



TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA DE SISTEMES DE TELECOMUNICACIONES

DISEÑO DE UN DISPOSITIVO IoT DE MEDIDA DE DISTANCIA

María Belén Núñez Franco

TUTORES: Roger Malet i Munté y Carles Ferrer Ramís

DEPARTAMENTO DE TELECOMUNICACIÓN E INGENIERÍA DE SISTEMAS

UNIVERSIDAD AUTÒNOMA DE BARCELONA

Bellaterra, Julio 02, 2023

Abstract

The core idea of the project is to create a smart device connected to the Internet of Things (IoT) that has the ability to accurately calculate distances. Furthermore, this device would be able to send information about the measured distance using Bluetooth Low Energy (BLE) technology.

In order to achieve the set objectives, it has been necessary to acquire detailed knowledge about the configuration and operation of each component used. These components refer to the different functionalities that the microcontroller can offer and that we have taken advantage of in this project. In particular, we have used the following modules: UART, Timer and BLE. Each of these modules plays a specific role in the operation of the device and it has been essential to understand how they work in order to achieve our objectives.

In the annexes section we find the BLE architecture explained extensively, the components of the Pioneer KIT which is the one that has been used to carry out the project, this KIT contains the PSoC 4 BLE with the BLE radio incorporated into the microcontroller and the BLE receiver. Other annexes such as the one dedicated to the SRM400 sonar module and finally the PSoC Creator 4.4 program, which is the one used to program the microcontroller used.

Resumen

La idea central del proyecto es crear un aparato inteligente conectado a Internet de las cosas (IoT) que tenga la capacidad de calcular distancias de forma precisa. Además, este dispositivo sería capaz de enviar la información sobre la distancia medida utilizando la tecnología de Bluetooth de bajo consumo(BLE).

Con el fin de alcanzar los objetivos establecidos, ha sido necesario adquirir un conocimiento detallado sobre la configuración y operación de cada componente utilizado. Estos componentes se refieren a las diferentes funcionalidades que el microcontrolador puede ofrecer y que hemos aprovechado en este proyecto. En particular, hemos utilizado los siguientes módulos: UART, Timer y BLE. Cada uno de estos módulos desempeña un papel específico en el funcionamiento del dispositivo y ha sido fundamental comprender su funcionamiento para lograr nuestros objetivos.

En el apartado de los anexos encontramos la arquitectura del BLE explicada extensamente, los componentes del KIT Pioneer que es el que ha sido usado para realizar el proyecto, este KIT contiene el PSoC 4 BLE con la radio BLE incorporada al microcontrolador y el receptor BLE. Otros anexos como el dedicado al módulo sónar SRM400 y finalmente el programa PSoC Creator 4.4 que es con el que se ha podido programar el microcontrolador utilizado.

Resum

La idea central del projecte és crear un aparell intel·ligent connectat a Internet de les coses (IoT) que tingui la capacitat de calcular distàncies de manera precisa. A més, aquest dispositiu seria capaç d'enviar la informació sobre la distància mesurada utilitzant la tecnologia de Bluetooth de baix consum(BLE).

Amb la finalitat d'aconseguir els objectius establerts, ha estat necessari adquirir un coneixement detallat sobre la configuració i operació de cada component utilitzat. Aquests components es refereixen a les diferents funcionalitats que el microcontrolador pot oferir i que hem aprofitat en aquest projecte. En particular, hem utilitzat els següents mòduls: UART, Timer i BLE. Cadascun d'aquests mòduls exerceix un paper específic en el funcionament del dispositiu i ha estat fonamental comprendre el seu funcionament per a aconseguir els nostres objectius.

En l'apartat dels annexos trobem l'arquitectura del BLE explicada extensament, els components del KIT Pioneer que és el que ha estat usat per a realitzar el projecte, aquest KIT conté el PSoC 4 BLE amb la ràdio BLE incorporada al microcontrolador i el receptor BLE. Altres annexos com el dedicat al mòdul sónar SRM400 i finalment el programa PSoC Creator 4.4 que és amb el qual s'ha pogut programar el microcontrolador utilitzat.

Índice general

Abstract	I
Resumen	III
Resum	V
Lista de Figuras	XIII
Lista de Tablas	XV
1. Introducción	1
1.1. Propósito	2
1.2. Motivación y objetivos	3
1.3. Metodología	3
2. Estado del arte	5
2.1. Clasificación de tecnologías de medición de nivel	5
2.2. Tecnologías para la medición sin contacto	6
2.2.1. Radar	6
2.2.2. Sensores ultrasónicos	8
2.2.3. Sensores radiométricos	9
2.3. Elección de tecnología de medición	9
3. ¿Qué es Bluetooth Low Energy?	13

3.1. Breve comparación entre Bluetooth Classic y Bluetooth Low Energy	13
3.2. Arquitectura del BLE	14
4. Metodología y Desarrollo	17
4.1. Método de trabajo	17
4.2. Materiales	18
4.3. Módulo UART	18
4.3.1. El protocolo UART	18
4.3.2. Tera Term	20
4.3.3. Programa en PSOC Creator	20
4.4. Encender el sensor y calcular la medida de distancia	23
4.4.1. Prueba para comprender el funcionamiento del Timer	24
4.4.2. Encendemos el módulo SRM400 y escuchamos la señal de ECHO	28
4.5. Configuración del módulo BLE	33
5. Resultados y Conclusiones	45
A. Módulo de alcance sonar SRM400	49
A.1. Especificaciones del módulo	50
A.2. El circuito integrado PW-0268	50
A.3. Funcionamiento del circuito integrado PW-0268	51
A.3.1. Ejemplo con gráficas	52
A.4. Características del circuito integrado PW-0268	53
A.5. Diagrama de bloques	54
A.6. Asignación de pines	54
A.6.1. Descripción de los pines	55
B. Arquitectura del BLE	57
B.1. Controlador	58

B.1.1. Capa física	58
B.1.2. Capa de enlace	60
B.1.3. La interfaz entre Host y Controlador:	61
B.2. Host	62
B.2.1. La lógica de conexión y protocolo de adaptación (L2CAP)	62
B.2.2. Protocolo de gestor de seguridad	63
B.2.3. El protocolo de atributos	63
B.2.4. Protocolo de gestión de Seguridad (SMP)	64
B.2.5. El perfil genérico de atributo (GATT)	65
B.2.6. El perfil genérico de acceso (GAP)	66
B.3. La capa de aplicación	67
C. Kit de desarrollo y empresa Cypress	69
C.1. CYPRESS	69
C.2. Detalles de la placa base	70
C.3. Módulo CY8CKIT-143A PSoC 4 BLE 256 KB	71
C.4. CYPRESS	73
C.5. PSoC	73
C.6. PSoC 4 BLE	74
C.7. PSoC Creator 4.4	75
D. Selección del microcontrolador	77

Índice de figuras

1.1. Diagrama funcional del dispositivo	2
3.1. Diferentes capas de la arquitectura BLE	15
4.1. Tera Term detecta el microcontrolador	20
4.2. Configuración de Tera Term	21
4.3. Configuración de UART	21
4.4. Módulo UART y esquema usado	22
4.5. Pines en el proyecto UART	22
4.6. Número en Tera Term	23
4.7. Circuito datasheet	24
4.8. Esquemático temporizador	25
4.9. Seleccionamos el modo Timer	25
4.10. Configuramos el Timer	26
4.11. Configuramos los pines del Timer	26
4.12. Resultado del Timer en Tera Term	28
4.13. Resultado en Tera Term con el valor del auxiliar	28
4.14. Dispositivo IoT	29
4.15. Esquema proyecto sensor	29
4.16. Configuración pestaña General del BLE	34
4.17. Configuración pestaña Perfil del BLE	34

4.18. Configuración característica Distancia	35
4.19. Configuración descriptor de característica del usuario	35
4.20. Configuración de la pestaña GAP Settings	36
4.21. Configuración de la pestaña GAP Settings	37
4.22. Configuración de la pestaña GAP Settings	37
4.23. Configuración de la pestaña GAP Settings	38
4.24. Configuración de la pestaña GAP Settings	38
4.25. Configuración de la pestaña L2CAP	39
4.26. Configuración de la pestaña Advanced	39
4.27. Configuración de la pestaña Built-in	40
5.1. Dispositivo IoT	45
5.2. Dispositivo IoT conexiones	46
5.3. Módulo SRM400	46
A.1. Pulso de control del μP	52
A.2. Señal de ráfaga de tonos	52
A.3. Carga del transductor	52
A.4. Preamplificador	52
A.5. Amplificador de 32 pasos	52
A.6. Salida digital de la señal ECHO	52
A.7. Diagrama de bloques [2]	54
A.8. Asignación de pines [2]	54
B.1. Capas que forman la arquitectura del BLE	58
B.2. Distribución de canales	59
B.3. Paquete de datos	60
B.4. Estados de la capa de enlace	61
B.5. Paquete L2CAP	63

Lista de figuras

C.1.	Contenido del Kit	69
C.2.	Placa base	70
C.3.	Placa PSoC 4 BLE	71
C.4.	Dongle	71
C.5.	Placa PSoC 4 BLE	75
C.6.	PSoC Creator 4.4	76

Índice de cuadros

2.1. Tabla comparativa de especificaciones del sensor de radar AWR1843 [4] y el módulo ultrasónico SRM400 [8].	11
A.1. Tabla descriptiva de las especificaciones	50
A.2. Tabla descriptiva de los pines del IC PW0268	55
C.1. Contenido del PIONEER KIT	69

Capítulo 1

Introducción

El proyecto consiste en el diseño e implementación de un dispositivo IoT que sea capaz de medir nivel y enviar la información utilizando Bluetooth Low Energy (BLE). Para llevar a cabo este proyecto utilizaremos el módulo **CY8CKIT-042-BLE**, que pertenece a la familia de microcontroladores desarrollados por la empresa Cypress, comercialmente es llamado **PSoC** (Programmable System on Chip).

Al PSoC le conectamos un pequeño circuito formado por el módulo SRM400 que está diseñado para aplicaciones de sónar, con él implementamos la función de medida de distancia. La programación del módulo se realiza mediante la herramienta **PSoC Creator 4.4**. Finalmente, los datos son enviados por Bluetooth al dispositivo móvil u ordenador donde veremos los resultados de la medición.

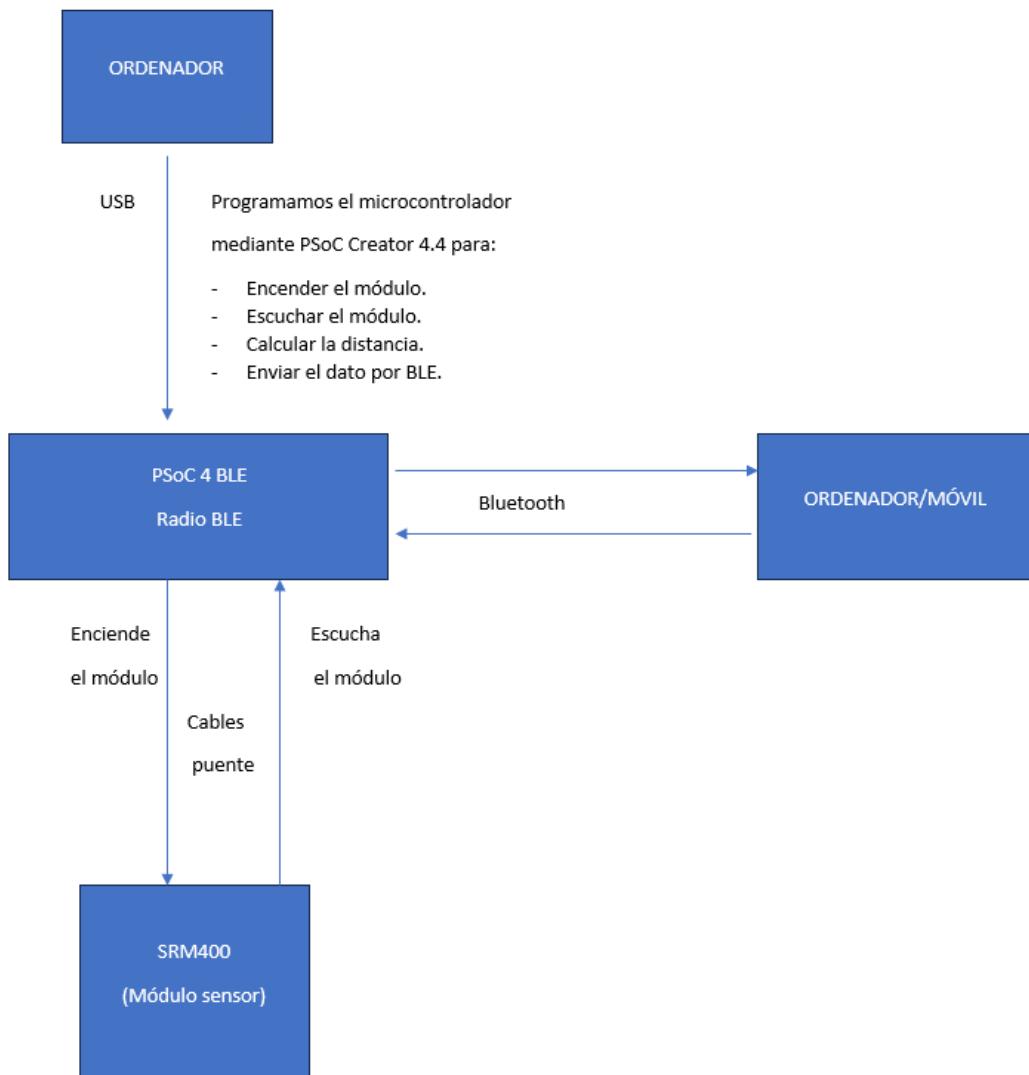


Figura 1.1: Diagrama funcional del dispositivo

1.1. Propósito

El propósito de este Trabajo de Fin de Grado es el diseño e implementación de un dispositivo IoT, capaz de comunicarse inalámbricamente, medir el nivel de sólidos y líquidos, y además que sea de bajo consumo. Para ello se empleará el PSoC 4 BLE que incorpora la tecnología Bluetooth Low Energy para la comunicación y la plataforma PSoC Creator para programar el microcontrolador.

1.2. Motivación y objetivos

La motivación principal para este proyecto es establecer una conexión, a través de la tecnología inalámbrica de baja potencia BLE, entre un dispositivo móvil y el microcontrolador PSoC 4 BLE, a través del cual se obtienen los datos de distancia, mediante el sensor de proximidad que van conectados a dicho microcontrolador. Para un desarrollo exitoso de este Trabajo de Fin de Grado, se plantearon los siguientes objetivos:

- Estudio de la tecnología BLE: Estudiar y comprender en qué consiste esta tecnología para posteriormente utilizar el conocimiento adquirido en el prototipo.
- Diseño y desarrollo del sensor de distancia: Diseño y desarrollo de un dispositivo IoT utilizando el Kit Pioneer, el módulo SRM400, transductores de ultrasonido, etc.
- Evaluación mediante experimentación del dispositivo final obtenido: La evaluación consistirá en diversas tomas de medidas de distancia con diferentes objetos o fluidos y el envío de estos datos a través de BLE.

1.3. Metodología

Pasos que se han llevado a cabo para poder obtener el prototipo:

- Estado del arte de los sistemas de medida inalámbricos.
- Estudio y análisis de la tecnología Bluetooth Low Energy.
- Estudio y selección de los componentes necesarios para el desarrollo del sensor.
- Desarrollo con PSOC Creator 4.4 dividiendo el proyecto en pequeños proyectos cumpliendo en cada uno de ellos un objetivo o característica del proyecto principal.
- Evaluación.
- Conclusión.

Capítulo 2

Estado del arte

2.1. Clasificación de tecnologías de medición de nivel

Los medidores de nivel se utilizan en procesos de líquidos y sólidos, principalmente se clasifican de la siguiente manera:

Medidores de punto fijo: se usan para marcar cierta altura cuando llega a una condición de nivel. Este tipo de sensor suele funcionar como una alarma, señalando cuando hay una condición de exceso de llenado o como una condición de bajo nivel.

Medidores continuos: se utilizan para medir el nivel en diferentes valores ya que son capaces de medir el nivel en un rango y no solamente un punto como los anteriores.

La siguiente clasificación que haremos será la de contacto o no contacto:

Medidores de nivel de contacto: cuando el material entra en contacto con el instrumento se activa una señal para indicar el nivel del material.

Medidores de nivel sin contacto: estos instrumentos no necesitan el contacto con el material para poder realizar la medición.

En el siguiente apartado analizaremos las tecnologías de medición sin contacto que son aquellas que son de interés para este proyecto, concretamente:

- Radar
- Ultrasónicos
- Radiométricos

2.2. Tecnologías para la medición sin contacto

2.2.1. Radar

El funcionamiento del Radar se basa en la emisión de una señal electromagnética, que se refleja en el objetivo y es recibida por el receptor que se encuentra en la misma posición que el emisor. La distancia se determina usando como referencia el tiempo en que la señal de alta frecuencia es transmitida y recibida además de la velocidad de la luz c_0 .

La operación a realizar es la siguiente:

$$d = \frac{t}{2}c_0 \quad (2.1)$$

- d= distancia
- t= tiempo
- c_0 = velocidad de la luz

Las ondas electromagnéticas del radar no necesitan de ningún medio para propagarse, pueden propagarse en el vacío, por esta razón la tecnología radar no se ve afectada por variaciones de temperatura y presión en el medio.

Hay diferentes tecnologías de Radar para la medición de nivel.

Radar de onda continua (CW): Utiliza una señal de radar de frecuencia constante para medir el nivel de los fluidos. Emite una señal de forma continua y mide la señal reflejada para determinar la distancia al líquido. [1].

Es una tecnología precisa y es usada en aplicaciones que requieren alta precisión. El radar de onda continua se utiliza comúnmente para medir niveles de líquidos en tanques y recipientes en la industria química, petroquímica, alimentaria y farmacéutica. Esta tecnología es adecuada para medir líquidos en tanques grandes y pequeños, y es capaz de medir niveles de líquidos en condiciones extremas, como altas temperaturas y presiones, y en líquidos corrosivos. [11].

Radar de frecuencia modulada (FMCW): Utiliza una señal de radar modulada en frecuencia para medir la distancia del objeto. En esta tecnología la frecuencia de la señal transmitida varía constantemente y se mide la frecuencia de la señal reflejada para determinar la distancia al objeto. [5].

El radar FMCW se utiliza comúnmente para medir niveles de líquidos en tanques y recipientes en la industria química, petroquímica, alimentaria y farmacéutica. Además de la medición de

2.2. Tecnologías para la medición sin contacto

niveles de líquidos, el radar FMCW se utiliza también para la medición de distancia, detección de objetos, y la monitorización del movimiento de personas y vehículos.

El radar FMCW es especialmente útil para la medición de niveles de líquidos en tanques y recipientes, ya que esta tecnología es capaz de medir con alta precisión el nivel de líquidos, incluso en condiciones extremas como altas temperaturas y presiones, y en líquidos corrosivos. Además, el radar FMCW es capaz de medir con alta precisión el nivel de líquidos en tanques con agitación, lo que puede ser un desafío para otras tecnologías de medición. [12].

El radar pulsado: utiliza impulsos de alta potencia y corta duración para transmitir la señal. El radar emite un pulso de energía electromagnética y luego espera a que la señal reflejada regrese al radar antes de enviar un nuevo pulso. La distancia al objetivo se calcula midiendo el tiempo que tarda la señal en viajar desde el radar hasta el objeto de regreso.

El radar pulsado se utiliza en una amplia variedad de aplicaciones, incluyendo la detección de objetos en el aire, en el mar y en tierra, la medición de la velocidad y la dirección del viento, la monitorización del movimiento de personas y vehículos, y en la medición de niveles de líquidos en tanques. [13].

El radar de banda ultra ancha (UWB): utiliza pulsos de radio de muy corta duración y ancho de banda extremadamente ancho. La característica principal es que su ancho de banda es mayor al 25 % de la frecuencia central utilizada y una duración de pulso inferior a los 2 nanosegundos.

El radar UWB tiene diversas aplicaciones, como la detección de obstáculos en vehículos, la medición de la distancia y la ubicación de objetos en entornos militares, la localización de personas y la detección de movimientos en entornos de seguridad, la medición de niveles en contenedores y la detección de movimientos de la tierra en aplicaciones geofísicas. También se utiliza en sistemas de comunicación de corto alcance y en aplicaciones médicas, como la detección de objetos extraños en el cuerpo humano y la monitorización de signos vitales. [10].

Ventajas:

- Medición sin contacto ni mantenimiento.
- Insensible a propiedades del producto como la densidad y la conductividad, en sólidos granulados también insensible al ruido de llenado y el polvo.
- Rango de medición de ajuste libre.
- Para altas temperaturas de hasta +450 °C/+842 °F.

2.2.2. Sensores ultrasónicos

Los sensores ultrasónicos emiten ondas mecánicas (ultrasonidos) que se reflejan cuando encuentran un cambio de densidad. El tiempo que tarda la onda emitida en volver al sensor, tiempo de vuelo, es medido por el transmisor y éste se utiliza para medir el nivel, de la misma forma que el funcionamiento explicado anteriormente del radar, la única diferencia es que aquí las ondas emitidas son ultrasónicas, las ondas de ultrasonidos son ondas mecánicas por lo que necesitan de un medio para propagarse.

La velocidad de propagación del sonido dependerá de las características del medio en el que viaja. Si el medio varía por cambios de temperatura, presión, composición, densidad, la velocidad de propagación del sonido puede variar. Como hemos comentado, la variación de temperatura en el medio hará variar la velocidad de propagación del sonido. La mayor parte de los sensores ultrasónicos disponen de compensación de temperatura por lo que este hecho no les afectará. [6].

Ventajas:

- Disponibilidad de sensores no intrusivos para evitar problemas de corrosión y contaminación. Medición continua y puntual.
- No posee partes móviles. Menor mantenimiento.
- Se utiliza para líquidos y sólidos, conductivos y no conductivos.

Desventajas:

- La medición puede ser afectada por movimiento del material en el tanque.
- La espuma del líquido puede absorber la señal transmitida.
- La presencia de partículas o vapor en el aire puede interferir la señal de los sensores de tipo no intrusivo.

2.2.3. Sensores radiométricos

Para poder realizar la medición usa una fuente radiactiva de rayos gamma. El sensor situado en el lado opuesto del depósito, recibe esa radiación. Basado en la atenuación de la radiación gamma al penetrar la materia, el sensor puede calcular el nivel, el nivel límite, la densidad o el caudal a partir de la intensidad de la radiación entrante.

La medición del nivel radiométrico mediante el uso de un contenedor de fuente radiactiva proporciona una fuente confiable de medición en situaciones en las cuales otros principios de medición no son viables debido a condiciones extremas de proceso o por limitaciones mecánicas, geométricas o constructivas. Los sensores radiométricos, que emplean la emisión de rayos gamma, están diseñados específicamente para detectar el nivel sin contacto y para llevar a cabo mediciones continuas de nivel, interfase y densidad en aplicaciones que involucran líquidos, sólidos, sólidos en suspensión o fangos.

La fuente de rayos gamma, un isótopo de cesio o cobalto emite radiación gamma que se atenúa al pasar a través de los materiales. El efecto de medición se obtiene a partir de la absorción de radiación del producto que se mide, causada por los cambios de nivel. El sistema de medición consiste en una fuente, un contenedor de fuente radiactiva y un transmisor compacto como receptor. [7].

Debido a que nuestro objetivo no serán las sustancias o materiales más complicados de medir, no nos extenderemos en el análisis de esta tecnología de nivel.

2.3. Elección de tecnología de medición

Finalmente, después del estudio sobre las tecnologías de medición sin contacto hemos llegado a la conclusión de que las tecnologías que debemos comparar son la ultrasónica y el radar por la similitud de características y las aplicaciones en las que suelen utilizarse.

Diferentes ondas transmisoras

El medidor de nivel radar es capaz de emitir ondas electromagnéticas que no requieren de un medio material para su propagación. Por otro lado, el sensor de nivel ultrasónico emplea ondas sonoras que sí requieren de un medio material para su propagación, por lo que resulta ineficaz en ambientes de vacío. En este sentido, es posible afirmar que la sonda de nivel de radar es capaz de proporcionar mediciones de nivel en entornos de vacío, lo que no es posible con el sensor de nivel ultrasónico.

Temperatura de operación

Es importante destacar que el medidor de nivel ultrasónico presenta limitaciones significativas

en cuanto a su capacidad de medición en medios con temperaturas superiores a los 80°C. Esto se debe, en gran medida, a que el sensor ultrasónico es altamente sensible a las condiciones ambientales, como la temperatura, que pueden afectar la velocidad de propagación de las ondas sonoras y, por ende, la precisión de las mediciones.

En contraste, los transmisores de nivel de radar presentan una mayor versatilidad en cuanto a su capacidad para operar en medios con temperaturas elevadas, ya que pueden funcionar adecuadamente en ambientes con temperaturas de hasta 250°C. Esto se debe a que la tecnología de radar no depende de la propagación de ondas sonoras, sino que se basa en la emisión y recepción de ondas electromagnéticas, lo que les permite funcionar en una amplia variedad de entornos y condiciones.

Ambiente polvoriento

Es importante destacar que el medidor de nivel ultrasónico presenta limitaciones significativas en entornos polvorrientos, lo que puede afectar negativamente la calidad y precisión de las mediciones obtenidas. En contraste, el medidor de nivel de radar ofrece un mejor rendimiento en condiciones ambientales polvorrientas, lo que lo convierte en una opción más adecuada para este tipo de entornos. En resumen, mientras que el medidor de nivel ultrasónico puede ser menos efectivo en ambientes con polvo, el medidor de nivel de radar puede proporcionar mediciones más precisas y confiables en estas condiciones.

Precisión

Es importante destacar que el medidor de nivel de radar ofrece una precisión significativamente mayor que la proporcionada por los sensores de nivel ultrasónicos. De hecho, cuando se trata de medir el nivel de los tanques de almacenamiento, los medidores de nivel de radar de alta precisión suelen ser la opción preferida debido a su capacidad para proporcionar mediciones altamente precisas y confiables.

En términos de coste, es importante señalar que el medidor de nivel ultrasónico suele ser más económico que el sensor de nivel de radar. Sin embargo, es importante tener en cuenta que el mayor costo del medidor de nivel de radar se justifica por su mayor precisión y fiabilidad, lo que puede resultar fundamental en aplicaciones críticas y exigentes en términos de medición de nivel. En este sentido, la elección del tipo de medidor de nivel dependerá de la aplicación específica y de las necesidades de medición de cada usuario.

Las conclusiones a las que llegamos tras el estudio de las tecnologías de medición con ultrasonidos y radar son las siguientes:

1. El medidor de nivel de radar tiene un rango de medición mucho más amplio que el sensor de nivel ultrasónico.

2.3. Elección de tecnología de medición

2. La precisión ultrasónica no es tan buena como el sensor de nivel de radar.
3. Los transmisores de nivel de radar son más caros que los transmisores de nivel ultrasónico.
4. Al usar el medidor de nivel de radar, debemos considerar la constante dieléctrica del medio.
5. La sonda de nivel ultrasónico no debe utilizarse en entornos de vacío, entornos de vapor o si hay espuma en la superficie del líquido.

Especificación	Módulo AWR1843	Módulo SRM400
Tecnología	Radar	Ultrasónico
Frecuencia de Operación	76-81 GHz	40 kHz
Alcance	Hasta 16 m	Hasta 1,5 m
Precisión	+/- 3 cm	+/- 1 cm
Resolución	1 cm	0,1 cm
Campo de Visión	120 grados	60 grados
Consumo de Energía	2 W	0,2 W
Tamaño	75 x 46 x 9,5 mm	55 x 45 x 20 mm
Coste	31,15	28,76

Cuadro 2.1: Tabla comparativa de especificaciones del sensor de radar AWR1843 [4] y el módulo ultrasónico SRM400 [8].

Las secciones de precisión y resolución pueden hacernos pensar que el de ultrasonidos es más preciso y con mayor resolución, sin embargo, estamos hablando en términos absolutos ya que no es lo mismo 1 cm en 16 m que 0,1 cm en 1,5 m y lo mismo sucede con la precisión.

Esta tabla comparativa deja claro que los radar tienen ventajas de medición ante los sensores de ultrasonidos, sin embargo, este proyecto se centraba en crear un dispositivo para la medición de nivel donde no es necesario tener un gran alcance, ya que por ej un caso de uso podría ser notificar cuando el líquido o sólido se encuentre a 0.5 metros del sensor es más que suficiente.

Otra de las razones de la elección de ultrasonidos ante el radar fue la característica del bajo consumo de energía ya que es uno de los objetivos de este proyecto, por tanto, observando los consumos de ambos módulos era conveniente la elección del módulo de ultrasonido ya que su consumo es 90 % menos que el de radar y además también tiene un menor coste.

Capítulo 3

¿Qué es Bluetooth Low Energy?

3.1. Breve comparación entre Bluetooth Classic y Bluetooth Low Energy

Para empezar a hablar de BLE (Bluetooth Low Energy) debemos empezar hablando de Bluetooth Classic, el cual llamaremos Bluetooth de aquí en adelante, que es su predecesor. Con el avance de la tecnología y aparecer más casos de uso, era necesario más ancho de banda, esto hizo que se añadieran radios más rápidas al ecosistema Bluetooth. La primera velocidad de Bluetooth fue la básica (BR) con un máximo de 1 Mbps, después la velocidad de datos mejorada (EDR) en la versión 2.0 con un máximo de 3 Mbps. Y así fue mejorando la velocidad hasta llegar a la actualidad con la versión 5.2 que, además de otros beneficios, nos permite una tasa de 50 Mbps.

Sin embargo en la versión de Bluetooth 4.0 o Bluetooth Smart después conocido como BLE no toma esta dirección, no quiere aumentar solamente la velocidad de datos sino también optimizar el uso de recursos para poder ofrecer un consumo de energía ultrabajo. Este cambio de dirección es debido a que el Bluetooth no puede cumplir los requisitos de bajo consumo exigidos a los dispositivos alimentados por pilas de botón.

Otro de los objetivos de BLE es poder desplegarse en volúmenes altos, en dispositivos que actualmente no disponen de ninguna tecnología inalámbrica, para poder lograr esto el coste debe ser extremadamente bajo. Es importante considerar también el diseño del sistema BLE desde la perspectiva del bajo coste, hay tres elementos clave que debemos mencionar:

- Banda ISM: La banda 2.4 GHz es una banda terrible para las tecnologías inalámbricas porque tiene pobres características de propagación, la energía radioeléctrica es absorbida fácilmente por todo, especialmente por el agua y el cuerpo humano está mayormente

formado por agua. Pero también hay que tener en cuenta que este espectro está disponible en todo el mundo además de que no es necesario una licencia. Evidentemente la ausencia de licencia va a hacer que otras tecnologías también la usen, por ejemplo la mayoría de radios WIFI, además de otras tecnologías hay limitaciones de potencia limitando el alcance. Sin embargo estas limitaciones e inconvenientes son más atractivas que pagar por el espectro bajo licencia, por tanto, usar la banda ISM abarata el coste.

- Licencia IP: Nokia trajo la tecnología Wibree o BLE a SIG (Special Interest Group) debido a su excelente reputación y la política de licencias de esta organización. Esto significó que los costes de las licencias de patentes se reducen significativamente para un dispositivo Bluetooth en comparación con una tecnología desarrollada en otro SIG o asociación que tenga una política FRAND⁴. Gracias al bajo coste de licencia, el coste por dispositivo también se reduce.
- Baja potencia: para diseñar un dispositivo de bajo coste debemos reducir los materiales necesarios para producirlo por ejemplo las pilas. Si la pila es grande la carcasa también tendrá que serlo, aumentando los costes. La sustitución de la pila también genera costes al tener que adquirir otra además de la pérdida de beneficios por la inactividad del dispositivo. Entonces diseñar la tecnología en torno al bajo consumo reduce costes.

Conclusión BLE se diseñó bajo la máxima de ser usado con pilas de botón, las más pequeñas, más baratas y el tipo de batería más fácil de adquirir por su alta disponibilidad. La mayor diferencia entre el Bluetooth y el BLE es la tasa de datos ya que si queremos un bajo consumo no podemos priorizar una alta velocidad de datos. Bluetooth tenía un objetivo de diseño de unos días en espera y unas horas de conversación para unos auriculares, mientras que BLE tiene un objetivo de diseño de unos pocos años para un sensor que mide la temperatura o la distancia que has caminado. ha caminado.

3.2. Arquitectura del BLE

La arquitectura de BLE es muy sencilla, se divide en tres partes básicas: controlador, host y aplicación.

El controlador suele ser un dispositivo físico que puede transmitir y recibir señales de radio y entender cómo estas señales pueden interpretarse como paquetes que contienen información.[9]

El host suele ser una pila de software que gestiona la comunicación entre dos o más dispositivos y cómo se pueden proporcionar varios servicios diferentes al mismo tiempo a través de las radios.

Las aplicaciones utilizan la pila de software, y por tanto el controlador, para habilitar un caso

3.2. Arquitectura del BLE

de uso. En la siguiente imagen podemos ver las diferentes capas que se encuentran en las partes que forman el protocolo BLE.

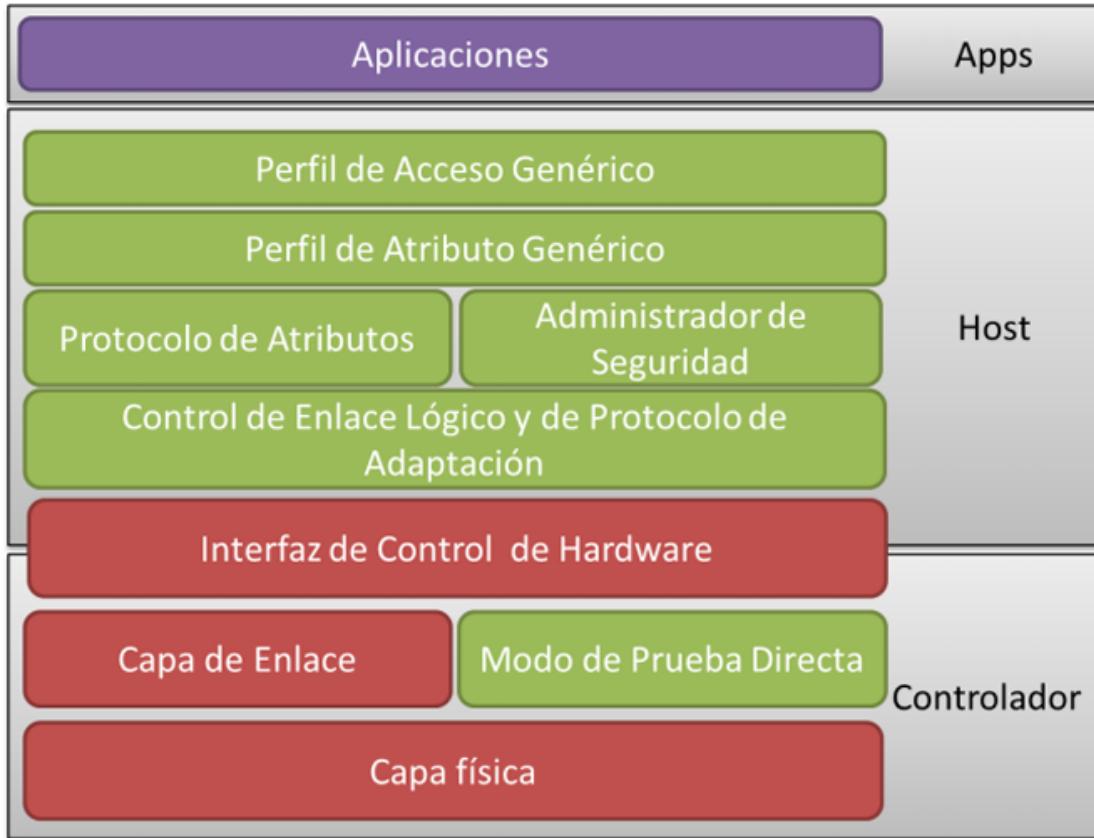


Figura 3.1: Diferentes capas de la arquitectura BLE

Capítulo 4

Metodología y Desarrollo

En este capítulo, abordaremos aspectos fundamentales relacionados con el desarrollo del dispositivo, centrándonos específicamente en el método de trabajo empleado, los materiales requeridos y las diferentes etapas del desarrollo. La fuente de información que hemos usado para comprender el funcionamiento de cada módulo han sido los datasheets de cada módulo que los proporciona PSoC Creator.

4.1. Método de trabajo

Para poder obtener los datos de medida y enviarlos por BLE, decidimos dividir el proyecto en diferentes partes con el objetivo de ir entendiendo los diferentes módulos y cubriendo las necesidades del proyecto.

Nuestro primer objetivo consistía en comprender el funcionamiento del módulo UART para poder visualizar los datos por pantalla, este dato lo utilizaríamos más adelante para poder comprobar que el dato enviado por el BLE es el correcto.

La segunda parte consistía en comprender el funcionamiento del módulo temporizador y poder obtener la medida precisa del tiempo y así poder calcular la distancia.

La tercera parte tenía como objetivo encender el sensor y obtener las medidas.

La última parte consistía en entender el módulo BLE y conseguir que el dispositivo se anunciara, conectara y fuera capaz de enviar datos por BLE.

Finalmente uniendo todas estas partes conseguíamos obtener el proyecto principal con todas las funciones que le proporcionan cada una de las partes.

4.2. Materiales

Hardware

- Ordenador con Windows 10.
- Placa base BLE Pioneer precargada con el módulo CY8CKIT-143A PSoC 4 BLE de 256 KB.
- CY5677 Receptor CySmart BLE 4.2 USB (receptor BLE).
- Módulo SRM400.
- Protoboard.
- Cables.
- Resistencia de $10\text{ k}\Omega$.

Software

- PSoC Creator 4.4
- CySmart 1.3
- Tera Term
- PSoC Programmer 3.29.5

4.3. Módulo UART

El módulo UART es el que nos permitirá visualizar los datos que enviamos por pantalla, tendrá dos funciones principales, el primero de ellos es poder ver el cálculo de la distancia que estamos realizando antes de poder enviarlo por BLE ya que el envío de datos por UART es más sencillo que por BLE y su segunda función será que nos permitirá comprobar que el dato que enviamos por BLE es el correcto.

4.3.1. El protocolo UART

El protocolo UART (Universal Asynchronous Receiver-Transmitter) es un estándar de comunicación que se utiliza para la transmisión de datos entre dos dispositivos de manera serial, los dispositivos que se van a comunicar en nuestro caso son el microcontrolador y el

4.3. Módulo UART

ordenador. La comunicación serial es la forma en la que se envían los bits, secuencialmente y a través de un único canal de comunicación, el canal de comunicación físico que usamos es cable serie.

Debido a que es una comunicación asíncrona no es necesario una señal de reloj compartida entre los dispositivos que se comunican entre sí, sin embargo, deben poder sincronizar de alguna manera el envío de datos para ello utiliza bits de inicio y bits de parada para poder diferenciar los paquetes de datos y así el receptor sea capaz de interpretarlos y sincronizar su recepción.

Formato de paquetes de la comunicación UART:

- Bits de inicio: el transmisor envía un bit con valor 0 indicando que es el inicio de un byte de datos.
- Bits de datos: la cantidad máxima de bits de datos es 8, previamente se configura el número de bits de datos que se van a enviar.
- Bit de paridad: es opcional, este bit se utiliza para comprobar que los datos no se hayan dañado durante el envío.
- Bits de parada: después de los bits de datos se envían uno o más bits de parada con valor de 1, indican el final de los datos.

Nuestro objetivo al implementar UART como se ha mencionado con anterioridad es enviar un número de 32 bits para no tener ningún tipo de problema al visualizar los datos de tiempo ni distancia de las medidas de nuestro sensor. El primer problema al que nos enfrentamos fue que con UART no podíamos enviar más de 8 bits por paquete, por tanto tuvimos que convertir nuestro número a caracteres y almacenarlo en una cadena de caracteres y finalmente enviar la cadena de caracteres mediante la función `Put_String` que nos proporciona UART. Empleamos la función `sprintf` de C que cumple la función exacta que necesitamos, es decir, transformar los enteros a caracteres.

El formato de la función es el siguiente:

```
sprintf(char *str, formato de datos, datos)
```

- `*str` es un puntero al buffer de caracteres donde almacenaremos nuestra cadena de caracteres.
- `formato de datos` indica el tipo de formato que le estamos proporcionando a la función.
- `datos` es el número que queremos pasar a caracteres.

Hasta aquí ya habíamos solucionado el problema de enviar el número, pero teníamos que poder ver el dato que enviábamos, para ello usamos Tera Term.

4.3.2. Tera Term

Es un programa que emula una terminal que se utiliza para acceder y administrar dispositivos remotos a través de conexiones seriales como puertos COM, conexiones de red, etc.

Es una interfaz simple que nos facilita la comunicación entre los dispositivos que se comunican mediante la comunicación serial, en nuestro caso el PSoC y el ordenador.

4.3.3. Programa en PSOC Creator

Importamos el módulo UART de PSOC Creator 4.4, configuramos los pines emisores y receptores mediante la información que nos proporcionaba la hoja de datos del módulo y establecimos los valores de la comunicación esto lo tuvimos que hacer tanto en PSOC Creator como en el Tera Term para que la comunicación fuera posible.

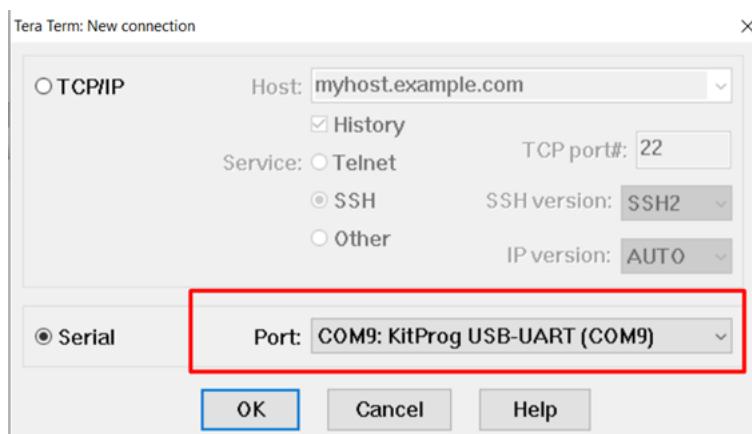


Figura 4.1: Tera Term detecta el microcontrolador

4.3. Módulo UART

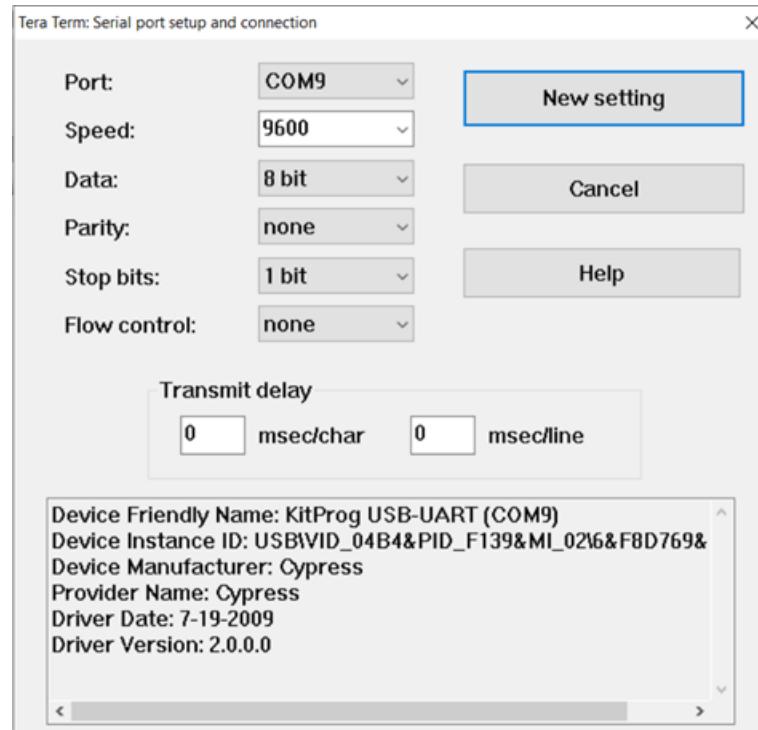


Figura 4.2: Configuración de Tera Term

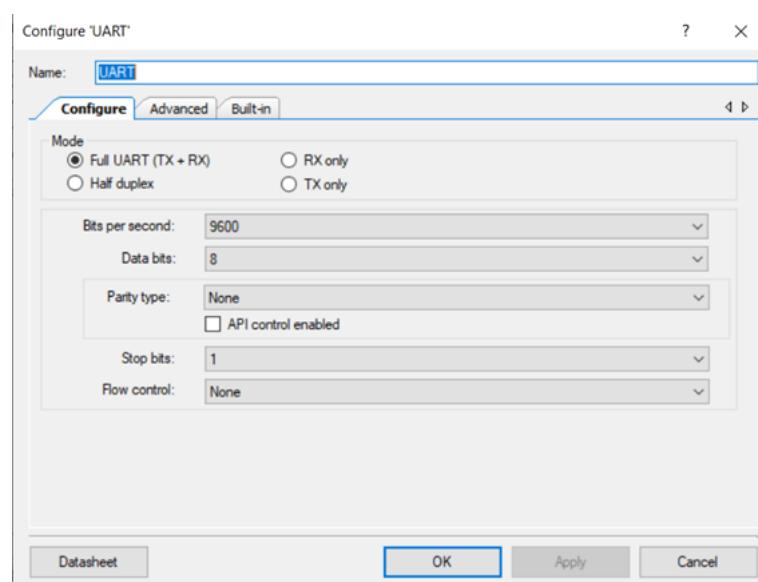


Figura 4.3: Configuración de UART

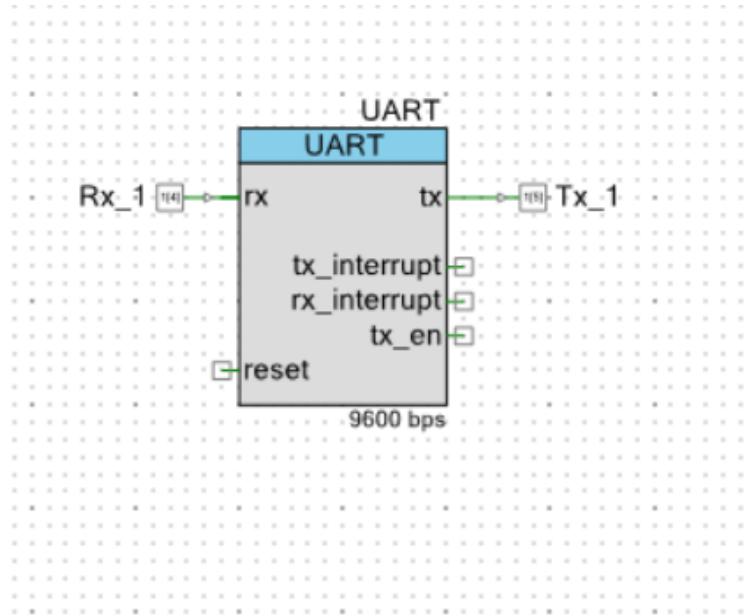


Figura 4.4: Módulo UART y esquema usado

Name	Port	Pin	Lock
Rx_1	P1[4]	32	<input checked="" type="checkbox"/>
Tx_1	P1[5]	33	<input checked="" type="checkbox"/>

Figura 4.5: Pines en el proyecto UART

El código consiste en admitir interrupciones globales, encender el UART, declarar el número a enviar, la cadena para guardar el dato y un texto para guiarlos. La función de UART que se usa para enviar el dato es: **PutString** que nos permite enviar una cadena de caracteres.

```
#include <project.h>
#include <stdio.h>

int main(void)
{
    CyGlobalIntEnable; //Permitimos interrupciones globales
    UART_Start(); //Iniciamos el modulo UART

    uint32_t num = (uint32_t)4294967295; //el numero mayor de 4 bytes
    char str[4]; // buffer para guardar la cadena
    char text[]={"La distancia es:"};

    for (int i=0;i<1;i++)
    {

```

4.4. Encender el sensor y calcular la medida de distancia

```
    sprintf(str , "%d" , num); // convierte el int en char
    UART_PutString(text);
    UART_PutString(str);

}
}
```

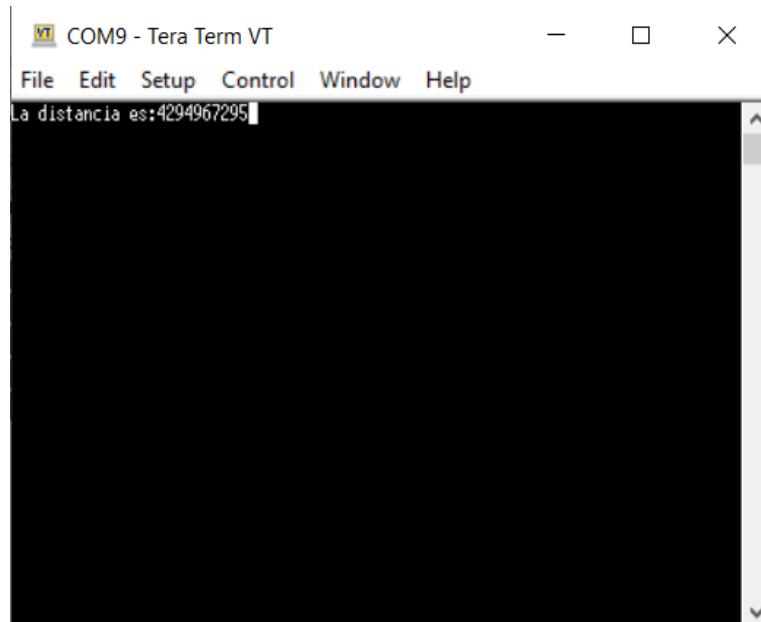


Figura 4.6: Número en Tera Term

4.4. Encender el sensor y calcular la medida de distancia

En este punto del desarrollo nuestro objetivo era encender el módulo SRM400, detectar la señal de ECHO y finalmente mostrar el cálculo de la distancia. La primera cuestión para resolver fue ¿Cómo le indicamos al módulo que debe encenderse? Para ello consultamos la hoja de datos y nos informaba que la señal del pin 1 del módulo debía estar activa durante 0,5 milisegundos. Además de ello nos informaba que era necesario utilizar un transistor para poder encender el módulo.

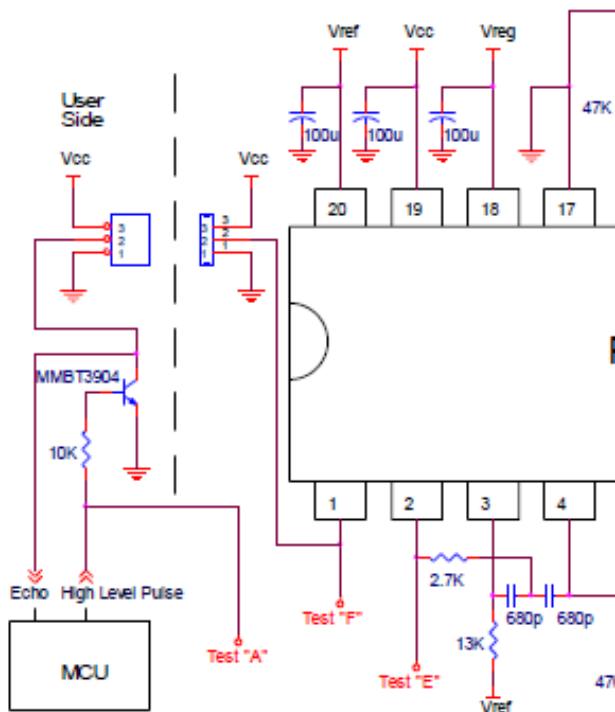


Figura 4.7: Circuito datasheet

Para poder realizar la medición de distancia no era suficiente con encender el módulo SRM400 también era necesario escuchar la señal de ECHO y guardar el tiempo desde que hemos iniciado el módulo hasta que recibimos la señal de ECHO.

4.4.1. Prueba para comprender el funcionamiento del Timer

Los objetivos esquematizados de la prueba son los siguientes:

- Controlar el inicio del temporizador.
- Forzar la unidad de medida del temporizador.
- Tener en cuenta el periodo del temporizador.
- Reiniciar el temporizador.
- Ver el dato del tiempo por pantalla mediante UART.

La siguiente figura nos muestra el esquemático utilizado.

4.4. Encender el sensor y calcular la medida de distancia

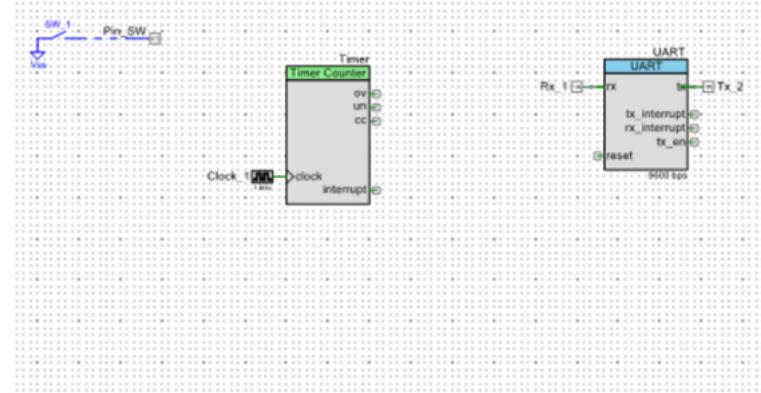


Figura 4.8: Esquemático temporizador

Podemos observar en el esquema que el reloj usado para el Timer es de 1KHz por tanto tomará valores cada milisecondo.

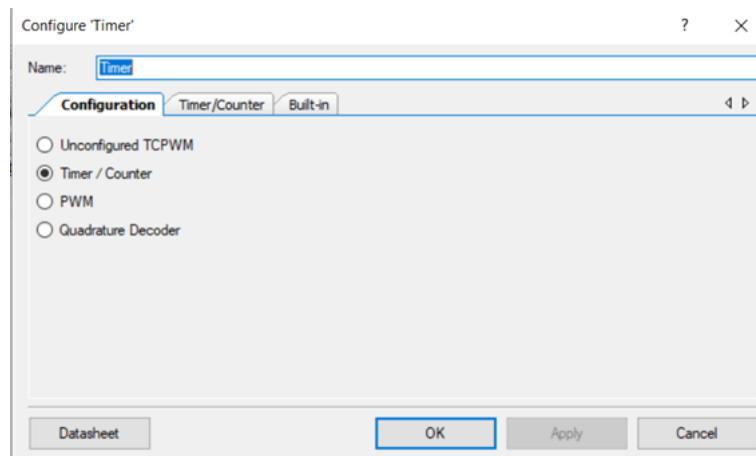


Figura 4.9: Seleccionamos el modo Timer

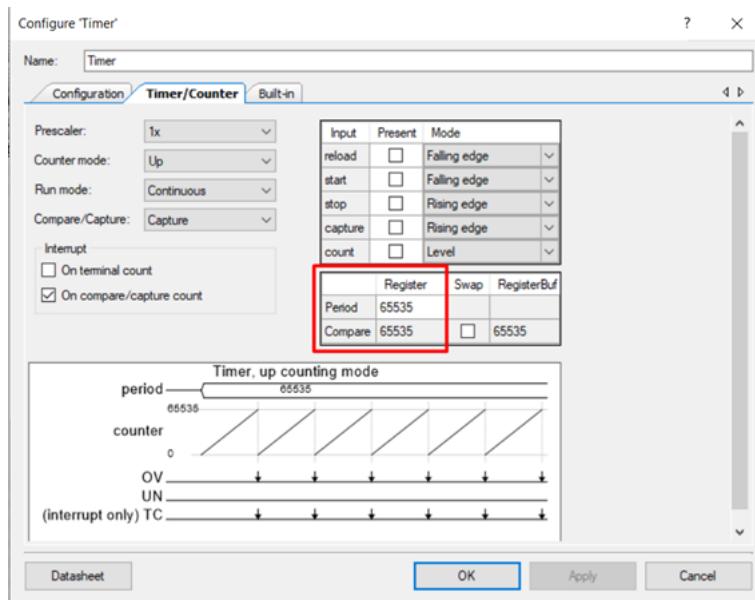


Figura 4.10: Configuramos el Timer

Establecemos como periodo del temporizador el máximo que podemos que es de 65535 milisegundos. En el esquema podemos observar el módulo UART para ver los datos, el módulo Timer que es el contador y un Pin digital que es el botón de usuario.

Name	/	Port	Pin	Lock
Pin_SW		P2[7]	44	<input checked="" type="checkbox"/>
Rx_1		P1[4]	32	<input checked="" type="checkbox"/>
Tx_2		P1[5]	33	<input checked="" type="checkbox"/>

Figura 4.11: Configuramos los pines del Timer

El **main.c** consiste en iniciar el UART después de permitir las interrupciones globales. La declaración de las variables que necesitamos. La función del botón de usuario está añadida para poder reiniciar el temporizador y lo forzamos con un if, bueno la función **Timer_Init** reinicia el valor del temporizador pero no le indica que inicie a contar, por eso justo después del if encontramos la función **Timer_Start** que sí le indica que empiece a contar. Finalmente, para leer el valor del contador usamos la función **Timer_ReadCounter**. Condicionamos el envío del tiempo cada segundo y usamos la función **enviarDatos** que habíamos creado previamente para enviar el dato del tiempo por UART. Debido a que el máximo periodo que tiene el contador es de 65 segundos añadimos una variable auxiliar que nos muestra cuántas veces se ha cumplido este periodo.

4.4. Encender el sensor y calcular la medida de distancia

```
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    UART_Start();

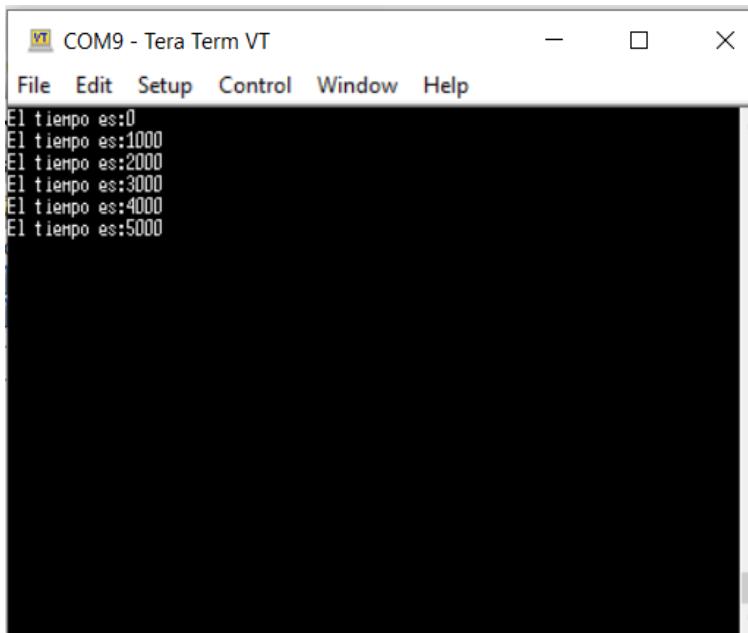
    uint32 contador=0;
    uint32 aux_tiempo=0;
    char8 texto []= {"El valor del tiempo es:"};

    for (;;)
    {
        if (Pin_SW_Read() == 0) //PULSAMOS BOTON DE USUARIO
        {
            Timer_Init(); //RESET

        }

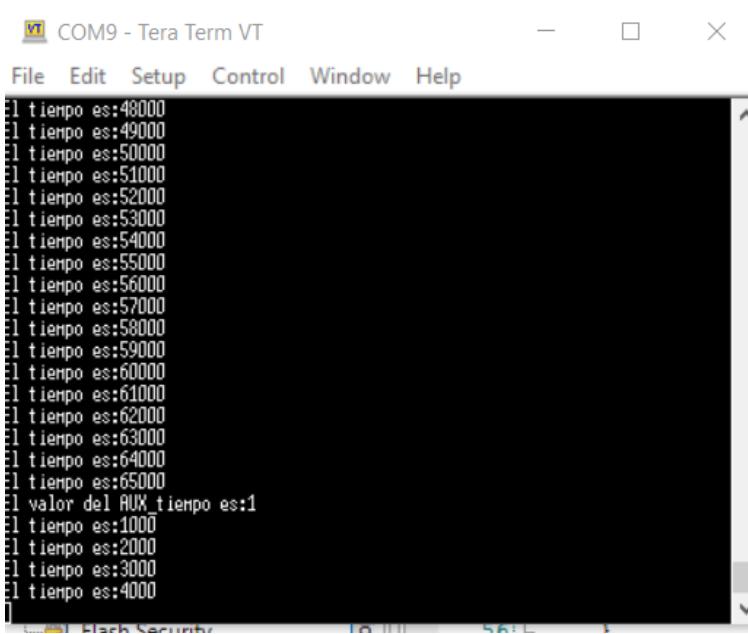
        Timer_Start(); // EMPIEZA A CONTAR
        contador = Timer_ReadCounter();
        if (contador % 1000 == 0)
        {
            char8 texto2 [] = {"El tiempo es:"};
            mostrarDatos(contador, texto2); //TIEMPO
        }

        if (contador == 65534)
        {
            char8 texto []= {"El valor del AUX_tiempo es:"};
            aux_tiempo= aux_tiempo + 1;
            mostrarDatos(aux_tiempo, texto);
        }
    }
}
```



A screenshot of a terminal window titled "COM9 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main text area displays the following text:
El tiempo es:0
El tiempo es:1000
El tiempo es:2000
El tiempo es:3000
El tiempo es:4000
El tiempo es:5000

Figura 4.12: Resultado del Timer en Tera Term



A screenshot of a terminal window titled "COM9 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main text area displays the following text:
l tiempo es:48000
l tiempo es:49000
l tiempo es:50000
l tiempo es:51000
l tiempo es:52000
l tiempo es:53000
l tiempo es:54000
l tiempo es:55000
l tiempo es:56000
l tiempo es:57000
l tiempo es:58000
l tiempo es:59000
l tiempo es:60000
l tiempo es:61000
l tiempo es:62000
l tiempo es:63000
l tiempo es:64000
l tiempo es:65000
l valor del AUX_tiempo es:1
l tiempo es:1000
l tiempo es:2000
l tiempo es:3000
l tiempo es:4000

Figura 4.13: Resultado en Tera Term con el valor del auxiliar

4.4.2. Encendemos el módulo SRM400 y escuchamos la señal de ECHO

La siguiente figura es el dispositivo montado donde podemos ver el PSoC 4 BLE conectado al módulo SRM400 además del transistor.

4.4. Encender el sensor y calcular la medida de distancia

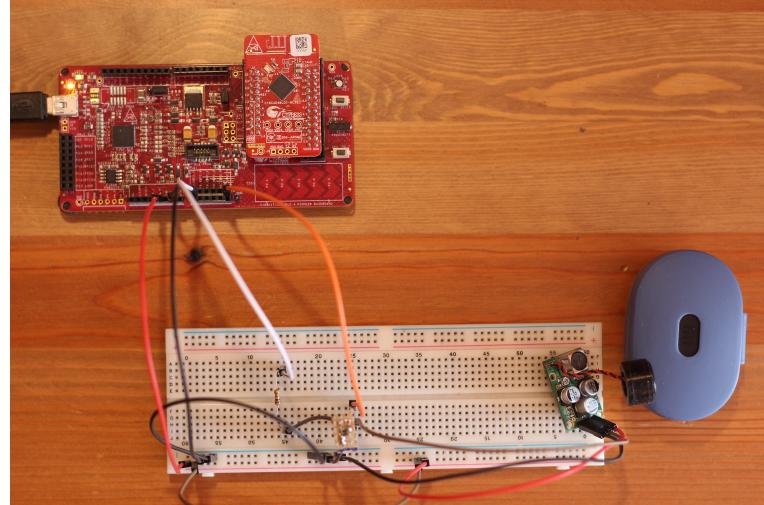


Figura 4.14: Dispositivo IoT

En este punto ya comprendíamos el Timer y ya podíamos guardar el valor del tiempo, entonces ya simplemente nos quedaba encender el módulo SRM400 y escuchar la señal de ECHO.

En la figura del esquemático podemos ver el UART que nos permitirá ver los datos y el Timer que nos permitirá tomar las medidas de tiempo.

Pero los protagonistas son los pines **vin** y **Pin_ECHO**, el primero lo usaremos para enviarle la señal de activación al **módulo SRM400** y el segundo lo usaremos para leer la señal que nos envía el módulo. Una vez teníamos montado el circuito nos quedaba programar nuestro microcontrolador para que cumpliera los objetivos.

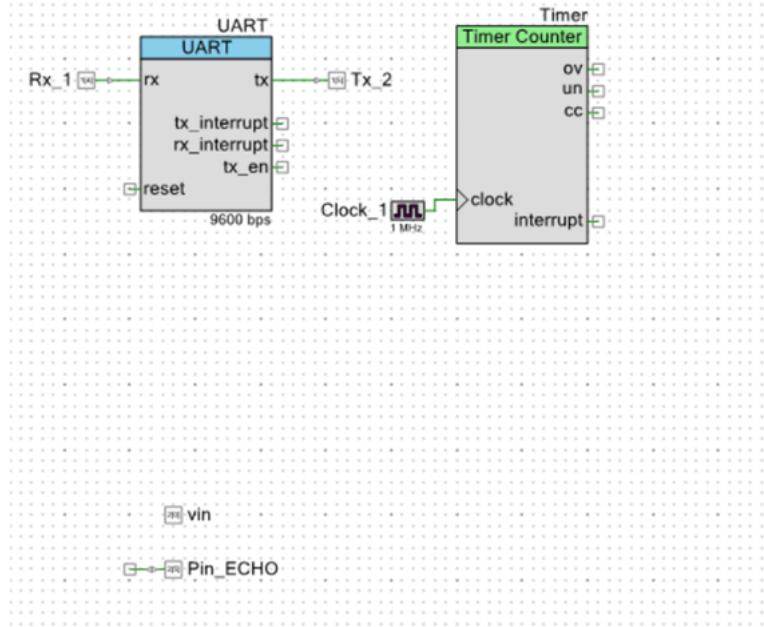


Figura 4.15: Esquema proyecto sensor

El siguiente código consiste en iniciar el módulo poniendo durante 0,5 milisegundos la señal de vin en 1 para luego dejarlo en 0. Como el módulo tiene un pequeño retraso que debemos tener en cuenta para no detectarlo como un falso ECHO añadimos el primer while con la condición **Pin_ECHO_Read() == 0**, donde la función Read() nos permite leer el estado de ese pin.

Debo mencionar que cuando nosotros enviamos un 1 por vin el módulo responde poniendo su pin 1 en 0 es por eso que debemos evitar leer ese 0 como respuesta del módulo.

Después de ese 0 el módulo se pone en 1 y cuando detecta la señal vuelve a ponerse en 0, y para eso tenemos el segundo while con la condición **Pin_ECHO_Read() == 1** donde lo que hacemos es esperar que cambie de estado a 0 para saltar el bucle. La medición del tiempo se realiza justo antes de encender el módulo y después de detectar la señal de ECHO.

4.4. Encender el sensor y calcular la medida de distancia

```
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    UART_Start();
    Timer_Start();

    float tiempoTotal=0;
    float distancia = 0;
    vin_Write(0);

    for (;;)
    {
        uint16 contadorAux1=0;
        uint16 contadorAux2=0;

        contadorAux1=Timer_ReadCounter(); // leemos el tiempo cuando iniciamos el ←
                                         modulo

        vin_Write(1);
        CyDelayUs(500);
        vin_Write(0);

        while( Pin_ECHO_Read() == 0 )
        {

        }
        while( Pin_ECHO_Read() == 1 )
        {

        }
        contadorAux2=Timer_ReadCounter(); // leemos el tiempo cuando leemos el ←
                                         echo

        tiempoTotal = contadorAux2 - contadorAux1;
        mostrarDatos(tiempoTotal , "El tiempo total es: ");
        distancia = calcularDistancia(tiempoTotal);
        mostrarDatos(distancia , "El valor de la distancia es: ");

    }

}
```

La función que usamos para realizar el cálculo de la distancia es el siguiente:

```
float calcularDistancia(float contador)
{
    float tiempo;
    float distancia;

    tiempo = (contador) / 2;
    distancia = tiempo * 0.0003432 * 100; //distancia esta en cm
    return distancia;
}
```

En la función nos llega por parámetro el valor del tiempo medido, desde que se ha encendido el módulo hasta que detecta la señal de ECHO, es por ello que se divide entre 2 porque sino tendríamos el doble de la distancia que realmente queremos medir.

Debemos tener en cuenta que el tiempo que medimos está en microsegundos es por eso que la velocidad del sonido está en m/microsegundos y multiplicamos por 100 para pasar nuestra distancia en cm.

La función que usamos para ver los datos es la siguiente:

```
void mostrarDatos(float numero, char8* texto)
{
    char contador_str[10];
    uint32 numero2;
    numero2 = numero;

    sprintf(contador_str, "%lu\r\n", numero2);
    UART_PutString(texto);
    UART_PutString(contador_str);
}
```

Como mencionamos en la parte de UART, debíamos convertir los número a caracteres para ello debíamos crear previamente un buffer donde almacenarlos, ese es **contador_str[10]**.

Tuvimos ciertos problemas al convertir floats a caracteres que solventamos convirtiéndolos a enteros previamente, para ello nos ayudamos de **numero2**. Y mediante la función **PutString** enviamos tanto el texto como los números.

4.5. Configuración del módulo BLE

El objetivo de esta última parte era entender el módulo BLE para poder anunciar el dispositivo, permitir la conexión con el ordenador o el móvil y enviar el dato de la medición.

Para configurar el módulo BLE lo primero que tuvimos que hacer es comprender cómo funcionaba el protocolo BLE para ello nos ayudamos de la hoja de datos del módulo BLE proporcionado por PSoC Creator, la documentación oficial del protocolo BLE y finalmente también del documento *Creating a BLE Custom Profile* que nos enseña cómo crear un perfil customizado del BLE en PSOC Creator.

En la primera pantalla debemos seleccionar el tipo de perfil que vamos a utilizar, en el protocolo BLE permiten definir características y funcionalidades de un dispositivo Bluetooth. Cada perfil BLE define un conjunto de atributos, servicios y características para que dos dispositivos puedan comunicarse mediante BLE y realizar funcionalidades específicas. Ejemplo de aplicaciones y casos de uso serían monitores de actividad física, dispositivos médicos etc.

Los perfiles BLE están compuestos por:

- Atributos: unidades básicas de datos que representan información específica.
- Características: son atributos individuales que forman parte de un servicio que representan una unidad lógica de datos. Cuando escogemos un perfil diseñado en BLE estamos configurando los atributos, servicios y características para la función específica que va a desarrollar, así los dispositivos BLE pueden interoperar y comunicarse de manera estándar, sin importar el fabricante.[3]
- Servicios: son conjuntos de atributos relacionados, un único servicio puede tener múltiples características y se usa para definir una parte específica de la funcionalidad global del dispositivo.

Para la configuración del módulo BLE hemos seguido la guía *Creating a BLE Custom Profile*. En nuestro caso le indicamos que el perfil que vamos a usar es personalizado, el rol que tomará el microcontrolador será el de servidor, aunque bueno esto es dinámico depende de si pide o da la información y finalmente el GAP rol que en cambio, sí es estático, será el de periférico ya que él no tendrá el poder de iniciar una conexión de eso se encargará el ordenador o el móvil que serán los maestros, los roles están explicado en el anexo B.

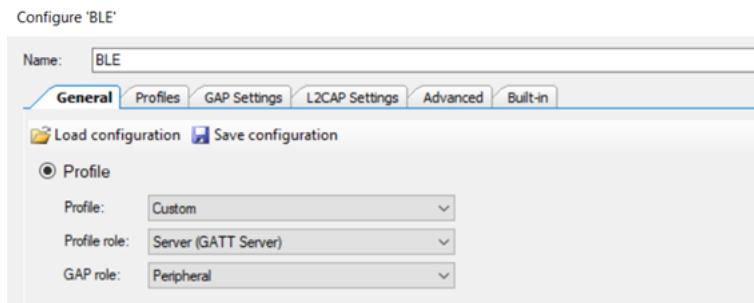


Figura 4.16: Configuración pestaña General del BLE

Una vez configurado el perfil, creamos el servicio y sus características. Tanto las características como los servicios tienen una UUID asignada y diferente.

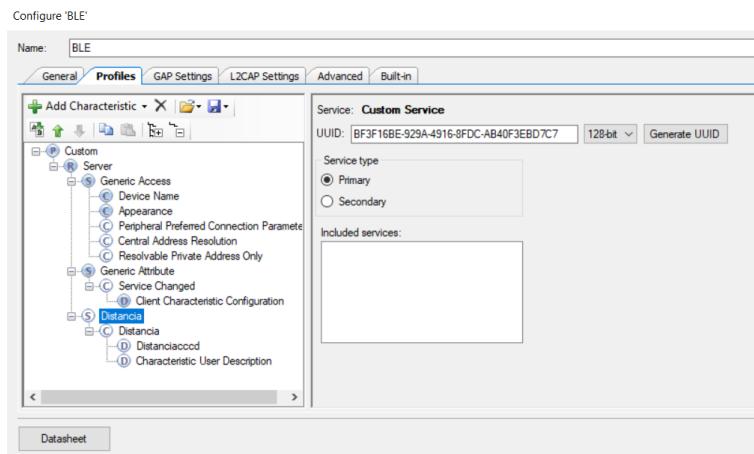


Figura 4.17: Configuración pestaña Perfil del BLE

Escogemos la opción de servicio primario porque es un servicio principal que ofrece nuestro dispositivo Bluetooth.

Nuestro servicio primario solo contiene un atributo y es el de Distancia, que es el que nos permitirá ver en el ordenador el valor de la distancia que estamos enviando. Los servicios primarios pueden agrupar diferentes atributos.

Los servicios secundarios, son servicios adicionales que ofrece un dispositivo que hacen referencia a otros servicios primarios, además no contiene más atributos, su función es la de proporcionar una referencia a otro servicio primario en el dispositivo, como ejemplo en un dispositivo de monitorización de actividad física como servicio primario podríamos ofrecer la medición de pasos y la frecuencia cardíaca y como secundario un servicio que hace referencia a un servicio primario en otro dispositivo remoto que proporcione información de ubicación GPS.

4.5. Configuración del módulo BLE

Lo siguiente que debemos configurar es la característica de nuestro servicio indicarle el tipo de dato y la longitud del dato que vamos a enviar, también debemos indicar las propiedades, es decir, cómo se accederá a este dato y los permisos de acceso a los datos. Los permisos se configuran para cifrado, autenticación y autorización. El uuid es una identificación única de la característica.

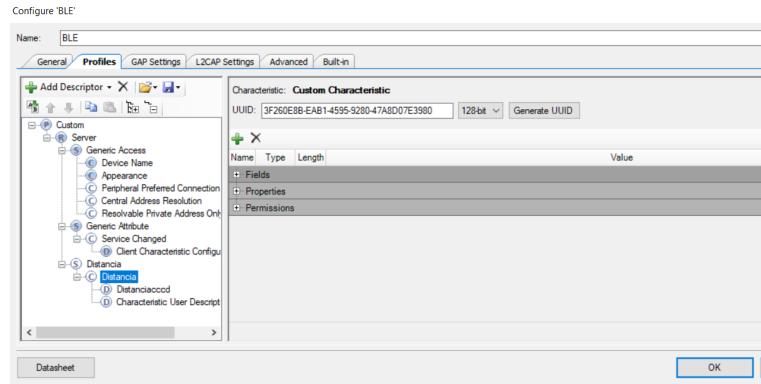


Figura 4.18: Configuración característica Distancia

Nosotros añadimos un descriptor que esto permite proporcionarle al usuario información sobre la característica, en nuestro caso informamos que el valor que se proporciona es de distancia.

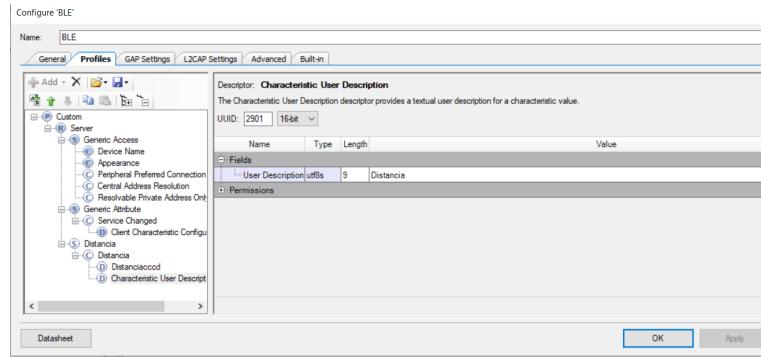


Figura 4.19: Configuración descriptor de característica del usuario

Una vez hemos configurado la pestaña de perfiles y características, configuraremos el GAP. Las configuraciones del GAP son todas aquellas relacionadas a la conexión como intervalos de conexión, tamaño del paquete de anuncio y el contenido etc.

La primera pantalla que vemos es la configuración general donde establecemos la dirección de 6 bytes del dispositivo BLE, es una dirección utilizada en la publicidad, al seleccionar la opción de silicio generamos los últimos bytes de la dirección automáticamente, también definimos el tamaño del MTU que nos indica el tamaño del PDU(Protocol Data Unit). Especificamos el nombre que verá el cliente GATT cuando nuestro dispositivo se anuncie, es decir cuando el

ordenador escanee verá TFG_BLE como un dispositivo que se está anunciando. La apariencia del dispositivo la hemos dejado como desconocido. También especificamos el nivel de potencia de la radio al anunciar los datos y especificamos el nivel de potencia de transmisión de la radio al enviar datos durante la conexión.

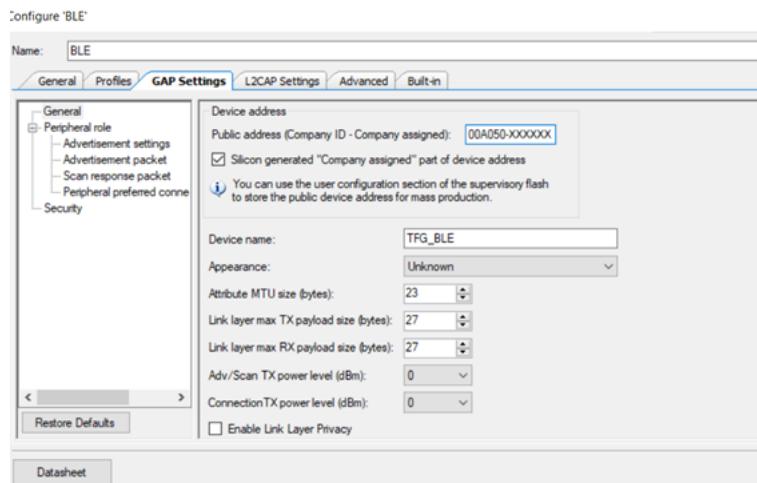


Figura 4.20: Configuración de la pestaña GAP Settings

En la pantalla de configuración de publicidad, indicamos que el dispositivo se anuncie de forma general para que cualquier dispositivo pueda encontrarlo, también indicamos que puede recibir solicitudes de conexión de cualquier central. También decimos qué canales de publicidad debe usar en nuestro caso le indicamos que todos.

Finalmente le indicamos que el intervalo de publicidad como mínimo entre evento de publicidad sea 80 milisegundos y máximo 200 milisegundos, el timeout es el máximo de tiempo que el dispositivo se mantendrá anunciando si no recibe ninguna petición de conexión, en nuestro caso hemos establecido el máximo que son 32 segundos.

4.5. Configuración del módulo BLE

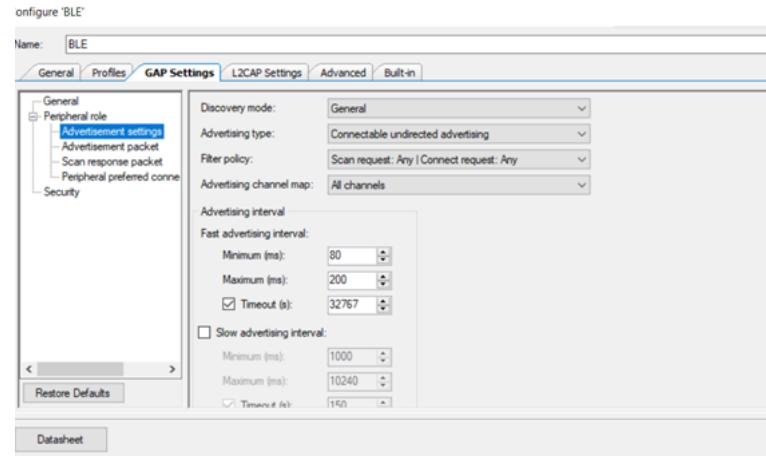


Figura 4.21: Configuración de la pestaña GAP Settings

Continuaremos con la configuración del paquete de publicidad donde le indicamos que queremos que envíe el nombre del dispositivo y un par de indicadores.

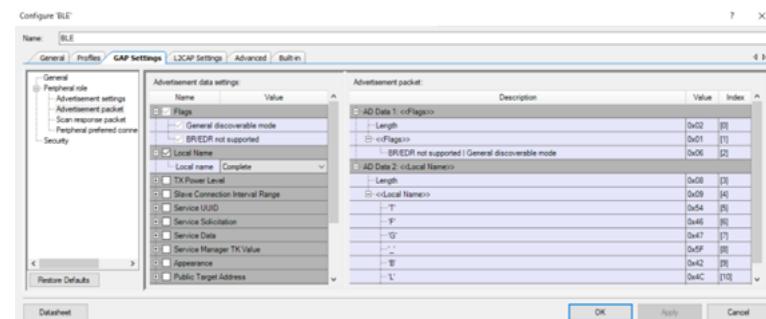


Figura 4.22: Configuración de la pestaña GAP Settings

En respuesta al escaneo le indicamos con qué paquete debe responder a las peticiones del central durante el escaneado.

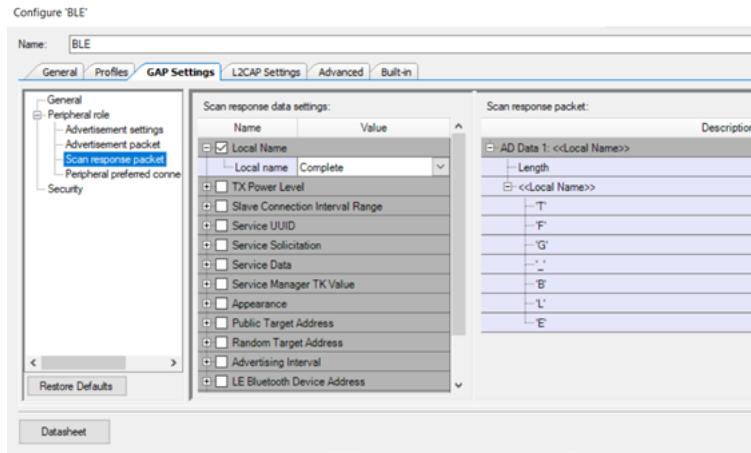


Figura 4.23: Configuración de la pestaña GAP Settings

En las configuraciones de conexión periférica preferente indicamos cuánto es el valor mínimo de la conexión una vez establecida, además del tiempo que la duración se mantendrá activa si no hay ningún tipo de intercambio entre los dispositivos en nuestro caso le hemos indicado que 2 segundos.

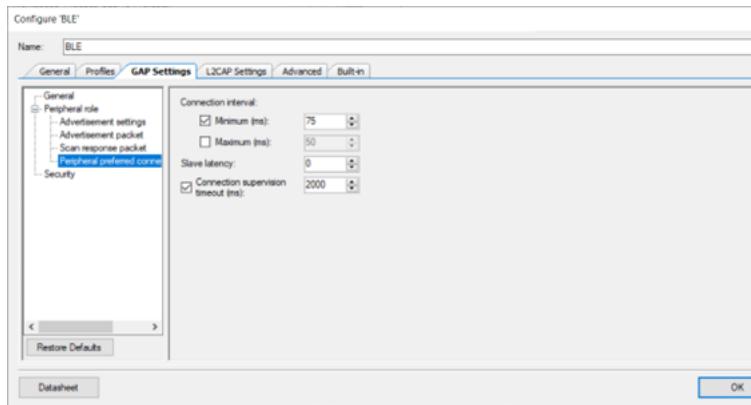


Figura 4.24: Configuración de la pestaña GAP Settings

En el caso de este proyecto en las demás pestañas dejamos la configuración predeterminada.

4.5. Configuración del módulo BLE

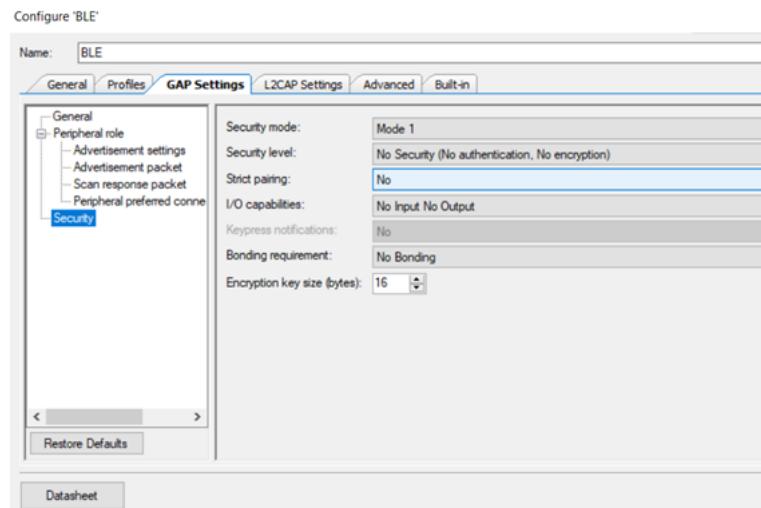


Figura 4.25: Configuración de la pestaña L2CAP

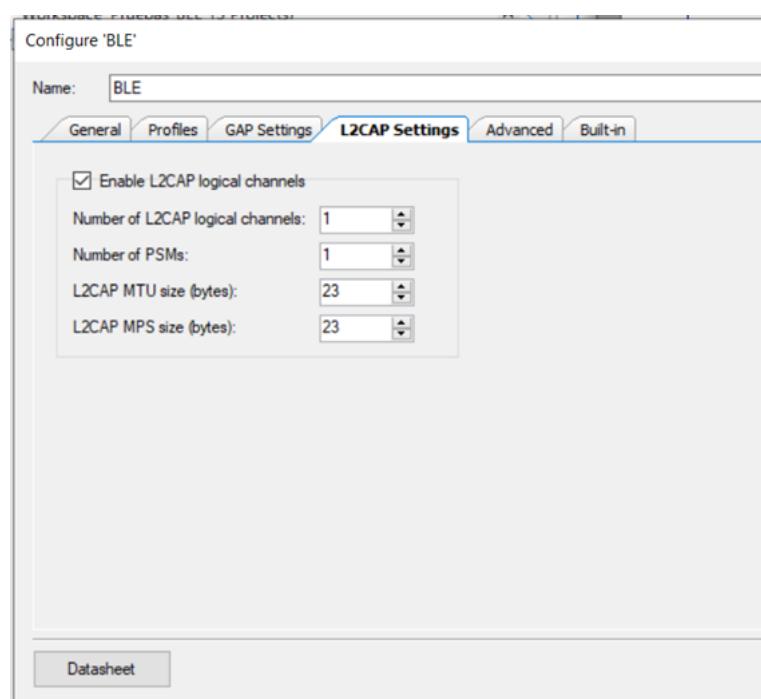


Figura 4.26: Configuración de la pestaña Advanced

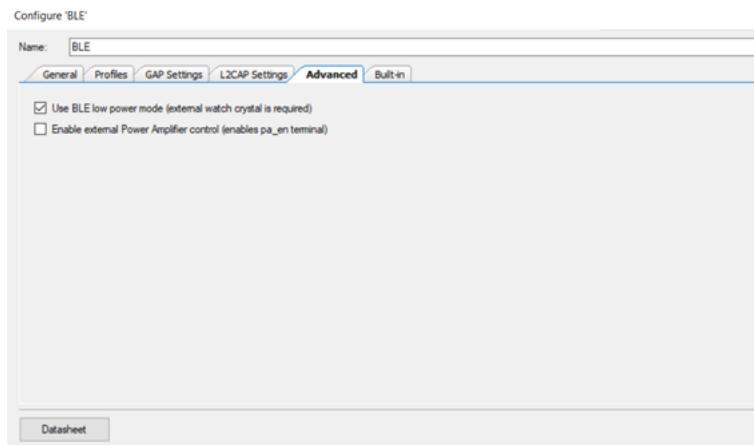


Figura 4.27: Configuración de la pestaña Built-in

Una vez configurado el módulo añadimos el código para poder establecer la conexión entre el PSOC y el ordenador. A continuación explicaré los dos archivos relacionados a BLE que he debido añadir al proyecto BLE_APP.c y BLE_APP.h además de los cambios añadidos al main.c necesarios para poder enviar el dato de la medida.

El primer archivo que vamos a explicar es BLE_APP.c. Se encarga de establecer la conexión y manejar los eventos relacionados con la comunicación BLE.

Primero definimos la variable deviceConnected que nos indicará si tenemos la conexión con algún dispositivo o no.

La función **BLECallBack** se encarga de manejar algunos eventos relacionados con la conexión y la comunicación BLE. Cuando recibe un evento lo evalúa y actúa según el caso, ya que cada evento tiene relacionado una acciones diferentes y determinadas que se manejan con un switch case. Nuestro código gestiona 3 casos:

- **CYBLE_EVT_GAP_DEVICE_DISCONNECTED:** En este caso la pila BLE indica que no hay ningún dispositivo conectado, entonces se reinicia la publicidad y establecemos la variable deviceConnected en 0.
- **CYBLE_EVT_GATT_CONNECT_IND:** Este evento nos indica que se ha establecido una conexión, entonces guardamos el identificador de conexión connectionHandle y establecemos la variable deviceConnected en 1 ya que hay un dispositivo conectado.
- **CYBLE_EVT_GATTS_WRITE_REQ:** Este evento tiene la función de gestionar las solicitudes de escritura de atributos.

Estos son los eventos que hemos querido gestionar nosotros, dependiendo de la aplicación que queremos podemos gestionar más eventos o menos.

4.5. Configuración del módulo BLE

En nuestro caso gestionamos los eventos relacionados con la conexión y la comunicación Bluetooth. En el archivo BLE_APP.h declaramos alguna variable y hacemos la declaración de la función BLECallBack.

Declaramos la variable deviceConnected como externa, implica que esta variable está declarada en otra parte del programa y se usa aquí para poder acceder a su valor. Solo mencionaré las partes esenciales relacionadas al BLE ya que todo lo demás ha sido explicado con anterioridad.

Declaramos las variables para enviar la notificación al maestro, concretamente notificatioHandle2 y connectionHandle, estas variables son especiales porque se definen a partir de una estructura que además dentro de esta estructura hay otras estructuras porque así se ha definido la programación de las variables BLE en PSOC pero en resumen notificationHandle2 contiene el valor que vamos a enviar y connectionHandle contiene la información de la conexión con el maestro.

La función CyBle_Start inicia la pila BLE y le añadimos la función de devolución de llamada BLECallBack. Para procesar los eventos BLE dentro del bucle infinito for declaramos CyBle_ProcessEvents. Realizamos la medición de distancia, comprobamos que el dispositivo esté conectado y enviamos el dato mediante CyBle_GattsNotification().

```

#include "project.h"
#include "main.h"
#include "BLE_APP.h"

// Variables referentes la conexion del dispositivo.
uint8 deviceConnected = 0;

CYBLE_CONN_HANDLE_T connectionHandle;

void BLECallBack ( uint32 event , void *eventParam ) //Funcion de devolucion de ←
llamada BLE.
{
    CYBLE_GATTS_WRITE_REQ_PARAM_T *wrReqParam; //Mantener los comandos ←
enviados por el cliente GATT.

    switch ( event )
    {
        //Gestionamos eventos de conexion.

        //Eventos Generales
        case CYBLE_EVT_STACK_ON:

            //Eventos GAP

        case CYBLE_EVT_GAP_DEVICE_DISCONNECTED: // Estado de ←
desconexion del dispositivo.
            CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST); // Reiniciar ←
publicidad.
            deviceConnected = 0; // Conexion del ←
dispositivo , no activa.
            break;
            //Eventos GATT
        case CYBLE_EVT_GATT_CONNECT_IND: // Estado de conexion←
establecida .
            connectionHandle = *(CYBLE_CONN_HANDLE_T *)eventParam;
            deviceConnected = 1; // Conexion del ←
dispositivo , activa.
            break;
    }
}
}

```

4.5. Configuración del módulo BLE

El siguiente código es el fragmento del main.c donde enviamos el dato por BLE.

```
if(deviceConnected){ //Solo si cumple esto el sensor captara los datos.

    if (proximidad!=proximidadant){ // Si el dispositivo ←
        se conecta

        proximidadant=proximidad; // Si la distancia ←
        leida no es igual al dato almacenado en la variable "←
        proximidadant".

        //Transmisión de datos de distancia por BLE.
        notificationHandle.attrHandle = ←
            CYBLE_DISTANCIA_DISTANCIA_CHAR_HANDLE;
        notificationHandle.value.val = (uint8*) &proximidadant;
        notificationHandle.value.len = 1;

        CyBle_GattsWriteAttributeValue(&notificationHandle,0,&←
            connectionHandle, CYBLE_GATT_DB_LOCALLY_INITIATED);

        CyBle_GattsNotification(connectionHandle, &notificationHandle);
    }
}
```


Capítulo 5

Resultados y Conclusiones

A continuación, presentaremos los objetivos planteados de manera de concisa con el fin de establecer una base inicial que nos permita profundizar en cada uno de ellos.

- Estudio de la tecnología BLE: Estudiar, comprender y aplicar al prototipo.
- Diseño y desarrollo del dispositivo IoT, capaz de medir de forma precisa y enviar el dato.
- Evaluación mediante experimentación del dispositivo.

En las siguientes figuras podemos observar el dispositivo desde diferentes perspectivas y el módulo SRM400.

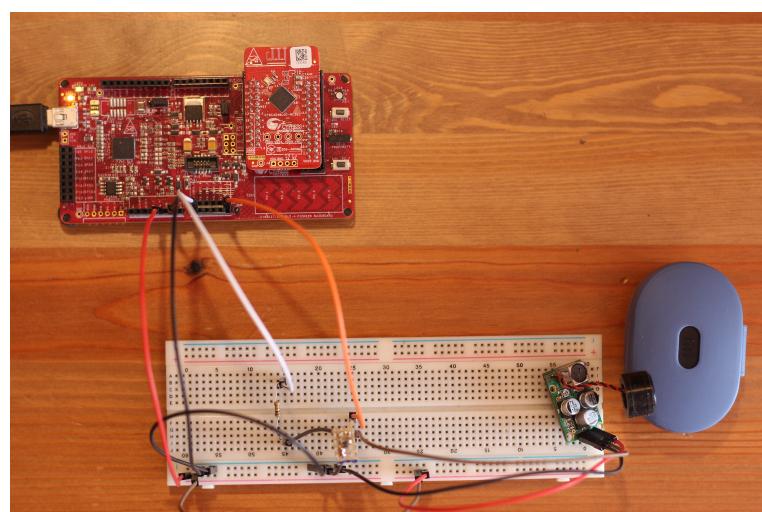


Figura 5.1: Dispositivo IoT

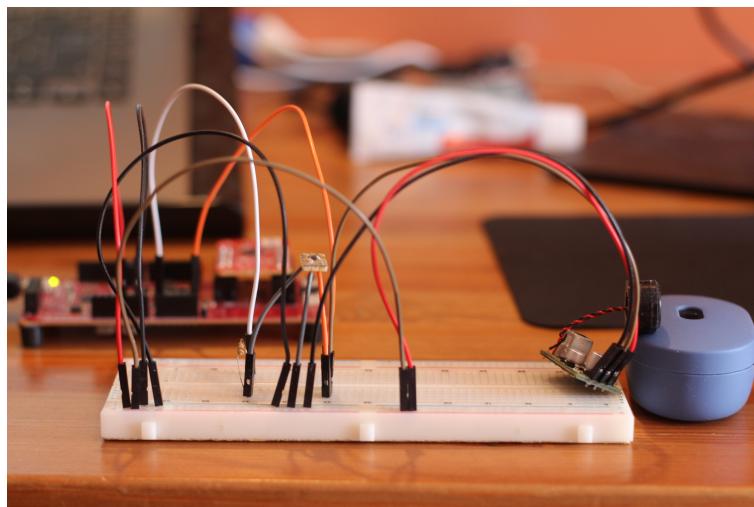


Figura 5.2: Dispositivo IoT conexiones

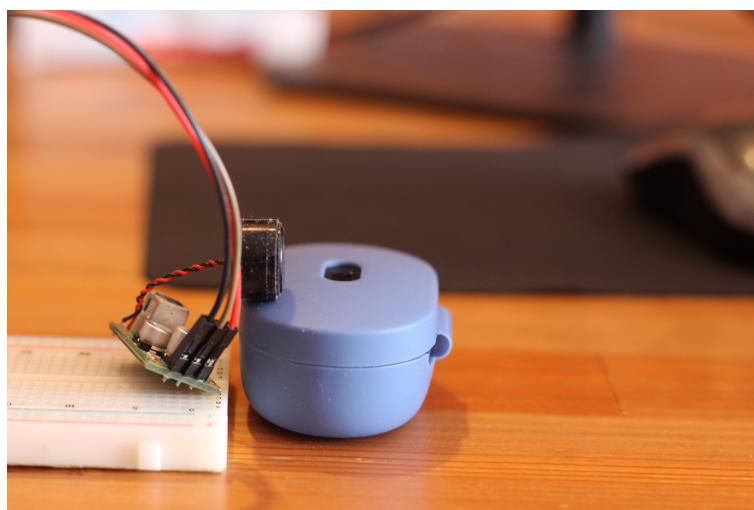


Figura 5.3: Módulo SRM400

El primer objetivo ha sido logrado de forma satisfactoria ya que hemos podido estudiar de forma exhaustiva y comprender correctamente la tecnología BLE (Bluetooth Low Energy). Esta afirmación es respaldada por el hecho de haber podido enviar la información por BLE.

Durante el desarrollo de la investigación, hemos dedicado un tiempo considerable a examinar y analizar los principios y características fundamentales de BLE. Hemos adquirido un conocimiento profundo de su funcionamiento, sus protocolos de comunicación, su arquitectura y sus limitaciones, aunque en este proyecto no hemos puesto aprueba ya que hemos evitado alejar el dispositivo y medir el alcance de la conexión además de que no enviamos grandes cantidades de datos.

Hemos sido capaces de aplicar efectivamente los conocimientos adquiridos. Hemos implementado con éxito la transmisión de información utilizando BLE, lo que demuestra nuestra comprensión sólida y la capacidad para operar dentro de este entorno tecnológico específico.

El segundo objetivo también ha sido logrado de forma satisfactoria ya que gracias a la comprensión del funcionamiento del programa PSoC Creator 4.4 hemos podido lograr la medida precisa de la distancia comprendiendo el funcionamiento del módulo temporizador y comprendiendo el funcionamiento del módulo sónar SRM400.

Estos son los objetivos principales que hemos podido cumplir con este proyecto, el tercer y último objetivo no se ha cumplido de forma satisfactoria ya que debido al momento de lograr los objetivos anteriores ya no había tiempo para poder realizar pruebas, sin embargo, hubiera sido interesante realizar un prototipo y ponerlo a prueba realmente con diferentes sustancias tanto líquidas como sólidos. Personalmente me hubiera gustado poder utilizar la función de deepSleep y así lograr que el dispositivo fuera aún más de bajo consumo.

Las pruebas realizadas se pueden ver en los siguientes vídeos:

El primer vídeo hace una medición de 30 cm y luego cambia la distancia a 25cm mostrándonos los resultados tanto por el puerto serial como por BLE.

<https://youtu.be/m52UpiPiwq8>.

El segundo vídeo nos muestra la distancia mínima que mide el sensor.

Con la siguiente prueba concluimos que la distancia mínima que mide nuestro sensor es de 26 cm, distancias menores de 26 cm provocan un error entre 1 y 2 cm. Finalmente distancias menores de 20 cm no se pueden medir ya que el dato que nos devuelve es aleatorio.

<https://youtu.be/00MAJXXW2ck>

Apéndice A

Módulo de alcance sonar SRM400

El módulo SRM400 es una herramienta útil para ingenieros de diseño que no poseen un amplio conocimiento en circuitos analógicos o el funcionamiento de transductores ultrasónicos para poder desarrollar sistemas de ayuda a la marcha atrás u otro sistemas de medición de distancia, estas características hacían del módulo SRM400 una elección perfecta para realizar el dispositivo de medición de nivel para cumplir los objetivos de este trabajo. Las facilidades que aporta el módulo nos permitirá centrarnos en el circuito digital, en el diseño del software y en cuestiones mecánicas.

A.1. Especificaciones del módulo

Voltaje de operación	DC 6-10 V
Corriente de operación	< 20 mA @DC10V
Frecuencia de oscilación	Oscilador RC variable
Ganancia del amplificador	
Preamplificador	14dB
Amplificador de 2 ^a etapa	30 dB
Amplificador de 32 paso controlado por tiempo	35dB
Filtro paso banda	$F_c : 38KHz$ Ancho de banda: 20 KHz Pérdidas de inserción: 1 dB
Voltaje de alimentación (sin carga)	130 Vpp Ancho de pulso 0.5 ms
Bidireccional E/S	
Señal de entrada	Colector abierto pull low
Salida	0.05*Vcc hasta 0.9*Vcc señales de echo digitales
Distancia medida	25-150 cm

Cuadro A.1: Tabla descriptiva de las especificaciones

A.2. El circuito integrado PW-0268

El circuito integrado PW-0268 ha sido especialmente diseñado para aplicaciones en sonar. Este dispositivo cuenta con un oscilador RC externo sintonizable, que puede compensar el desplazamiento de la frecuencia de resonancia del transductor utilizado causado por el cambio de temperatura. Además, dispone de una etapa de preamplificación de ganancia fija que se ajusta de forma externa, un amplificador de ganancia controlada a 32 pasos, un filtro paso banda, y un comparador que convierte la señal analógica en señal digital TTL para el procesamiento externo por el microprocesador.

A.3. Funcionamiento del circuito integrado PW-0268

Asimismo, este circuito integrado cuenta con un temporizador personalizable que actúa como referencia del reloj que contiene el dispositivo. [2]

A.3. Funcionamiento del circuito integrado PW-0268

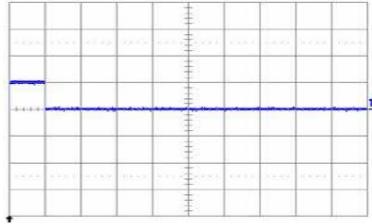
El pin 1, conocido como I_O, es un pin de entrada\salida bidireccional que posee una conexión de colector abierto con una resistencia interna pull alta. Cuando este pin es bajado por un transistor externo, se genera una señal de tono en el pin 11, conocido como DRIVER_O, para la etapa de salida de potencia a través del oscilador RC. Asimismo, si se detecta una señal de eco efectiva, el pin I_O pasa a nivel bajo.

La señal de eco reflejada se encarga de alimentar las primeras etapas del preamplificador a través del pin 10, conocido como ECHO, cuya ganancia varía en función de la sensibilidad del transductor utilizado. Esta sensibilidad puede ser ajustada mediante una resistencia externa entre ECHO y GR_I (pin 9). El amplificador de ganancia controlado por tiempo de 32 pasos desde TCG_I (pin 7) a TCG_O (pin 6) se sincroniza con el final de la señal del pulso de control y se restablece al principio del siguiente pulso de control. Para el filtro de paso de banda activo, que consiste en una etapa de paso bajo entre LP_I (pin 5) y LP_O (pin 4) y otra etapa de filtro de paso alto de HP_I (pin 3) a HP_O (pin 2), solo se requieren unos pocos componentes pasivos. La frecuencia central y el ancho de banda se determinan según los tipos de transductores ultrasónicos y aplicaciones utilizados.

Después del filtro de paso de banda, la señal de eco amplificada se dirige a un comparador, que convierte la señal analógica en una señal digital de salida a la frecuencia central en el pin I_O (pin 1) para su posterior manejo por el microprocesador (μ P). Una función única de oscilador controlado por temperatura rastrea la deriva de la frecuencia de resonancia de los transductores ultrasónicos causada por el cambio de temperatura del entorno. Para ello, se añade un diodo doble y una resistencia entre DRIVER_O (pin 11) y Ftrace (pin 12).

A.3.1. Ejemplo con gráficas

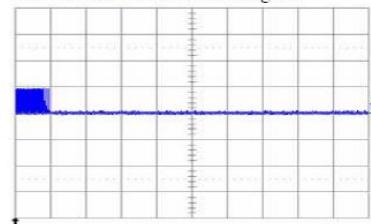
“A” Point: Control Pulse (from MCU)



H: 0.5ms/div
V: 5V/div

Figura A.1: Pulso de control del μP

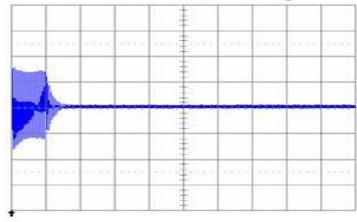
“B” Point: Tone bursts Signal



H: 0.5ms/div
V: 5V/div

Figura A.2: Señal de ráfaga de tonos

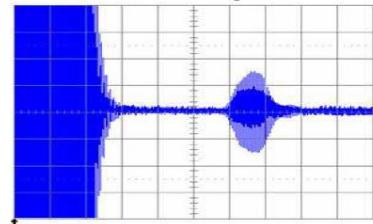
“C” Point: Transducer loading



H: 0.5ms/div
V: 50V/div

Figura A.3: Carga del transductor

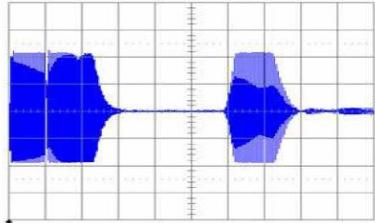
“D” Point: 1st Pre-Amplifier



H: 0.5ms/div
V: 20mV/div

Figura A.4: Preamplificador

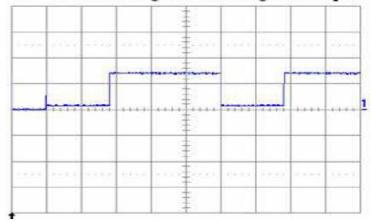
“E” Point: Main 32 Steps TCG Amplifier



H: 0.5ms/div
V: 1V/div

Figura A.5: Amplificador de 32 pasos

“F” Point: Digital Echo signal Output



H: 0.5ms/div
V: 5V/div

Figura A.6: Salida digital de la señal ECHO

En la primera figura podemos observar la señal enviada por el μP de duración 0.5 ms y voltaje de 5 V, en la segunda figura podemos ver que hemos generado una ráfaga de la misma duración y mismo voltaje.

En la tercera figura podemos ver la salida del transductor que ha amplificado la ráfaga de tonos que ha recibido del oscilador RC.

En la cuarta figura podemos ver la señal de ECHO que hemos recibido por el pin 11 amplificado por la etapa de preamplificación, su voltaje es de 20mV.

A.4. Características del circuito integrado PW-0268

En la quinta figura podemos ver la salida de la etapa de amplificación de 32 pasos donde el voltaje ha pasado de 20mV a 1V.

Finalmente podemos observar la señal de ECHO digitalizada.

A.4. Características del circuito integrado PW-0268

1. Funciona en tensión de 6 - 10 V.
2. Amplia gama de frecuencias de funcionamiento hasta 250KHz.
3. El oscilador controlado por temperatura traza la frecuencia de resonancia deriva de los transductores.
4. Amplificador de ganancia controlado por tiempo de 32 pasos incorporado.
5. Filtro pasa banda incorporado de menos componentes externos sintonizados.
6. E/S bidireccional para impulso de conducción y eco.
7. Reloj del sistema ajustable para diversos rangos de detección.
8. Adecuado para aplicaciones de sensor de distancias para coches y otras aplicaciones sonar.
9. Temperatura de operación -40 a +85

A.5. Diagrama de bloques

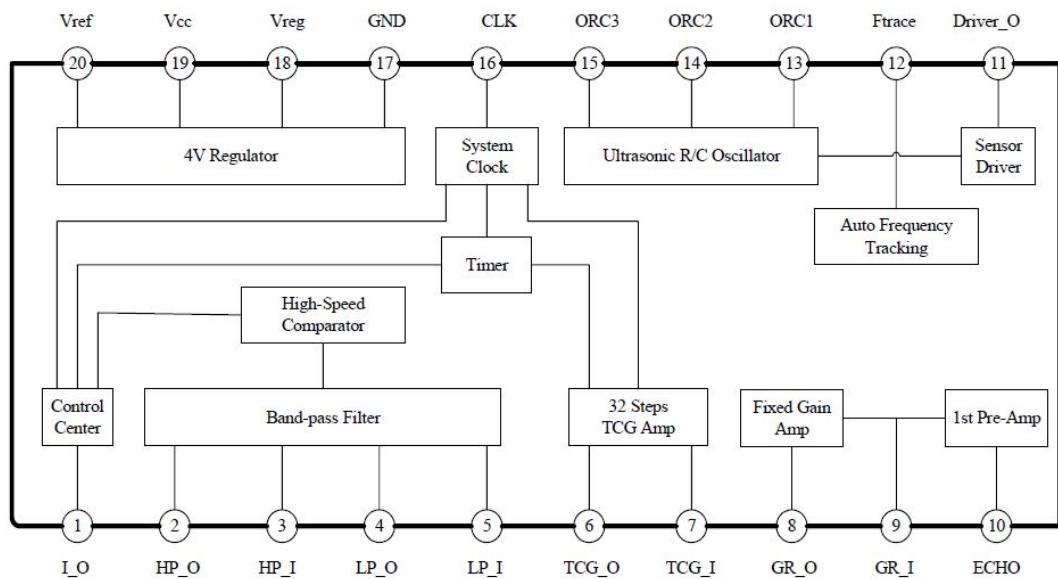


Figura A.7: Diagrama de bloques [2]

A.6. Asignación de pines

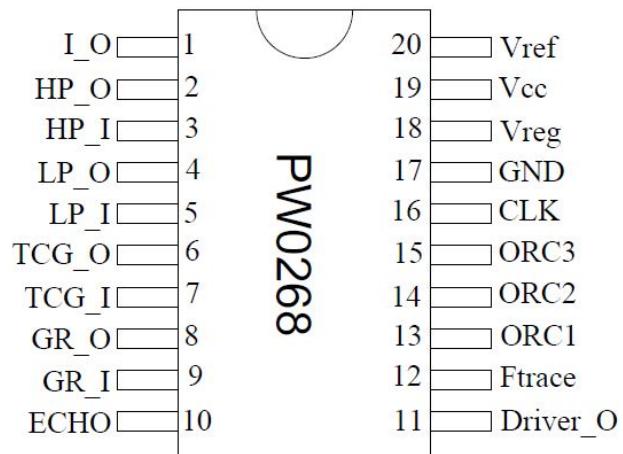


Figura A.8: Asignación de pines [2]

A.6. Asignación de pines

A.6.1. Descripción de los pines

Pin	Nombre	Descripción	Pin	Nombre	Descripción
1	I.O	Entrada/Salida	11	Driver_O	Salida de conducción del transductor
2	HP_O	Salida del filtro de paso alto	12	Ftrace	Entrada de trazado de frecuencia
3	HP_I	Entrada de filtro de paso alto	13	ORC1	Oscilador RC: borne 1
4	LP_O	Salida del filtro de paso bajo	14	ORC2	Oscilador RC: borne 2
5	LP_I	Entrada del filtro de paso bajo	15	ORC3	Oscilador RC: borne 3
6	TCG_O	Salida de ganancia controlada por tiempo	16	CLK	Reloj del sistema
7	TCG_I	Entrada de ganancia controlada por tiempo	17	GND	Tierra
8	GR_O	Salida de ganancia ajustable externa	18	Vreg	Tensión regulada para dispositivos analógicos
9	GR_I	Entrada de ganancia ajustable externa	19	Vcc	Alimentación
10	ECHO	Recepción de entrada de eco	20	Vref	Salida de tensión de referencia

Cuadro A.2: Tabla descriptiva de los pines del IC PW0268

Apéndice B

Arquitectura del BLE

Bluetooth Low Energy (BLE) es una tecnología inalámbrica diseñada específicamente para aplicaciones de corto alcance. Se caracteriza por tener una velocidad de transferencia de datos de 1 Mbps. Uno de los principales objetivos de BLE es optimizar el consumo de energía. Permite enviar pequeñas cantidades de datos a velocidades más bajas, lo que reduce el consumo de energía en comparación con otras tecnologías similares. Esto es especialmente beneficioso para dispositivos alimentados por batería, ya que prolonga la duración de la batería. Además, una ventaja notable de BLE es que el acceso a la documentación oficial es gratuito, a diferencia de otras tecnologías competidoras. Esto facilita el desarrollo de dispositivos y aplicaciones basados en BLE, ya que los fabricantes y desarrolladores pueden acceder a recursos y herramientas sin aumentar costos adicionales. En términos de precio, los dispositivos que utilizan BLE tienden a ser más asequibles en comparación con aquellos que utilizan otras tecnologías competidoras. Esto se debe a la amplia adopción y disponibilidad de componentes BLE en el mercado, lo que ayuda a reducir los costos de fabricación. Otra ventaja significativa de BLE es su compatibilidad con la mayoría de los teléfonos inteligentes disponibles en el mercado. Esto se traduce en una amplia base de usuarios potenciales para dispositivos y aplicaciones basados en BLE. En contraste, tecnologías como ZigBee, Z-Wave y Thread pueden tener limitaciones de compatibilidad con ciertos modelos de teléfonos inteligentes. En cuanto a las especificaciones técnicas, BLE opera en el rango de frecuencia de 2,400 a 2,4835 GHz. Tiene un alcance que puede superar los 100 metros en condiciones óptimas y su consumo de corriente durante la transmisión de radio es relativamente bajo, con una corriente pico de 15 mA para un conjunto de chips típico. En resumen, BLE es una tecnología inalámbrica diseñada para aplicaciones de corto alcance centrada en la eficiencia energética, accesibilidad gratuita a la documentación, precios competitivos, compatibilidad con teléfonos inteligentes y especificaciones técnicas adecuadas para diversas aplicaciones.

Capas de la arquitectura BLE. Capas de la arquitectura BLE. La arquitectura de Bluetooth de

baja energía es fundamentalmente muy sencilla. Se divide en tres partes básicas:

- Controlador
- Host
- Aplicación

El controlador suele ser un dispositivo físico que puede transmitir y recibir señales de radio y entender cómo estas señales pueden interpretarse como paquetes que contienen información.

El host suele ser una pila de software que gestiona cómo dos o más dispositivos se comunican entre sí y cómo se pueden proporcionar varios servicios diferentes al mismo tiempo a través de las radios. Las aplicaciones utilizan la pila de software, y por tanto el controlador, para habilitar un caso de uso.

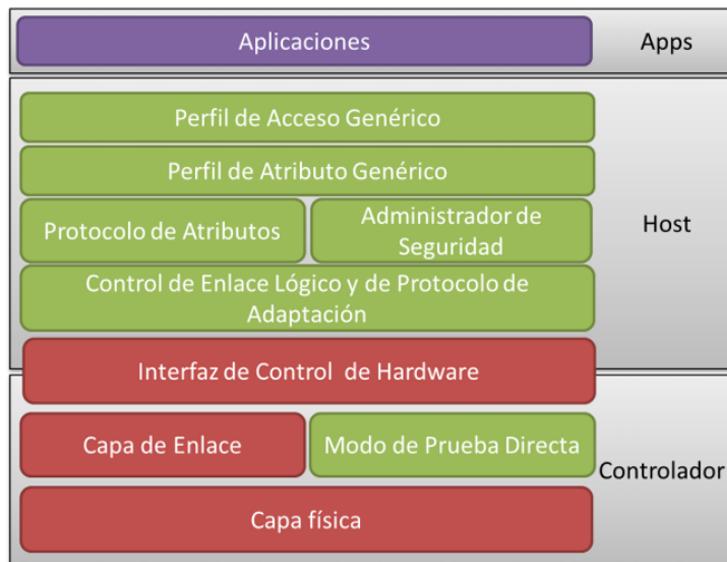


Figura B.1: Capas que forman la arquitectura del BLE

B.1. Controlador

B.1.1. Capa física

La capa física es la encargada de transmitir y recibir bits usando ondas de radio de frecuencia 2,4 GHz, esta capa tiene todo el circuito de comunicaciones encargado de la modulación y la demodulación de señales analógicas para después realizar la transformación a símbolos de tipo digitales. Las ondas de radio pueden transmitir información variando la amplitud, la frecuencia

B.1. Controlador

o fase de la onda dentro de una banda de frecuencia determinada. En el BLE la frecuencia de las ondas de radio varía para enviar un 1 o un 0 mediante el esquema de modulación por desplazamiento de frecuencia Gaussiana (GFSK). En este esquema el 1 es representado mediante una desviación positiva(incremento) de la frecuencia de la onda portadora y un 0 mediante la desviación negativa(decremento) de la onda portadora. En esta modulación lo que ocurre es que se varía la frecuencia dependiendo de si enviamos un uno o un cero, pero hay que tener cuidado con el desplazamiento brusco de la frecuencia , ya que esto produce un pulso de energía que se extiende por una gama más amplia de frecuencias, para evitar la propagación de la energía se utiliza un filtro, en esta codificación el filtro usado tiene forma de campana de Gauss, a diferencia del Bluetooth Classic el filtro usado en BLE no es tan estricto, es por ello que la señal de radio de BLE es más ancha, este ensanchamiento permite que entre dentro de la normativa de espectro ensanchado. La normativa de radio de espectro ensanchado permite que una radio transmita en menos frecuencias que las regulaciones de radio de salto de frecuencia. Sin la forma de filtro más relajada, la radio Bluetooth de BLE no podría anunciararse en sólo tres canales, sino que tendría que utilizar muchos más canales, lo que haría que el sistema consumiera mayor potencia. El índice de modulación describe la amplitud de las frecuencias superior e inferior utilizadas en torno a la frecuencia central de un canal. Cuando se transmite la señal de radio, una desviación de frecuencia positiva de más de 185 kHz de la frecuencia central representa un bit con el valor 1; una desviación de frecuencia negativa de más de 185 kHz representa un bit con el valor 0. Teniendo en cuenta que no es la única tecnología que utiliza esta banda de frecuencias BLE evita las interferencias de la siguiente manera, divide la banda de 2,4 GHz en 40 canales de RF separados, cada canal es de 2 MHz. La capa física transmite un bit de datos cada microsegundo, por tanto, enviamos 1Mb/s. Para poder realizar el envío de las señales tenemos 2 tipos de canales, los de publicidad y los de datos, la elección de canales de publicidad se hizo de tal manera que la superposición con los canales WIFI fuera mínima. En la siguiente imagen podemos ver que los canales de publicidad son 37, 38 y 39 además de evitar los canales de WIFI.

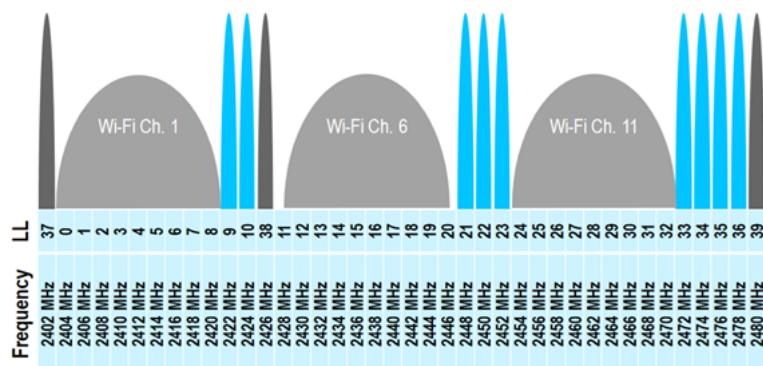


Figura B.2: Distribución de canales

B.1.2. Capa de enlace

La capa de enlace es la parte más compleja de la arquitectura BLE porque se encarga de la publicidad, la exploración, el mantenimiento y la creación de conexiones. Otra de sus responsabilidades es garantizar que los paquetes se estructuren de la forma adecuada, con los valores de comprobación y secuencias de cifrado correctamente calculados. Tres conceptos son necesarios para realizar todas estas tareas: canales, paquetes y procedimientos. Hay dos tipos de canales en la capa de enlace, los canales de publicidad y los de datos. Los canales de publicidad los utilizan dispositivos que no están en una conexión para enviar datos. Hay 3 tipos de canales de publicidad. Estos canales son usados para difusión de la información, para anunciar que pueden ser conectables, descubribles, para escaneare e iniciar conexiones. Los canales de datos son solo usados cuando se ha realizado la conexión. Hay 37 canales de datos, utilizan un motor adaptativo de salto de frecuencia para garantizar la robustez. Los canales de datos permiten enviar datos de un dispositivo a otro, también retransmitirlos. Los canales de datos pueden cifrarse y autenticarse por paquete. Para poder enviar la información entre emisor y receptor. Los paquetes incluyen información para identificar al receptor, además de una suma de comprobación que garantiza la validez del paquete. La estructura básica de los paquetes es idéntica entre los canales de publicidad y los canales de datos, con un mínimo de 80 bits de direccionamiento, cabecera e información de comprobación incluidos en todos y cada uno de los paquetes.



Figura B.3: Paquete de datos

Los paquetes son optimizados para incrementar su robustez usando 8 bits de preámbulo que es suficientemente grande para permitir al receptor sincronizar la temporización de bits y ajustar el control automático de ganancia de la radio; una dirección de acceso de 32 bits que es fija para los paquetes publicitarios, pero aleatoria y privada para los paquetes de datos. Una cabecera de 8 bits para describir el contenido del paquete. Un campo de 8 bits para describir la longitud de la carga útil. Finalmente una comprobación de redundancia cíclica(CRC) de 24 bits para garantizar que no hay errores de bits en los paquetes recibidos. El paquete más corto que se puede enviar en un paquete vacío de 80 microsegundos de longitud y el paquete más largo es un completamente cargado de 376 microsegundos. Los paquetes publicitarios suelen tener una duración de 128 microsegundos y los de datos 144 microsegundos. Para realizar la conexión son necesarios diferentes procedimientos como el escaneo del dispositivo, establecer y administrar la conexión con el otro dispositivo y una vez que se ha establecido la conexión el

B.1. Controlador

envío de datos a través de la conexión.

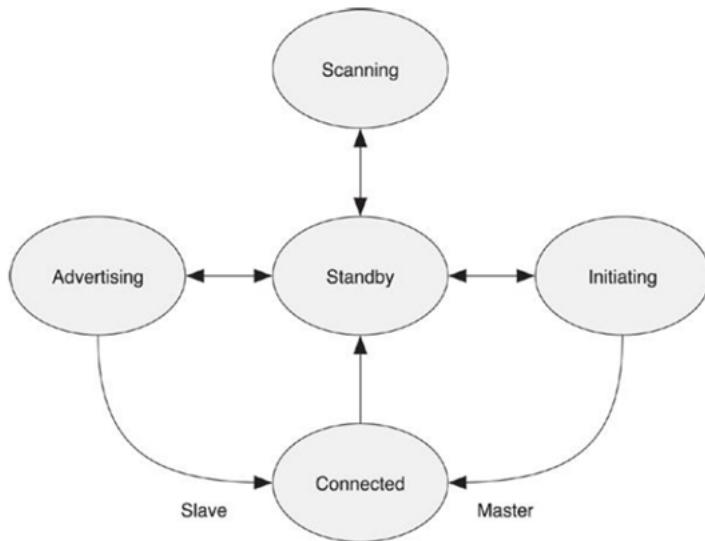


Figura B.4: Estados de la capa de enlace

- Standby: es el estado predeterminado en el que está la capa de enlace desde que funciona hasta que las capas del controlador le envían otra orden.
- Advertising: Aquí es donde los paquetes publicitarios son enviados para avisar a los maestros de que pueden solicitar una conexión, estos paquetes son enviados por el esclavo.
- Scanning: En este estado se reciben los paquetes de publicidad.
- Initiating: En este estado se puede establecer la conexión, aunque es el maestro el que envía paquetes de petición de conexión.
- Connected: Estado en el que la conexión se ha establecido, y es donde se intercambian paquetes los dispositivos conectados.

B.1.3. La interfaz entre Host y Controlador:

Para muchos dispositivos, se proporcionará una interfaz host/controlador (HCI) que permite al host comunicarse con el controlador a través de una interfaz estandarizada. Esta división arquitectónica ha demostrado ser muy popular en el Bluetooth clásico, en el que más del 60 % de todos los controladores Bluetooth se utilizan a través de la interfaz HCI. Permite a un host enviar comandos y datos al controlador y al controlador envíe eventos y datos al host. En realidad, se compone de dos partes separadas: la interfaz lógica y la interfaz física. La interfaz

lógica define los comandos y eventos y su comportamiento asociado. La interfaz lógica puede transmitirse a través de cualquiera de los transportes físicos o mediante una interfaz de programación de aplicaciones (API) local en el controlador, lo que permite integrarla en el controlador. La interfaz física define cómo se transportan los comandos, eventos y datos a través de diferentes tecnologías de conexión. Las interfaces físicas definidas incluyen USB 1 SDIO, y dos variantes de la UART. La mayoría de los controladores sólo soportarán una o posiblemente dos interfaces. También hay que tener en cuenta que para implementar una interfaz USB se necesita mucho hardware, y la interfaz no es la interfaz de bajo consumo, por lo que normalmente no se proporcionaría en un dispositivo Bluetooth de bajo consumo. Dado que la interfaz de controlador de host debe existir tanto en el controlador como en el host, la parte que se encuentra en el controlador suele denominarse interfaz de controlador de host inferior; la parte que se encuentra en el host se suele denominarse interfaz de controlador de host superior.

B.2. Host

El host contiene capas encargadas de multiplexar, protocolos y procedimientos para realizar varias funciones necesarias para llevar a cabo la comunicación mediante BLE. El host se construye por encima de la interfaz de usuario. Por encima la interfaz entre host y controlador nos encontramos la capa de control de la lógica de conexión y protocolo de adaptación (L2CAP) una capa de multiplexación. Después de esta capa encontramos la capa de gestión de la seguridad que se encarga de la autenticación y establecer conexiones seguras, a la misma altura, se encuentra la capa de protocolo de atributos(ATT) se encarga de exponer el estado del dispositivo. Finalmente encontramos la capa de perfil de acceso genérico (GAP) que define cómo los dispositivos se encuentran y se conectan entre ellos.

B.2.1. La lógica de conexión y protocolo de adaptación (L2CAP)

Es la capa de multiplexación para el BLE. Esta capa define dos conceptos básicos, los canales L2CAP y los comandos de señalización L2CAP. Un canal de L2CAP es un único canal de datos bidireccional que se termina en un protocolo o perfil concreto en el dispositivo par. Cada canal es independiente y puede tener su propio control de flujo y otra información de configuración asociada. Bluetooth clásico utiliza la mayoría de las características de L2CAP, incluidos identificadores de canal dinámicos, multiplexores de servicio de protocolo, retransmisión mejorada, etc. multiplexores, retransmisión mejorada y modos de transmisión. BLE sólo utiliza el mínimo absoluto de L2CAP. En Bluetooth de baja energía, sólo se utilizan canales fijos: uno para el canal de señalización, otro para el de seguridad y otro para el protocolo de atributos. Sólo hay un formato de trama, la trama B; Esta tiene un campo de longitud de dos octetos y un campo identificador de canal de dos octetos, como se ilustra en la

B.2. Host

siguiente figura. Este es el mismo formato de trama que el L2CAP clásico utiliza para cada canal hasta que los formatos de trama se negocien a algo más complejo.

Por ejemplo, en Bluetooth clásico, es posible tener formatos de trama que incluyan secuenciación de tramas y comprobaciones adicionales. Estas no son necesarias en Bluetooth low energy porque las comprobaciones en la capa de enlace son lo suficientemente potentes como para no necesitarlas, y el protocolo de atributo sencillo no necesita entregar paquetes de varios canales fuera de orden. Al simplificar los protocolos y realizar suficientes comprobaciones, sólo se necesitó un formato de trama.



Figura B.5: Paquete L2CAP

B.2.2. Protocolo de gestor de seguridad

El Gestor de Seguridad define un protocolo sencillo para el emparejamiento y la distribución de claves. El emparejamiento es el proceso de intentar confiar en otro dispositivo, normalmente autenticando el otro dispositivo. Al emparejamiento suele ir seguido de la encriptación del enlace y la distribución de claves. Mediante la distribución de claves secretos compartidos de un esclavo a un maestro, de modo que cuando estos dos dispositivos vuelvan a conectarse más adelante, puedan demostrar rápidamente su autenticidad. más tarde, puedan demostrar rápidamente su autenticidad encriptando con los secretos compartidos previamente distribuidos. previamente distribuidos. El gestor de seguridad también proporciona una caja de herramientas de seguridad para generar hashes de datos, generar valores de confirmación y claves a corto plazo utilizadas durante el emparejamiento.

B.2.3. El protocolo de atributos

El Protocolo de Atributos define un conjunto de reglas para acceder a los datos, define la estructura de los datos y la forma en que estos son presentados a un cliente, los datos están estructurados como atributos. No importa que un dispositivo sea maestro o esclavo, puede desempeñar el rol de cliente o servidor, porque solo depende la dirección en la que debe ir los datos, es decir, si el esclavo pide datos al maestro, el esclavo está siendo cliente y viceversa. Los datos se almacenan en un servidor de atributos en "atributos" que un cliente de atributos puede leer y escribir. El cliente envía solicitudes al servidor y éste responde con mensajes de

respuesta. El cliente puede utilizar estas solicitudes para encontrar todos los atributos en un servidor y luego leer y escribir estos atributos. El protocolo de atributos define seis tipos de mensajes: 1) peticiones enviadas del cliente al servidor; 2) respuestas enviadas del servidor al cliente en respuesta a una solicitud; 3) comandos enviados desde el cliente al servidor que no tienen respuesta; 4) notificaciones enviadas el servidor al cliente que no tienen confirmación. 5) indicaciones enviadas desde el servidor al cliente; 6) confirmaciones enviadas desde el cliente al servidor en respuesta a una indicación. Así, tanto el cliente como el servidor pueden iniciar la comunicación con mensajes que requieren una respuesta, o con mensajes que no requieren respuesta. Los atributos son direcciones, bits de datos etiquetados. Cada atributo tiene un manejador único que lo identifica, un tipo que identifica los datos almacenados en él y un valor. Por ejemplo, un atributo con el tipo Temperatura que tiene el valor 20,5º podría estar contenido dentro de un atributo con valor 0x01CE. El Protocolo de Atributos no define ningún tipo de atributo, aunque sí define que algunos atributos se pueden agrupar, y su semántica de grupo se puede descubrir a través del protocolo de atributos. El Protocolo de Atributos también define que algunos atributos tienen permisos, permisos para permitir a un cliente leer o escribir el valor de un atributo, o sólo permitir el acceso al valor del atributo si el cliente se ha autenticado o ha sido autorizado por el servidor. No es posible descubrir explícitamente los permisos de un atributo; eso sólo puede hacerse implícitamente enviando una petición y recibiendo un error como respuesta, indicando por qué no se puede completar la solicitud. El protocolo de atributos por sí solo es sin estado, cada transacción individual por ejemplo una única petición de lectura y una respuesta de lectura, no causa que el estado se guarde en el servidor. Esto implica que el protocolo por sí solo necesita muy poca memoria. Pero hay una excepción la petición de preparar y ejecutar escritura. Esta petición guarda un conjunto de valores que deben ser escritos en el servidor y luego ejecutados todos en secuencia, en una simple transición

B.2.4. Protocolo de gestión de Seguridad (SMP)

El Protocolo de Seguridad de la Capa de Enlace (SMP, por sus siglas en inglés) es la capa responsable de garantizar la seguridad en las comunicaciones de Bluetooth Low Energy (BLE). Esta capa facilita el intercambio y generación de claves de seguridad, las cuales son utilizadas para asegurar la confidencialidad de la comunicación mediante el cifrado de los datos. Esto ayuda a proteger la dirección pública del dispositivo, evitando que dispositivos no autorizados puedan detectar o acceder al mismo. Dentro del SMP se pueden distinguir dos roles diferentes:

[label=-]Iniciador: Este rol pertenece a la entidad central del Protocolo de Acceso a la Capa de Enlace (GAP) y también al dispositivo que actúa como maestro en la comunicación de enlace. Respondedor: Este rol pertenece al dispositivo periférico del Protocolo de Acceso a la Capa de Enlace (GAP) y también al dispositivo que actúa como

esclavo en la comunicación de enlace.

Estos roles desempeñan funciones específicas en el proceso de establecimiento de una conexión segura en BLE, contribuyendo a garantizar la integridad y confidencialidad de los datos transmitidos.

B.2.5. El perfil genérico de atributo (GATT)

El perfil genérico de atributos se sitúa por encima del protocolo de atributos. Define los tipos de atributos y cómo se utilizan. Introduce una serie de conceptos, como "características", "servicios", relaciones de inclusión entre servicios y "descriptores" de características. También define una serie procedimientos que pueden utilizarse para descubrir los servicios, las características y las relaciones entre servicios, así como para leer y escribir valores de características. Un servicio es una especie de "paquete" que contiene una tarea específica que puede hacer un dispositivo. Cuando decimos que es "inmutable", nos referimos a que una vez que se crea un servicio, no se puede cambiar. Esto es importante porque si queremos reutilizar un servicio, no podemos modificarlo. Si cambiamos el comportamiento de un servicio, necesitaríamos hacer actualizaciones y configuraciones complicadas, lo cual va en contra del objetivo de tener un modelo de conexión sencilla como el de BLE. Cuando hablamos de "encapsular", nos referimos a resumir las características de algo de manera concisa. En el caso de un servicio, toda la información relacionada se guarda y se presenta a través de un conjunto de atributos en un servidor de atributos. Al conocer los límites de un servicio en el servidor de atributos, podemos entender qué información abarca ese servicio. Por otro lado, "atómico" significa que algo es una unidad indivisible o un componente esencial de un sistema más grande. Los servicios atómicos son importantes porque cuando un servidor es más pequeño, tiene más posibilidades de ser reutilizable en otros contextos. Si creáramos servicios complejos que tuvieran múltiples comportamientos, posiblemente relacionados, la posibilidad de que éstos fueran reutilizados se reduce significativamente. Comportamiento significa la forma en que algo actúa en respuesta a una situación o estímulo particular. En servicios, el comportamiento significa lo que ocurre cuando se lee o escribe un atributo, o lo que hace que el atributo se notifique al cliente. Definir explícitamente el comportamiento es muy importante para la interoperabilidad. Si un servicio se especifica con un comportamiento mal definido, cada cliente podría actuar de forma diferente al interactuar con el servicio. Los servicios podrían actuar de forma diferente según el cliente que se conecte o, lo que es más importante, según el mismo cliente, o, lo que es más importante, el mismo servicio en distintos dispositivos actuará de forma diferente. En cuanto en los dispositivos, se destruye la interoperabilidad. Por lo tanto, un comportamiento explícitamente definido y comprobable, incluso para interacciones erróneas, favorece la interoperabilidad. Las relaciones entre servicios son clave para los

complejos comportamientos que exponen los dispositivos. Un servicio es atómico por naturaleza. Los comportamientos complejos no deben exponerse en un único servicio. Tomemos, por ejemplo, un dispositivo que puede medir la temperatura ambiente exponiendo un servicio de temperatura. El dispositivo podría estar alimentado por una batería, por lo que expondría un servicio de batería. Sin embargo, si la batería también tiene un sensor de temperatura, deberíamos poder exponer otra instancia del servicio de temperatura en el dispositivo. Para dar cabida a comportamientos complejos y relaciones entre servicios, éstos se dividen en dos tipos: servicios primarios y servicios secundarios. El tipo de un servicio no suele depender del servicio en sí, sino de cómo se utiliza ese servicio en un dispositivo. Un servicio primario es aquel que expone lo que hace el dispositivo, desde la perspectiva del usuario. Un servicio secundario es aquel que es utilizado por un servicio primario u otro servicio secundario para permitirle proporcionar su comportamiento completo. En el ejemplo anterior, el primer servicio de temperatura sería un servicio primario, el servicio de la batería también sería un servicio primario, mientras que el servicio de la temperatura de la batería sería un servicio secundario referenciado desde el servicio de batería.

B.2.6. El perfil genérico de acceso (GAP)

Es un elemento principal para que se pueda establecer una conexión entre dos dispositivos a través de bluetooth. Hay un marco a seguir para toda implementación de BLE que permite la realización de las operaciones principales a través de un estándar mundialmente extendido. El perfil de acceso genérico define cómo los dispositivos descubren, se conectan y presentan información útil a los usuarios. También define cómo los dispositivos pueden crear una relación permanente, denominada vinculación. Para esto, el perfil define cómo los dispositivos pueden ser detectables, conectables y vinculables. También describe cómo los dispositivos pueden utilizar procedimientos para descubrir otros dispositivos, conectarse a otros dispositivos, leer su nombre de dispositivo y vincularse con ellos. Esta capa también introduce el concepto de privacidad mediante el uso de direcciones privadas resolubles. La privacidad es importante para los dispositivos que anuncian constantemente su presencia para que otros dispositivos puedan descubrirlos y conectarse a ellos. Los dispositivos que quieren ser privados, sin embargo, deben transmitir utilizando una dirección aleatoria que cambia constantemente para que otros dispositivos no puedan determinar qué dispositivo es escuchando. Sin embargo, para permitir que dispositivos de confianza determinar si está cerca y permitir conexiones, la dirección privada debe poder resolverse, ser resoluble. Por lo tanto, el perfil de acceso genérico no sólo define cómo se pueden resolver las direcciones privadas, sino también cómo conectarse a dispositivos privados.

B.3. La capa de aplicación

Por encima del controlador y del host se encuentra la Capa de Aplicación. La capa de aplicación define tres tipos de especificaciones: característica, servicio y perfil. Cada una de estas especificaciones se basa en perfil genérico de atributos. El perfil genérico de atributos define atributos de agrupación para características y servicios, y las aplicaciones definen las especificaciones que utilizan estos grupos de atributos.

Apéndice C

Kit de desarrollo y empresa Cypress

C.1. CYPRESS



Figura C.1: Contenido del Kit

1	Placa base del BLE Pioneer kit, con módulo CY8CKIT-143A PSoC 4 BLE 256 KB
2	Dongle BLE. USB dongle CY5677 CySmart BLE 4.2
3	2 cable de sensor de proximidad (5 pulgadas cada uno)
4	4 cables de puente (4 pulgadas cada uno)
5	Guía de comienzo rápido
6	Cable USB estándar-A a mini-B
7	Pila de moneda (3V CR2032)

Cuadro C.1: Contenido del PIONEER KIT

C.2. Detalles de la placa base

- RGB LED
- BLE device reset button
- CapSense proximity header
- User button
- CapSense slider
- Arduino-compatible I/O headers (J2/J3/J4)
- Arduino-compatible power header (J1)
- Digilent Pmod-compatible I/O header (J5)
- Cypress F-RAM 1 Mb (FM24V10-G)
- PSoC 5LP I/O header (J8)
- PSoC 5LP programmer and debugger (CY8C5868LTI-LP039)
- Coin cell holder (bottom side)
- USB connector (J13)
- Power LED and Status LED
- System power supply jumper (J16) - LDO 1.9 V~5 V
- BLE power supply jumper / current measurement (J15)
- BLE module headers (J10/J11)

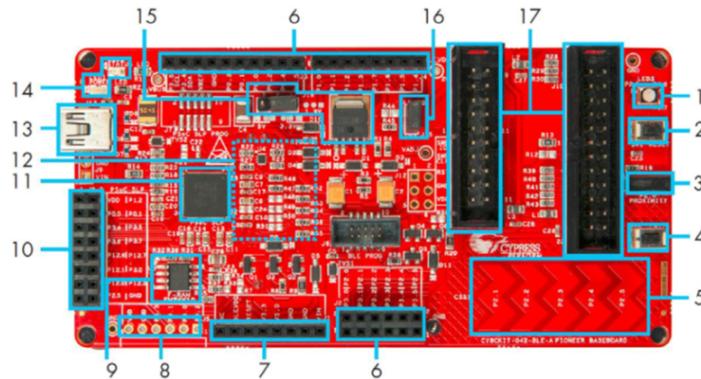


Figura C.2: Placa base

C.3. Módulo CY8CKIT-143A PSoC 4 BLE 256 KB

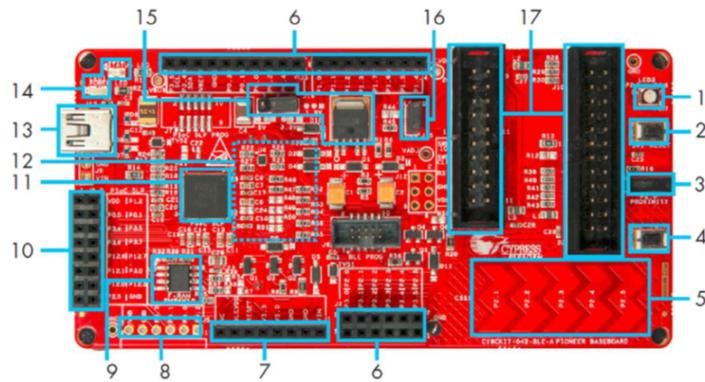


Figura C.3: Placa PSoC 4 BLE

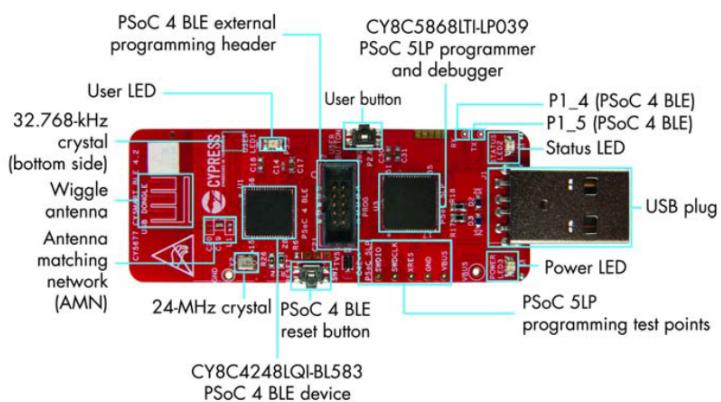


Figura C.4: Dongle

Núcleo CPU	ARM Cortex M0
CapSense	Si
ADC dedicado	SAR (1, 12-bit @ 1 msps)
EEPROM (KB)	0
Flash (KB)	256
Conductor directo LCD	Si
Frecuencia máxima para trabajar (MHz)	48
Temperatura máxima para trabajar (ºC)	85
Temperatura mínima para trabajar (ºC)	-40
Voltaje máximo para trabajar (V)	5.5
Voltaje mínimo para trabajar (V)	1.9
Número controladores CAN	0
Número bloques de tiempo continuo	2
Número canales DMA	8
Número de comparadores	2
Número de bloques de tiempo/contadores/PWM	4
Número de GPIOs	36
Número de amplificadores operacionales	4
Número de SIO	0
Número de bloques de comunicación serial	2
Número de salidas/entradas USB	0
Número de bloques analógicos universales	0
Número de bloques digitales universales	4
PLL	No
SRAM (KB)	32
Serie	PSoC 4200BL
Tipo USB	Ninguno

C.4. CYPRESS

Cypress Semiconductor Corporation era una empresa estadounidense que se especializaba en el diseño y fabricación de componentes electrónicos llamados semiconductores. Estos semiconductores incluían diversos productos como memorias flash, microcontroladores, chips para la gestión de energía, controladores capacitivos sensibles al tacto y soluciones de conectividad Bluetooth y USB. En 2019, Infineon Technologies, otra empresa del sector de semiconductores, anunció su intención de comprar Cypress por una suma de 9.400 millones de dólares. Esta adquisición se completó en abril de 2020, lo que permitió que Infineon se convirtiera en uno de los diez mayores fabricantes de semiconductores del mundo. Cypress tenía su sede principal en San José, California, y operaba en varios países como Estados Unidos, Irlanda, India y Filipinas. Algunos de sus competidores más importantes incluían empresas como Microchip Technology, Integrated Device Technology, Samsung Electronics y Xilinx.

C.5. PSoC

PSoC (Programmable System-on-Chip) es un tipo de componente electrónico desarrollado por Cypress Semiconductor Corporation que ofrece la capacidad de generar diversas funciones a través de hardware utilizando un código mínimo en lenguaje de programación C. Esta característica permite una solución más eficiente en comparación con el enfoque basado en software. Al requerir menos código, se ahorra espacio y tiempo, y se pueden desarrollar múltiples funciones sin afectar el procesamiento del CPU, dejándolo libre para tareas más importantes. Además, PSoC permite implementaciones de hardware que de otra manera serían complicadas. La arquitectura única de PSoC y sus propiedades son responsables de que Cypress sea líder mundial en sensores capacitivos. Dentro de la línea de productos PSoC, existen diferentes familias. PSoC 1 fue el primer sistema analógico programable que logró un rendimiento óptimo, superando intentos anteriores que eran limitados y lentos, Cypress comenzó a distribuir cantidades comerciales del PSoC 1 2002. Luego, se lanzaron las familias PSoC 3, PSoC 4 y PSoC 5, las cuales demandaron años de desarrollo meticuloso y presentaron precios altos debido a su avanzada tecnología en ese momento. En abril de 2013 Cypress lanzó la cuarta generación, el PSoC 4 que cuenta con una CPU ARM Cortex-M0 de 32 bits, con bloques analógicos programables (amplificadores operacionales y comparadores), bloques digitales programables (UDBs basados en PLD), enrutamiento programable y GPIO flexible (enrutar cualquier función a cualquier pin), un bloque de comunicación serie (para SPI, UART, I²C), un bloque temporizador/contador/PWM y mucho más. PSoC 4 fue diseñado para adaptarse a las necesidades de diversos clientes, y se subdividió en varias familias, como la 4000, 4100, 4200, L, M o BLE. Esta amplia variedad de opciones permite que PSoC satisfaga las necesidades de cualquier cliente. Por último, se introdujo la familia PSoC 6, que

mejoró ciertos aspectos de las familias anteriores, aunque no se profundizará en esos detalles, ya que no se utilizaron en el desarrollo del proyecto mencionado. En términos de programación, cada PSoC tiene registros USB propios, ya que funciona como un pequeño coprocesador. Esta es la principal diferencia entre PSoC y otros microcontroladores. Otra diferencia significativa se encuentra en la parte analógica, donde PSoC supera a otros microcontroladores disponibles en el mercado.

PSoC se utiliza en productos electrónicos de consumo como teléfonos inteligentes, tabletas, relojes inteligentes y sistemas de audio. En estos dispositivos, PSoC puede proporcionar funcionalidades de control de energía, gestión de sensores, interfaces táctiles y conectividad inalámbrica. En la industria automotriz para aplicaciones como control de sistemas de iluminación, control de climatización, control de motores y sistemas de información y entretenimiento en el automóvil. PSoC ofrece capacidades de procesamiento y control de señales analógicas que son esenciales para estas aplicaciones. Y como es el caso de este proyecto, PSoC es ampliamente utilizado en aplicaciones de IoT, debido a su capacidad de integrar múltiples funciones en un solo chip. Puede ser utilizado en dispositivos IoT como sensores inteligentes, gateways de comunicación y sistemas de control doméstico inteligente.

C.6. PSoC 4 BLE

El PSoC 4 BLE (Bluetooth Low Energy) está diseñado para aplicaciones de bajo consumo de energía y conectividad inalámbrica a través de la tecnología Bluetooth. El PSoC 4 BLE combina un microcontrolador de 32 bits con un sistema de radio Bluetooth integrado únicamente en un chip, gracias a ello, los desarrolladores pueden crear soluciones de IoT completas y dispositivos portátiles que tienen la capacidad de comunicarse inalámbricamente con otros dispositivos compatibles con Bluetooth. Características destacables:

- Núcleo ARM Cortex-M0: Obtenemos un rendimiento eficiente y un consumo de energía optimizado para aplicaciones de baja potencia. Además de la CPU, tiene hasta 256 KB de memoria flash con un acelerador de lectura y 32 KB de SRAM.
- Controlador CapSense con SmartSense con capacidad de panel táctil.
- Periféricos programables: tiene una amplia gama de periféricos programables, como DC (convertidor analógico-digital), UART (receptor/transmisor universal asíncrono), PWM (modulación por ancho de pulso), I2C (interfaz de comunicación en serie de dos hilos) y SPI (interfaz de comunicación en serie).
- Capacidades de procesamiento de señales analógicas: El PSoC 4 BLE nos permite procesar señales analógicas ya que tiene bloques de procesamiento de señales analógicas

ASP (Analog Signal Processing), lo que permite a los diseñadores realizar funciones de acondicionamiento y procesamiento de señales analógicas de manera flexible y personalizada.

- Sus dimensiones son muy pequeñas, 10mm x 10mm x 1.8mm.
- Módulo Bluetooth Low Energy (BLE): El PSoC 4 BLE incluye un sistema de radio BLE que permite la comunicación inalámbrica de bajo consumo con otros dispositivos habilitados para Bluetooth, como teléfonos inteligentes, tabletas y otros dispositivos IoT.
- Herramientas de desarrollo: Podemos programar el microcontrolador con PSoC Creator, que es un entorno de desarrollo integrado (IDE) que nos permite diseñar, programar y depurar las aplicaciones para PSoC 4 BLE de manera sencilla y eficiente.

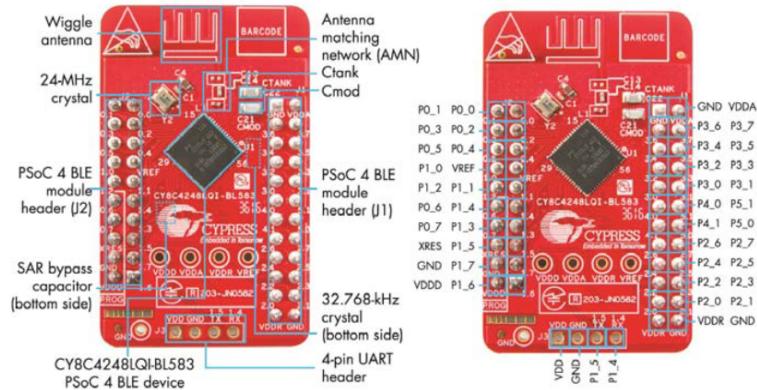


Figura C.5: Placa PSoC 4 BLE

C.7. PSoC Creator 4.4

Es el entorno de diseño integrado que nos permite realizar diseños hardware y software para PSoC, con componente de PSoC que ayudan a crear diseños de hardware por el método de esquemas clásicos. El programa se organiza en diferentes tipos de contenedores de información:

Es el entorno de diseño integrado que nos permite realizar diseños hardware y software para PSoC, con componente de PSoC que ayudan a crear diseños de hardware por el método de esquemas clásicos. El programa se organiza en diferentes tipos de contenedores de información:

- Espacio de trabajo (*Workspace*): Es el contenedor de nivel superior ya que almacena los diferentes proyectos que creamos o hemos creado.
- Proyecto (*Project*): El proyecto almacena los archivos hexadecimales, el diseño, el esquema del proyecto y los ficheros con el código fuente. Los proyectos siempre formarán

parte de un espacio de trabajo. Dentro de un mismo proyecto encontraremos proyectos de diseño y proyectos de biblioteca.

- Proyecto de diseño: Primero tenemos el esquema que está formado por los componentes que formarán parte de nuestro proyecto que arrastraremos y soltaremos, además de configurarlos. En este mismo proyecto escribiremos el código C. Una vez configurado todo lo anterior, construimos el proyecto creando los archivos hexadecimales y archivos del proyecto, finalmente ejecutamos para programar el dispositivo BLE.
- Proyecto de biblioteca: Es el conjunto de uno o varios componentes y su código asociado. Estos proyectos se podrán guardar en las bibliotecas para posteriormente poder utilizarlos en algún diseño.

Un proyecto es una mezcla de los dos tipos de proyectos que hemos explicado previamente. Los proyectos se guardan con la extensión “.cyprj”, y los archivos del proyecto se guardan en un directorio con el mismo nombre que el proyecto y la extensión “.cydsn”.

Descripción de las herramientas de PSoC Creator 4.4:

1. Organización del espacio de trabajo, el proyecto y sus archivos.
2. Espacio de esquema con los componentes de nuestro proyecto.
3. Listado de componentes.
4. Ventana de advertencia de errores.

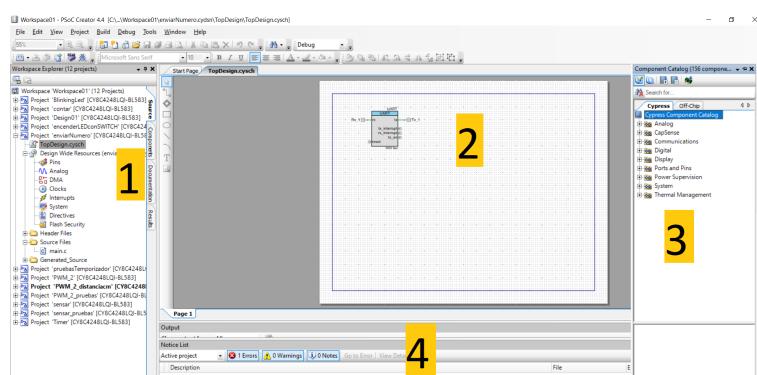


Figura C.6: PSoC Creator 4.4

Apéndice D

Selección del microcontrolador

Para realizar la elección de microcontrolador buscamos diferentes dispositivos de las marcas más populares del mercado con un precio menor de 50 euros. Los siguientes parámetros son los que vamos a tener en cuenta para la elección del microcontrolador.

- Procesador
- Precio
- Memoria
- Voltajes de operación
- Comunicación

PSoC 6

Procesador: CPU Arm Cortex-M4F (CM4) de 150 MHz y 32 bits con multiplicación de ciclo único, punto flotante y unidad de protección de memoria (MPU).

Precio: 30 euros.

Memoria:

- Flash de aplicación de 1 MB, flash auxiliar de 32 KB (AUXflash) y flash de supervisión de 32 KB (SFlash); soporte de lectura mientras se escribe (RWW). Dos cachés flash de 8-KB, una para cada CPU.
- SRAM de 288-KB con control de alimentación y retención de datos.
- 1-Kb programable una sola vez (OTP).

Voltajes de operación: de 1.7 V a 3.6 V

Comunicación: SPI, I2C, UART, USB.

PSoC 5

Procesador: El procesador que tiene es el ARM Cortex-M3 80 MHz de 32 bits además de un controlador DMA y procesador de filtro digital.

Precio: 20 euros.

Memoria:

- Hasta 256 KB de flash de programa, con caché y funciones de seguridad.
- Hasta 32 KB de flash adicional para código de corrección de errores (ECC).
- Hasta 64 KB de RAM.
- 2 KB de EEPROM.

Voltajes de operación: de 0.5 a 5.5.

Comunicación: SPI, I2C, UART y USB.

PSoC 4 BLE:

Procesador: ARM Cortex-M0 hasta 48 MHz.

Precio: 12 euros.

Memoria: Hasta 256 KB de flash, 32 KB de RAM.

Voltajes de operación: 1.7 V a 5.5 V.

Comunicación: Serial I2C, SPI, UART e inalámbrica BLE.

Arduino UNO

Procesador: Microcontrolador ATmega328P AVR de 8-bit CPU AVR hasta 16 MHz.

Precio: 30 euros.

Memoria:

- 32KB Flash.
- 1KB EEPROM.
- 2KB SRAM.

Voltajes de operación: 2.7-5.5 volts

Comunicación: UART, SPI e I2C y USB

Arduino Leonardo

Procesador: ATmega32u4

Precio: 25 euros.

Memoria:

- 16/32KB Flash
- 1.25/2.5KB SRAM
- 512Bytes/1KB EEPROM

Voltajes de operación: 2.7-5.5 volts

Comunicación: UART, 12C, SPI.

STM32WL55JC

Procesador: ARM Cortex M4 32 bits 48 MHz Precio: 48 euros Memoria:

- Hasta 256 Kbyte Flash
- Hasta 64 Kbyte SRAM

Voltajes de operación: 1.8V – 3.6 V

Comunicación: SPI, I2C, LPUART, USART,

STM32L151CCT6

Procesador: ARM Cortex-M3 32-bit de 32 kHz hasta 32 MHz

Precio: 9 euros

- Memoria Flash: Hasta 256 Kbyte
- 32 Kbyte de RAM
- 8 KB de EEPROM

Voltajes de operación: 1.65 V to 3.6 V

Comunicación: USB, USARTS, SPI e I2C

Una consideración importante en todo diseño es el coste, en nuestro caso, buscamos obtener un buen rendimiento a un costo razonable. Por lo tanto, descartamos inicialmente el microcontrolador STM32WL55JC debido a su precio.

La siguiente característica que observamos fue la memoria de los microcontroladores. Para tener cierto criterio en el descarte por memoria hicimos una pequeña búsqueda y las conclusiones esquematizadas fueron las siguientes:

- RAM:
 - Memoria de gran velocidad, pero guarda los datos de forma temporal, por lo que cuando el dispositivo se apaga, los datos se borran.
 - La RAM sirve para poder utilizar varias aplicaciones de forma simultánea.
- EEPROM:
 - Chip de memoria que retiene su contenido sin energía.
 - Funciona como RAM no volátil, pero grabar en EEPROM es mucho más lento que hacerlo en RAM.
- Flash:
 - Almacenar grandes cantidades de información y archivos en un espacio reducido.
 - Permite la lectura y escritura de múltiples posiciones.
 - Permite conseguir velocidades de funcionamiento superiores frente a la tecnología EEPROM.
 - Se utiliza principalmente para poder almacenar y transferir datos entre diferentes dispositivos digitales, como desde un ordenador a un smartphone o Tablet.

Esta fue la característica por la que descartamos las placas de Arduino ya que las placas de PSoC ofrecen mayor memoria a un precio inferior, y no solo mayor memoria, sino que los procesadores de los demás microcontroladores eran más potentes ya que ofrecían una frecuencia de reloj superior que les permite una capacidad superior tanto de cálculo como de ejecución de instrucciones.

El último filtro que utilizamos para la elección del microcontrolador fue la función de comunicación inalámbrica de nuestro dispositivo ya que es una de las características principales y este problema nos lo solucionaba tanto el PSoC 6 como el PSoC 4 porque incorporan radios BLE, por tanto, descartamos todos los demás.

Pero a pesar de que el PSoC 6 tiene mejores prestaciones en términos de memoria que el PSoC 4, nos decantamos por el PSoC 4 ya que para los requisitos de nuestro proyecto las

características de memoria eran más que suficientes y el hecho de que incorpora una radio BLE nos facilitaba el desarrollo del proyecto.

Otras razones por las que el PSoC 4 BLE era una buena elección para nuestro proyecto es que tiene la capacidad de emulación de dispositivos USB, esto nos permitiría interactuar con el ordenador u otro dispositivo mediante una conexión USB. También la amplia gama de periféricos que ofrece como ADC, PWM, UART, SPI e I2C.

El bajo consumo de energía también fue una característica clave, ya que el PSoC 4 BLE está diseñado para ser eficiente en cuanto a consumo de energía, haciéndolo adecuado para aplicaciones alimentadas con batería un requisito deseable para nuestro dispositivo.

Bibliografía

- [1] Bindicator. *Radar Onda Continua - Bindicator*. Ago. de 2020. URL: <https://es.bindicator.com.br/radar-onda-continua.html>.
- [2] Pro-Wave Electronics Corp. *Ultrasonic Sonar Ranging IC - PW0268*. URL: <http://www.prowave.com.tw>.
- [3] Cypress. *Creating a BLE Custom Profile*. Nov. de 2019. URL: <https://www.infineon.com/documentation/application-notes/an91162-creating-ble-custom-profile>.
- [4] Texas Instruments. *AWR1843 Single-Chip 77- to 79-GHz FMCW Radar Sensor*. URL: <https://www.ti.com/>.
- [5] KROHNE Group. *Medición de Nivel Radar FMCW – comparación de tecnología de 24 GHz y 80 GHz*. Jul. de 2017. URL: https://www.youtube.com/watch?v=dN6dG4wkJ9w&ab_channel=KROHNEGroup.
- [6] Lana Sarrate. “Medición de nivel sin contacto: ultrasonidos vs radar - Lana Sarrate”. En: *Lana Sarrate* (nov. de 2021). URL: <https://www.lanasarrate.es/blog/medicion-de-nivel-sin-contacto-ultrasonidos-vs-radar>.
- [7] *Sensores radiométricos para la medición de nivel*. Mar. de 2023. URL: <https://www.es.endress.com/es/instrumentacion-campo/medicion-nivel/medicion-nivel-radiometrica>.
- [8] MDAS Sensors. *Sonar Ranging Module SRM400*. URL: <https://es.farnell.com/>.
- [9] Kevin Townsend et al. *Getting Started with Bluetooth Low Energy*. O'Reilly Media, Inc., mayo de 2014.
- [10] Dipl.- Ing. (fh) Christian Wolff. *Fundamentos de radar - Radar de banda ultraancha*. Ene. de 2023. URL: <https://www.radartutorial.eu/02.basics/rp21.es.html>.
- [11] Dipl.- Ing. (fh) Christian Wolff. *Fundamentos de radar - Radar de onda continua*. Feb. de 2023. URL: <https://www.radartutorial.eu/02.basics/rp07.es.html#abs2>.
- [12] Dipl.- Ing. (fh) Christian Wolff. *Fundamentos de radar - Radar FMCW*. Feb. de 2023. URL: <https://www.radartutorial.eu/02.basics/rp08.es.html>.

- [13] Dipl.- Ing. (fh) Christian Wolff. *Fundamentos de radar - Radar pulsado*. Feb. de 2023.
URL: <https://www.radartutorial.eu/02.basics/rp05.es.html>.