las distintas opciones de microcontrolador aptas para el desarrollo de dispositivos, siendo finalmente elegida la plataforma Arduino por su versatilidad. Por ende, la programación del controlador fue realizada en el lenguaje C++.

Para garantizar la idoneidad y facilidad de uso del dispositivo, se han realizado dos nuevos diseños PCB de módulos electrónicos específicos para el presente proyecto, incluyendo un módulo emisor (alarma) y un módulo receptor (pulsera-brazalete).

En base a la tecnología de comunicación elegida, RF, caracterizada por su simplicidad y bajo coste, los chips FS1000A y XY-MK-5V fueron elegidos para el envío y recepción de las señales de alarma, respectivamente. La comunicación entre los módulos emisor y receptor se modeló utilizando el paradigma REST, el cual ha permitido una fácil interpretación de los intercambios de mensajes, así como su potencial versatilidad de integración en entornos Web e IoT.

Para dotar al prototipo de un mayor grado de realismo, también fueron incluidos diversos componentes electrónicos adicionales. En módulo de alarma se equipó con un sensor de humo y un sensor de temperatura, los cuales son utilizados como entrada para determinar el estado de alarma. El módulo pulsera-brazalete fue equipado con un motor vibrador, el cual actúa como indicador de la alarma. Ambos módulos incluyen emisores LED, que cumplen diversas funciones, incluyendo depuración de la conexión e indicación de la alarma.

El prototipo fue puesto a prueba con éxito, mostrando la correcta funcionalidad del sistema de alarma ideado. El alcance de la comunicación inalámbrica

resultó ser de hasta 50 metros, suficiente para ciertos entornos industriales y correcto dada la naturaleza prototípica de los dispositivos.

El coste total del prototipo asciende en torno a 40€. Teniendo en cuenta el probable ahorro de una fabricación en masa, esto muestra la viabilidad económica de soluciones similares a la presentada para las empresas del sector industrial.

Estos resultados son alentadores, mostrando el amplio abanico de posibilidades aun existentes para la investigación y desarrollo de soluciones tecnológicas que faciliten la integración de las personas con diversidad funcional en la sociedad.

Finalmente, los posibles puntos de mejora y líneas de investigación futura fueron descritos, dejando abierta la posible continuación del presente proyecto.

7 BIBLIOGRAFÍA

[1] Referencias Arduino - https://www.arduino.cc/reference/en/

[2] Definición de REST - https://developer.mozilla.org/es/docs/Glossary/REST

[3] REST Guidelines - https://docs.microsoft.com/es-es/azure/architecture/best-practices/api-design

[4] APIs Web - https://developer.mozilla.org/es/docs/Learn/JavaScript/Client-side_web_APIs/

[5] ESP32 Hardware Design Guidelines - https://espressif.com/sites/default/files/documentation/esp32_hardware_design_guidelines_en.pdf

[6] ESP32 Technical Reference Manual - https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[7] ESP32 Datasheet - https://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[8] Repositorio Arduino Core ESP32 - https://github.com/espressif/arduino-esp32

[9] Datasheet Atmega328p - https://www.microchip.com/content/dam/mchp/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[10] Esquemático Arduino Nano - https://www.arduino.cc/en/uploads/Main/Arduino_Nano-Rev3.2-SCH.pdf

[11] TP4056 Design Notes - http://www.tp4056.com/datasheet/

[12] Power Management Guide 2018, Texas Instruments - https://www.ti.com/lit/sg/slvt145r/slvt145r.pdf?ts=1630460609733

[13] Webench(c) Power Designer - Texas Instruments - https://www.ti.com/design-resources/design-tools-simulation/webench-power-designer.html

[14] CH330 post - https://hackaday.com/2018/10/03/new-part-day-the-fifty-cent-usb-chip/

[15] BS870 datasheet - https://www.diodes.com/assets/Datasheets/ds11302.pdf

[16] B59052D1070A040 (PTC) datasheet - https://product.tdk.com/system/files/dam/doc/product/sensor/ptc/limit-temp/data_sheet/55/db/ptc/ptc_sensors_d1052.pdf

[17] AP1117 datasheet - https://www.diodes.com/assets/Datasheets/AP1117_R21.pdf

[18] MQ-02 Module datasheet - https://www.pololu.com/file/0J309/MQ2.pdf

[19] LM321 datasheet - https://www.ti.com/lit/ds/symlink/lm321.pdf

[20] Ley de Integración Social del Minusválido - https://www.boe.es/eli/es/rd/1985/12/04/2273