Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Estudio y análisis de la tarjeta Thunderboard Sense y su aplicación en la adquisición de señales ultrasónicas

Autor: Carlos Alonso Lucea

Tutor: José Manuel Villadangos Carrizo

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

Estudio y análisis de la tarjeta Thunderboard Sense y su aplicación en la adquisición de señales ultrasónicas

Autor: Carlos Alonso Lucea

Tutor: José Manuel Villadangos Carrizo

TRIBUNAL:

Presidente: Julio Pastor Mendoza

Vocal 1º: Mª del Carmen Pérez Rubio

Vocal 2º: José Manuel Villadangos Carrizo

FECHA: 8 de enero de 2019

Agradecimientos

Quiero dar las gracias especialmente a mis padres Ángel y Pilar porque gracias a sus esfuerzos he llegado hasta aquí y sin ellos nada de esto habría sido posible. También a mi hermana Blanca que ha sido mi referente para querer dedicarme al mundo de la ingeniería. A Julia por creer siempre en mí y apoyarme todo este tiempo porque sin ella no habría podido con todo.

Por último, darle las gracias también a mi tutor José Manuel Villadangos por darme la oportunidad de trabajar en este proyecto con él y por todo el tiempo y ayuda prestado a lo largo de los últimos meses.

Índice

Índice de figuras

Índice de tablas

Índice de códigos

1. Resumen

El objetivo de este proyecto es estudiar el funcionamiento de la tarjeta de evaluación Thunderboard Sense y realizar una aplicación utilizando alguno de los sensores que ofrece.

La aplicación desarrollada será un sistema para determinar la orientación y posición de la placa en tiempo real. Para determinar la orientación se utilizará un sensor inercial IMU mientras que la posición se obtendrá a partir de de un sistema de balizamiento empleando balizas ultrasónicas codificadas.

Los datos de orientación y posición son adquiridos continuamente, y enviados vía bluetooth a un PC donde se mostrará una representación de la tarjeta en 3D.

2. Abstract

The objetive of this project is to study the operation of the Thunderboard Sense evaluation package and to make an application using some of its sensors.

The developed application will obtain the orientation and position of the package in real time. In order to get the orientation we will use an IMU inertial sensor while the position will be obtained using codified ultrasonic beacons.

The orientation and position data will be continuolly adquired and send by bluetooth to a PC where a representation of the package in 3D will be shown.

3. Palabras clave

Thunderboard Sense, sensor inercial IMU, Sistema de Posicionamiento Local Ultrasónico (U-LPS), Bluetooth

4. Memoria

4.1. Introducción

El presente documento puede dividirse en 3 grandes bloques principalmente. El primero se centrará en la placa Thunderboard Sense, se explicarán los distintos componentes que la forman centrándonos sobre todo en el núcleo de la misma, el SoC EFR32. En el segundo bloque se basará en la orientación. Se explicará cómo determinar la orientación de un objeto, así como el funcionamiento de los elementos utilizados para ello. En el tercero nos centraremos en la localización, se describirán los distintos métodos de obtención de la posición de un objeto y se explicará en detalle los elementos de la placa utilizados. En un último apartado, se tratará el proceso seguido para establecer la comunicación con la placa y el envío de datos desde la misma a través de un módulo bluetooth.

4.2. Placa Thunderboard Sense

En este primer capítulo se va a realizar una descripción detallada de las principales características que incluye la Thunderboard Sense. Es una placa desarrollada por Silicon Labs y basada en el SoC EFR32 Mighty Gecko. Es una placa pequeña y de bajo coste ideal para conectar con dispositivos IoT de bajo consumo. Soporta protocolos como ZigBee, Thread y Bluetooth de Baja Energía. Contiene numerosos sensores que se agrupan en diferentes bloques controlados por un microcontrolador. Los principales elementos de hardware que incluye son:

- EFR32 Mighty Gecko Wireless SoC con una antena cerámica de 2.4 GHz
- Microcontrolador EFM8 Sleepy Bee
- Sensor de temperatura y humedad Si7021
- Sensor de índice ultravioleta y luz ambiental Si1133
- Sensor de presión barométrica BMP280
- Sensor de calidad del aire interior CCS811
- Sensor inercial de 6 ejes ICM-20648
- Micrófono MEMS SPV1840
- 4 leds RGB, un led bicolor y 2 botones
- Memoria Flash SPI de 8 Mbit
- Depurador J-Link con un puerto virtual COM a través de un puerto USB Micro-B
- Conector Mini Simplicity

- 20 pines para conexiones externas
- Botón de reset
- Conectores para pila de botón CR2032 y para batería externa

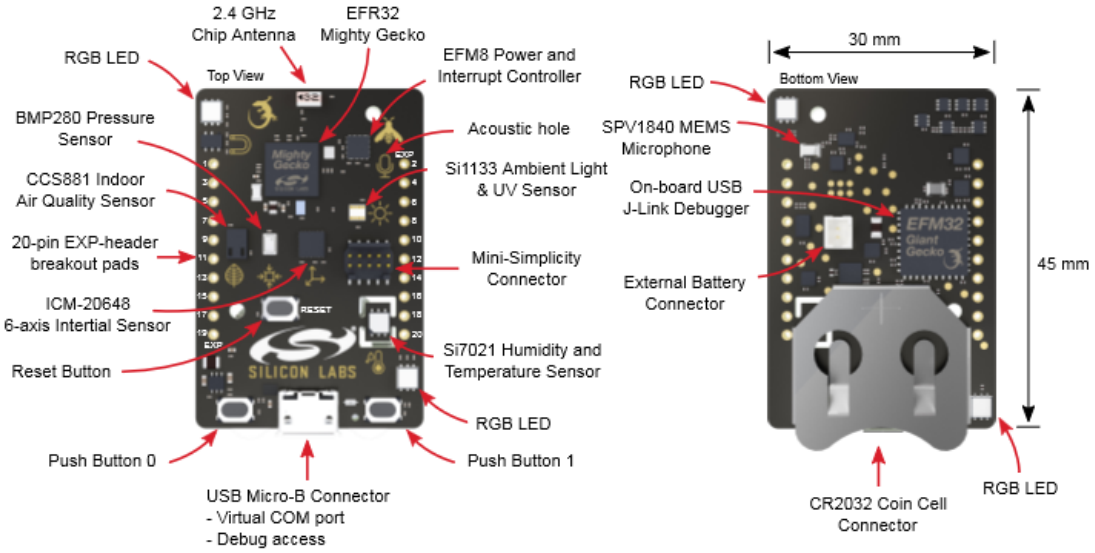

Figura 1. Diseño hardware de la Thunderboard Sense

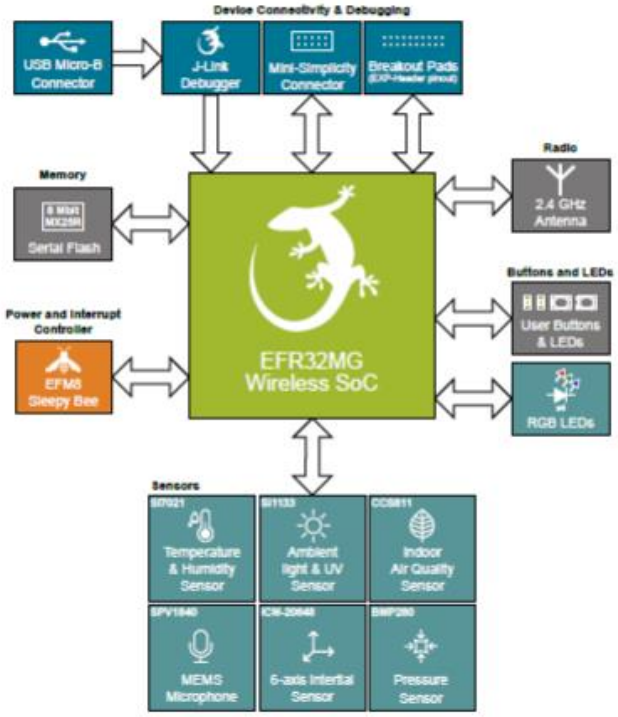

Figura 2. Diagrama de bloques de la Thunderboard Sense

4.2.1. Especificaciones

En las tablas 1 y 2 podemos ver los valores para los distintos parámetros de la Thunderboard Sense así como los recomendados por el fabricante.

Parameter	Symbol	Min	Typ	Max	Unit
USB Input Voltage	$V_{USB-MAX}$	0		+5.5	V
Supply Voltage VMCU	V_{VDDMAX}	0		+3.8	V
LDO output current	$I_{VREG-LOAD}$			300	mA
Voltage on any I/O pin	V_{DIGPIN}	-0.3		VMCU+0.3	V
Current per I/O pin (sink)	I_{IOMAX}			50	mA
Current per I/O pin (source)	I_{IOMAX}			50	mA
Current for all I/O pins (sink)	$I_{IOALLMAX}$			200	mA
Current for all I/O pins (source)	$I_{IOALLMAX}$			200	mA
Storage Temperature	T_{stg}	-40		+85	°C
ESD Susceptibility HBM (Human Body Model)	V_{ESD}			2	kV

Tabla 1. Valores máximos absolutos

Parameter	Symbol	Min	Typ	Max	Unit
Supply Input Voltage	V_{USB}	+4.5	+5.0	+5.5	V
Supply Input Voltage	V_{VBAT}	+2.0		+3.8	V
Supply Input Voltage (VMCU supplied externally)	V_{VMCU}	+2.0		+3.8	V
LDO Output Voltage	V_{REG}	+3.0	+3.3	+3.6	V
Operating Temperature	T_{OP}	0		70	°C

Tabla 2. Condiciones recomendadas

4.2.2. Sistema en Chip EFR32MG

El elemento principal de la placa es el sistema en chip (SoC) EFR32MG de 32 bits desarrollado por Silicon Labs y basado en el núcleo ARM Cortex-M4 con 40 Mhz de máxima frecuencia de operación.

Las principales características del SoC son:

- Sistema flexible de administración de energía que incluye 5 modos (EM0 a EM4) que proveen flexibilidad entre alto rendimiento y bajo consumo
- Memoria Flash de 256 kB

- Memoria RAM de 32 kB
- Pines GPIO configurables como entrada o salida en modos push-pull, drenador abierto o con resistencia de pull-up/pull-down
- Controlador Direct Memory Access (DMA) de 8 canales
- Peripheral Reflex System (PRS) de 12 canales
- Interfaces de comunicación
    - Universal Synchronous/Asynchronous Receiver/Transmitter (USART)
    - UART de baja energía
    - Interfaz I^2C
- Timers
    - 2 Timers de 16 bits
    - Timer de baja energía de 16 bits (LETIMER)
    - Timer de ultra baja energía de 32 bits (CRYOTIMER)
    - Real-Time Clock and Calendar (RTCC) de 32 bits
    - Contador de pulsos (PCNT)
    - Watchdog (WDOG)
- Periféricos analógicos
    - Conversor Analógico-Digital de 12 bits y 8 canales (ADC)
    - Conversor Digital-Analógico (DAC)
    - 2 comparadores analógicos (ACMP)
    - Puerto analógico (APORT)
- 6 osciladores integrados de mínimo consumo controlados por la CMU (Clock Management Unit)

Oscilador	Frecuencia
HFXO	38 MHz – 40 MHz
HFRCO	1 MHz – 38 MHz
AUXHFRCO	1 MHz – 38 MHz
LFRCO	32768 Hz
LFXO	32768 Hz
ULFRCO	1000 Hz

Tabla 3. Osciladores del EFR32

De las características comentadas, para este proyecto se van a utilizar 1 pin externo de la placa configurado como interrupción por flanco de bajada, el DMA, el PRS, las comunicaciones USART e I^2C, un Timer y el ADC con una frecuencia de muestreo de 100 kHz.

El oscilador principial utilizado será el HFXO a una frecuencia de 38.4 MHz que servirá como fuente para el reloj del Timer y las comunicaciones síncronas. Se utilizará

tambien el oscilador AUXHFRCO a una frecuancia de 19 MHz como fuente para el reloj de conversión del ADC.

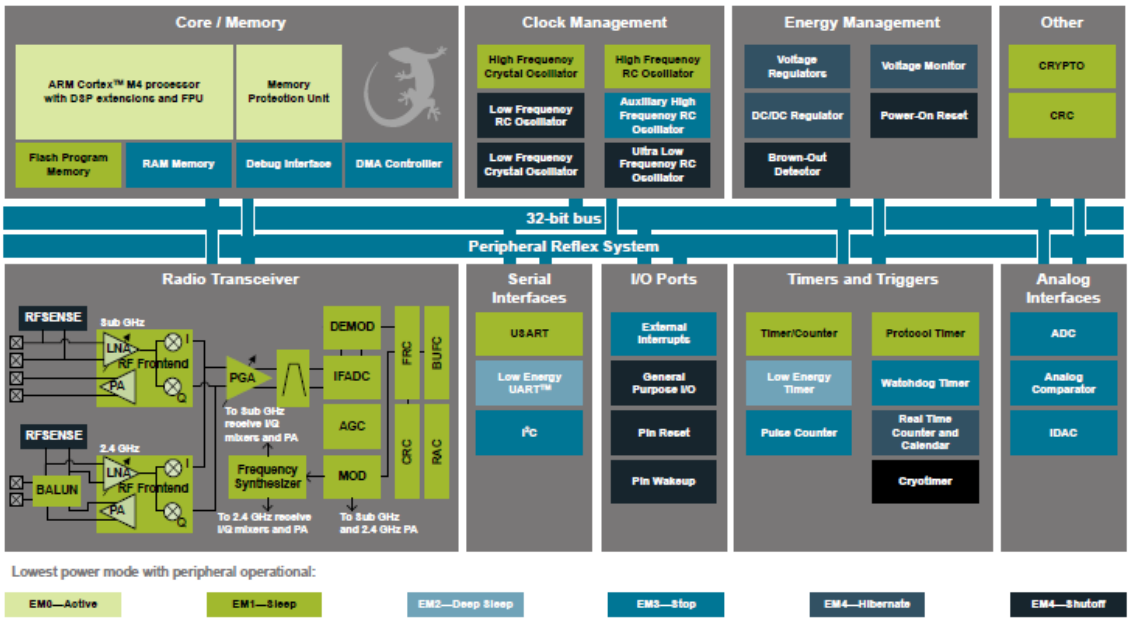

Figura 3. Diagrama de bloques del SoC EFR32

4.2.2.1. Entrada y Salida de Propósito General (GPIO)

Los pines GPIO en el EFR32 se organizan en puertos (desde A hasta F) de hasta 16 pines cada uno configurables como entrada o salida. Se pueden configurar hasta 16 pines como interrupción activa por flanco de subida o bajada.

Los registros que utilizaremos para su configuración son los siguientes:

- Con GPIO_Px_MODEL y GPIO_Px_MODEH podemos configurar el modo de los 16 pines (8 en cada registro) del puerto x.

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0				0x0				0x0				0x0				0x0				0x0				0x0				0x0			
Access	RW				RW				RW				RW				RW				RW				RW				RW			
Name	MODE7				MODE6				MODE5				MODE4				MODE3				MODE2				MODE1				MODE0			

Figura 4. Registro GPIO_Px_MODEL del EFR32

MODE0	0x0	RW	**Pin 0 Mode**
Configure mode for pin 0.			

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set
2	INPUTPULL	Input enabled. DOUT determines pull direction
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction
4	PUSHPULL	Push-pull output
5	PUSHPULLALT	Push-pull using alternate control
6	WIREDOR	Wired-or output
7	WIREDORPULLDOWN	Wired-or output with pull-down
8	WIREDAND	Open-drain output
9	WIREDANDFILTER	Open-drain output with filter
10	WIREDANDPULLUP	Open-drain output with pullup
11	WIREDANDPULLUP-FILTER	Open-drain output with filter and pullup
12	WIREDANDALT	Open-drain output using alternate control
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter
14	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output uisng alternate control with filter and pullup

Figura 5. Distintos modos de configuración de un pin GPIO

- Con GPIO_EXTIRISE y GPIO_EXTIFALL podemos configurar las interrupciones como activas por flanco de subida o bajada respectivamente. Cada bit n de EXTIRISE y EXTIFALL corresponderá con la interrupción n.

Offset	Bit Position																													
0x410	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0000							
Access																									RW							
Name																									EXTIRISE							

Figura 6. Registro GPIO_ EXTIRISE del EFR32

Offset	Bit Position																													
0x414	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0000								
Access																								RW								
Name																								EXTIFALL								

Figura 7. Registro GPIO_EXTIFALL del EFR32

Para configurar los distintos pines GPIO utilizaremos las siguientes funciones:

- *GPIO_PinModeSet()*: configura el modo de un pin GPIO. Recibe como argumentos el puerto, el número de pin, el modo deseado y un 0 para dejar el puerto a nivel bajo o un 1 para dejarlo a nivel alto.
- *GPIO_IntConfig():* configura la interrupción de un pin. Recibe como argumentos el puerto, el número de pin y 3 variables booleanas. La primera nos permite elegir si queremos que se active la interrupción en flancos de subida, la segunda si queremos que lo haga en flancos de bajada, y la última nos permite habilitar o no la interrupción en el momento de configurarla.
- *GPIO_PinOutSet():* fija el estado de un pin a nivel alto. Recibe como argumentos el puerto y el número de pin.
- *GPIO_PinOutClear():* fija el estado de un pin a nivel bajo. Recibe como argumentos el puerto y el número de pin.
- *GPIO_PinInGet()*: lee el valor de un pin GPIO y devuelve su valor. Recibe como argumentos el puerto y el número de pin.

4.2.2.2. Bus I^2C

El I^2C (Inter-Integrated Circuit) es un estandar de comunicación desarrollado por Philips en 1982. El bus I^2C requiere unicamente de dos cables para su funcionamiento, uno para la señal de reloj (CLK) y otro para el envío de datos (SDA).

Tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro inicia la comunicación con los esclavos pudiendo mandar o recibir datos de los mismos. Los esclavos no pueden iniciar la comunicación ni hablar entre ellos. Además, cada dispositivo conectado al bus debe tener una dirección única.

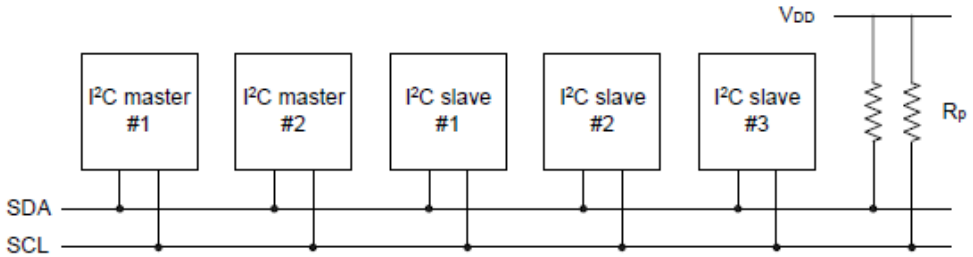

Figura 8. Esquema de comunicación por bus I²C

El proceso de comunicación a través del bus I²C es el siguiente:

- Comienza con una condición de START que se produce cuando la línea SDA pasa de nivel alto a nivel bajo mientras la línea SCL se mantiene a nivel alto
- Envío de un byte cuyos 7 bits más significativos contienen la dirección del esclavo. El bit restante indica si queremos leer o escribir (R/W)
- Envío de un bit de validación (ACK) que fija a nivel bajo la línea SDA durante un pulso de SCL
- Envío de los bytes de datos con su correspondiente bit de validación
- Finaliza con una condición de STOP que sucede cuando la línea SDA pasa de nivel bajo a nivel alto mientras la línea SCL se mantien a nivel alto

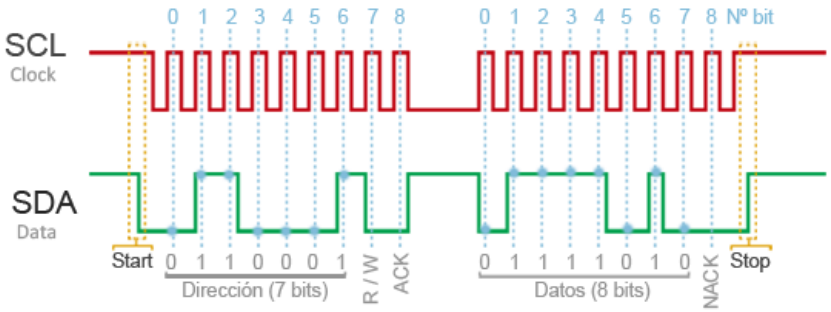

Figura 9. Ejemplo de un proceso de comunicación I²C

El bus I²C se utilizará para la comunicación entre el EFR32, que actuará como maestro, y el EFM8 que lo hará como esclavo. A través de este bus podremos habilitar los distintos sensores y leer las interrupciones de estos. En el EFR32, las líneas SDA y SCL están asignadas a los pines PC10 y PC11 respectivamente.

La configuración de los registros del bus I²C del EFR32 es la siguiente:

- Con el registro I2C_CTRL habilitaremos el módulo I²C. Para ello fijaremos a 1 el bit 0 (EN).

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0		0		0x0				0x0		0	0	0	0	0	0	0	0
Access															RW		RW		RW				RW		RW	RW	RW	RW	RW	RW	RW	RW
Name															CLTO		GIBITO		BITO				CLHR		TXBIL	GCAMEN	ARBDIS	AUTOSN	AUTOSE	AUTOACK	SLAVE	EN

Figura 10. Registro I2C_CTRL del EFR32

- Con los bits 1 (SCLPEN) y 0 (SDAPEN) del registro I2C_ROUTPEN fijados a 1 habilitaremos los pines SCL y SDA respectivamente.

Offset	Bit Position																													
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0	0
Access																															RW	RW
Name																															SCLPEN	SDAPEN

Figura 11. Registro I2C_ROUTPEN del EFR32

- Con el registro I2C_ROUTELOC0 elegiremos la localización de los pines SCL y SDA con los bits 13:8 (SCLLOC) y 5:0 (SDALOC). Para el caso de I²C la localización es la 15 por lo que escribiremos 0xF en SCLLOC y SDALOC.

Offset	Bit Position																													
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x00								0x00					
Access																			RW								RW					
Name																			SCLLOC								SDALOC					

Figura 12. Registro I2C_ROUTELOC0 del EFR32

Las funciones utilizadas para la comunicación I^2C son:

- *I2CSPM_Init()*: recibe como argumento un puntero a una estructura *I2CSPM_Init_TypeDef* como la que vemos a continuación e inicializa los registros del I2C según los valores dados a la estructura.

```
typedef struct
{
  I2C_TypeDef           *port;          // Peripheral port
  GPIO_Port_TypeDef     sclPort;        // SCL pin port number
  uint8_t               sclPin;         // SCL pin number
  GPIO_Port_TypeDef     sdaPort;        // SDA pin port number
  uint8_t               sdaPin;         // SDA pin number
  uint8_t               portLocationScl; //Port location of SCL signal
  uint8_t               portLocationSda; //Port location of SDA signal
  uint32_t              i2cRefFreq;     // I2C reference clock
} I2CSPM_Init_TypeDef;
```

Código 1. Estructura de inicialización I^2C

- *I2CSPM_Transfer()*: realiza una transferencia a través de I^2C. Recibe como argumentos un puntero al puerto especificado en la estructura *I2CSPM_Init_TypeDef* y un puntero a una estructura con los datos de la transferencia como se ve en el código 2.

```
typedef struct
{
  uint16_t addr;              // address to use after start
  uint16_t flags;              //flags defining sequence type R/W
  struct
  {
    uint8_t  *data;           // data bufer
    uint16_t len;             // number of bytes
  } buf[2];
} I2C_TransferSeq_TypeDef;
```

Código 2. Estructura de una transferencia I^2C

4.2.2.3. Protocolo USART

El protocolo USART (Universal Synchronous/Asynchronous Receiver Transmitter) es un protocolo de comunicaciones con la capacidad de transmitir y recibir datos simultáneamente. Los datos se transmiten en grupos de bits (trama) de manera serial, es decir, bit a bit. Estas tramas están formadas por 1 bit de inicio, varios bits de datos y uno o dos bits de parada. Además, se tiene la posibilidad de incluir o no un bit de paridad para detección y corrección de errores.

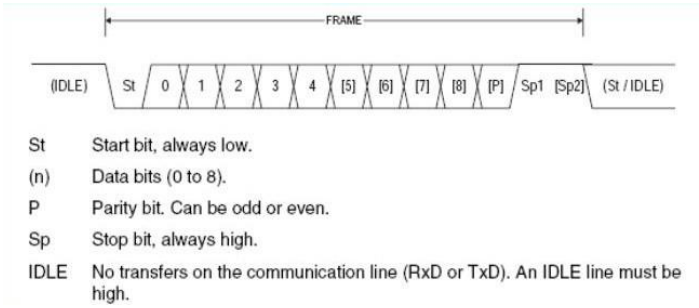

Figura 13. Esquema de una trama

El proceso de envío de una trama es el siguiente:

- Envío del bit de inicio (START) representado como a nivel bajo
- Envío de os bits de datos (8 en nuestro caso) de menos significativo a más significativo
- Posible envío de bit de paridad
- Envío de bit de parada (STOP) representado como a nivel alto
- Cuando no se estén enviando tramas, la línea de comunicación estará a nivel alto

También es posible configurar la velocidad de transmisión de los bits. La velocidad de transmisión (baudrate) se mide en baudios que equivalen al número de símbolos enviados por segundo. Para nuestro caso cada símbolo codifica un bit por lo que un baudio equivale a un bit por segundo (bps).

El bus de comunicación está formado por 2 líneas, Tx y Rx, de tal manera que conecten el pin Tx (transmisión) de uno de los dispositivos, con el pin Rx (recepción) del otro dispositivo y viceversa. En la Thunderboard Sense las líneas Tx y Rx corresponden a los pines PA0 y PA1 respectivamente.

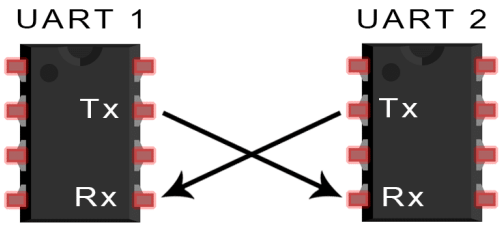

Figura 14. Conexión USART entre dos dispositivos

En el EFR32, la USART puede funcionar en modo síncrono o asíncrono. En modo síncrono, además de los datos, se trasmite una señal de reloj y ambos dispositivos envían

los datos en función de este reloj. En modo asíncrono, no se transmite ninguna señal de reloj a través del bus.

SYNC	Communication Mode	Supported Protocols
0	Asynchronous	RS-232, RS-485 (w/external driver), IrDA, ISO 7816
1	Synchronous	SPI, MicroWire, 3-wire

Tabla 4. Modos de la USART

Para este proyecto se van a emplear dos USART. La USART0 se utilizará para transmitir los datos de posición y orientación obtenidos al módulo bluetooth HC-06 para su posterior envío al PC. Se configurará en modo asíncrono a una velocidad de 460800 baudios, 8 bits de datos, un bit de parada y sin bit de paridad.

El baudrate en modo asíncrono se obtiene de la siguiente fórmula:

$$br = \frac{f_{HFPERCLK}}{(oversample * \left(1 + \frac{CKDIV}{256}\right))}$$

Donde $f_{HFPERCLK}$ es la frecuencia del HFPERCLK y $oversample$ es el ratio de oversampling, que en nuestro caso será 16. El HFPERCLK es una versión prescalada de HFXO que tendrá una frecuencia de 19 MHz. Despejando de la fórmula anterior:

$$CLKDIV = 256 * \left(\frac{f_{HFPERCLK}}{(oversample * br)} - 1\right) = 256 * (\frac{19000000}{16 * 460800} - 1) \cong 404$$

La USART1 se va a utilizar para la comunicación entre el EFR32 y el sensor inercial ICM-20648 mediante un bus SPI. Para ello se configurará en modo síncrono a una velocidad de 1000000 baudios y 8 bits de datos.

El baudrate en modo síncrono se obtiene de la siguiente fórmula:

$$br = \frac{f_{HFPERCLK}}{(2 * \left(1 + \frac{CLKDIV}{256}\right))}$$

De nuevo despejando:

$$CLKDIV = 256 * \left(\frac{f_{HFPERCLK}}{(2 * br)} - 1\right) = 256 * \left(\frac{19000000}{2 * 1000000} - 1\right) = 2176$$

El bus SPI es similar al bus I^2C pues tiene una arquitectura de tipo maestro-esclavo siendo el maestro el único que puede iniciar la comunicación. El intercambio de datos entre maestro y esclavos se realiza en dos líneas independientes, una del maestro a los esclavos (MOSI), y otra de los esclavos al maestro (MISO) por lo tanto el maestro puede enviar y recibir datos simultáneamente (comunicación Full Duplex). El maestro proporciona la señal de reloj (SCK) que mantiene a todos los dispositivos sincronizados. Además, se requiere una línea adicional (CS) por cada dispositivo esclavo conectado con el objetivo de seleccionar el dispositivo con el que se va a realizar la comunicación.

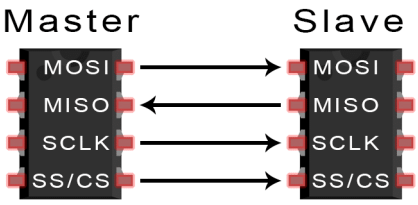

Figura 15. Conexión SPI

Por defecto, el maestro mantiene las líneas CS a nivel alto. Cuando quiere iniciar una comunicación con un esclavo, pone a nivel bajo la línea CS correspondiente, se habilita dicho esclavo y se inicia la comunicación. En cada pulso de la señal de reloj el maestro envía un bit a la vez que puede recibir otro del esclavo.

El bus SPI puede funcionar de 4 modos distintos dependiendo de la fase (CLKPHA) y la polaridad (CLKPOL) de la señal de reloj que determinará cuando se produce el envío del bit:

- Modo 0: mientras no se envíen datos el reloj se encuentra a nivel bajo y el envío se produce en el flanco ascendente
- Modo 1: mientras no se envíen datos el reloj se encuentra a nivel bajo y el envío se produce en el flanco descendente
- Modo 2: mientras no se envíen datos el reloj se encuentra a nivel alto y el envío se produce en el flanco descendente
- Modo 3: mientras no se envíen datos el reloj se encuentra a nivel alto y el envío se produce en el flanco descendente

SPI mode	CLKPOL	CLKPHA	Leading edge	Trailing edge
0	0	0	Rising, sample	Falling, set-up
1	0	1	Rising, set-up	Falling, sample
2	1	0	Falling, sample	Rising, set-up
3	1	1	Falling, set-up	Rising, sample

Tabla 5. Distintos modos en SPI

A continuación, vamos a ver los registros utilizados para configurar las USART:

- El bit 0 (SYNC) del registro USARTn_CTRL nos permite seleccionar el modo de funcionamiento de la USART. Fijándolo a 0 elegiremos el modo asíncrono y fijándolo a 1 el modo síncrono. Con los bits 6:5 (OVS) fijados a 0 seleccionaremos un oversampling de 16. Por último, mediante los bits 9 (CLKPHA) y 8 (CLKPOL) podemos elegir el modo de funcionamiento del bus SPI (USART1) según la tabla 5. Para nuestra aplicación será el modo 0 por lo que fijaremos a 0 dichos bits.

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0x0		0	0	0	0	0
Access	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW		RW	RW	RW	RW	RW
Name	SMSDELAY	MVDIS	AUTOTX	BYTESWAP			SSSEARLY	ERRSTX	ERRSRX	ERRSDMA	BIT8DV	SKIPPERRF	SCRETRANS	SCMODE	AUTOTRI	AUTOCS	CSINV	TXINV	RXINV	TXBIL	CSMA	MSBF	CLKPHA	CLKPOL		OVS		MPAB	MPM	CCEN	LOOPBK	SYNC

Figura 16. Registro USARTn_CTRL del EFR32

- Con el registro USARTn_FRAME configuraremos las características de las tramas. Escribiremos un 0x1 en los bits 13:12 (STOPBITS) para determinar que se envíe solo 1 bit de STOP, fijaremos a 0 los bits 9:8 (PARITY) para no enviar bits de paridad y escribiremos un 0x5 en los bits 3:0 (DATABITS) para que se envíen 8 bits de datos por trama.

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x1				0x0						0x5			
Access																			RW				RW						RW			
Name																			STOPBITS				PARITY						DATABITS			

Figura 17. Registro USARTn_CTRL del EFR32

- Con el registro USARTn_CMD habilitaremos el envío y recepción de datos fijando a 1 los bits 2 (TXEN) y 0 (RXEN).

Offset	Bit Position																															
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0	0	0	0	0	0	0	0	0	0	0	0
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CLEARRX	CLEARTX	TXTRIDIS	TXTRIEN	RXBLOCKDIS	RXBLOCKEN	MASTERDIS	MASTEREN	TXDIS	TXEN	RXDIS	RXEN

Figura 18. Registro USARTn_CMD del EFR32

- En el registro USARTn_CLKDIV configuraremos al baudrate escribiendo en los bits 22:3 (DIV) los valores de $CLKDIV$ calculados anteriormente.

Offset	Bit Position																															
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0									0x00000																						
Access	RW									RWH																						
Name	AUTOBAUDEN									DIV																						

Figura 19. Registro USARTn_CLKDIV del EFR32

- El registro USARTn_ROUTEPEN nos permite habilitar los pines utilizados por la USART. Fijando a 1 el bit 1 (TXPEN) habilitaremos el pin TX y fijando a 1 el bit 0 (RXPEN) habilitaremos el pin RX.

Offset	Bit Position																															
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											RTSPEN	CTSPEN	CLKPEN	CSPEN	TXPEN	RXPEN

Figura 20. Registro USARTn_ROUTEPEN del EFR32

- Con el registro USARTn_ROUTELOC0 elegiremos la localización de los pines TX y RX con los bits 13:8 (SCLLOC) y 5:0 (SDALOC). Para el caso de la USART0 la

localización de los pines es la 0 por lo que escribiremos un 0x0 en SCLLOC y SDDALOC y para el caso de la USART1 la localización es la 11 por lo que escribiremos 0xB.

Offset	Bit Position																													
0x078	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CLKLOC								CSLOC								TXLOC								RXLOC					

Figura 21. Registro USARTn_ROUTELOC0 del EFR32

Las funciones utilizadas para la comunicación usart serán las siguientes:

- *USART_InitAsync()*: inicializa una USART en modo asíncrono. Recibe como argumentos un puntero a la dirección de la USART a inicializar y un puntero a una estructura como la que se ve en el código 3.

```
typedef struct
{
  /** Specifies whether TX and/or RX shall be enabled when init completed. */
  USART_Enable_TypeDef    enable;
  uint32_t                baudrate;          // Desired baudrate
  USART_OVS_TypeDef       oversampling;      // Oversampling used
  USART_Databits_TypeDef  databits;          // Number of databits in frame
  USART_Parity_TypeDef    parity;            // Parity mode to use
  USART_Stopbits_TypeDef  stopbits;          // Number of stopbits to use
  bool                    prsRxEnable;       //Enable USART Rx via PRS

  /** Select PRS channel for USART Rx. (Only valid if prsRxEnable is true). */
  USART_PrsRxCh_TypeDef  prsRxCh;
} USART_InitAsync_TypeDef
```

Código 3. Estructura para inicializar una USART en modo asíncrono

- *USART_InitSync()*: inicializa una USART en modo síncrono. Recibe como argumentos un puntero a la dirección de la USART a inicializar y un puntero a una estructura como la que se ve en el código 3.

```
typedef struct
{
  /** Specifies whether TX and/or RX shall be enabled when init completed. */
  USART_Enable_TypeDef    enable;
  uint32_t                baudrate;          // Desired baudrate
  USART_Databits_TypeDef  databits;          // Number of databits in frame

  //Select if to operate in master or slave mode
  bool                    master;

  //Select if to send most or least significant bit first
  bool                    msbf;
  USART_ClockMode_TypeDef clockMode;         // Clock polarity/phase mode
  bool                    prsRxEnable;       //Enable USART Rx via PRS
```

```
  /** Select PRS channel for USART Rx. (Only valid if prsRxEnable is true). */
  USART_PrsRxCh_TypeDef   prsRxCh;
} USART_InitSync_TypeDef;
```

Código 4. Estructura para inicializar una USART en modo síncrono

- *USART_Tx()*: transmite una trama de 4 a 9 bits. Recibe como argumentos un puntero a la dirección de la USART ya inicializada y los datos a transmitir.
- *USART_Rx()*: función para recibir una trama de 4 a 9 bits. Se le pasa como argumento un puntero a la dirección de la USART ya inicializada y devuelve los datos transmitidos.

4.2.3. Microcontrolador EFM8SB

El EFM8 es un microcontrolador (MCU) de 8 bits desarrollado por Silicon Labs ideal para sistemas de bajo consumo y con una frecuencia de operación de hasta 25 MHz.

Las principales características que incluye son:

- Memoria Flash de 8 kB
- Memoria RAM de 512 bytes
- Pines GPIO
- Interfaces de comunicación
  - Universal Asynchronous Receiver/Transmitter (UART)
  - Bus SPI
  - System Management Bus (SMB) compatible con interfaz I^2C
- Timers
  - Real-Time Clock (RTC) de 32 bits
  - Programmable Counter Array (PCA)
  - 4 Timers de 16 bits
  - Watchdog (WDT)
- Periféricos analógicos
  - Conversor Analógico-Digital (ADC) de 12 bits
  - Comparador (CMP)
- 3 osciladores internos, incluyendo el del RTC cuya frecuencia es elegible entre la que ofrece un cristal externo de 32 kHz o un oscilador interno de 16.4 kHz

Oscilador	Frecuencia
LPOSC	20 MHz
HFOSC	24.5 MHz
RTCOSC	32 kHz o 16.4 kHz

Tabla 6. Osciladores del EFM8

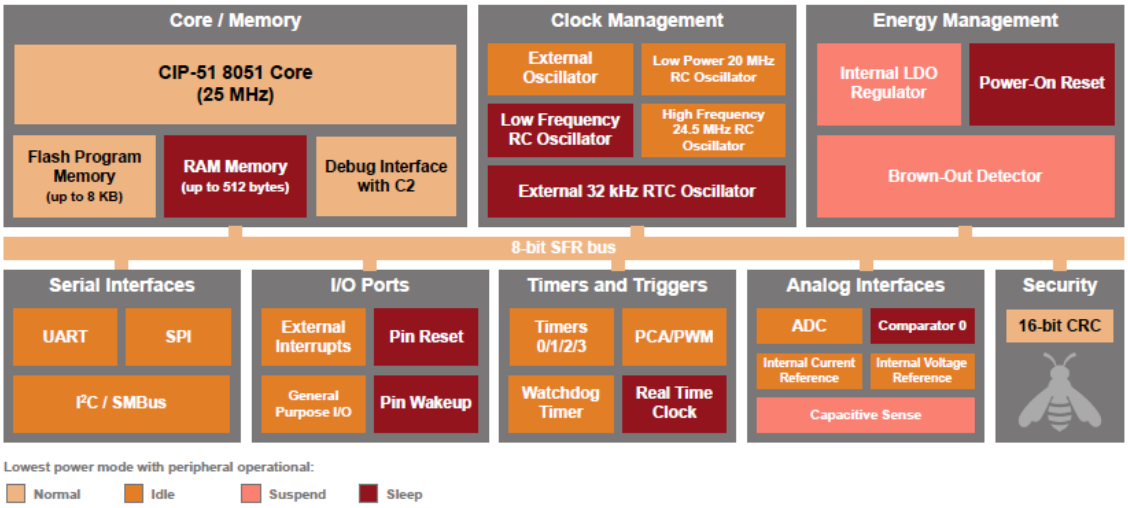

Figura 22. Diagrama de bloques del MCU EFM8SB

En la Thunderboard Sense, el EFM8 actúa como Controlador de Alimentación e Interrupción para los sensores y los leds RGB. Se comunica con el EFR32 mediante un bus I^2C. Con el objetivo de ahorrar energía, cuando no se esté utilizando el EFM8 estará dormido. Para despertarlo, el EFR32 debe enviar un pulso a nivel bajo de mínimo 4 µs a través del pin INT/WAKE (PD10). El EFM8 permanecerá activo mientras la línea INT/WAKE este a nivel bajo y durante 1 ms más después de pasar a nivel alto. Si hay actividad en el bus I^2C mientras está activo, seguirá activo hasta 1 ms después de que haya cesado la actividad.

Una vez despierto, el EFM8 puede habilitar o deshabilitar los diferentes sensores y LEDs RGB de la placa. Cuando se produce una interrupción en alguno de los sensores, el EFM8 notifica al EFR32 con un pulso a nivel bajo a través del mismo pin INT/WAKE.

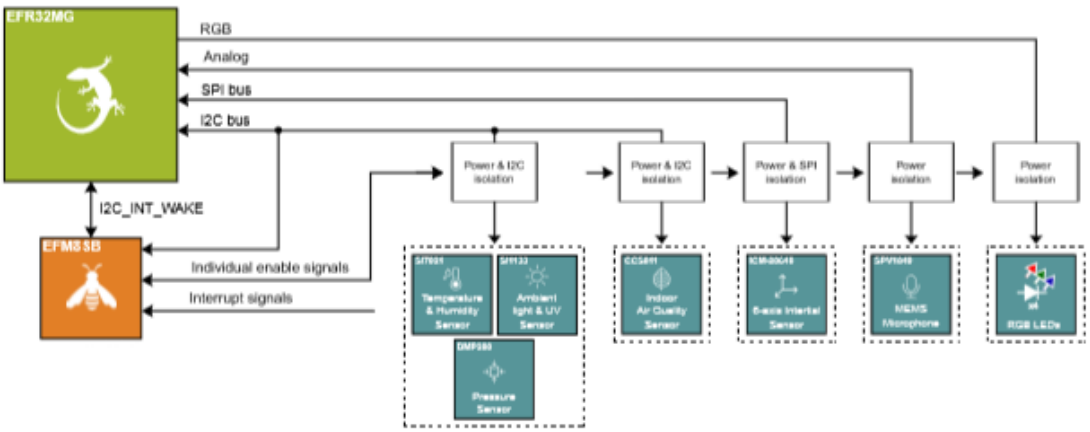

Figura 23. Diagrama de bloques del Controlador de Alimentación e Interrupción

El EFM8 presenta una interfaz de registros que pueden ser leídos y escritos por el EFR32. Para habilitar los sensores tan solo se debe escribir un 1 en el bit 0 del registro de control correspondiente al sensor.

Addr	Name	Type	Description	Reset
0x00	IMU_CTRL	RW	**Control register for the inertial sensor** • Bits 7:1 - Reserved • Bit 0 - Write 1 to this bit to enable power and SPI access	0x00
0x01	ENV_SENSOR_CTRL	RW	**Environmental sensor group control register** • Bits 7:1 - Reserved • Bit 0 - Write 1 to this bit to enable power and I²C access to the environmental sensor group	0x00
0x02	MIC_CTRL	RW	**Microphone control register** • Bits 7:1 - Reserved • Bit 0 - Write 1 to this bit to enable power to the microphone and microphone gain/filter circuit	0x00
0x03	CCS_CTRL	RW	**Indoor air quality sensor control register** • Bits 7:2 - Reserved • Bit 1 - Write 1 to this bit to wake up the indoor air quality sensor • Bit 0 - Write 1 to this to enable power and I²C access to the indoor air quality sensor	0x00
0x04	LED_CTRL	RW	**Control register for the RGB LEDs** • Bits 7:4 - Individual enable bit for each RGB LED • Bits 3:1 - Reserved • Bit 0 - Write 1 to this bit to enable the RGB LED voltage regulator	0x00
0x05	INT_ENABLE	RW	**Interrupt enable register** • Bits 7:3 - Reserved • Bit 2 - UV_ALS interrupt enable • Bit 1 - IMU interrupt enable • Bit 0 - CCS811 interrupt enable	0x00
0x06	INT_CLEAR	W	**Interrupt clear register** • Bits 7:3 - Reserved • Bit 2 - Clears UV_ALS interrupt status flag • Bit 1 - Clears IMU interrupt status flag • Bit 0 - Clears CCS811 interrupt status flag	0x00
0x07	INT_STATUS	R	**Interrupt status register** • Bits 7:3 - Reserved • Bit 2 - UV_ALS interrupt status flag • Bit 1 - IMU interrupt status flag • Bit 0 - CCS811 interrupt status flag	0x00

Tabla 7. Mapa de registros del EFM8

En este proyecto tan solo se han utilizado el sensor inercial y el micrófono. Vamos a ver a continuación el procedimiento que se ha seguido para habilitarlos.

Lo primero que haremos será configurar el pin INT/WAKE (PD10) como drenador abierto mediante la función *GPIO_PinModeSet()*. Se ha configurado de este modo puesto que tanto el EFR32 como el EFM8 deben poder cambiar el estado del pin, el EFR32 para despertar al EFM8 y el EFM8 para avisar de que se ha producido una interrupción en uno de los sensores.

Una vez configurado el pin, enviaremos el pulso a nivel bajo para iniciar la comunicación. Para poder escribir en los registros se debe enviar a través del bus la dirección 0x90 con el bit de R/W a 0. A continuación se enviará la dirección del registro que se desea habilitar, y un 1 que se escribirá en el bit 0 del registro para habilitarlo. Como se ve en la tabla 7, para el sensor inercial se enviará la dirección 0x00 (IMU_CTRL) y para el micrófono la 0x02 (MIC_CTRL).

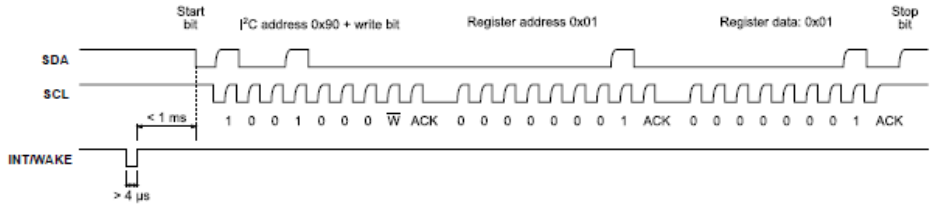

Figura 24. Ejemplo de una escritura vía I^2C

El EFM8 notifica al EFR32 con un pulso a nivel bajo cuando se produce una interrupción en un sensor, por lo tanto, se va a configurar el pin INT/WAKE como interrupción activa por flanco de bajada mediante la función *GPIO_IntConfig()*. Mediante el registro INT_ENABLE se habilitan las interrupciones de los sensores. En nuestra aplicación solo necesitaremos habilitar la interrupción del sensor inercial (bit 1 del registro INT_ENABLE). Para activarla se sigue el mismo procedimiento que para habilitar los sensores, se enviará la dirección 0x90, seguida de la dirección del registro INT_ENABLE (dirección 0x05) y por último se enviará un 0x2 para escribir un 1 en el bit 1.

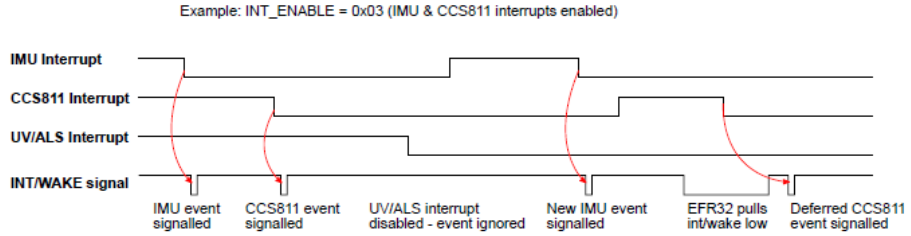

Figura 25. Ejemplo de funcionamiento del Controlador de Interrupción

Para un correcto funcionamiento de las interrupciones, se recomienda seguir el siguiente procedimiento a la hora de activar el EFM8 a través del pin INT/WAKE:

- Deshabilitar la interrupción por flanco de bajada en el pin INT/WAKE (PD10)
- Poner el pin a nivel bajo
- Esperar un mínimo 4 µs
- Habilitar la interrupción por flanco de bajada
- Poner el pin a nivel alto

Para eliminar los flags de las interrupciones se debe escribir un 1 en el bit correspondiente al sensor que generó la interrupción en el registro INT_CLEAR siguiendo el mismo procedimiento ya comentado.

Las funciones utilizadas para poder llevar a cabo los procedimientos comentados son las siguientes:

- *picWake()*: realiza el proceso de despertar al EFM8 a través del pin INT/WAKE.

```
static inline void picWake( void )
{
   volatile int delay;

   GPIO_IntDisable( 1 << BOARD_PIC_INT_WAKE_PIN );

   GPIO_PinOutClear( BOARD_PIC_INT_WAKE_PORT, BOARD_PIC_INT_WAKE_PIN );

   /* Minimum 5 us delay */
   delay = 27;
   while(--delay);

   GPIO_IntClear( 1 << BOARD_PIC_INT_WAKE_PIN );
   GPIO_IntEnable( 1 << BOARD_PIC_INT_WAKE_PIN );

   GPIO_PinOutSet( BOARD_PIC_INT_WAKE_PORT, BOARD_PIC_INT_WAKE_PIN );


   return;
}
```

Código 5. Función *picWake()*

- *picSleep()*: pone al EFM8 en estado dormido.

```
static inline void picSleep( void )
{
   GPIO_IntClear( 1 << BOARD_PIC_INT_WAKE_PIN );
   GPIO_IntEnable( 1 << BOARD_PIC_INT_WAKE_PIN );

   GPIO_PinOutSet( BOARD_PIC_INT_WAKE_PORT, BOARD_PIC_INT_WAKE_PIN );

   return;
}
```

Código 6. Función *picSleep()*

- *picWriteReg()*: escribe en un registro del Controlador de Alimentación e Interrupción. Recibe como argumentos la dirección del registro y los datos a escribir.

```
static uint32_t picWriteReg( uint8_t addr, uint8_t value )
{
   uint32_t status;

   I2C_TransferSeq_TypeDef seq;

   seq.addr        = 0x90;
   seq.flags       = I2C_FLAG_WRITE_WRITE;
   seq.buf[0].len  = 1;
   seq.buf[0].data = &addr;
   seq.buf[1].len  = 1;
   seq.buf[1].data = &value;

   I2CSPM_Transfer( i2cDevice, &seq );

   return;
}
```

Código 7. Función *picWriteReg()*

4.2.4. Sensores

Por defecto, todos los sensores están desactivados cuando se enciende la placa y, además, para obtener un bajo consumo de energía, se desactivan mientras no se estén utilizando. Vamos a proceder a comentar las principales características de cada uno de los sensores que componen la placa.

4.2.4.1. Si7021 – Sensor de temperatura y humedad relativa

El Si7021 es un sensor monolítico CMOS IC que incluye, además de los elementos de sensor de temperatura y humedad, un conversor analógico-digital, procesamiento de señal, calibración de datos y una interfaz I^2C.

Para utilizar el sensor debemos escribir un 0x01 en el registro ENV_SENSOR_CTRL por medio del EFM8. Esto provocará que se conecte el bus I^2C del sensor al bus I^2C principal. El Si7021 comparte bus I^2C con los sensores Si1133 y BMP280 por lo que, al activarlo, también se activaran estos últimos.

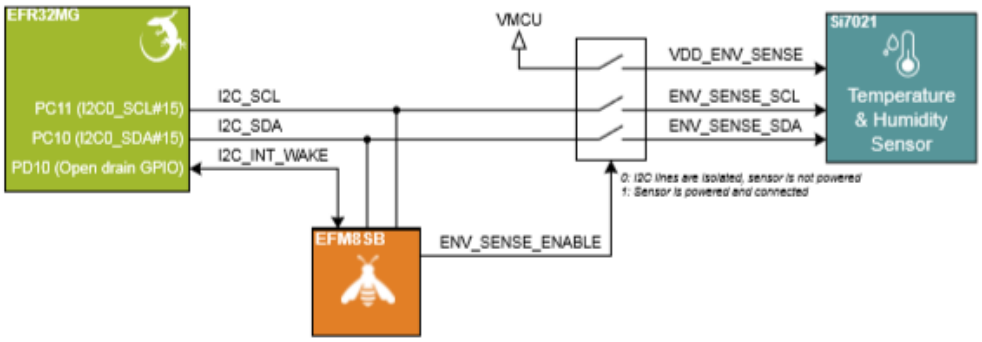

Figura 26. Diagrama de bloques del Si7021

4.2.4.2. Si1133 - Sensor de índice ultravioleta y luz ambiental

El Si1133 posee una interfaz digital I^2C e incluye convertidores analógico-digitales de 23 bits, fotodiodos de alta sensibilidad para luz ultravioleta, visible e infrarroja, y un procesador digital de señal.

Como se ha comentado anteriormente, para activarlo escribiremos un 0x01 en el registro ENV_SENSOR_CTRL, lo que conectará el bus I^2C del sensor al bus I^2C principal. Además, este sensor envía una interrupción al EFM8 cada vez que obtenga una nueva muestra.

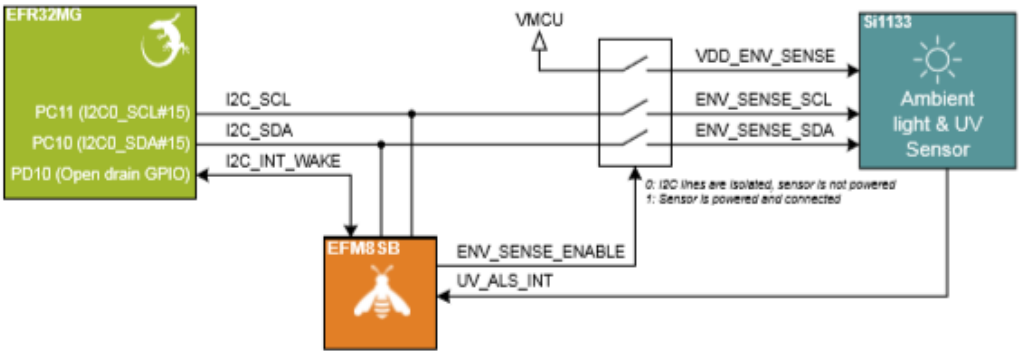

Figura 27. Diagrama de bloques del Si1133

4.2.4.3. BMP280 – Sensor de presión barométrica

El BMP280 combina un sensor de presión barométrica y un sensor de temperatura. Posee una interfaz digital que soporta tanto SPI como I^2C, además de un

ADC integrado. Permite elegir la frecuencia de oversampling lo que nos ofrece una solución de compromiso entre bajo consumo y alta resolución.

Para activarlo, al igual que con los sensores Si7021 y Si1133, escribiremos un 0x01 en el registro ENV_SENSOR_CTRL.

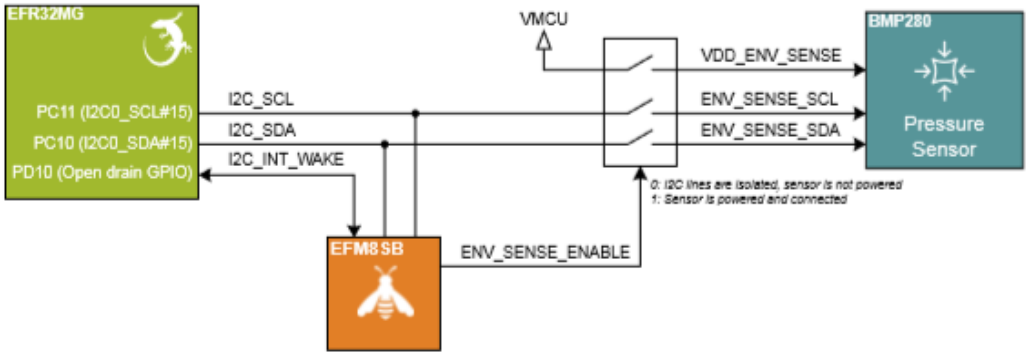

Figura 28. Diagrama de bloques del BMP280

4.2.4.4. CCS811 – Sensor de calidad del aire interior

El CCS811 incluye una interfaz I^2C, un conversor analógico-digital y un detector de gas MOX capaz de detectar gases tóxicos tales como el monóxido de carbono y una amplia variedad de compuestos orgánicos volátiles (VOCs).

Para poder utilizar el sensor, antes de activarlo, debemos despertarlo mediante el pin *WAKE*. Para ello escribiremos un 0x10 en el registro CCS_CTRL y ahora, una vez despierto, para activarlo escribiremos un 0x01 en el mismo registro CCS_CTRL.

Al igual que el sensor Si1133, el CCS811 posee un pin de interrupción para notificar el EFM8 cuando disponga de una nueva muestra para enviar.

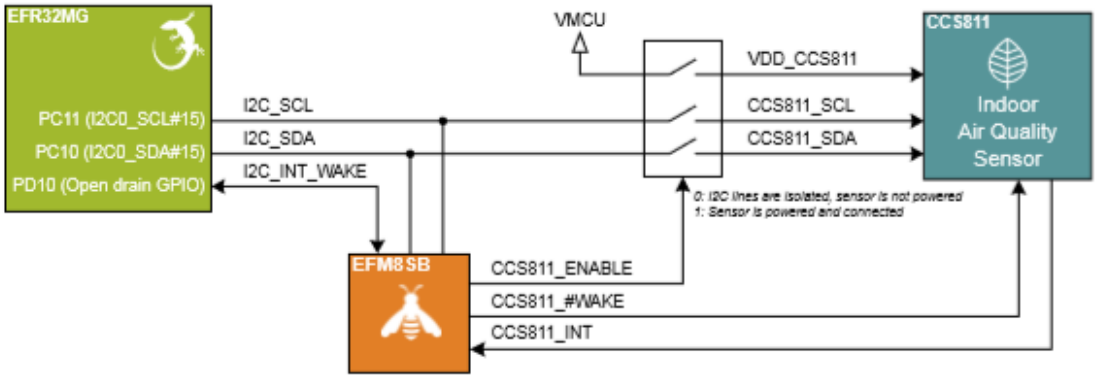

Figura 29. Diagrama de bloques del CCS811

4.2.4.5. ICM-20648 – Sensor inercial de 6 ejes

El ICM-20648 está formado por un giroscopio y un acelerómetro de 3 ejes cada uno. Soporta tanto interfaz SPI como i^2C. El sensor está colocado en el centro de la placa y la rotación sigue la regla de la mano derecha.

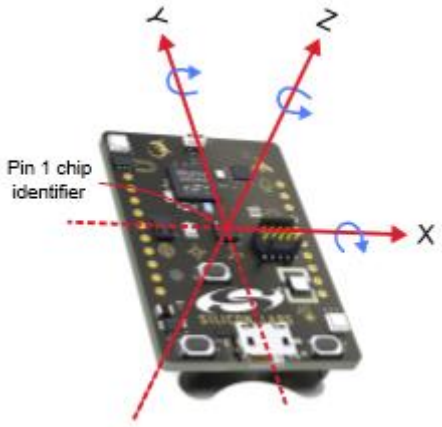

Figura 30. Distribución de los ejes del sensor ICM-20648

Para activarlo escribimos un 0x01 en el registro IMU_CTRL del EFM8 y se conectarán las líneas SPI del sensor al bus SPI principal. También tiene disponible un pin que generará una interrupción a través de la línea IMU_INT cuando disponga de una nueva muestra.

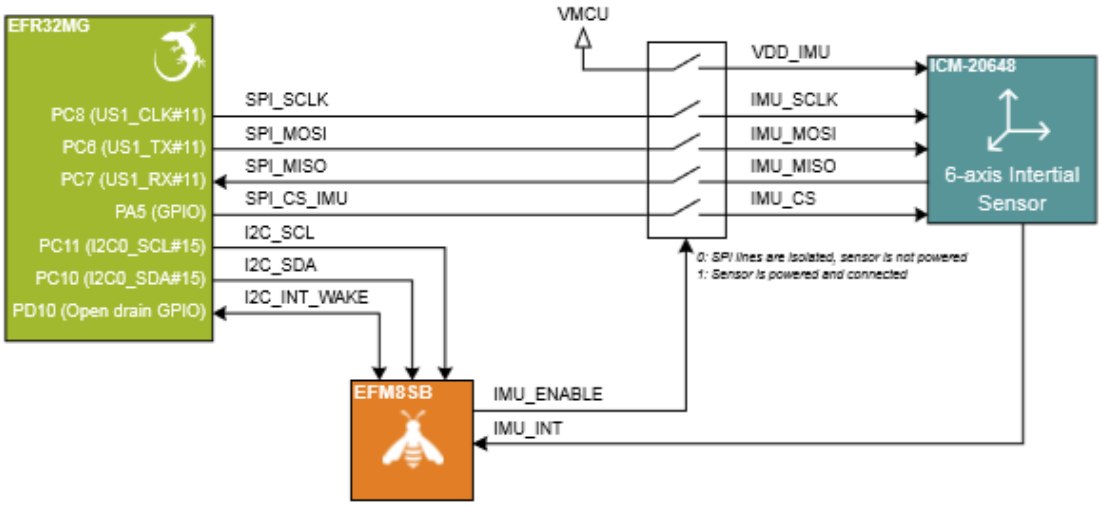

Figura 31. Diagrama de bloques del ICM-20648

4.2.4.6. SPV1840 – Micrófono MEMS

El SPV1840 es un micrófono MEMS omnidireccional con un alto rendimiento y un bajo consumo de energía de dimensiones 3.75x1.85x0.90 mm. Incluye un sensor acústico, un búfer a la entrada de bajo ruido y un amplificador de salida.

El micrófono está situado en la parte trasera de la placa, pero cuenta con un orificio de ventilación acústica a través del cual las ondas de sonido pueden llegar desde la parte delantera de la placa. A la salida del micrófono, disponemos de una etapa de procesamiento de señal con un amplificador de ganancia 32.1 dB y un filtro paso bajo de primer orden con una frecuencia de corte de 10 kHz. Para este proyecto, el filtro paso bajo se ha sustituido por un filtro paso alto de Sallen-Key.

Para activarlo debemos escribir un 0x01 en el registro MIC_CTRL del EFM8. La señal analógica del micrófono es transmitida al ADC (pin PF7) del EFR32 como se ve en la figura 32.

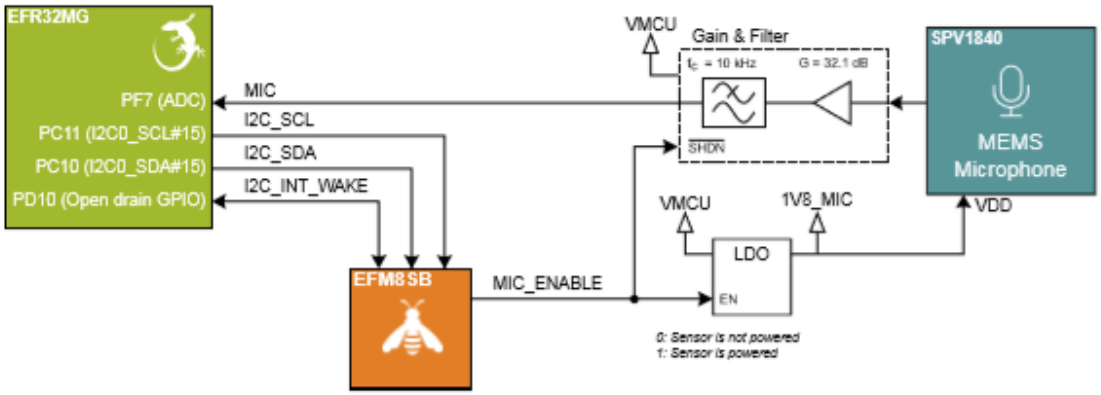

Figura 32. Diagrama de bloques del SPV1840

4.2.5. LEDs

La placa contiene un LED bicolor rojo/verde de baja energía y cuatro LEDs RGB de alta luminosidad. Todos comparten los mismos pines de entrada/salida, pero los LEDs RGB pueden ser activados o desactivados individualmente a través del EFM8.

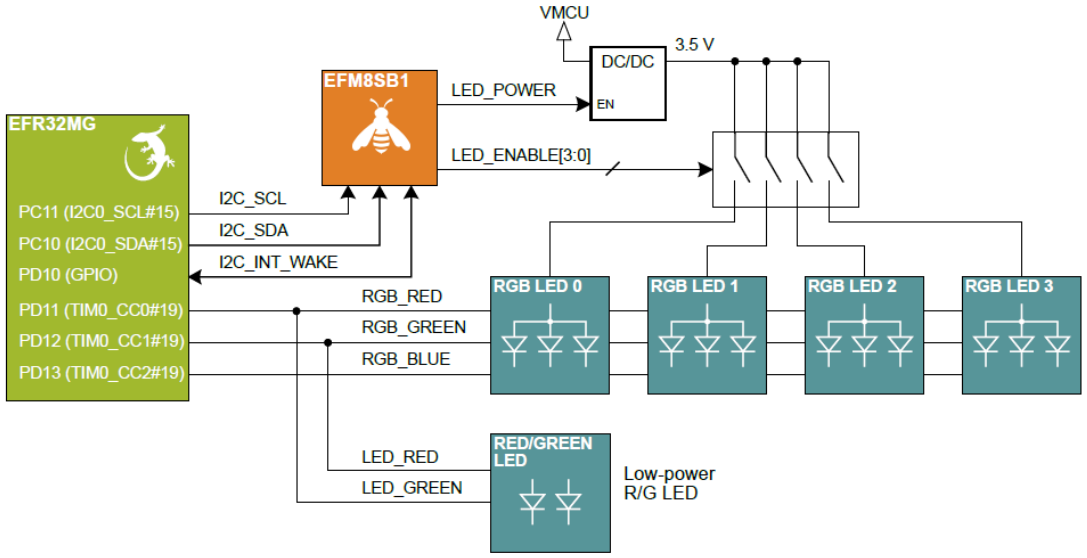

Figura 33. Diagrama de bloques de los LEDs de la placa

El LED bicolor está directamente conectado a PD11 y PD12 por lo que no puede desactivarse si estas líneas están a nivel alto. El consumo de corriente es notablemente

inferior que en el caso de los LEDs RGB. El LED rojo consume 0.8 mA a 3.3 V mientras que el verde consume 0.7 mA a 3.3 V.

Los LEDs RGB están alimentados por un convertidor Boost de 3.5 V a su salida. Tanto el convertidor Boost como cada uno de los LEDs se habilitan mediante el registro LED_CTRL del EFM8. Dado a su elevado consumo de corriente, no es posible utilizar estos LEDs en aplicaciones en las que la alimentación de la placa sea a través de una pila de botón.

Color	1x LED	2x LEDs	3x LEDs	4x LEDs
Red, 100% duty cycle	13.4 mA	26.8 mA	40.2 mA	53.6 mA
Green, 100% duty cycle	8.0 mA	16.0 mA	24.0 mA	32.0 mA
Blue, 100% duty cycle	8.5 mA	17.0 mA	25.5 mA	34.0 mA
All, 100% duty cycle (white)	29.9 mA	59.8 mA	89.7 mA	119.6 mA

Tabla 8. Consumo de corriente de los LEDs RGB para VMCU=3.3 V

4.2.6. Botones

La Thunderboard Sense incluye dos botones situados en cada una de las esquinas inferiores de la misma. Están conectados a los pines PD14 y PD15 y son activos a nivel bajo.

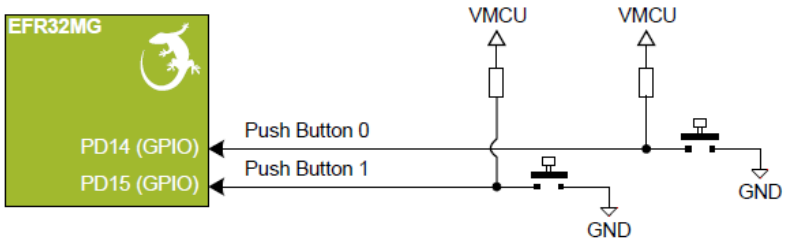

Figura 34. Diagrama de bloques de los botones de la placa

4.2.7. Memoria MX25R

La placa está equipada con una memoria Flash SPI de 8 Mbit conectada directamente con el EFR32. Comparte el bus SPI con el sensor inercial ICM-20648. La serie MX25R son dispositivos flash de bajo consumo.

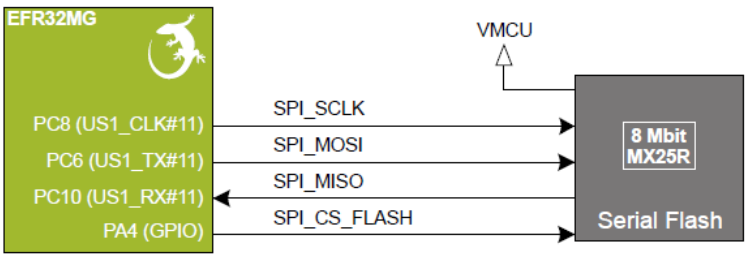

Figura 35. Diagrama de bloques de la MX25R

4.2.8. Alimentación

Existen diferentes opciones a la hora de alimentar la Thunderboard Sense como son mediante batería, conexión a través del puerto USB micro-B o a través del conector Mini Simplicity.

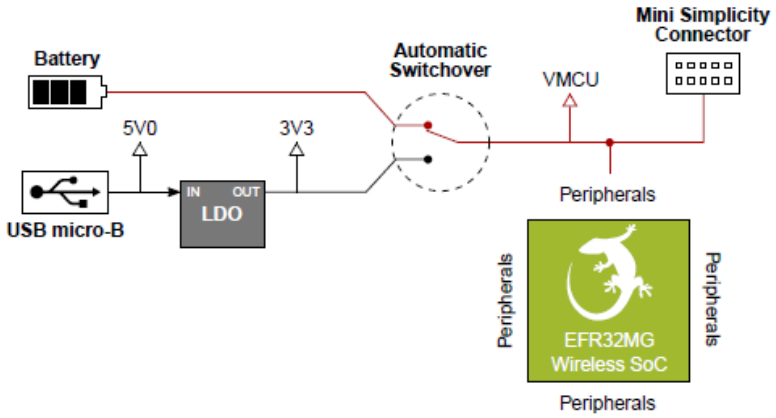

Figura 36. Esquema de alimentación de la Thunderboard Sense

Si alimentamos la placa por medio de USB, los 5 voltios que obtenemos se regulan a 3.3 voltios utilizando un regulador de bajo diferencial (LDO). En el caso de alimentación por batería, esta puede realizarse utilizando una pila de botón CR2032 o el conector de batería externa.

Como se ve en la figura 36, existe un circuito automático que alterna entre alimentación por batería y alimentación por medio del USB para evitar que pueda cargarse la batería si estamos utilizando los dos tipos de conexiones al mismo tiempo. En el momento en que se conecte el USB, este circuito automático cambiará su posición y la alimentación se producirá a través del puerto USB.

Sin embargo, no existe un circuito para el caso de la conexión a través del conector Mini Simplicity por lo que al utilizar esta opción, se requiere que ningún otro tipo de alimentación esté activa.

Supply mode	VIN	VMCU	3V3	5V0
USB power	4.5 - 5.5 V	On-board regulator	On-board regulator	USB VBUS
CR2032 battery	2.0 - 3.8 V	Battery voltage	Turned off and isolated	No voltage present
External battery	2.0 - 3.8 V	Battery voltage	Turned off and isolated	No voltage present
Mini Simplicity	2.0 - 3.8 V	Debugger dependent	Turned off and isolated	No voltage present

Tabla 9. Características de las opciones de alimentación de la Thunderboard Sense

Para nuestra aplicación lo ideal será utilizar una conexión que no requiera estar conectado a un PC u otras fuentes de alimentación no portátiles y nos permita movernos libremente por la sala. Esto podemos conseguirlo utilizando la alimentación por batería ya sea a través de una pila de botón o del conector de batería externa, o bien podemos usar un power bank conectado a la placa a través del puerto USB.

4.3. Orientación

En este capítulo, se va a estudiar cómo determinar la orientación de un objeto y se explicará en detalle los elementos de la Thunderboard Sense utilizados para ello.

En geometría, la orientación de un objeto se refiere a como está colocado dicho objeto en el espacio que ocupa sin cambiar un punto fijo de referencia. Normalmente, la orientación se da en relación con un sistema de referencia , generalmente especificado por un sistema de coordenadas cartesiano.

4.3.1. Representaciones matemáticas de la orientación

Existen varias representaciones matemáticas para definir la orientación del objeto respecto al sistema de referencia definidas a continuación.

4.3.1.1. Ángulos de Euler

Los ángulos de Euler constituyen un conjunto de tres coordenadas angulares que especifican la orientación de un sistema de referencia de ejes ortogonales móvil (X, Y, Z), respecto a un sistema de referencia de ejes ortogonales fijo (x, y, z). Se denotan típicamente como α, β, γ o φ, θ, Ψ.

Para llevar el sistema de referencia móvil a su posición final se llevan a cabo tres rotaciones y pueden realizarse sobre el sistema fijo (rotación extrínseca) o sobre el propio sistema móvil (rotación intrínseca). Los ángulos de Euler representan estas rotaciones.

Se tiene libertad a la hora de elegir los ejes sobre los que se realizarán las rotaciones y en qué orden, pero cuando hablamos de ángulos de Euler se debe cumplir que el eje escogido para la primera rotación debe ser el mismo que para la última.

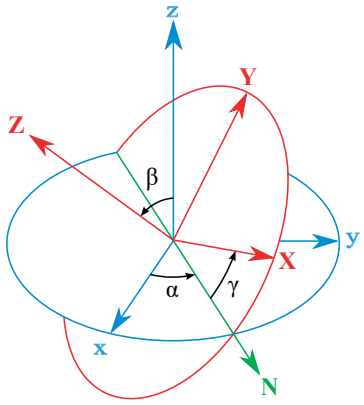

Figura 37. Representación de los ángulos de Euler

La elección más habitual de rotaciones suele ser la secuencia Z-X-Z y de manera intrínseca. De esta forma, partiendo de ambos sistemas de referencia iguales, en la misma posición, se realiza una rotación del sistema XYZ respecto al eje Z en α, a continuación, se rota respecto al eje X en β y, por último, se rota de nuevo respecto a Z en γ.

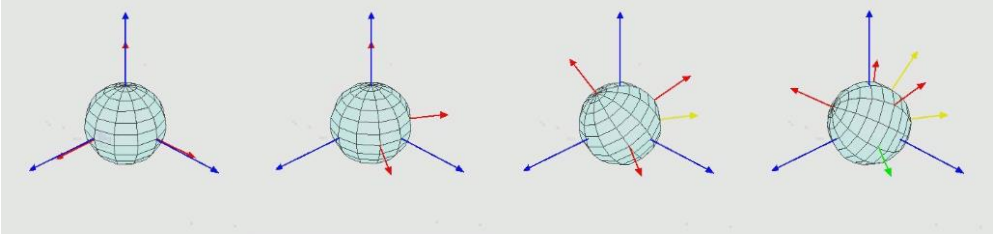

Figura 38. Rotación intrínseca de un objeto siguiendo la secuencia Z-X-Z

4.3.1.2. Ángulos de navegación

Los ángulos de navegación o ángulos de Tait-Bryan son una representación similar a los ángulos de Euler salvo que a la hora de elegir los ejes sobre los que se realizarán las tres rotaciones, no se debe repetir ningún eje como ocurre con los ángulos de Euler. Además, tienen la ventaja de ser conmutativos, es decir, la orientación final obtenida es independiente de orden en el que apliquemos la rotación. Sin embargo, tienen una desventaja denominada bloqueo cardán (gimbal lock) que sufren todos los sistemas de ángulos de Euler. Este bloqueo ocurre cuando dos de los ejes coinciden por lo que el sistema pierde una magnitud.

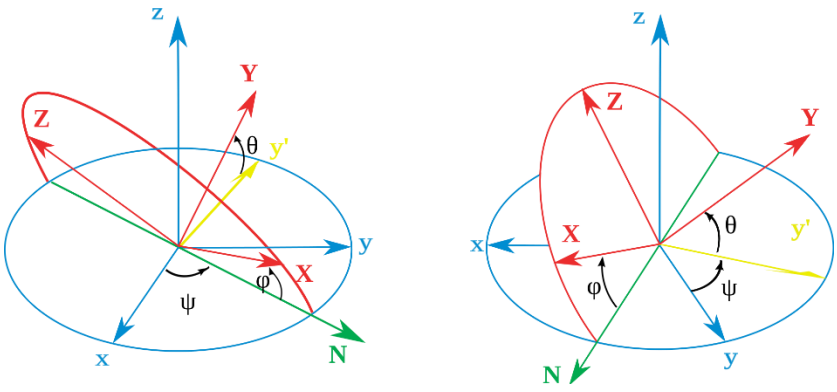

Figura 39. Ángulos de navegación según convención ZXY

Los ángulos de navegación se denominan Roll, Pitch y Yaw y se utilizan sobre todo en aplicaciones aeroespaciales, así como para describir la orientación en los aviones.

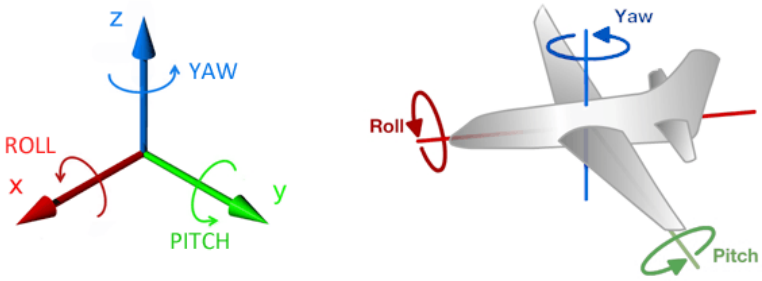

Figura 40. Representación de los ángulos Roll, Pitch y Yaw

4.3.1.3. Matriz de rotación

La matriz de rotación o Direction Cosine Matrix (DCM) permite indicar la orientación de un objeto O con respecto a un sistema de referencia M mediante las coordenadas en el sistema de referencia de los vectores unitarios en la dirección de los ejes del objeto. Es una matriz 3x3 cuyos 9 elementos escalares son las coordenadas de los vectores del objeto en el sistema de referencia.

La matriz de rotación de un objeto O respecto a un sistema de referencia M es la que se ve a continuación:

$$ {}^{\mathrm{M}}\mathbf{Rot}_{\mathrm{O}} = \left[\boldsymbol{x}_{\mathrm{O}}^{\mathrm{M}} \quad \boldsymbol{y}_{\mathrm{O}}^{\mathrm{M}} \quad \boldsymbol{z}_{\mathrm{O}}^{\mathrm{M}} \right] = \begin{bmatrix} x_{x_{\mathrm{O}}^{\mathrm{M}}} & x_{y_{\mathrm{O}}^{\mathrm{M}}} & x_{z_{\mathrm{O}}^{\mathrm{M}}} \\ y_{x_{\mathrm{O}}^{\mathrm{M}}} & y_{y_{\mathrm{O}}^{\mathrm{M}}} & y_{z_{\mathrm{O}}^{\mathrm{M}}} \\ z_{x_{\mathrm{O}}^{\mathrm{M}}} & z_{y_{\mathrm{O}}^{\mathrm{M}}} & z_{z_{\mathrm{O}}^{\mathrm{M}}} \end{bmatrix} $$

4.3.1.4. Cuaterniones

Los cuaterniones proporcionan una notación matemática para representar las orientaciones y las rotaciones de objetos en tres dimensiones. Son una extensión de los números complejos introducida por Hamilton en 1843, que emplean tres unidades imaginarias i, j, k.

$$ i^2 = j^2 = k^2 = i \cdot j \cdot k = -1 $$

Al igual que los números complejos pueden usarse para representar vectores en el plano, un cuaternión permite expresar un vector en el espacio tridimensional. Para expresar la rotación de un punto, un ángulo α, en torno a un vector arbitrario de coordenadas E_x, E_y, E_z el cuaternión resultante adopta la siguiente expresión:

$$q = \cos(\frac{\alpha}{2}) + i \cdot (E_x \cdot \sin(\frac{\alpha}{2})) + j \cdot (E_y \cdot \sin(\frac{\alpha}{2})) + k \cdot (E_z \cdot \sin(\frac{\alpha}{2}))$$

4.3.2. Unidad de Medición Inercial

Una unidad de medición inercial (IMU) es un dispositivo electrónico que mide e informa acerca de la velocidad, orientación y fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros y giróscopos. Las IMU son normalmente usadas para maniobrar aviones y naves espaciales, incluyendo satélites, aterrizadores y transbordadores.

Al hablar de IMUs es frecuente hacer referencia a la cantidad de grados de libertad (DOF) que dispone. Los DOF de un sensor representan la cantidad de magnitudes independientes que es capaz de medir. Así, un acelerómetro de 3 ejes ortogonales es un sensor de 3 DOF, y un giroscopio que mida ángulos en 3 ejes ortogonales es un sensor de 3 DOF.

La mayoría de los IMUs parten de la combinación de un acelerómetro y un giroscopio debido a que combinan muy bien y compensan las limitaciones del otro. Los IMUs más frecuentes son:

- IMU de 6 DOF, combinando un acelerómetro de 3 ejes y un giroscopio de 3 ejes.
- IMU de 9 DOF, añadiendo un magnetómetro de 3 ejes.
- IMU de 10 DOF, que añade un barómetro para la estimación de la altura del sensor.

Siendo el más habitual el IMU de 6 DOF, como es el caso del sensor ICM-20648 utilizado en este proyecto.

4.3.2.1. Acelerómetro

Un acelerómetro es un dispositivo que permite medir la aceleración a la que está sometido. La aceleración es la variación de la velocidad respecto del tiempo, expresado matemáticamente:

$$\vec{a} = \frac{\partial \vec{v}}{\partial t} = \frac{\partial^2 \vec{r}}{\partial t^2}$$

Y que por la primera ley de Newton:

$$\vec{F} = m \cdot \vec{a}$$

Es decir, que cualquier cuerpo con una masa m requiere una cierta fuerza para variar su velocidad. Equivalentemente, cualquier cuerpo sometido a una aceleración experimentará una cierta fuerza.

El funcionamiento depende de su tipo, ya que las composiciones son distintas, pero siempre cumplen una función principal que es medir el desplazamiento de placas, resortes, etc. En el caso de un acelerómetro capacitivo estos están compuestos por capas capacitivas internas, ya sean fijas o que contengan unos pequeños resortes. Al aplicar una fuerza de aceleración sobre el sensor, las placas se mueven una a otra ocasionando que la capacitancia entre ellas cambie. Midiendo este cambio se obtiene una señal a partir de la cual, al digitalizarla utilizando un ADC, podemos determinar la aceleración.

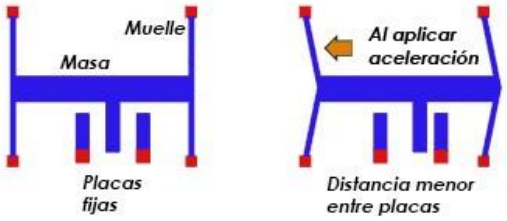

Figura 41. Funcionamiento de un acelerómetro

Los acelerómetros disponibles normalmente son de 3 ejes, es decir, son capaces de medir la aceleración a la que está sometido el sensor en X, Y y Z independientemente, lo que permite saber simultáneamente la magnitud y dirección de la aceleración medida.

La capacidad de medir la aceleración de un sistema proporciona en si misma funcionalidades interesantes, como registrar vibraciones o golpes, pero no es la única función que podemos obtener de un acelerómetro. El sensor se ve afectado por la gravedad terrestre, que supone una aceleración de aproximadamente $9.81\ m/s^2$ en la superficie de la tierra, que por supuesto es registrada constantemente por el sensor.

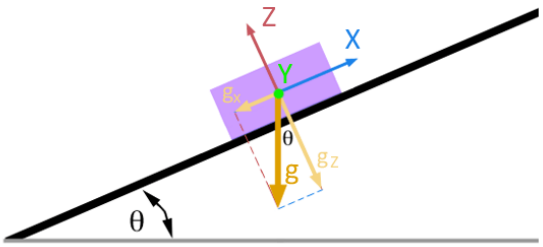

Figura 42. Representación del vector de gravedad *g* en un sistema 2D

Dado que podemos registrar la aceleración en tres ejes y en ausencia de otras aceleraciones, podremos a partir de la medición de la gravedad, determinar la orientación del sensor calculando los ángulos θ mediante ecuaciones trigonométricas.

Los acelerómetros no tienen deriva (drift) a medio o largo plazo, ya que realizan la medición medida absoluta del ángulo que forma el sensor con la dirección vertical, marcada por la gravedad. Sin embargo, se ven influenciados por los movimientos del sensor y el ruido por lo que no son fiables a corto plazo.

4.3.2.2. Giroscopio

Un giroscopio o giróscopo es un dispositivo que permite medir el ángulo de rotación girado por un determinado mecanismo. Son dispositivos puramente diferenciales, es decir, no existe una referencia absoluta si no que siempre medimos ángulos relativos a una referencia arbitraria.

Existen diversos tipos de giroscopios (mecánicos, de anillo láser, de fibra óptica), los que se emplean en MEMS son denominados giroscopios vibratorios de efecto Coriolis (CVG) que a diferencia de otros tipos de giroscopios no registran el ángulo girado si no la velocidad angular. La fuerza de Coriolis es una fuerza ficticia que aparece sobre un cuerpo en movimiento cuando se encuentra en un sistema en rotación.

$$\vec{F}_c = -2m\,(\vec{w} \cdot \vec{v}\,)$$

El funcionamiento del giroscopio consiste en medir la fuerza Coriolis que ejerce un objeto vibratorio sobre un soporte para determinar la rotación a la que está sometido. Ciertas partes del soporte se someten a vibración por resonancia y el efecto

de la fuerza de Coriolis deforma la estructura, lo cuál puede ser medido por la variación de la capacitancia del sistema. La señal obtenida es amplificada, demodulada y filtrada para producir un voltaje proporcional a la velocidad angular. Este voltaje se digitaliza mediante un ADC muestreando cada eje.

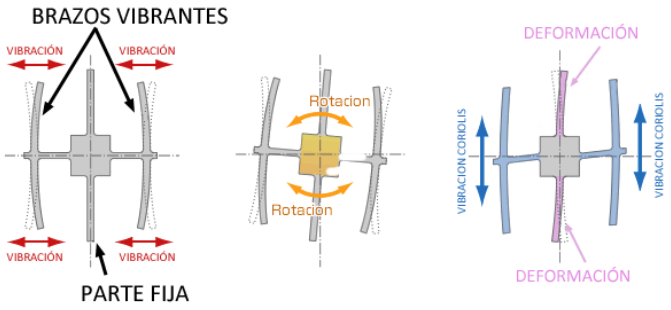

Figura 43. Funcionamiento de un giroscopio

Lo habitual es que los giroscopios que empleemos sean de 3 ejes, es decir, que registran de forma independiente la rotación en X, Y, y Z, lo cual permite determinar la magnitud y dirección de la rotación.

Los giroscopios funcionan muy bien para movimientos cortos o bruscos, pero al usar giroscopios de vibración que realmente miden la velocidad angular, y obtienen el ángulo por integración respecto al tiempo, acumulan los errores y el ruido en la medición, por lo que a medio o largo plazo tienen deriva.

4.3.3. Filtrado de la señal

La mayoría de los IMUs incorporan un acelerómetro y un giroscopio debido a que cada uno suple las carencias del otro. El giroscopio funciona muy bien a corto plazo mientras que el acelerómetro funciona mejor a largo plazo. Por tanto, combinar valores obtenidos de ambos dispositivos nos permite obtener mediciones de la orientación muy precisas.

Para determinar la orientación de un objeto, necesitamos conocer tres parámetros. En este proyecto vamos a utilizar los ángulos de navegación, representados como tres rotaciones en torno a los ejes X (Roll), Y (Pitch) y Z (Yaw). Para hallar estos ángulos, combinaremos los valores resultantes de las medidas del acelerómetro y del giroscopio mediante el Filtro Complementario.

Si tenemos en cuenta que la única fuerza que actúa sobre el sensor es la fuerza de la gravedad ($9.8\ \mathrm{m/s^2}$), los valores que obtenemos en las componentes del

acelerómetro corresponden a la gravedad y los ángulos de la resultante serán la inclinación del plano del sensor, puesto que la gravedad siempre es vertical.

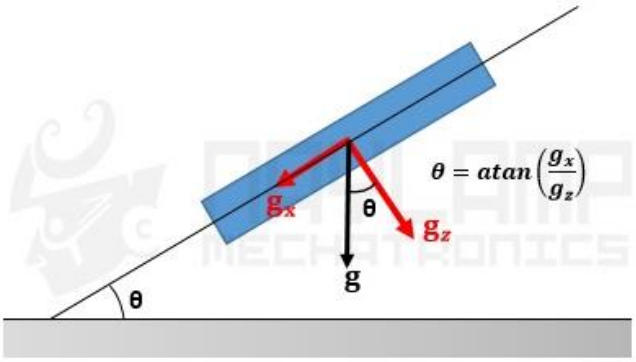

Figura 44. Cálculo del ángulo θ a partir del vector *g*

Lo que vemos en la figura 44 nos sirve para calcular el ángulo en un plano 2D, pero para calcular los ángulos de inclinación en un espacio 3D tanto en X como en Y usamos las siguientes ecuaciones trigonométricas:

$$\theta_{acelerometro_x} = \tan^{-1}\frac{y}{\sqrt{x^2 + z^2}}$$

$$\theta_{acelerometro_y} = \tan^{-1}\frac{x}{\sqrt{z^2 + z^2}}$$

Pero dado que calculamos estos valores a partir de la aceleración de la gravedad, no es posible aplicar la misma fórmula en el caso del eje Z. Para hallar su valor necesitaríamos un magnetómetro digital, aunque no va a ser necesario conocer dicho valor a la hora de aplicar el filtro complementario.

El giroscopio mide la velocidad angular (ω), es decir, el ángulo girado en una unidad de tiempo.

$$\omega = \frac{\partial\theta}{\partial t}$$

Sabiendo esto, para hallar el valor actual del ángulo en un eje nos basta con conocer el valor anterior del propio ángulo medido por el giroscopio en función del tiempo transcurrido. Aplicaremos la fórmula a cada uno de los tres ejes X, Y, Z.

$$\theta_{giroscopio} = \theta_{anterior} + \omega * \Delta t$$

Una vez obtenidos los valores de los ángulos a partir del acelerómetro y el giroscopio, aplicaremos el Filtro Complementario. Este filtro es una combinación de un filtro paso bajo y un filtro paso alto y es ideal para nuestra aplicación debido a su bajo coste, fácil utilización y buena precisión. El filtro paso alto lo aplicaremos a las medidas del giroscopio mientras que el filtro paso bajo se aplicará a las medidas del acelerómetro. De esta forma conseguimos que la señal del giroscopio mande a corto plazo y la señal del acelerómetro mande a largo plazo que es lo que buscábamos al combinar ambos dispositivos con el objetivo de compensar las limitaciones de cada uno.

La ecuación del Filtro Complementario la aplicaremos para calcular los valores de Roll y Pitch. Dado que no podemos obtener el valor de la aceleración en el eje Z sin un magnetómetro, el valor de Yaw lo obtendremos directamente del valor obtenido del giroscopio en el eje Z.

$$Roll = A * \theta_{giroscopio_x} + B * \theta_{acelerometro_x}$$

$$Pitch = A * \theta_{giroscopio_y} + B * \theta_{acelerometro_y}$$

$$Yaw = \theta_{giroscopio_z}$$

Donde los valores de A y B pueden ser cualesquiera siempre que la suma de ambos sea 1. En nuestro caso vamos a utilizar los valores $A = 0.98$ y $B = 0.02$.

Una vez hemos explicado el procedimiento a seguir para obtener los ángulos de navegación, vamos a ver cómo aplicarlo a nuestro proyecto y como obtener dichos ángulos utilizando el sensor IMU ICM-20648 que incorpora la Thunderboard Sense.

4.3.4. Sensor IMU ICM-20648

El ICM-20648 es un sensor inercial fabricado por TDK InvenSense de 6 DOF formado por un giroscopio MEMS de 3 ejes y un acelerómetro MEMS de 3 ejes. Sus dimensiones son 3x3x0.9 mm y dispone de 24 pines.

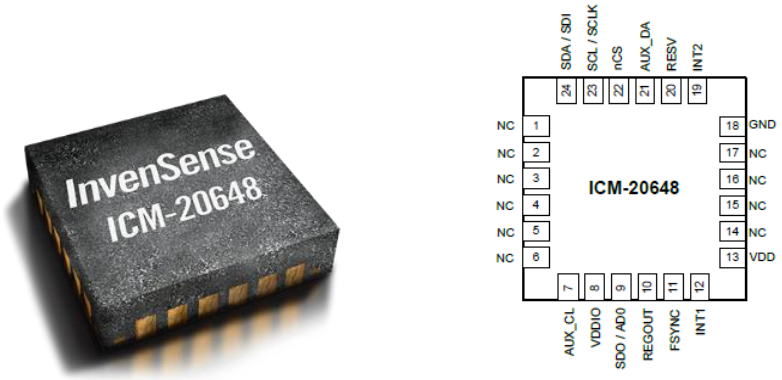

Figura 45. Sensor ICM-20648

PIN NUMBER	PIN NAME	PIN DESCRIPTION
7	AUX_CL	I²C Master serial clock, for connecting to external sensors.
8	VDDIO	Digital I/O supply voltage.
9	AD0 / SDO	I²C Slave Address LSB (AD0); SPI serial data output (SDO).
10	REGOUT	Regulator filter capacitor connection.
11	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	INT1	Interrupt 1.
13	VDD	Power supply voltage.
18	GND	Power supply ground.
19	INT2	Interrupt 2.
20	RESV	Reserved. Connect to GND.
21	AUX_DA	I²C master serial data, for connecting to external sensors.
22	nCS	Chip select (SPI mode only).
23	SCL / SCLK	I²C serial clock (SCL); SPI serial clock (SCLK).
24	SDA / SDI	I²C serial data (SDA); SPI serial data input (SDI).
1 – 6, 14 - 17	NC	No Connect pins. Do not connect.

Tabla 10. Descripción de los pines del ICM-20648

Las principales características que nos indica el fabricante son las siguientes:

- Tensión de alimentación de 1.71 V a 3.6 V
- Corriente típica con giroscopio habilitado de 1.23 mA
- Corriente típica con acelerómetro habilitado de 68.9 µA
- Corriente típica con giroscopio y acelerómetro habilitados de 1.27 mA
- Interfaz I2C de hasta 100 kHz en modo estándar o hasta 400 kHz en modo rápido
- Interfaz SPI de hasta 7 MHz
- Relojes
    - Oscilador de relajación interno de 20 MHz
    - Lazo de seguimiento de fase (PLL) con el oscilador del giroscopio MEMS
- ADC de 16 bits
- Temperatura operativa de -40 ˚C a 85 ˚C

Como se ha explicado en anteriormente en este documento, el sensor está controlado por el EFM8 a través de Controlador de Alimentación e Interrupción y está deshabilitado por defecto. Una vez lo habilitemos, se activará la línea IMU_ENABLE del EFM8 y se conectarán las líneas del bus SPI del sensor ICM-20648 con las líneas del bus SPI del EFR32 como se ve en la figura 46.

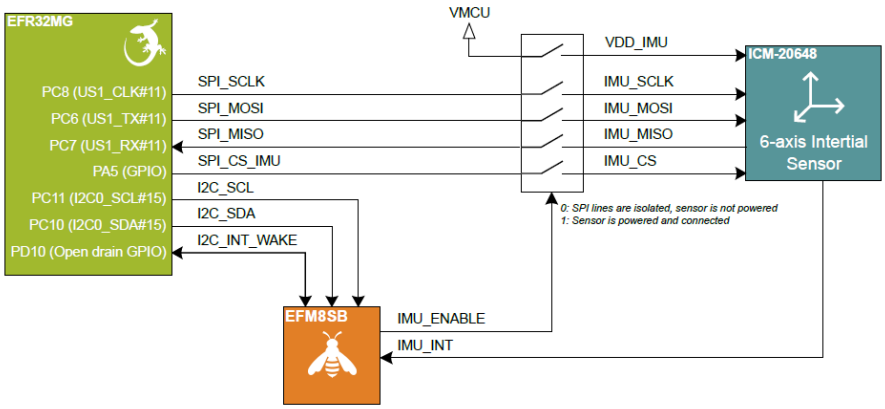

Figura 46. Diagrama de bloques del ICM-20648

A continuación, vamos a configurar la comunicación SPI entre sensor y EFR32. En la figura 46 podemos ver los pines utilizados en la interfaz SPI por parte del EFR32 y los configuraremos mediante la función *GPIO_PinModeSet()* de la siguiente manera:

- Pin PC8: como salida en modo push-pull
- Pin PC6: como salida en modo push-pull
- Pin PC7: como entrada
- Pin PA5: como salida en modo push-pull

Una vez establecida la comunicación SPI, vamos a proceder a la configuración del sensor. Los registros del ICM-20648 se agrupan en 4 bancos distintos y para comunicarnos con ellos a través del bus SPI se utilizarán las siguientes funciones:

- *ICM20648_chipSelect()*: esta función controla el estado del pin CS que habilita o deshabilita la comunicación entre maestro y esclavo. Recibe como argumento el estado deseado del pin como variable booleana.

```
static void ICM20648_chipSelectSet( bool select )
{
   if( select )
      GPIO_PinOutClear( ICM20648_PORT_SPI_CS, ICM20648_PIN_SPI_CS );

   else
      GPIO_PinOutSet( ICM20648_PORT_SPI_CS, ICM20648_PIN_SPI_CS );

   return;
}
```

Código 8. Función *ICM20648_chipSelectSet()*

- *ICM20648_bankSelect()*: función para seleccionar un banco del ICM-20648. Recibe como argumento la dirección del banco (0 a 3)

```
void ICM20648_bankSelect( uint8_t bank )
{
   /* Enable chip select */
   ICM20648_chipSelectSet( true );

   /* Select the Bank Select register */
   USART_Tx( ICM20648_SPI_USART, ICM20648_REG_BANK_SEL );
   USART_Rx( ICM20648_SPI_USART );

   /* Write the desired bank address 0..3 */
   USART_Tx( ICM20648_SPI_USART, ( bank << 4 ) );
   USART_Rx( ICM20648_SPI_USART );

   /* Disable chip select */
   ICM20648_chipSelectSet( false );

   return
}
```

Código 9. Función *ICM20648_bankSelect()*

- *ICM20648_registerRead()*: lee un registro del ICM-20648. Recibe como argumentos la dirección del registro, el número de bytes a leer y una variable *data* donde se almacenarán los datos leídos.

```
void ICM20648_registerRead( uint16_t addr, int numBytes, uint8_t *data )
{
   uint8_t regAddr;
   uint8_t bank;

   regAddr = (uint8_t) ( addr & 0x7F );
   bank = (uint8_t) ( addr >> 7 );

   ICM20648_bankSelect( bank );

   /* Enable chip select */
   ICM20648_chipSelectSet( true );

   /* Set R/W bit to 1 - read */
   USART_Tx( ICM20648_SPI_USART, ( regAddr | 0x80 ) );
   USART_Rx( ICM20648_SPI_USART );

   /* Transmit 0's to provide clock and read the data */
   while( numBytes-- ) {
      USART_Tx( ICM20648_SPI_USART, 0x00 );
      *data++ = USART_Rx( ICM20648_SPI_USART );
   }

   /* Disable chip select */
   ICM20648_chipSelectSet( false );

   return;
}
```

Código 10. Función *ICM20648_registerRead()*

- *ICM20648_registerWrite()*: escribe en un registro del ICM-20648. Recibe como argumentos la dirección del registro y los datos a escribir.

```
void ICM20648_registerWrite( uint16_t addr, uint8_t data )
{
   uint8_t regAddr;
   uint8_t bank;

   regAddr = (uint8_t) ( addr & 0x7F );
   bank = (uint8_t) ( addr >> 7 );

   ICM20648_bankSelect( bank );

   /* Enable chip select */
   ICM20648_chipSelectSet( true );

   /* clear R/W bit - write, send the address */
   USART_Tx( ICM20648_SPI_USART, ( regAddr & 0x7F ) );
   USART_Rx( ICM20648_SPI_USART );

   /* Send the data */
   USART_Tx( ICM20648_SPI_USART, data );
   USART_Rx( ICM20648_SPI_USART );

   /* Disable chip select */
   ICM20648_chipSelectSet( false );

   return;
}
```

Código 11. Función *ICM20648_registerWrite()*

A continuación, se explica la configuración de los principales registros utilizados:

- El fabricante nos recomienda deshabilitar la interfaz I^2C cuando utilicemos la interfaz SPI. Para ello fijaremos a 1 el bit 4 (I2C_IF_DIS) del registro USER_CTRL.

Name: USER_CTRL
Address: 3 (03h)
Type: USR0
Bank: 0
Serial IF: R/W
Reset Value: 0x00

BIT	NAME	FUNCTION
7	DMP_EN	1 – Enables DMP features. 0 – DMP features are disabled after the current processing round has completed.
6	FIFO_EN	1 – Enable FIFO operation mode. 0 – Disable FIFO access from serial interface. To disable FIFO writes by DMA, use FIFO_EN register. To disable possible FIFO writes from DMP, disable the DMP.
5	I2C_MST_EN	1 – Enable the I²C Master I/F module; pins ES_DA and ES_SCL are isolated from pins SDA/SDI and SCL/ SCLK. 0 – Disable I²C Master I/F module; pins ES_DA and ES_SCL are logically driven by pins SDA/SDI and SCL/ SCLK.
4	I2C_IF_DIS	1 – Reset I²C Slave module and put the serial interface in SPI mode only.
3	DMP_RST	1 – Reset DMP module. Reset is asynchronous. This bit auto clears after one clock cycle of the internal 20 MHz clock.
2	SRAM_RST	1 – Reset SRAM module. Reset is asynchronous. This bit auto clears after one clock cycle of the internal 20 MHz clock.
1	I2C_MST_RST	1 – Reset I²C Master module. Reset is asynchronous. This bit auto clears after one clock cycle of the internal 20 MHz clock. **Note:** This bit should only be set when the I²C master has hung. If this bit is set during an active I²C master transaction, the I²C slave will hang, which will require the host to reset the slave.
0	-	Reserved

Figura 47. Registro USER_CTRL del ICM-20648

- Además, para obtener los valores especificados en el datasheet, nos recomiendan que fijemos los bits CLKSEL[2:0] del registro PWR_MGMT_1 a 1. Haciendo esto, dejamos la elección de la fuente de reloj al sensor, priorizando el PLL si es posible.

Name: PWR_MGMT_1
Address: 6 (06h)
Type: USR0
Bank: 0
Serial IF: R/W
Reset Value: 0x41

BIT	NAME	FUNCTION
7	DEVICE_RESET	1 – Reset the internal registers and restores the default settings. Write a 1 to set the reset, the bit will auto clear.
6	SLEEP	When set, the chip is set to sleep mode (in sleep mode all analog is powered off). Clearing the bit wakes the chip from sleep mode.
5	LP_EN	The LP_EN only affects the digital circuitry, it helps to reduce the digital current when sensors are in LP mode. Please note that the sensors themselves are set in LP mode by the LP_CONFIG register settings. Sensors in LP mode, and use of LP_EN bit together help to reduce overall current. The bit settings are: 1: Turn on low power feature 0: Turn off low power feature LP_EN has no effect when the sensors are in low-noise mode.
4	-	Reserved.
3	TEMP_DIS	When set to 1, this bit disables the temperature sensor.
2:0	CLKSEL[2:0]	**Code: Clock Source** 0: Internal 20 MHz oscillator 1-5: Auto selects the best available clock source – PLL if ready, else use the Internal oscillator 6: Internal 20 MHz oscillator 7: Stops the clock and keeps timing generator in reset **Note:** CLKSEL[2:0] should be set to 1~5 to achieve full gyroscope performance.

Figura 48. Registro PWR_MGMT_1 del ICM-20648

- El pin utilizado para avisar de una nueva muestra es el INT1 (pin 12) que se conecta el EFM8 a través de la línea IMU_INT (figura 46). Se ha configurado en modo drenador abierto y activo a nivel bajo fijando a 1 los bits 6 (INT1_OPEN) y 7 (INT1_ACTL) del registro INT_PIN_CFG. El bit 5 (INT1_LATCH_EN) de este mismo registro se ha fijado a 0 para que cuando se genere la interrupción asociada al pin INT1, se envíe un pulso de 50 µs a través de este.

Name: INT_PIN_CFG
Address: 15 (0Fh)
Type: USR0
Bank: 0
Serial IF: R/W
Reset Value: 0x00

BIT	NAME	FUNCTION
7	INT1_ACTL	1 – The logic level for INT1 pin is active low. 0 – The logic level for INT1 pin is active high.
6	INT1_OPEN	1 – INT1 pin is configured as open drain. 0 – INT1 pin is configured as push-pull.
5	INT1_LATCH__EN	1 – INT1 pin level held until interrupt status is cleared. 0 – INT1 pin indicates interrupt pulse is width 50 µs.
4	INT_ANYRD_2CLEAR	1 – Interrupt status in INT_STATUS is cleared (set to 0) if any read operation is performed. 0 – Interrupt status in INT_STATUS is cleared (set to 0) only by reading INT_STATUS register. This bit only affects the interrupt status bits that are contained in the register INT_STATUS, and the corresponding hardware interrupt. This bit does not affect the interrupt status bits that are contained in registers INT_STATUS_1, INT_STATUS_2, INT_STATUS_3, and the corresponding hardware interrupt.
3	ACTL_FSYNC	1 – The logic level for the FSYNC pin as an interrupt to the ICM-20648 is active low. 0 – The logic level for the FSYNC pin as an interrupt to the ICM-20648 is active high.
2	FSYNC_INT_MODE_EN	1 – This enables the FSYNC pin to be used as an interrupt. A transition to the active level described by the ACTL_FSYNC bit will cause an interrupt. The status of the interrupt is read in the I²C Master Status register PASS_THROUGH bit. 0 – This disables the FSYNC pin from causing an interrupt.
1	BYPASS_EN	When asserted, the I2C_MASTER interface pins (ES_CL and ES_DA) will go into 'bypass mode' when the I2C master interface is disabled.
0	-	Reserved

Figura 49. Registro INT_PIN_CFG del ICM-20648

- La interrupción asociada al pin INT1 se producirá cuando el sensor disponga de nuevos datos. Para ello fijaremos a 1 el bit 0 (RAW_DATA_0_RDY_EN) del registro INT_ENABLE_1

Name: INT_ENABLE_1
Address: 17 (11h)
Type: USR0
Bank: 0
Serial IF: R/W
Reset Value: 0x00

BIT	NAME	FUNCTION
7	INT2_ACTL	1 – The logic level for INT2 pin is active low. 0 – The logic level for INT2 pin is active high.
6	INT2_OPEN	1 – INT2 pin is configured as open drain. 0 – INT2 pin is configured as push-pull.
5	INT2_LATCH_EN	1 – INT2 pin level held until interrupt status is cleared. 0 – INT2 pin indicates interrupt pulse is width 50 µs.
4:1	-	Reserved
0	RAW_DATA_0_RDY_EN	1 – Enable raw data ready interrupt from any sensor to propagate to interrupt pin 1. 0 – Function is disabled.

Figura 50. Registro INT_ENABLE_1 del ICM-20648

- Mediante los bits GYRO_FS_SEL[1:0] y GYRO_FS_SEL[1:0] de los registros GYRO_CONFIG_1 y ACCEL_CONFIG podemos seleccionar la sensibilidad del giroscopio y acelerómetro respectivamente entre 4 valores. Para este proyecto se han elegido los valores de ±250dps para el giroscopio y ±2g para el acelerómetro

Name: GYRO_CONFIG_1
Address: 1 (01h)
Type: USR2
Bank: 2
Serial IF: R/W
Reset Value: 0x01

BIT	NAME	FUNCTION
7:6	-	Reserved
5:3	GYRO_DLPFCFG[2:0]	Gyro low pass filter configuration as shown in the table below
2:1	GYRO_FS_SEL[1:0]	Gyro Full Scale Select: 00 = ±250 dps 01= ±500 dps 10 = ±1000 dps 11 = ±2000 dps
0	GYRO_FCHOICE	0 – bypass gyro DLPF 1 – enable gyro DLPF

Figura 51. Registro GYRO_CONFIG_1 del ICM-20648

Name: ACCEL_CONFIG
Address: 20 (14h)
Type: USR2
Bank: 2
Serial IF: R/W
Reset Value: 0x01

BIT	NAME	FUNCTION
7:6	-	Reserved
5:3	ACCEL_DLPFCFG[2:0]	Accelerometer low pass filter configuration as shown in the table below
2:1	ACCEL_FS_SEL[1:0]	Accelerometer Full Scale Select: 00: ±2*g* 01: ±4*g* 10: ±8*g* 11: ±16*g*
0	ACCEL_FCHOICE	0: bypass accel DLPF 1: enable accel DLPF

Figura 52. Registro ACEL_CONFIG del ICM-20648

Una vez configurados los registros, vamos a explicar cómo leer los datos medidos.

El ADC del sensor ICM digitaliza las señales obtenidas por el giroscopio y el acelerómetro muestreando cada eje por separado. Por tanto, a la salida de dicho ADC tendremos 2 bytes por cada eje de acelerómetro y giroscopio. Estos bytes se guardan en registros diferentes según el eje y el byte. Así, el registro GYRO_XOUT_H almacena el byte más significativo del eje X del giroscopio y el registro GYRO_XOUT_L el byte menos significativo. Para el caso de los ejes Y, Z serán los registros GYRO_YOUT_H, GYRO_YOUT_L, GYRO_ZOUT_H y GYRO_ZOUT_L. En cuanto al acelerómetro ocurre de forma similar con los registros ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_H y ACCEL_ZOUT_L.

Dado que todos estos registros están almacenados en memoria en posiciones consecutivas, para obtener los datos del acelerómetro vamos a leer 6 bytes a partir de la dirección del registro ACCEL_XOUT_H mediante la función ICM20648_registerRead(). Esta función recibe como argumentos la dirección del registro a leer, el número de bytes que se van a leer y la variable donde se almacenarán dichos bytes. Para los datos del giroscopio leeremos a partir del registro GYRO_XOUT_H.

Name: ACCEL_XOUT_H
Address: 45 (2Dh)
Type: USR0
Bank: 0
Serial IF: R
Reset Value: 0x00

BIT	NAME	FUNCTION
7:0	ACCEL_XOUT_H[7:0]	High Byte of Accelerometer X-axis data

Figura 53. Registro ACCEL_XOUT_H del ICM-20648

Name: GYRO_XOUT_H
Address: 51 (33h)
Type: USR0
Bank: 0
Serial IF: R
Reset Value: 0x00

BIT	NAME	FUNCTION
7:0	GYRO_XOUT_H[7:0]	High Byte of Gyroscope X-axis data

Figura 54. Registro GYRO_XOUT_H del ICM-20648

Los valores obtenidos de estos registros van a estar comprendidos en el intervalo de -32768 a +32767 puesto que se trata de un ADC de 16 bits. Para obtener los datos de aceleración y velocidad angular, necesitamos calcular la resolución del ADC según la sensibilidad del acelerómetro (±2g) y del giroscopio (±250dps). Para ello utilizaremos la siguiente fórmula:

$$Resolucion = \frac{V_{max}}{32767}$$

Donde V_{max} va a ser el valor máximo de sensibilidad de cada dispositivo.

$$Resolucion_{acel} = \frac{2\ g}{32767} = 61.04 \cdot 10^{-6}\ g$$

$$Resolucion_{giro} = \frac{250\ dps}{32767} = 7.63 \cdot 10^{-3}\ dps$$

A la hora de aplicar las ecuaciones vistas en el apartado de Filtrado de la señal, para calcular el ángulo de inclinación del acelerómetro, no se aplicará la resolución puesto que no se necesitan los valores de la aceleración. En cambio, para calcular el ángulo de rotación del giroscopio, sí que se necesitan los datos de velocidad angular por lo que los datos leídos de los registros del giroscopio se multiplicarán por la resolución del mismo.

Las funciones utilizadas para obtener los datos son las siguientes:

- *ICM20648_accelDataRead()*: función que lee los valores obtenidos por el acelerómetro y calcula el ángulo de inclinación del acelerómetro. Recibe como argumento un array de 3 elementos (*accel)* donde se almacenarán los valores de aceleración de los 3 ejes.

```
void ICM20648 accelDataRead( float *accel )
{

   uint8_t rawData[6];

   /* Read the six raw data registers into data array */
   ICM20648_registerRead( ICM20648_REG_ACCEL_XOUT_H_SH, 6, &rawData[0] );

   accel[0] = ( (int16_t) rawData[0] << 8 ) | rawData[1];
   accel[1] = ( (int16_t) rawData[2] << 8 ) | rawData[3];
   accel[2] = ( (int16_t) rawData[4] << 8 ) | rawData[5];

   accel_ang_x = atan(accel[1]/sqrt(pow(acc[0],2) + pow(accel[2],2))) *
(180/3.141592);

   accel_ang_y = atan(-accel[0]/sqrt(pow(acc[1],2) + pow(accel[2],2))) *
(180/3.141592);

   return;

}
```

Código 12. Función *ICM20648_accelDataRead()*

- *ICM20648_gyroDataRead()*: función que lee los valores obtenidos por el giroscopio, los convierte a valores dps utilizando la resolución establecida y calcula los ángulos de navegación Roll, Pitch y Yaw. Recibe como argumento un array de 3 elementos (*gyro)* donde se almacenarán los valores de los 3 ejes.

```
void ICM20648_gyroDataRead( float *gyro )
{

   uint8_t rawData[6];
```

```
    float gyroRes = 250.0/32767.0;
    int16_t temp;

    /* Read the six raw data registers into data array */
    ICM20648_registerRead( ICM20648_REG_GYRO_XOUT_H_SH, 6, &rawData[0] );

    /* Convert the MSB and LSB into a signed 16-bit value and multiply by the
    resolution to get the dps value */
    temp = ( (int16_t) rawData[0] << 8 ) | rawData[1];
    gyro[0] = (float) temp * gyroRes;
    temp = ( (int16_t) rawData[2] << 8 ) | rawData[3];
    gyro[1] = (float) temp * gyroRes;
    temp = ( (int16_t) rawData[4] << 8 ) | rawData[5];
    gyro[2] = (float) temp * gyroRes;

    ang_x = 0.98*(ang_x_prev+(gyro[0]*0.08)) + 0.02*accel_ang_x;    // Roll
    ang_y = 0.98*(ang_y_prev+(gyro[1]*0.08)) + 0.02*accel_ang_y;    // Pitch
    ang_z = ang_z_prev+(gyro[2]*0.08);                              // Yaw

    ang_x_prev = ang_x;
    ang_y_prev = ang_y;
    ang_z_prev = ang_z;

    return;

}
```

Código 13. Función *ICM20648_gyroDataRead ()*

Una vez obtenidos los ángulos Roll, Pitch y Yaw, se enviarán por comunicación USART al módulo bluetooth para su posterior envío al PC. Cada ángulo ocupará 2 bytes por lo que se enviarán 6 muestras por cada medida.

Las funciones utilizadas para el envío de las muestras son las siguientes:

- *getAngles()*: lee los valores de los ángulos de navegación de cada eje y los almacena en el array de 3 elementos *angles*

```
void getAngles ( int16_t angles[3])
{
  angles[0] = (int16_t) ang_x;
  angles[1] = (int16_t) ang_y;
  angles[2] = (int16_t) ang_z;

  return;
}
```

Código 14. Función *getAngles()*

- *UINT_TO_BITSTREAM()*: convierte un dato de 16 bits en 2 datos de 8 bits

```
#define UINT16_TO_BITSTREAM(p, n)
{
  *(p)++ = (uint8_t)(n);
  *(p)++ = (uint8_t)((n) >> 8);
}
```

Código 15. Función *UINT_TO_BITSTREAM()*

- *sendOri()*: envía los 6 datos de los ángulos de navegación a través de la USART0 al módulo bluetooth

```
void sendOri(void)
{
   uint8_t buffer[6];
   uint8_t *p;
   int16_t angles[3];

   p = buffer;

   getAngles(angles);
   UINT16_TO_BITSTREAM(p, (uint16_t)angles[0]);
   UINT16_TO_BITSTREAM(p, (uint16_t)angles[1]);
   UINT16_TO_BITSTREAM(p, (uint16_t)angles[2]);

   for(j=0;j<6;j++)
     USART_Tx(usart, buffer[j]);

   return;

}
```

Código 16. Función *sendOri()*

4.4. Localización

En este tercer bloque trataremos los distintos métodos de localización centrándonos en la localización en interiores. Al igual que el bloque anterior, se explicarán detalladamente los distintos elementos de la Thunderboard Sense utilizados para determinar la posición de la misma.

Las tecnologías satélites son sin duda las más utilizadas en los sistemas de localización modernos, destacándose el sistema americano GPS (Global Positioning System) y el ruso GLONASS (Global Orbiting Navigation Satellite System). Estas tecnologías resuelven el problema de localización en espacios abiertos, pero se quedan cortas en espacios cerrados, debido a la atenuación de las señales dentro de las edificaciones.

El GPS es la solución "casi universal" para obtener localización precisa y rápida en cualquier punto del planeta. Lamentablemente, en determinados entornos exteriores, y en la mayoría de los interiores, el GPS no es operativo. Sin embargo, en dichos lugares transcurre gran parte de la actividad humana

Un escenario de localización está formado por un conjunto de mínimo tres nodos de referencia o balizas cuya posición exacta es conocida, al estar dotados de tecnología para ello, que se comunican con uno o más nodos móviles cuya posición es desconocida. Los nodos pueden actuar como emisores o receptores de las señales transmitidas entre ellos.

4.4.1. Técnicas de localización

Existen diferentes técnicas de localización que podemos agrupar en dos tipos, basadas en distancias o rango (range-based) y basadas en proximidad o libres de distancia (range-free).

4.4.1.1. Técnicas de localización basadas en distancias

La localización en las técnicas basadas en distancias se realiza en dos pasos. Primero se calcula la distancia y/o los ángulos entre los nodos emisor y receptor y a continuación, se determina la posición mediante algoritmos de estimación de la posición.

Comenzaremos definiendo las diferentes técnicas para determinar la distancia entre nodos.

4.4.1.1.1. Técnicas de estimación de distancias

4.4.1.1.1.1. Tiempo de llegada (ToA)

Esta técnica se basa en la medición del tiempo que tarda una señal en viajar de un nodo a otro a una velocidad conocida. Para ello además de la velocidad de la señal, es necesario conocer el instante de tiempo exacto en el que se envía la señal y en el que se recibe por lo que requiere de sincronización estricta entre emisor y receptor. La distancia se halla aplicando la siguiente ecuación:

$$d = v * (t_{llegada} - t_{envío})$$

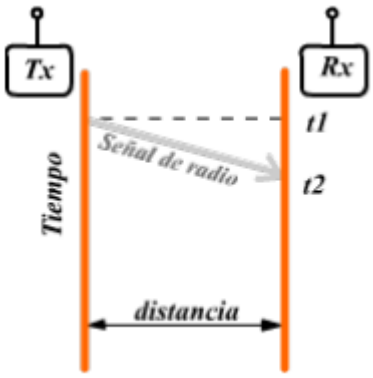

Figura 55. Medida de distancia con ToA

4.4.1.1.1.2. Diferencias de tiempos de llegada (TDoA)

Esta técnica utiliza la medición de la diferencia de tiempos que tardan en llegar un pulso acústico o ultrasónico y un pulso de radio del nodo emisor al receptor conociendo la velocidad de ambos pulsos. Para hallar la distancia se aplica la siguiente ecuación:

$$d = (V_r - V_s) * (t_s - t_r - t_{retardo})$$

Donde V_r y t_r son la velocidad y el tiempo de propagación de la señal de radio respectivamente, V_sy t_s son la velocidad de propagación y el tiempo de la señal acústica o ultrasónica y $t_{retardo}$ es el tiempo fijo de espera.

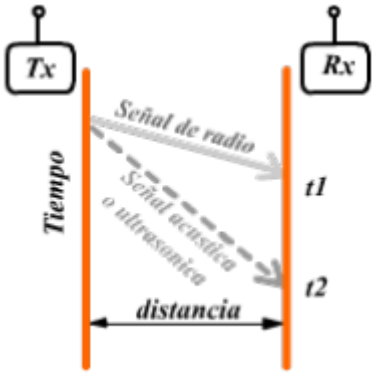

Figura 56. Medida de distancia con TDoA

4.4.1.1.1.3. Ángulo de llegada (AoA)

El ángulo de llegada se define como el ángulo que forma la dirección de propagación de una onda incidente y una determinada dirección de referencia conocida como orientación. Al igual que ocurre con las técnicas ToA y TDoA, el cálculo de AoA requiere dotar a la red de un costoso hardware adicional.

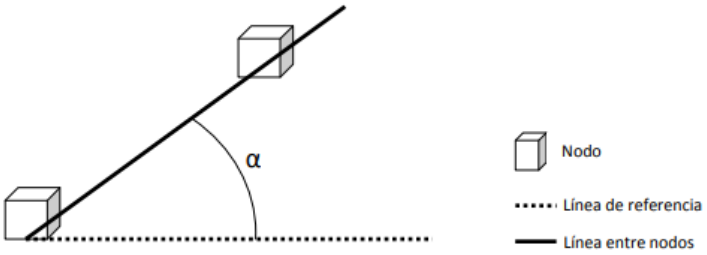

Figura 57. Medida del AoA entre dos nodos

4.4.1.1.1.4. Indicador de fuerza de la señal recibida (RSSI)

Conociendo la intensidad de potencia de una señal en un receptor, se podría estimar la distancia a la cual está ubicado el emisor. Esta técnica utiliza las propiedades

de atenuación de la señal de radio para modelar la distancia entre nodos como una función del indicador de la fuerza de la señal recibida.

Aunque pareciera simple, esta técnica presenta significativos errores ocasionados por las pérdidas de trayecto dependientes de las características físicas del entorno por lo que su eficiencia depende mucho de la distancia entre nodos, funcionando mejor en distancias cortas.

Una vez calculadas las distancias y/o ángulos entres los nodos emisor y receptor, se procede a determinar la posición del nodo desconocido mediante técnicas de estimación de la posición que definiremos a continuación.

4.4.1.1.2. Técnicas de estimación de posición

4.4.1.1.2.1. Multilateración esférica

La multilateración esférica es una técnica que calcula la posición de un nodo a partir de las distancias medidas desde el propio nodo hasta varios nodos de referencia o balizas. Si se trabaja en dos dimensiones, se requieren al menos las distancias desde tres balizas no colineales (en diferentes líneas) y se conoce como trilateración. Para tres dimensiones serán necesarias cuatro balizas.

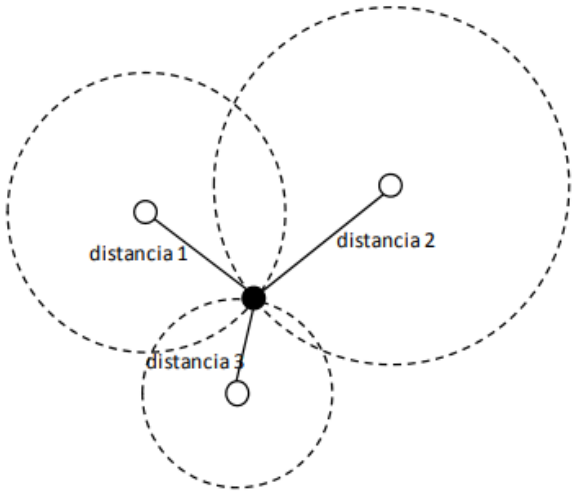

Figura 58. Técnica de trilateración

Para calcular estas distancias, suelen utilizarse técnicas ToA por lo que se requiere que todos los nodos estén sincronizados. Una vez calculada, cada nodo baliza

traza una circunferencia de radio igual a la distancia obtenida. El punto de intersección de las circunferencias será la posición donde se encuentre el receptor.

4.4.1.1.2.2. Multilateración hiperbólica

La multilateración hiperbólica es una técnica que utiliza TDoA para calcular la distancias entre los nodos baliza y el nodo móvil desconocido, por lo que no requiere sincronización entre ellos. A diferencia de la multilateración esférica, la medida de la diferencia de tiempos de llegada da como resultado un número infinito de localizaciones que satisfacen las condiciones. Cuando se trazan estas posibles localizaciones se encuentra que forman una curva hiperbólica.

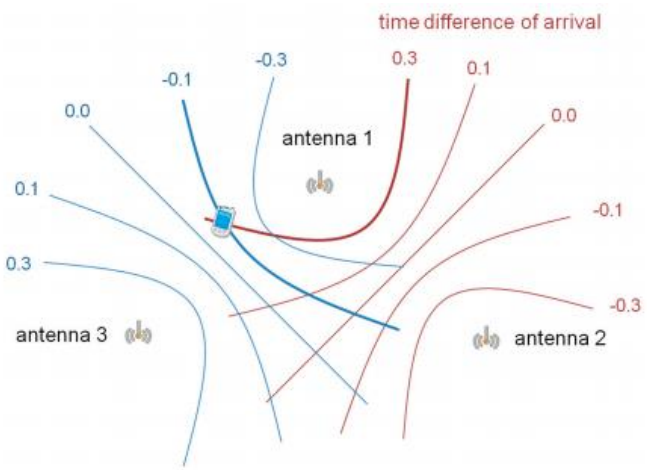

Figura 59. Multilateración hiperbólica

El procedimiento es similar al caso de la multilateración esférica, cada nodo baliza calcula la distancia al nodo móvil mediante TDoA y traza una curva hiperbólica de las posibles localizaciones. La intersección de todas las curvas será la posición del nodo receptor.

Para el caso bidimensional se requieren tres nodos baliza, mientras que para el caso tridimensional se requieren cuatro.

4.4.1.1.2.3. Triangulación

La triangulación es una técnica similar a la trilateración salvo que utiliza ángulos en lugar de distancias para determinar la posición del nodo desconocido. En un entorno bidimensional, se requieren dos ángulos y la distancia entre dos puntos de referencia.

En el caso de tres dimensiones se necesitan dos ángulos, la distancia entre los nodos y un azimut.

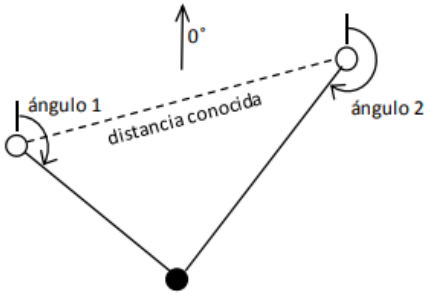

Figura 60. Triangulación

4.4.1.2. Técnicas de localización libres de distancias

Las técnicas de localización libres de distancias no requieren calcular la distancia o ángulos entre los nodos emisor y receptor. Se aplican de forma alternativa o paralela a las técnicas basadas en distancias.

4.4.1.2.1. Técnicas basadas en el análisis del escenario

Las técnicas basadas en el análisis del escenario (fingerprinting) utilizan datos previamente almacenados y se basan en mediciones de la potencia de la señal recibida (RSS). Se desarrollan en dos fases, en la primera, denominada fase off-line o fase de entrenamiento, se realizan estas mediciones RSS en puntos de referencia predefinidos y se genera una matriz o mapa de potencia con los datos obtenidos. A continuación, en la fase on-line o fase de posicionamiento, se mide la potencia al nodo móvil ubicado dentro del mapa de potencia y se compara con las existentes en el propio mapa para estimar la posición del nodo móvil.

4.4.1.2.2. Técnicas basadas en proximidad

Las técnicas basadas en proximidad proporcionan información de la posición relativa o simbólica, basándose en una densa red de antenas o sensores. Se asume que el nodo móvil tiene la misma posición que el sensor o antena más cercano al mismo, que será el que reciba su señal con más potencia.

Su bajo costo y fácil implementación las han convertido en una opción para aplicaciones donde no se requiere de una localización demasiado exacta.

4.4.2. Sistemas de Posicionamiento Local

Un Sistema de Posicionamiento Local (LPS) es una tecnología utilizada para obtener la posición o información de la ubicación de un objeto en un área local. Es similar a un GPS, salvo que trabaja a nivel local utilizando balizas de señalización de corto alcance en lugar de satélites.

Los LPS suelen utilizarse para extender el alcance de los GPS proporcionando información de posición en áreas inaccesibles para sus señales o bien para mejorar la precisión de los GPS. Aunque se ha realizado numerosos estudios de investigación durante los últimos años, ninguna de las soluciones propuestas ha alcanzado el éxito de los sistemas de localización empleados en exteriores debido a razones técnicas y sobre todo económicas. La localización en interiores plantea retos tecnológicos muy superiores a la localización en el exterior y los sistemas que emplea utilizan una gran cantidad de elementos que incrementan el coste.

4.4.2.1. Identificación por radiofrecuencia (RFID)

RFID (Radio Frecuencia Identificación) es una tecnología inalámbrica capaz de identificar de forma única los objetos y transmitir esta información al sistema a través de señales radio. Para dar servicios de localización de buena precisión con esta tecnología se requiere un alto número de sensores.

Esta tecnología se basa en etiquetas de radiofrecuencia que contienen una antena que al ser excitada por un transmisor emite una señal. Así, un usuario que necesitase conocer su posición en un edificio tendría que tener cerca de él un número de etiquetas de radiofrecuencia. El propio usuario tendría un lector de etiquetas RFID y leyendo las etiquetas cercanas a él, se puede llegar a conocer su posición.

Este método puede localizar de manera satisfactoria, pero es aún un método caro debido a la gran cantidad de lectores RFID que se necesitan para una localización correcta, ya que el alcance de estas señales es muy reducido. Además, es necesaria la utilización de otras tecnologías de soporte para la implantación de estos sistemas.

4.4.2.2. Infrarrojos

La radiación infrarroja, o radiación IR, es un tipo de radiación electromagnética y térmica, de mayor longitud de onda que la luz visible pero menor que la de las microondas. Su rango de longitudes de onda va desde aproximadamente 0,7 micrómetros hasta los 1000 micrómetros y es emitida por cualquier cuerpo cuya temperatura sea mayor que 0º Kelvin.

En la localización por infrarrojos se utilizan etiquetas (tags) que emiten radiación infrarroja en modo difusivo, es decir, de forma radial, no en modo punto a punto como es habitual en los sistemas IR empleados en comunicaciones. Se trata de un sistema de detección más que de localización y se basa en proximidad. La posición del elemento etiquetado se deduce de la posición fija y conocida de los elementos sensores que lo detectan.

Al ser un método de corto alcance (unos dos metros) habría que incluir una cantidad importante de emisores de infrarrojos, con el consiguiente encarecimiento de la instalación.

4.4.2.3. Ultrasonidos

Los ultrasonidos son ondas acústicas cuya frecuencia está por encima del umbral de audición del oído humano, esto incluye cualquier frecuencia superior a 20 kHz.

Los sistemas basados en ultrasonidos presentan una ventaja importante frente a los basados en radiofrecuencia ya que su velocidad de propagación es mucho menor. La velocidad de los ultrasonidos es aproximadamente la velocidad del sonido (343 m/s) mientras las ondas radio viajan a la velocidad de la luz (10^8 m/s) lo que permite enviar desde el emisor un mensaje radio y un pulso de ultrasonidos en el mismo instante de tiempo y así calcular con mayor precisión la localización mediante técnicas TDoA.

La tecnología de ultrasonidos obtiene gran precisión en la localización, pero presenta algunos inconvenientes como los multitrayectos que se producen en señales debido a reflexiones. Si no existe línea de visión directa entre emisor y receptor, el alcance es mucho menor debido a la disipación de energía de las ondas acústicas al rebotar en superficies. Esto conlleva a la necesidad de utilizar un gran número de sensores lo que incrementa su coste.

4.4.2.4. Banda ultra-ancha (UWB)

Estas tecnologías se basan en radiación UWB que tiene anchos de banda relativos superiores al 20% o anchos de banda absolutos de as de 500 MHz. Con un amplio ancho de banda se consigue que la distribución de energía se reparta por un rango de frecuencias muy grande, con lo que la densidad espectral es muy pequeña reduciendo la interferencia con otros sistemas y la posibilidad de interceptación de la señal.

UWB emplea ráfagas de potencia más bajas que las otras técnicas mencionadas, con duración de picosegundos y velocidades de transferencia de datos mucho mayores. En este tipo de sistemas pueden utilizarse técnicas ToA y TDoA con las que se obtiene

una localización con gran precisión o técnicas RSSI de menor precisión, pero más económicas.

4.4.2.5. Wifi

La tecnología WiFi (Wireless-Fidelilty) es una tecnología de gran éxito comercial basada en el estándar 802.11. La comunicación típica sigue un modelo centralizado lo que implica que la red consta de uno o varios puntos de acceso y multitud de clientes conectados a uno de estos puntos. Para localizar el móvil, los puntos de acceso emiten periódicamente mensajes que son recibidos por los usuarios, los cuales pueden medir la potencia recibida e informar a una estación base para el cómputo de la localización.

La mayoría de las soluciones de localización con Wifi están basadas en técnicas de fingerprinting. Estas técnicas tienen como ventajas su sencillez y rapidez de despliegue, la ausencia de equipamiento adicional a la de una red LAN y su coste asociado.

La triangulación Wifi evita los requerimientos de línea de visión entre emisor y receptor, pero sufre los efectos del multicamino debido a las reflexiones en el entorno. Usando fingerprinting se mejoran las estimaciones teniendo en cuenta los efectos que causan los diferentes obstáculos, así como las reflexiones y atenuación. Aun así, pero los cambios en el medio causan importantes fluctuaciones en las señales Wifi en un mismo punto con el paso del tiempo, por lo que las medidas obtenidas en la fase de entrenamiento se quedan rápidamente obsoletas.

La ventaja principal respecto a otras tecnologías es la existencia de infraestructura ya instalada en muchos edificios contribuyendo así a que el coste hardware sea prácticamente nulo. Las principales desventajas son el alto consumo de energía de los dispositivos y la vulnerabilidad a las interferencias.

4.4.2.6. Bluetooth

Bluetooth es el nombre común de la especificación de IEEE 802.15.1. En esta tecnología se utilizan técnicas de triangulación basadas en la información proporcionada por la potencia de la señal recibida (RSSI). El sistema consta de uno o varios dispositivos a localizar que utilizan la información del RSSI de otros dispositivos que actúan como balizas. Estas balizas tienen una localización fija en el entorno y su posición es perfectamente conocida. El sistema tendrá en cuenta tanto los valores de RSSI recibidos por el dispositivo a localizar, como los recibidos entre balizas.

Las ventajas de esta tecnología se encuentran en su bajo coste y su amplio despliegue en dispositivos comerciales. El mayor inconveniente radica en que el RSSI no es preciso debido a la heterogeneidad de hardware en los dispositivos disponibles.

4.4.2.7. Redes de telefonía móvil

La red GSM (Sistema Global de comunicaciones Móviles) es el estándar más usado de Europa. Se denomina estándar de segunda generación (2G) porque las comunicaciones se producen de un modo completamente digital a diferencia de la primera generación de teléfonos móviles que se producían de forma analógica. Su extensión 3G se denomina UMTS (Sistema Universal de Telecomunicaciones Móviles) y difiere en su mayor velocidad de transmisión, el uso de una arquitectura de red ligeramente distinta y empleo de diferentes protocolos radio.

La red GSM está formada por estaciones base (BTS) que proporcionan cobertura a una o varias células. La localización se basa en la detección de la célula a la que está conectada el teléfono móvil por lo que su exactitud depende del tamaño de la propia célula. En las zonas urbanas, pobladas por numerosas células, la exactitud puede ser de decenas de metros. En las zonas rurales, donde se necesitan menos células, la exactitud es mucho menor debido al que la zona de cobertura de las células es mucho mayor.

4.4.3. Sistema de Posicionamiento Local Ultrasónico

En este proyecto se va a trabajar con un Sistema de Posicionamiento Local Ultrasónico (U-LPS). Para ello se utilizará una baliza colocada en el techo compuesta por un sistema central y un set de 5 transductores situados en posiciones conocidas y orientados hacia el suelo. El sistema central se encargará de generar las señales eléctricas y de sincronizar las emisiones. Los transductores transforman las señales eléctricas en señales ultrasónicas con una frecuencia aproximada de 40 kHz.

Para identificar a que transductor pertenece la señal recibida, se va a emplear la técnica de acceso múltiple. Las transmisiones ultrasónicas son codificadas por 5 secuencias Kasami de 255 bits y se empleará modulación BPSK (Binary Phase Shift Keying). La distancia entre los transductores y la tarjeta se calculará utilizando técnicas ToA y la posición mediante trilateración esférica.

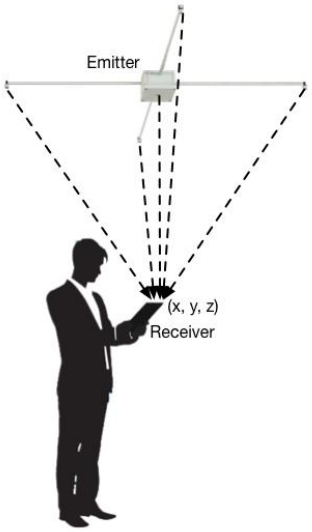

Figura 61. Estructura del U-LPS

La estructura de la baliza se compone de 4 brazos de aluminio y 50 cm de longitud, colocados en cruz y apoyados sobre una caja de derivación que contiene el sistema central. Los transductores están colocados, 4 de ellos en los extremos de cada brazo y el quinto en el centro de la baliza.

El sistema central dispone de un microcontrolador encargado de generar las señales ya moduladas, una etapa de acondicionamiento de la señal y un demultiplexor que selecciona el transductor al que se le va a enviar la señal generada. El microcontrolador utilizado es el LPC1768 de 32 bits basado en un núcleo ARM Cortex-M3, con una frecuencia de reloj de 96 MHz, 64 kB de memoria SRAM y 512 kB de memoria Flash y un DAC de 10 bits.

4.4.3.1. Técnicas de acceso múltiple

Las técnicas de acceso múltiple utilizadas en comunicación digital permiten que varios emisores puedan transmitir información utilizando el mismo canal de comunicación y permiten al receptor separar y recuperar de forma independiente cada señal sin verse afectada por el resto.

Existen tres técnicas principales:

- Acceso Múltiple por División de Frecuencia (FDMA): asigna a cada emisor una banda de frecuencia diferente, esto permite a varias balizas emitir simultáneamente. Para separar cada una de las señales, el receptor debe filtrar

la señal recibida teniendo en cuenta el ancho de banda ocupado por la señal del transmisor que desea obtener.

- Acceso Múltiple por División de Tiempo (TDMA): permite a diferentes emisores compartir una misma banda de frecuencias, para ello restringe la transmisión de cada uno a un intervalo de tiempo denominado ranura.
- Acceso Múltiple por División de Código (CDMA): asigna a cada emisor un código diferente, de esta manera permite emitir a varias balizas simultáneamente y utilizar la misma banda de frecuencias.

Para este proyecto se va a utilizar la técnica TDMA que hará que la baliza transmita de forma secuencial las 5 señales de los transductores y además se asigna a cada uno un código identificativo.

4.4.3.2. Secuencias Kasami

El conjunto de secuencias o códigos Kasami es uno de los tipos más importantes de secuencia binarias por su mayor amplitud en el pico de autocorrelación y sus bajos valores de correlación cruzada.

Un conjunto de códigos Kasami de 255 bits nos proporciona 16 códigos. A cada transductor se le asignará un código por lo que solo se emplearán 5. Al transductor central se le va a asignar el código 1 y a los transductores de los brazos los códigos 4, 5, 6 y 7.

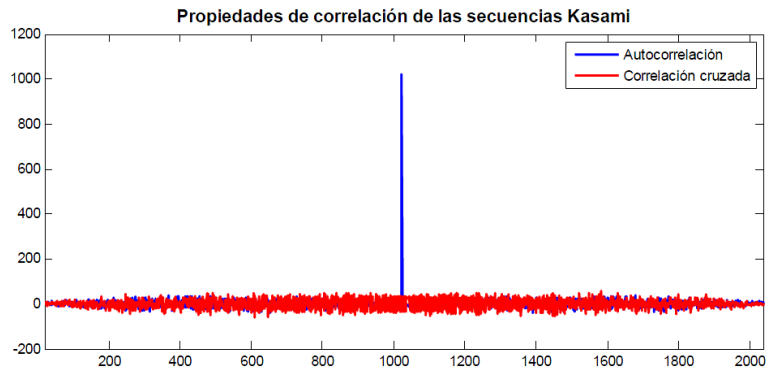

Figura 62. Autocorrelación y correlación cruzada de códigos Kasami

4.4.3.3. Modulación y emisión periódica de la secuencia

Para enviar los códigos digitales mediante una señal ultrasónica se utilizan técnicas de modulación, que se encargan de transportar dicha información sobre una portadora, típicamente una onda sinusoidal.

La modulación binaria por desplazamiento de fase (BPSK) destaca por su fácil implementación y adecuación al ancho de banda del transductor de ultrasonidos, además de ser bastante inmune a las interferencias de ruido de tipo impulsivo.

Es un tipo de modulación digital donde se utiliza un determinado número de ciclos de una señal sinusoidal a los que se les introduce un desfase dependiendo del valor a modular. En el caso binario, donde la constelación está formada únicamente por dos símbolos o dígitos, los desfases introducidos son de 0° para un valor de 0 binario y 180° para un valor de 1. La frecuencia de la portadora que se utiliza en este caso es de 40kHz.

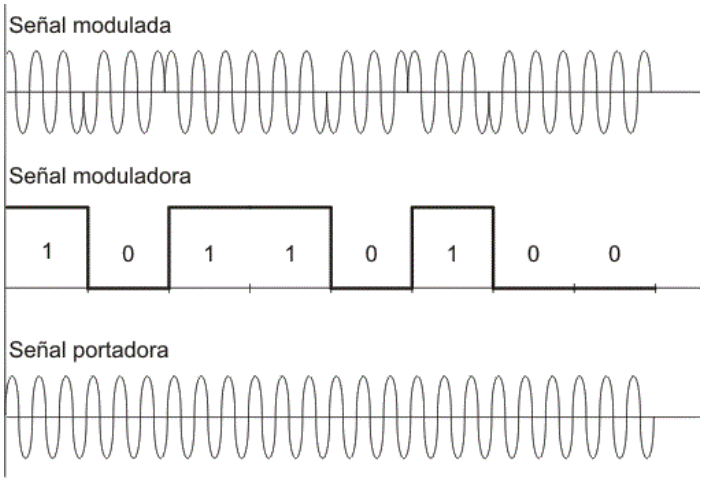

Figura 63. Modulación BPSK

Para evitar que los multitrayectos de una emisión puedan afectar a la siguiente, las emisiones se espaciarán dejando un tiempo de guarda (t_0). La frecuencia de emisión de los códigos (f_E) seleccionada es de 50 Hz, por tanto:

$$f_E = \frac{1}{t_E} = \frac{1}{50} Hz$$

$$t_E = 20\ ms$$

La frecuencia de trabajo del DAC es de 500 kHz y se seleccionan 12 muestras por cada ciclo de la señal portadora por lo que la frecuencia de la señal portadora será:

$$f_P = \frac{500\ kHz}{12\ muestras/ciclo} = 41.66\ kHz$$

A partir de la frecuencia de la señal portadora, podemos obtener el tiempo que tarda en emitirse un código (t_c) calculando previamente el periodo de la portadora (T_P) y el tiempo necesario para enviar un bit (t_n):

$$T_P = \frac{1}{f_P} = 24\ \mu s$$

$$t_n = T_p * m = 24\ \mu s * 2\ ciclos/bit = 48\ \mu s$$

$$t_C = t_n * n = 48\ \mu s * 255\ bits = 12.24\ ms$$

Para evitar que se solapen las emisiones el tiempo entre el envío de las mismas debe ser superior al tiempo que tarda en emitir cada una:

$$t_E = t_C + t_0$$

$$t_0 = t_E - t_C = 7.76\ ms$$

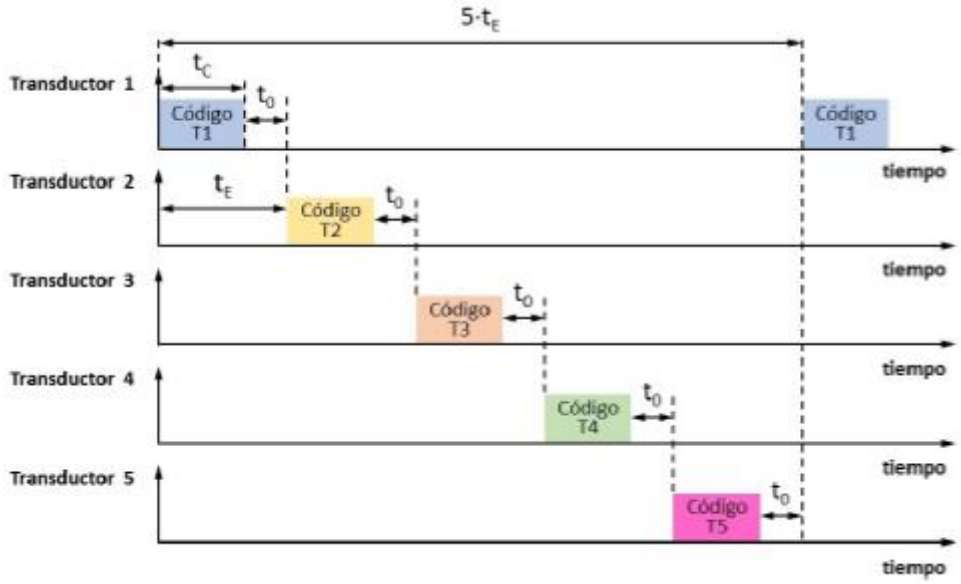

Figura 64. Secuencia de emisión de la baliza

El inicio de la emisión periódica se sincroniza con una señal infrarroja cuya portadora es de 500 kHz y una duración de 1 ms.

Esta información se ha extraído del capítulo 1.4. Sistema de Posicionamiento Local Ultrasónico del TFG “Diseño de un módulo receptor para un PLS ultrasónico” por J. F. Arango Vargas.

4.4.4. Bloque de recepción

El bloque de recepción se encarga de captar las señales emitidas por la baliza y procesarlas para obtener la posición actual de la placa. El micrófono será el encargado de recoger estas señales y transmitirlas al ADC. Una vez convertidas a digital, se enviarán en un búfer de 10000 muestras a un PC vía bluetooth que realizará las operaciones de correlación y los procedimientos de localización mediante ToA y trilateración esférica.

Para realizar el proceso de correlación, el PC genera localmente una réplica de las señales emitidas (señales patrón). Una vez generadas, se realiza la correlación de estas señales patrón con el búfer recibido del ADC y se obtiene la posición en muestras de los picos de autocorrelación. Tomando como referencia el código del transductor 1, se calculan las diferencias en muestras entre la posición de los picos del resto de transductores y el de referencia. Las diferencias en muestras se convierten en diferencias de tiempo restando el tiempo transcurrido entre las emisiones de señal de referencia con el resto. Por último, estas diferencias de tiempo se convierten en diferencias de distancia y a partir de ellas se calcula la posición mediante trilateración esférica.

4.4.4.1. Micrófono MEMS

Los micrófonos MEMS (Microelectromechanical-System) son dispositivos transductores de audio que convierten las ondas de presión acústica en señales eléctricas. Están fabricados utilizando tecnología MEMS, que nos permite obtener micrófonos más pequeños y la posibilidad de integrarse con electrónica analógica y digital en el mismo dispositivo. De esta manera, los micrófonos pueden integrar toda la etapa de acondicionamiento y digitalización de la señal y ofrecen a la salida una representación eléctrica del nivel de presión sonora a su entrada de manera más aprovechable en sistemas electrónicos.

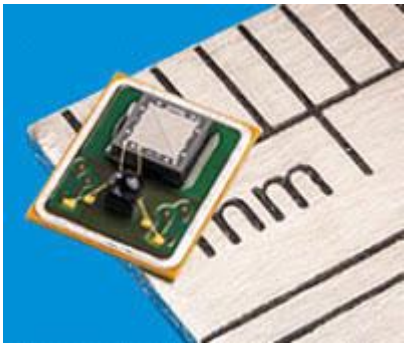

Figura 64. Micrófono MEMS

4.4.4.1.1. Características del SPV1840

El SPV1840 es un micrófono MEMS omnidireccional y con salida analógica fabricado por Knowles. Sus dimensiones son 3.75x1.85x0.99 mm.

Las principales características especificadas por el fabricante son las siguientes:

- Voltaje de alimentación de 1.5 V a 3.6 V
- Corriente típica de 45 µA para una tensión de 1.8 V
- Corriente típica de 52 µA para una tensión de 3.6 V
- Sensibilidad típica de -38 dB
- Relación Señal/Ruido de 62.5 dB
- Distorsión Armónica Total de 0.2 %
- Temperatura operativa de -40 ˚C a 100 ˚C

Como se comentó en el apartado del EFM8, para habilitar la comunicación entre el micrófono y el ADC, escribiremos un 0x01 en el registro MIC_CTRL del Controlador de Alimentación e Interrupción.

4.4.4.2. Conversor Analógico Digital

Un Conversor Analógico Digital (ADC) en un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador.

Las principales características que incluye el ADC del EFR32 son las siguientes:

- Resolución programable entre 6/8/12 bits
- Tiempo de adquisición configurable
- Prescaler integrado con un factor de división elegible en un rango de 1 a 128
- Soporta hasta 144 entradas de canales externos y 11 internos
- Modo de conversión Single y Scan
- Reloj en modo síncrono o asíncrono

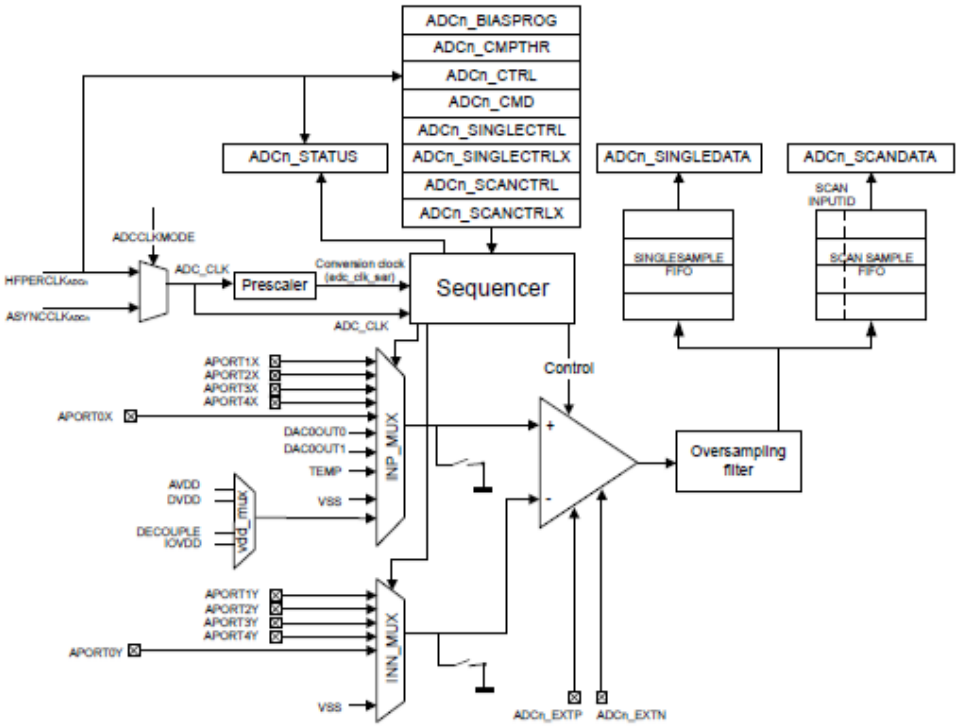

Figura 65. Esquema del ADC del EFR32

El reloj principal del ADC es el ADC_CLK que puede funcionar en modo síncrono (HFPERCLK) o modo asíncrono (ASYNCCLK). Dispone también de un prescaler que divide el ADC_CLK por un factor entre 1 y 128 generando el reloj de conversión (adc_clk_sar) del ADC. Puede funcionar en modo de bajo consumo, pero para ello se debe escoger el modo asíncrono, ASYNCCLK, y que su fuente sea el oscilador AUXHFRCO.

Una conversión consiste en dos fases, una de adquisición y otra de aproximación. Durante la fase de adquisición se muestrea la entrada y durante la fase de aproximación se convierte el valor muestreado a digital. El tiempo de adquisición es configurable y tendrá una duración de 1, 2, 3 o potencias de 2 hasta 256, ciclos de adc_clk_sar. El ADC utiliza un ciclo de adc_clk_sar por bit de salida en la fase de aproximación más un ciclo extra.

$$T_{conv} = \left(T_{acq} + (N+1) * T_{adc\_clk\_sar}\right) * OVSRSEL$$

Donde T_{acq} es el tiempo de adquisición elegido, N es la resolución en bits y $OVSRSEL$ es el ratio de oversampling si se utilizase. Para nuestra aplicación, el tiempo de adquisición va a ser de 1 ciclo de adc_clk_sar, la resolución será de 8 bits y no se utilizará oversampling.

$$T_{conv} = \left(1 + (8+1) * T_{adc\_clk\_sar}\right) = 10\, T_{adc\_clk\_sar}$$

El ADC se configurará utilizando el modo Scan que nos permite, si fuera necesario, utilizar más de un canal del ADC para realizar las conversiones, aunque finalmente solo se utilizará uno. El trigger utilizado para iniciar la conversión va a ser una señal PRS que se va a configurar en modo PULSED. Cuando se reciba un pulso de la señal, se habilitará el ADC_CLK, comenzará el periodo de adquisición y se realizará la conversión.

Existen 2 multiplexores a la entrada del ADC que lo conectan a las señales externas a través de 5 puertos analógicos (APORT0-APORT4). Los puertos 1 a 4 están compuestos de 32 canales divididos en un bus X de 16 canales y un bus Y de 16 canales.

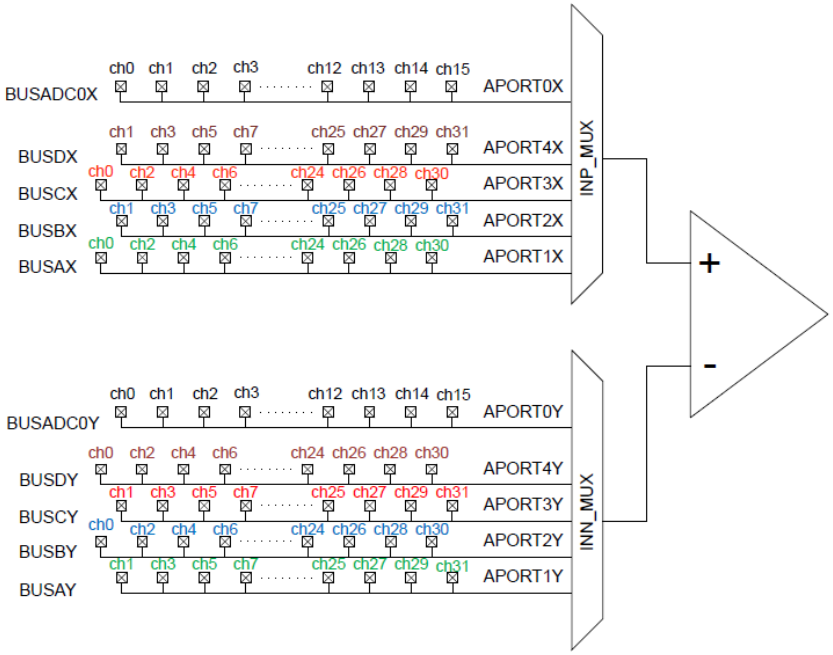

Figura 66. Conexiones de los puertos APORT con el ADC

Como se ve en la figura 66, los canales pares de los puertos APORT1 y APORT3 se conectarán al bus X mientras que los impares lo harán al bus Y. En los puertos APORT2 y APORT 4 ocurre al contrario. El puerto APORT0 no es un recurso compartido.

En la figura 67 vemos el mapa de las conexiones externas con las entradas del ADC. La salida del micrófono de la Thunderboard Sense está conectada al pin PF7 del EFR32 por lo que utilizaremos el canal 23, conectado al bus X del APORT2.

ADC Port	APORT0		APORT1		APORT2		APORT3		APORT4	
Polarity	X	Y	X	Y	X	Y	X	Y	X	Y
Shared Bus	n/a		BUSAX	BUSAY	BUSBX	BUSBY	BUSCX	BUSCY	BUSDX	BUSDY
CH31								PB15	PB15	
CH30							PB14			PB14
CH29								PB13	PB13	
CH28							PB12			PB12
CH27								PB11	PB11	
CH26										
CH25										
CH24										
CH23				PF7	PF7					
CH22			PF6			PF6				
CH21				PF5	PF5					
CH20			PF4			PF4				
CH19				PF3	PF3					
CH18			PF2			PF2				
CH17				PF1	PF1					
CH16			PF0			PF0				
CH15										
CH14										
CH13								PA5	PA5	
CH12							PA4			PA4
CH11				PC11	PC11			PA3	PA3	
CH10			PC10			PC10	PA2			PA2
CH9				PC9	PC9			PA1	PA1	
CH8			PC8			PC8	PA0			PA0
CH7				PC7	PC7			PD15	PD15	
CH6			PC6			PC6	PD14			PD14
CH5								PD13	PD13	
CH4							PD12			PD12
CH3								PD11	PD11	
CH2							PD10			PD10
CH1										
CH0										

Figura 67. Mapa de pines conectados al ADC

Los registros del ADC que se utilizarán son los siguientes:

- El registro ADC0_CTRL nos permite elegir el prescaler del reloj de conversión o el modo del reloj ADC_CLK entre otras cosas. En los bits 14:8 (PRESC) escribiremos un 1, el bit 7 (ADCCLKMODE) lo fijaremos a 1 para que funcione en modo asíncrono y el bit 6 (ASYNCCLKEN) a 0 para que el reloj ADC_CLK solo esté habilitado durante la conversión. Por último, el bit 3 (SCANDMAWU) lo fijaremos a 1 para que el ADC funcione como trigger del controlador DMA tras cada conversión.

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0		0x0					0x1F								0x00							0	0		0	0	0	0x0	
Access			RW		RW					RW								RW							RW	RW		RW	RW	RW	RW	
Name			CHCONMODE		OVSRSEL					TIMEBASE								PRESC							ADCCLKMODE	ASYNCCLKEN		TAILGATE	SCANDMAWU	SINGLEDMAWU	WARMUPMODE	

Figura 68. Registro ADC0_CTRL del EFR32

- En el registro ADC0_SCANCTRL fijaremos a 1 el bit 29 (PRSEN) para que el PRS actúe como trigger del muestreo. En los bits 7:5 (REF) escribiremos 0x2 para elegir la referencia del ADC que será VDD y escribiendo 0x1 en los bits 4:3 (RES) escogeremos la resolución de 8 bits.

Offset	Bit Position																															
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0		0		0x0																				0x0			0x0		0	0	0
Access	RW		RW		RW																				RW			RW		RW	RW	RW
Name	CMPEN		PRSEN		AT																				REF			RES		ADJ	DIFF	REP

Figura 69. Registro ADC0_SCANCTRL del EFR32

- ADC_SCANCTRLX para elegir el canal PRS y el modo de la señal PRS. Fijaremos a 0 los bits 20:17 (PRSSEL) para seleccionar el canal 0 y el bit 16 (PRSMODE) para elegir el modo PULSED.

Offset	Bit Position																															
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0	0x0						0x0				0		0	0x0		0x0				0x0				0		0x0	
Access					RW	RW						RW				RW		RW	RW		RW				RW				RW		RW	
Name					CONVSTARTDELAYEN	CONVSTARTDELAY						PRSSEL				PRSMODE		FIFOOFACT	DVL		VINATT				VREFATT				VREFATTFIX		VREFSEL	

Figura 70. Registro ADC0_SCANCTRLX del EFR32

- El registro ADC0_SCANMASK nos permite seleccionar los canales que utilizará el ADC. Fijando a 1 el bit 7 elegiremos el canal 7.

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	SCANINPUTEN																															

Figura 71. Registro ADC0_SCANMASK del EFR32

- Con el registro ADC0_SCANINPUTSEL podemos elegir las distintas entradas que utilizará el ADC. Escribiremos 0x0A en los bits 4:0 (INPUT0TO7SEL) para seleccionar el puerto APORT2 y los canales del 16 al 23 (APORT2CH16TO23).

Offset	Bit Position																															
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x00								0x00								0x00								0x00				
Access				RW								RW								RW								RW				
Name				INPUT24TO31SEL								INPUT16TO23SEL								INPUT8TO15SEL								INPUT0TO7SEL				

Figura 72. Registro ADC0_SCANINPUTSEL del EFR32

INPUT0TO7SEL	0x00	RW	**Inputs chosen for ADCn_INPUT7-ADCn_INPUT0 as referred in SCANMASK**

Mode	Value	Description
APORT0CH0TO7	0	Select APORT0's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
APORT0CH8TO15	1	Select APORT0's CH8-CH15 as ADCn_INPUT0-ADCn_INPUT7
APORT1CH0TO7	4	Select APORT1's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
APORT1CH8TO15	5	Select APORT1's CH8-CH15 as ADCn_INPUT0-ADCn_INPUT7
APORT1CH16TO23	6	Select APORT1's CH16-CH23 as ADCn_INPUT0-ADCn_INPUT7
APORT1CH24TO31	7	Select APORT1's CH24-CH31 as ADCn_INPUT0-ADCn_INPUT7
APORT2CH0TO7	8	Select APORT2's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
...	.	.....
APORT3CH0TO7	12	Select APORT3's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
...	.	.....
APORT4CH0TO7	16	Select APORT4's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
...	.	.....

Figura 73. INPUT0TO7SEL del registro ADC0_SCANINPUTSEL

El ADC muestreará las señales ultrasónicas a una frecuencia de 100 kHz. Para conseguir esta frecuencia de muestreo se ha utilizado uno de los Timers de 16 bits que nos proporciona el EFR32.

4.4.4.3. Timer

Para conseguir esa frecuencia de muestreo de 100 kHz vamos a configurar el Timer 0 en modo Up-count. En este modo, el contador funcionará de manera ascendente empezando por el valor 0 hasta un valor límite TOP. Cuando alcance este valor límite comenzará la cuenta de nuevo desde 0 provocando un overflow.

El reloj utilizado para el contador va a ser el HFXO con una frecuencia de 38.4 MHz. Para hallar el valor de TOP nos basta con aplicar la siguiente fórmula:

$$TOP = \frac{f_{HFXO}}{f_{deseada}} - 1 = \frac{38.4\ MHz}{100\ kHz} - 1 = 383$$

Vamos a ver a continuación los registros utilizados para configurar el Timer:

- El registro TIMER0_CTRL nos permite, entre otras cosas, elegir el modo de funcionamiento del Timer mediante los bits 1:0 (MODE). Está configurado por defecto en modo ascendente por lo que no se va a modificar nada de este registro.

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0	0	0x0										0x0				0		0x0		0x0		0	0	0	0	0		0x0	
Access			RW	RW	RW										RW				RW		RW		RW		RW	RW	RW	RW	RW		RW	
Name			RSSCOIST	ATI	PRESC										CLKSEL				X2CNT		FALLA		RISEA		DMACLRACT	DEBUGRUN	QDM	OSMEN	SYNC		MODE	

Figura 74. Registro TIMER0_CTRL del EFR32

- Con el registro TIMER0_CMD, podemos iniciar o detener el contador fijando a 1 los bits 0 (START) y 1 (STOP) respectivamente.

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0	0
Access																															W1	W1
Name																															STOP	START

Figura 75. Registro TIMER0_CMD del EFR32

- Mediante el registro TIMER0_TOP podemos elegir el valor límite del contador escribiendo dicho valor en los bits 15:0 (TOP). En nuestro caso va a ser 0x17F que se corresponde con 383 en hexadecimal.

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0xFFFF															
Access																	RWH															
Name																	TOP															

Figura 76. Registro TIMER0_TOP del EFR32

Configurado de este modo, se producirá un overflow con una frecuencia de 100 kHz, momento en que el ADC tomará una muestra. Para conseguir esto vamos a utilizar el Peripheral Reflex System como trigger del ADC.

4.4.4.4. Peripheral Reflex System (PRS)

El PRS permite comunicación rápida, autónoma y configurable entre diferentes periféricos del EFR32 sin hacer uso de la CPU. Un periférico actuará como productor de señales Reflex que el PRS encaminará a través de canales Reflex a un periférico consumidor que ejecutará distintas acciones en función de la señal recibida.

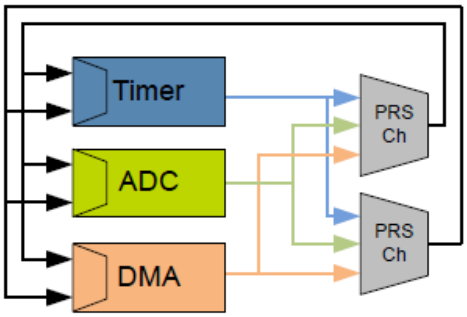

Figura 77. Peripheral Reflex System

El EFR32 dispone de 12 canales Reflex configurables. Cada canal puede conectarse a cualquier periférico productor incluyendo otros canales Reflex y los consumidores pueden elegir a que canal escucharán. Para nuestra aplicación vamos a utilizar el canal 0, el productor de la señal Reflex va a ser el Timer 0 y el consumidor será el ADC.

Mediante el registro PRS_CH0_CTRL podemos elegir tanto el productor como la señal que habilitará el canal PRS. Para seleccionar el Timer 0, escribiremos un 0x1C en los bits 14:8 (SOURCESEL).

Offset	Bit Position																														
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0		0	0	0	0				0x0										0x00									0x0		
Access		RW		RW	RW	RW	RW				RW										RW									RW		
Name		ASYNC		ANDNEXT	ORPREV	INV	STRETCH				EDSEL										SOURCESEL									SIGSEL		

Figura 78. Registro PRS_CH0_CTRL del EFR32

La elección de la señal Reflex se realiza con SIGSEL (bits 2:0) y dependerá del productor que hayamos seleccionado en SOURCESEL. En el caso de Timer 0 tenemos las opciones que vemos en la figura 79 entre las que escogeremos TIMER0OF que habilitará

al PRS cuando se produzca un overflow en el Timer. Por tanto, escribiremos un 0x01 en los bits 2:0 (SIGSEL).

SOURCESEL = 0b0011100 (TIMER0)		
0b000	TIMER0UF	Timer 0 Underflow TIMER0UF
0b001	TIMER0OF	Timer 0 Overflow TIMER0OF
0b010	TIMER0CC0	Timer 0 Compare/Capture 0 TIMER0CC0
0b011	TIMER0CC1	Timer 0 Compare/Capture 1 TIMER0CC1
0b100	TIMER0CC2	Timer 0 Compare/Capture 2 TIMER0CC2

Figura 79. SIGSEL para SOURCESEL=TIMER

4.4.4.5. Linked Direct Memory Access (LDMA)

El controlador LDMA nos permite realizar transferencias de datos sin la intervención de la CPU. Las transferencias pueden ser:

- Memoria a memoria
- Memoria a periférico
- Periférico a memoria
- Periférico a periférico

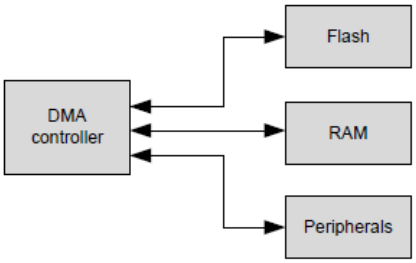

Figura 80. Controlador DMA

En nuestra aplicación vamos a utilizar el LDMA para transferir las muestras obtenidas por el ADC a un búfer *sampleBuffer* en memoria. Para llevar a cabo el proceso de posicionamiento de la placa se requieren 10000 muestras de la señal ultrasónica por lo que cada vez que se detecte la señal infrarroja, el LDMA deberá transferir esas 10000 muestras al búfer.

Los registros del LDMA utilizados son los siguientes:

- El registro LDMA_CHEN nos permite habilitar los distintos canales del DMA. Fijando a 1 el bit 0 habilitaremos el canal 0.

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x00				
Access																												RWH				
Name																												CHEN				

Figura 81. Registro LDMA_CHEN del EFR32

- Con el registro LDMA_IEN habilitaremos que se genere una interrupción cuando se complete la transferencia fijando a 1 el bit 0.

Offset	Bit Position																													
0x06C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0																											0x00				
Access	RW																											RW				
Name	ERROR																											DONE				

Figura 82. Registro LDMA_IEN del EFR32

- En el registro LDMA_CH0_REQSEL seleccionaremos la fuente para la transferencia de datos. Fijando el bit 19 a 1 seleccionaremos el ADC y fijando el bit 0 a 1 el modo SCAN del ADC.

Offset	Bit Position																													
0x080	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x00																0x0			
Access													RW																RW			
Name													SOURCESEL																SIGSEL			

Figura 83. Registro LDMA_CH0_REQSEL del EFR32

SOURCESEL 0x00 RW **Source Select**

Select input source to DMA channel.

Value	Mode	Description
0b000000	NONE	No source selected
0b000001	PRS	Peripheral Reflex System
0b001000	ADC0	Analog to Digital Converter 0
0b001100	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter 0
0b001101	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter 1
0b010000	LEUART0	Low Energy UART 0
0b010100	I2C0	I2C 0
0b011000	TIMER0	Timer 0
0b011001	TIMER1	Timer 1
0b110000	MSC	Memory System Controller
0b110001	CRYPTO	Advanced Encryption Standard Accelerator

Figura 84. Posibles señales fuente para LDMA

SOURCESEL = 0b001000 (ADC0)		
0b0000	ADC0SINGLE	ADC0SINGLE REQ/SREQ
0b0001	ADC0SCAN	ADC0SCAN REQ/SREQ

Figura 85. SIGSEL para SOURCESEL=ADC0

- Del registro LDMA_CH0_CTRL vamos a utilizar los siguientes bits:
  - DSTINC (bits 29:28) lo fijaremos a 0 para que la dirección destino, el búfer, se incremente 1 byte con cada transferencia de un dato.
  - SIZE (bits 27:26) lo fijaremos a 0 para que cada dato transferido sea de 1 byte.
  - SRCINC (bits 25:24) tendrá valor 0x3 para que la dirección origen no se incremente en cada transferencia.
  - DONEIFSEN (bit 20) fijado a 1 para que se active el flag de interrupción cuando se complete la transferencia de los datos.
  - XFERCNT (bits 14:4) indica el número de datos a transferir. El número máximo de datos que podemos transferir será de 2048 puesto que solo dispone de 11 bits. Como necesitamos transferir 10000 muestras, vamos a realizar 5 transferencias DMA de 2000 datos cada una por lo que XFERCNT tendrá el valor de 0x7CF (1999 en decimal ya que se incluye el 0).

Offset	Bit Position																															
0x08C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0x0		0x0		0x0		0	0	0	0	0x0				0	0x000											0		0x0	
Access	R	R	RWH		RWH		RWH		RWH	RWH	RWH	RWH	RWH				RWH	RWH											W1		R	
Name	DSTMODE	SRCMODE	DSTINC		SIZE		SRCINC		IGNORESREQ	DECLOOPCNT	REQMODE	DONEIFSEN	BLOCKSIZE				BYTESWAP	XFERCNT											STRUCTREQ		STRUCTTYPE	

Figura 86. Registro LDMA_CH0_CTRL del EFR32

- En el registro LDMA_CH0_SRC indicaremos la dirección de origen, que va a ser la dirección del registro ADC_SCANDATA (0x4000204C) que contiene el resultado de la conversión del ADC.

Offset	Bit Position																															
0x090	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x00000000															
Access																	RWH															
Name																	SRCADDR															

Figura 87. Registro LDMA_CH0_SRC del EFR32

- En el registro LDMA_CH0_DST indicaremos la dirección destino, que va a ser la dirección del búfer. Con cada transferencia de un dato, el registro aumentará en uno la dirección.

Offset	Bit Position																															
0x094	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x00000000															
Access																	RWH															
Name																	DSTADDR															

Figura 88. Registro LDMA_CH0_DST del EFR32

Como se ha comentado, para realizar las 10000 transferencias se van a realizar 5 transferencias LDMA de 2000 muestras cada una. Para ello deberemos inicializar el LDMA 5 veces cambiando la dirección destino de la transferencia.

La primera vez que se inicialice el LDMA, cuando se reciba la señal infrarroja, el registro LDMA_CH0_DST contendrá la dirección de la posición 0 del búfer *sampleBuffer*. Por cada transferencia de un byte que se realice, el valor del registro aumentará un byte apuntando cada vez a la siguiente posición del búfer hasta llegar a la 1999. Al comienzo de la segunda transferencia LDMA, el registro deberá contener la posición 2000 del búfer *sampleBuffer*, por lo que el valor del registro LDMA_CH0_DST deberá ser *sampleBuffer+*2000. Es decir, con cada transferencia se debe aumentar el valor del registro en 2000.

Para conseguir esto escribiremos *sampleBuffer+(2000*cnt)* en el registro LDMA_CH0_DST donde *cnt* es una variable cuyo valor inicial será 0 y aumentará en 1 con cada transferencia LDMA completa hasta un valor máximo de 4.

Para iniciar la transferencia de las 10000 muestras, se utiliza la función *MIC_start()*. Esta función habilita la transferencia LDMA periférico-memoria mediante la función *DMADRV_PeripheralMemory()* y el ADC con *adcEnable()*. Recibe como argumento la variable *cnt* para modificar la dirección inicial del destino de la transferencia.

```
void MIC_start(int cnt)
{
    /* Configure DMA */
    DMADRV_PeripheralMemory(dmadrvPeripheralSignal_ADC0_SCAN,     // señal de trigger
                            (void *) (sampleBuffer+(2000*cnt)),   // direccion destino
                            (void *) &(ADC0->SCANDATA),           // direccion origen
                            true,                                 // incrementar destino
                            2000,                                 // nº transferencias
                            dmadrvDataSize1,                      // tamaño 1 byte
                            dmaCompleteCallback);             // función de interrupcion
    adcEnable(true);

  return;
}
```

Código 17. Función *MIC_start()*

```
void adcEnable(bool enable)
{

  if (enable)
    timer->CMD = TIMER_CMD_START;

  else
    timer->CMD = TIMER_CMD_STOP;

  return;
}
```

Código 18. Función *adcEnable()*

Una vez completadas las 2000 transferencias, se activará la función de interrupción *dmaCompleteCallback()*, se aumentará en uno la variable *cnt* y se volverá llamar a la función *MIC_start()*. Este proceso se repetirá hasta que *cnt* tome el valor 5, momento en que se deshabilitará el ADC y se enviarán las 10000 muestras al módulo bluetooth a través de la USART0 mediante la función *USART_Tx()*.

```
static bool dmaCompleteCallback()
{

  cnt++;
  if(cnt == 5)
  {
      adcEnable(false);

      sampleBuffer = &micSampleBuffer[0];

      for(i=0;i<sampleBufferLen;i++)
          USART_Tx(Usart, micSampleBuffer[i] );

      sendsignal(false);
      cnt=0;
  }
  else
    MIC_start(cnt);

  return false;
}
```

Código 19. Función *dmaCompleteCallback()*

La función *sendsignal()* se utiliza para saber si se está desarrollando el proceso de localización o no. Cuando se recibe la señal infrarroja y se llama a la función *MIC_start()*, se activa un *flagS*, y cuando se terminan de enviar las 10000 muestras al módulo bluetooth, se desactiva.

```
void sendsignal(bool en)
{
  if(en)
    flagS = 1;
  else
    flagS = 0;
}
```

Código 20. Función *sendsignal()*

4.4.4.6. Sensor infrarrojo TSOP48

Un sensor infrarrojo es un dispositivo optoelectrónico capaz de medir la radiación electromagnética infrarroja de los cuerpos en su campo de visión. Esta

radiación resulta invisible para nosotros, pero no para estos dispositivos ya que se encuentran en el rango del espectro justo por debajo de la luz visible.

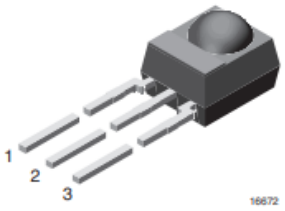

Figura 89. Sensor infrarrojo

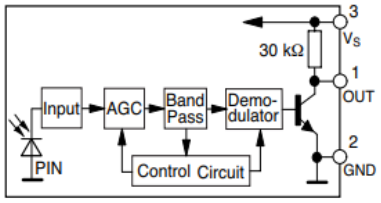

Figura 90. Diagrama de bloques

El sensor infrarrojo se alimentará a 3.3 V y será el encargado de indicar al ADC el momento exacto de iniciar el muestreo de las señales captadas por el micrófono. Cada vez que el sensor detecte una señal infrarroja, generará una interrupción que habilitará al ADC, iniciará el muestreo de la señal ultrasónica y activará el DMA para la transferencia de las muestras del ADC.

Las conexiones con la Thunderboard Sense son las que se ven en la figura 91. La placa proporcionará al sensor la alimentación a 3.3 V y la masa a través de los pines 18 y 1 de la placa.

La salida del sensor (OUT) se conectará a la placa a través del pin 5 de la misma. Este pin lleva asociado el pin PA3 del EFR32 que se ha configurado como entrada mediante la función *GPIO_PinModeSet()* y con interrupción activa por flanco de bajada mediante la función *GPIO_ExtIntConfig()*.

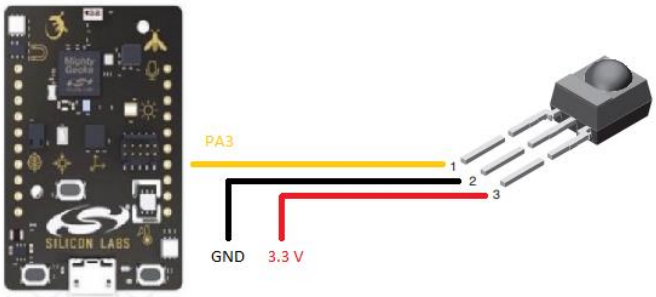

Figura 91. Conexiones del sensor infrarrojo

La rutina de atención a la interrupción provocada por el pin 5 de la placa será la función *GPIO_ODD_IRQHandler()* que atiende solo a los pines impares. Primero se borra el flag asociado a la interrupción, se activa el *flagS* para indicar que ha comenzado el proceso de localización y a continuación se llama a la función *MIC_start()* que habilitará al ADC y al LDMA.

```
void GPIO_ODD_IRQHandler(void)
{
  uint32_t iflags;

  /* Get all odd interrupts. */
  iflags = GPIO_IntGetEnabled() & 0x0000AAAA;

  /* Clean only odd interrupts. */
  GPIO_IntClear(iflags);

  GPIOINT_IRQDispatcher(iflags);

  sendsignal(true);
  MIC_start(0);
}
```

Código 21. Función *GPIO_ODD_IRQHandler()*

4.5. Envío y recepción de datos

4.5.1. Módulo bluetooth HC-06

El HC-06 es un módulo Bluetooth muy usado debido a su versatilidad y bajo coste. Solo puede funcionar como esclavo esperando peticiones de conexión, no puede iniciar una comunicación.

Figura 92. Modulo Bluetooth HC-06

El módulo está formado por 4 pines que se conectarán a la Thunderboard Sense del siguiente modo:

- Vcc: voltaje positivo de alimentación en un rango de 3.6V a 6V, se conectará al pin 18 (5V) de la placa
- GND: voltaje negativo de alimentación, se conectará al pin 1 (GND) de la placa
- TX: pin de transmisión, no se utilizará
- RX: pin de recepción, se conectará al pin 12 (TX) de la placa

El pin TX del HC-06 no se va a utilizar puesto que solo van a enviarse datos en una dirección desde la placa al módulo Bluetooth.

Como ya se ha comentado, la comunicación entre el módulo y la placa se hará a través de la USART0 a una velocidad de 460800 baudios, por tanto, debemos configurar el HC-06 a esa velocidad.

Para modificar las características del módulo se utilizan comandos AT. Por defecto el módulo viene configurado de la siguiente forma:

- Nombre: HC-06
- Código de emparejamiento: 1234
- Velocidad: 9600 baudios

El módulo dispone de dos estados diferentes, modo AT (desconectado) y modo conectado. Tan pronto como se alimenta el módulo y cuando no se ha establecido conexión bluetooth el HC-06 estará en modo AT. En este modo el led del módulo estará parpadeando y se pueden enviar comandos AT. Cuando se establece una conexión bluetooth, el HC-06 pasará a modo conectado. El led permanecerá encendido sin parpadear y los datos que se reciban a través del pin RX se enviarán vía bluetooth al dispositivo al que esté conectado y los que se reciban vía bluetooth se enviarán a través del pin TX. En este modo no se pueden utilizar los comandos AT.

Para enviar los comandos, se utilizará el Monitor Serie del programa Arduino IDE. El módulo se conectará a un PC a través de un conversor USB-Serial PL2303.

Figura 93. USB-Serial PL2303

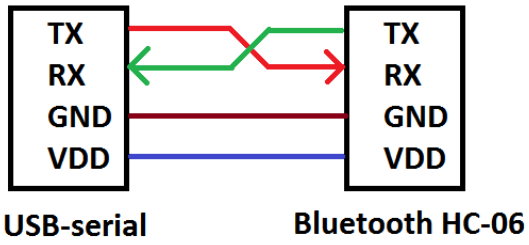

Figura 94. Conexión entre el HC-06 y el PL2303

Dado que por defecto el módulo está configurado a una velocidad de 9600 baudios, debemos elegir dicha velocidad en el Monitor Serie.

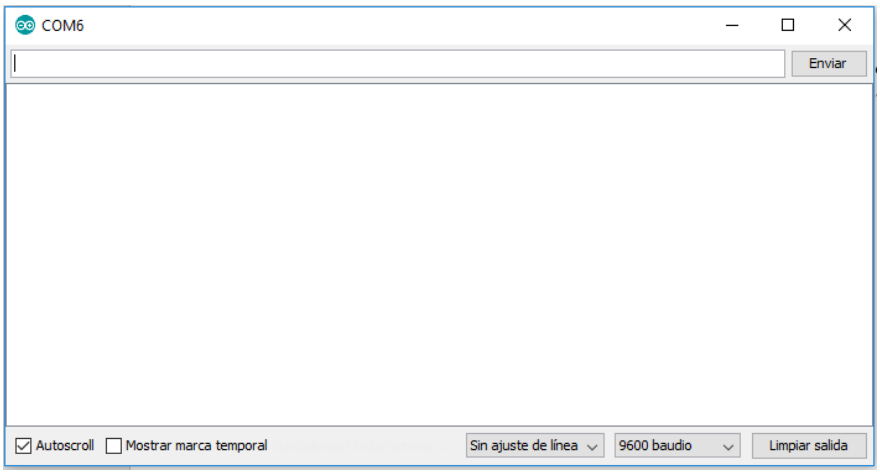

Figura 95. Ventana del Monitor Serie

Para comprobar que podemos comunicarnos con el módulo, escribiremos el comando *AT* a lo que el módulo deberá responder con un *OK*.

Para cambiar la velocidad de comunicación se utiliza el comando *AT+BAUD<numero>* donde *<numero>* equivale a la velocidad deseada como se ve a continuación:

<numero>	Velocidad
1	1200
2	2400
3	4800
4	9600
5	19200
6	38400
7	57600
8	115200
9	230400
A	460800
B	921600
C	1382400

Tabla 11. Valores de la velocidad de comunicación

Como queremos configurarlo a una velocidad de 460800 baudios, escribiremos el comando *AT+BAUDA* a lo que el módulo nos responderá con *OK*. Para comprobar que se ha configurado correctamente, cambiaremos la velocidad del Monitor Serie de 9600 a 460800 y enviaremos de nuevo el comando *AT* esperando como respuesta un *OK*.

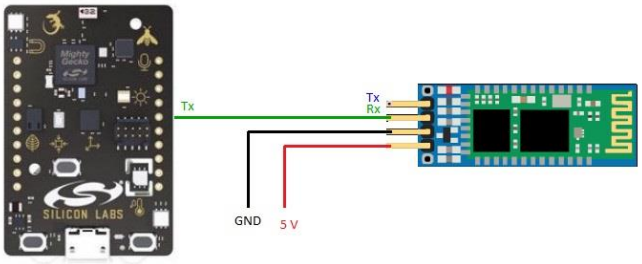

Figura 96. Conexión USART entre la Thunderboard Sense y el HC-06

4.5.2. Recepción de las muestras

El HC-06 enviará los datos vía bluetooth a un PC desde el que se calcularán la posición y orientación de la placa y se mostrarán utilizando el software Matlab en un gráfico 3D en tiempo real. Para ello se va a utilizar el siguiente código:

```
%% VARIABLES
x = [0,0];
y = [0,0];
z = [0,0];
flag1=0;
flag2=0;
Buffer_size=10000;
f=100e3;
c=343/f;
UMBRAL_AMPLITUD_BAL_MAX=1e7;
PORCENTAJE_AMPLITUD_BAL_MIN=0.10;
UMBRAL_MUESTRAS_MARGEN=400; %Diff de distancias máxima de 400*343/100e3=1.37m.
Teóricamente no puede dar una distancia mayor a 1m (digagonal de la
estructura)
MAX_DISTAHP=2;%2
MUESTRAS_SECUENCIA=1224;
MUESTRAS_REPETICION=380; %Muestras de espera entre la baliza 5 y la 1 de nuevo
%z=0;

%% ALGORITMO
load('set_kasami_255_s12_c2_TDMA_mod_down.mat');
set_kasami_mod_down=set_kasami_mod_down_tdma;

LPS=[0 0 2.672; ...
      0.51 0 2.818; ...
      0 0.53 2.730; ...
      -0.55 0 2.814; ...
      0 -0.50 2.764];

bt = Bluetooth ('Robot',1);
```

```
set(bt, 'InputBufferSize', Buffer_size,'TimeOut',8);

fopen(bt);
get(bt);

%% Prepare Figures

Ax(2) = axes('CameraPosition',[10 -10 10]);
grid;
hold on;
axis([-3 3 -3 3 -1 4]);

%% Read and plot the data from Arduino
i = 1;
T(i)=0;
CubH = [];
Angles = zeros(1,3);
X=[0 0 0]

while(1)
    if((flag1==1) )
        flag1=0;
        flag2=0;
        flushinput(bt);
       data1 = fread(bt, Buffer_size);
       data1 = (data1-mean(data1));


       corr(1,:)=abs(hilbert(conv(data1,fliplr(set_kasami_mod_down(1,:))))).^2;
       corr(2,:)=abs(hilbert(conv(data1,fliplr(set_kasami_mod_down(4,:))))).^2;
       corr(3,:)=abs(hilbert(conv(data1,fliplr(set_kasami_mod_down(5,:))))).^2;
       corr(4,:)=abs(hilbert(conv(data1,fliplr(set_kasami_mod_down(6,:))))).^2;
       corr(5,:)=abs(hilbert(conv(data1,fliplr(set_kasami_mod_down(7,:))))).^2;

       [maximos(1),pos(1)]=max(corr(1,:));
       [maximos(2),pos(2)]=max(corr(2,:));
       [maximos(3),pos(3)]=max(corr(3,:));
       [maximos(4),pos(4)]=max(corr(4,:));
       [maximos(5),pos(5)]=max(corr(5,:));

       [max_abs,pos_abs]=max(maximos);

       TDOA=inf*ones(1,5);

       if (max_abs>UMBRAL_AMPLITUD_BAL_MAX)
             UMBRAL_AMPLITUD_BAL_MIN=PORCENTAJE_AMPLITUD_BAL_MIN*max_abs;
             LPST=[];
             j=0;
             dista=[];
             for i=1:1:5
                 %Si el pico entra dentro del umbral, calculo su distancia y
                 %añado las coordenadas de la baliza correspondiente.
                 if maximos(i)>UMBRAL_AMPLITUD_BAL_MIN
                    j=j+1;
                    dista(j,1)=(pos(i)-MUESTRAS_SECUENCIA*i)*c;
                    LPST(j,:)=LPS(i,:);
                 end
             end
              %Obtengo la posición si conozco al menos 3 distancias
             if (length(dista)>2)
                 Y=X;
                 X=Gauss_Newton_esf_3d(LPST',dista,X);
```

```
            end

            if (X(1)>3 || X(1)<-3 || X(2)>3 || X(2)<-3 || X(3)>4 || X(3)<-1 ||
(X(1)==0 && X(2)==0 && X(3)==1.3360))

               X=Y;
            end

        end

        hold off;

     end

    sms = 0;
    idx = [];
    Angles = 0;

    while ((numel(Angles)~=3)&& (flag2==0))
        flushinput(bt);
        orient = fread(bt, 6);
        if((orient(1) == 's') && (orient(2) == 'i') && (orient(3) == 'g') &&
(orient(4) == 'n') && (orient(5) == 'a') && (orient(6) == 'l'))

            flag1=1;
            flag2=1;
        else

           hex1 = dec2hex(orient(1));
            hex2 = dec2hex(orient(2));
            hex = strcat(hex2,hex1);
            x = typecast(uint16(base2dec(hex, 16)), 'int16');
            Yaw = single(x);
            Angles(1) = Yaw;

            hex1 = dec2hex(orient(3));
            hex2 = dec2hex(orient(4));
            hex = strcat(hex2,hex1);
            y = typecast(uint16(base2dec(hex, 16)), 'int16');
            Pitch = single(y);
            Angles(2) = Pitch;

            hex1 = dec2hex(orient(5));
            hex2 = dec2hex(orient(6));
            hex = strcat(hex2,hex1);
            z = typecast(uint16(base2dec(hex, 16)), 'int16');
            Roll = single(z);
            Angles(3) = Roll;

        end
           CubH = Plot_Cube(deg2rad(Yaw),deg2rad(Pitch),deg2rad(-
Roll),CubH,X(1),X(2),X(3));

           drawnow;
           flushinput(bt);
    end
end
```

Código 22. Script de Matlab para representar posición y orientación de la placa

```
function CUBEH = Plot_Cube(yaw,pitch,roll,CUBEH,x,y,z)
if ~exist('CUBEH','var')
    CUBEH = [];
end
C1=[0 0 0];
C2=[1 1 1];
C3=[0 0 1];
C4=[0 1 0];
C5=[1 0 0];
C6=[1 1 0];

%Height
H = 1;
%Width
W = 1;
%Depth
D = 1;
R{3} = [1 0 0;0 cos(yaw) -sin(yaw);0 sin(yaw) cos(yaw)];
R{1} = [cos(pitch) 0 sin(pitch);0 1 0;-sin(pitch) 0 cos(pitch)];
R{2} = [cos(roll) -sin(roll) 0;sin(roll) cos(roll) 0;0 0 1];

    Vertex=[-D -W -H;
            +D -W -H;
            +D +W -H;
            -D +W -H;%Cambio
            -D -W +H;
            +D -W +H;
            +D +W +H;
            -D +W +H]/2;
    for rot = 1:3
        for v = 1:8
            Vertex(v,:) = Vertex(v,:)*R{rot};
        end
    end
    %Base
    idx1 = [1 2 3 4 1];
%     idx2 = [1 5 6 2 3 7 8 4];
%     idx3 = [5 6 7 8 5];
    %Culo
    idx2 = [5 6 7 8 5];
    %Lados
    idx3 = [1 5 6 2];
    idx4 = [2 6 7 3];
    idx5 = [3 7 8 4];
    idx6 = [4 8 5 1];

    if isempty(CUBEH)
        CUBEH(1) = fill3(Vertex(idx1,1),Vertex(idx1,2),Vertex(idx1,3),C1);
        CUBEH(2) = fill3(Vertex(idx2,1),Vertex(idx2,2),Vertex(idx2,3),C2);
        CUBEH(3) = fill3(Vertex(idx3,1),Vertex(idx3,2),Vertex(idx3,3),C3);
        CUBEH(4) = fill3(Vertex(idx4,1),Vertex(idx4,2),Vertex(idx4,3),C4);
        CUBEH(5) = fill3(Vertex(idx5,1),Vertex(idx5,2),Vertex(idx5,3),C5);
        CUBEH(6) = fill3(Vertex(idx6,1),Vertex(idx6,2),Vertex(idx6,3),C6);
        alpha(0.4);
    else

set(CUBEH(1),'XData',x+Vertex(idx1,1),'YData',y+Vertex(idx1,2),'ZData',z+Verte
x(idx1,3));

set(CUBEH(2),'XData',x+Vertex(idx2,1),'YData',y+Vertex(idx2,2),'ZData',z+Verte
x(idx2,3));
```

```
set(CUBEH(3),'XData',x+Vertex(idx3,1),'YData',y+Vertex(idx3,2),'ZData',z+Verte
x(idx3,3));

set(CUBEH(4),'XData',x+Vertex(idx4,1),'YData',y+Vertex(idx4,2),'ZData',z+Verte
x(idx4,3));

set(CUBEH(5),'XData',x+Vertex(idx5,1),'YData',y+Vertex(idx5,2),'ZData',z+Verte
x(idx5,3));

set(CUBEH(6),'XData',x+Vertex(idx6,1),'YData',y+Vertex(idx6,2),'ZData',z+Verte
x(idx6,3));
    end
end
```

Código 23. Script de Matlab de la función *Plot_Cube*

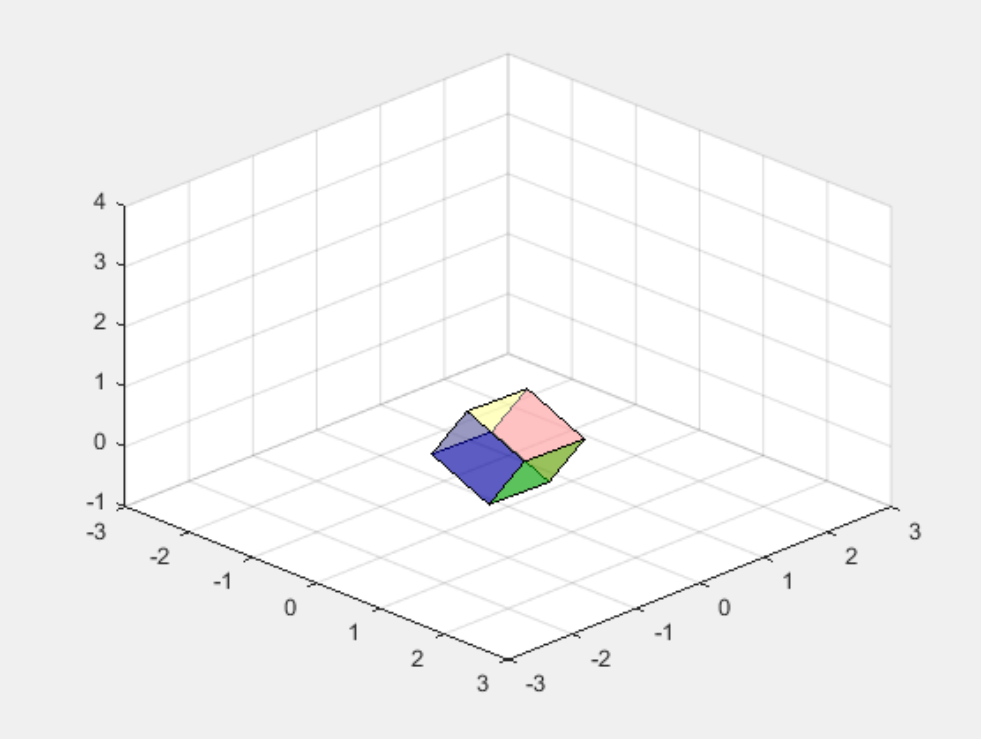

Figura 97. Ejemplo de una muestra de orientación y posición obtenidas

5. Desarrollo del programa

En este apartado se va a explicar el funcionamiento global de la aplicación que nos ofrecerá en tiempo real la posición y orientación de la placa en una habitación.

Una vez alimentemos la Thunderboard Sense, lo primero que haremos será establecer la comunicación bluetooth entre el módulo HC-06 y el PC. Cuando el led del módulo deje de parpadear y se quede encendido la comunicación se habrá establecido.

El programa puede dividirse en dos etapas, la de obtención y envío de los valores de orientación y la de obtención y envío de los valores de posición. Para este proyecto se busca la mayor precisión en la representación de la orientación por lo que la aplicación estará constantemente midiendo y representado los datos relativos a la orientación de la placa. La posición se actualizará cuando se reciba el pulso de la señal infrarroja cuyo periodo es configurable.

Cabe recordar que cuando el sensor inercial obtiene una nueva muestra, genera una interrupción a través de su pin INT que está conectado al EFM8. Cuando el EFM8 registra que se ha producido una interrupción, lo notifica al EFR32 a través del pin INT/WAKE. Por lo tanto, para obtener los datos de orientación el programa leerá continuamente el valor del pin INT/WAKE a través de la función *IMU_isDataReadyFlag()* que devuelve true si el pin se encuentra a nivel bajo. Cuando esto ocurra, se borrará el flag de la interrupción asociada al IMU del registro INT_CLEAR y se procederá al cálculo de los ángulos de navegación mediante la función *IMU_update()* para su posterior envío a través del HC-06.

Cuando se recibe el pulso de la señal infrarroja, se activa un *flagS* para que deje de desarrollarse el procedimiento de orientación y comience el de posicionamiento. Se enviará un código de 6 bytes (*"signal"*) contenido el búfer *buff* para indicarle al receptor que se van a enviar los 10000 bytes para obtener la posición en lugar de los 6 bytes de orientación. A continuación, se realiza el proceso de obtención de las 10000 muestras de las señales emitidas por las balizas ultrasónicas y se envían a través del HC-06.

En el momento en que terminen de enviarse las muestras, se desactivará el *flagS* por lo que se realizará de nuevo la etapa de orientación hasta que se reciba un nuevo pulso de la señal infrarroja.

En el receptor se estará calculando y representando la orientación de la tarjeta a partir de los 6 bytes recibidos. Cuando se reciba el código de 6 bytes "*signal"*, se cambiará el búfer de recepción a 10000 bytes y se calculará y representará la posición obtenida. Se cambiará de nuevo el búfer de recepción a 6 bytes y se calculará y representará la orientación hasta que se reciba de nuevo el código *"signal"*.

```
while (1)
{
    if(flagS==1 && flag==0)
    {
      flag=1;
      for(k=0;k<6;k++)
            USART_Tx(usart0, buff[k] );
    }

    if(flagS==0)
    {
       flag3=0;
       if( IMU_isDataReadyFlag() )
        {
            BOARD_picWriteReg( BOARD_PIC_REG_INT_CLEAR, 0 );
            IMU_update();
            sendOri(usart0);
        }
    }
}
```

Código 24. Función principal de la aplicación

```
bool IMU_isDataReadyFlag( void )
{
   bool ready;
   unsigned int pin;

      pin = GPIO_PinInGet( BOARD_PIC_INT_WAKE_PORT, BOARD_PIC_INT_WAKE_PIN );
      if( pin == 0 ) {
         ready = true;
   }
   return ready;
}
```

Código 25. Función IMU_isDataReady()

```
void IMU_update( void )
{

   IMU_getAccelerometerData( accel );
   IMU_getGyroData( gyro );

   return;

}
```

Código 26. Función IMU_update()

6. Análisis de costes

En este apartado se analizará el coste estimado del proyecto en cuanto a hardware, software y mano de obra.

6.1. Coste total de equipo y software

Concepto	Precio	Periodo de amortización (meses)	Uso del producto (meses)	Coste para el proyecto
Ordenador portátil (incluye S.O.)	699€	48	6	87.38€
IAR Embedded Workbench IDE (versión de evaluación)	0€	N/A	N/A	0€
Microsoft Office 365 (licencia universitaria)	0€	48	6	0€
Licencia Matlab (licencia universitaria)	0€	N/A	N/A	0€
Coste total de equipo y software				87.38€

Tabla 12. Desglose del coste de equipo y software

6.2. Coste de material

Concepto	Precio	Unidades	Coste para el proyecto
Tarjeta de trabajo Thunderboard Sense	32€	1	32€
Adaptador USB-Serie PL2303	2€	1	2€
Módulo bluetooth HC-06	8€	1	8€
Sensor infrarrojo TSOP48	1.5€	1	1.5€
Power Bank	6€	1	6€
Otros componentes (conectores, resistores..)	N/A	N/A	5€
Coste total del material			54.5€

Tabla 13. Desglose del coste de material

6.3. Coste de mano de obra

Concepto	Categoría	€/hora	Total horas	Coste para el proyecto
Carlos Alonso Lucea	Ingeniero Junior	20€	240	4800€
Coste total del material				4800€

Tabla 14. Desglose del coste de mano de obra

6.4. Presupuesto de ejecución material

Concepto	Precio
Coste de equipo y software	87.38€
Coste material	54.5€
Coste de mano de obra	4800€
Total	4941.88€

Tabla 15. Desglose del presupuesto de ejecución material

6.5. Presupuesto de contrata

Concepto	Precio
Presupuesto de ejecución material	4941.88€
Gastos generales (17%)	840.12€
Beneficio industrial (6%)	296.51€
Total	6078.51€

Tabla 16. Desglose del presupuesto de contrata

6.6. Presupuesto total

Concepto	Precio
Presupuesto de contrata	6078.51€
I.V.A. (21%)	1276.49€
Presupuesto Total	7355.0€

Tabla 17. Desglose del presupuesto total

7. Conclusiones y trabajo futuro

El objetivo principal de este proyecto era estudiar la tarjeta Thunderboard Sense y crear una aplicación capaz de representar la orientación y posición de la tarjeta utilizando elementos de la misma.

Los objetivos alcanzados son los siguientes:

- Comprensión del funcionamiento de la Thunderboard Sense así como de los sensores que incluye
- Utilización de distintas interfaces de comunicación
- Obtención de la orientación de un objeto a partir de un sensor IMU
- Obtención de la posición de un objeto a partir de balizas ultrasónicas
- Envío de datos mediante un módulo bluetooth

Aunque el resultado final ha sido satisfactorio, se podrían realizar algunas mejoras en el futuro:

- Utilización de la tecnología Bluetooth de Baja Energía para el envío de datos. En un principio la idea era utilizar esta tecnología para enviar las muestras de la tarjeta al PC evitando el uso de un módulo bluetooth externo. Al final se rechazó esta idea debido a que el software utilizado para la recepción de datos en el PC, Matlab, no reconocía los dispositivos BLE siendo imposible establecer una conexión entre ellos.
- Añadir un magnetómetro. Añadiendo un magnetómetro para calcular el ángulo Yaw se podría obtener de manera más precisa la orientación de la tarjeta
- Utilización de otros sensores de la Thunderboard Sense. Ahora que se conoce el funcionamiento de la tarjeta, se obtener distintos datos del entorno

8. Bibliografía

[1] Thunderboard Sense User´s Guide: https://www.silabs.com/documents/public/user-guides/ug250-tb001-user-guide.pdf

[2] Reference Manual EFR32: https://www.silabs.com/documents/public/reference-manuals/efr32xg1-rm.pdf

[3] Reference Manual EFM8: https://www.silabs.com/documents/public/reference-manuals/EFM8UB2-RM.pdf

[4] Datasheet ICM-20648: https://www.google.com/search?q=icm-20648+datasheet&oq=icm+20648+datashee&aqs=chrome.1.69i57j35i39j0l2.11273j1j4&sourceid=chrome&ie=UTF-8

[5] Datasheet SPV1840: https://www.mouser.es/datasheet/2/218/pv1840lr5h-b-rev-b-datasheet-1510007.pdf

[6] Medir la inclinación con IMU: https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/

[7] Utilizar un acelerómetro en proyectos de Arduino: https://www.luisllamas.es/como-usar-un-acelerometro-arduino/

[8] Utilizar un giroscopio en proyectos de Arduino: https://www.luisllamas.es/como-usar-un-giroscopio-arduino/

[9] Tutorial del MPU-6050: https://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/

[10] Protocolo USART: http://www.coffeebrain.org/wiki/index.php?title=USART_B%C3%A1sico

[11] Introduction to SPI and I2C protocols: https://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/

[12] García Polo E. M.; TÉCNICAS DE LOCALIZACIÓN EN REDES INALÁMBRICAS DE SENSORES; Universidad de Castilla-La Mancha.

[13] Técnicas de localización basadas en medidas de rango: http://bibing.us.es/proyectos/abreproy/12028/fichero/04_T%C3%A9cnicas+de+localizaci%C3%B3n+basadas+en+medidas+de+rango.pdf

[14] Rugeles J.; TÉCNICAS DE LOCALIZACIÓN DE NODOS INALÁMBRICOS MEDIANTE REDES DE SENSORES; Universidad Militar Nueva Granada

[15] Arango Vargas J. F.; DISEÑO DE UN MÓDULO RECEPTOR PARA UN LPS ULTRASÓNICO; Trabajo fin de grado, Universidad de Alcalá, 2016

[16] Lluva Plaza S.; SISTEMA REMOTO DE ADQUISICIÓN DE VARIABLES METEOROLÓGICAS CON MONITORIZACIÓN EN UN ENTORNO WEB; Trabajo fin de grado, Universidad de Alcalá, 2018

9. Anexo

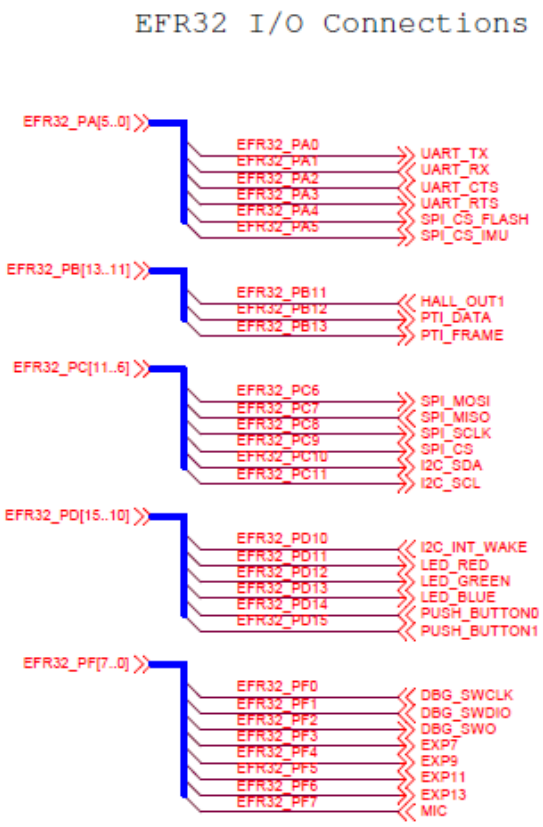

Figura 97. Esquemático de los pines GPIO del EFR32

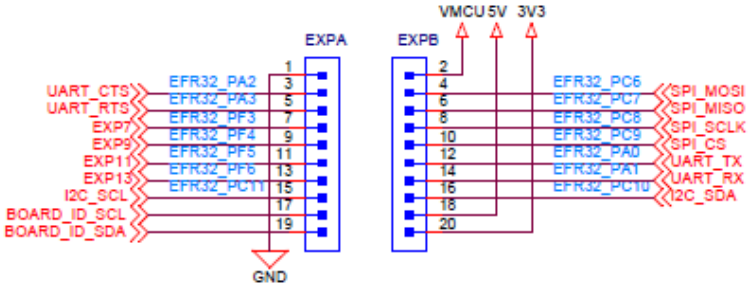

Figura 98. Esquemático de los pines de la Thunderboard Sense

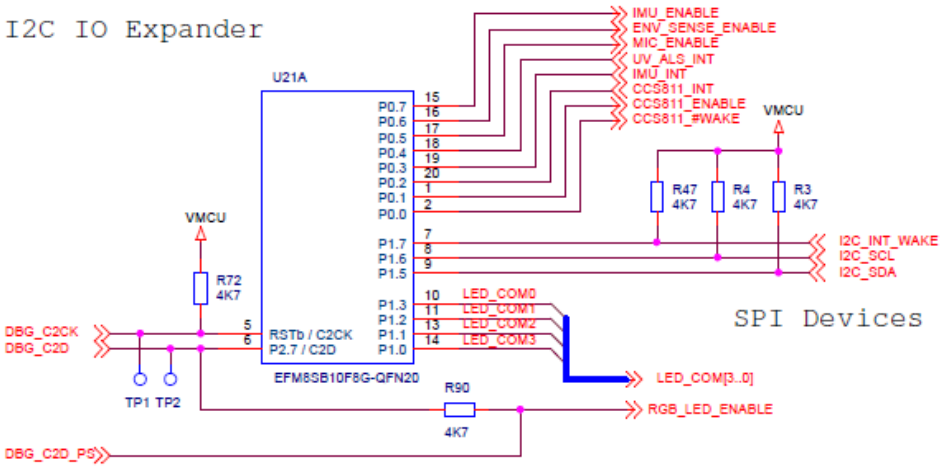

Figura 99. Esquemático del Controlador de Alimentación e Interrupción

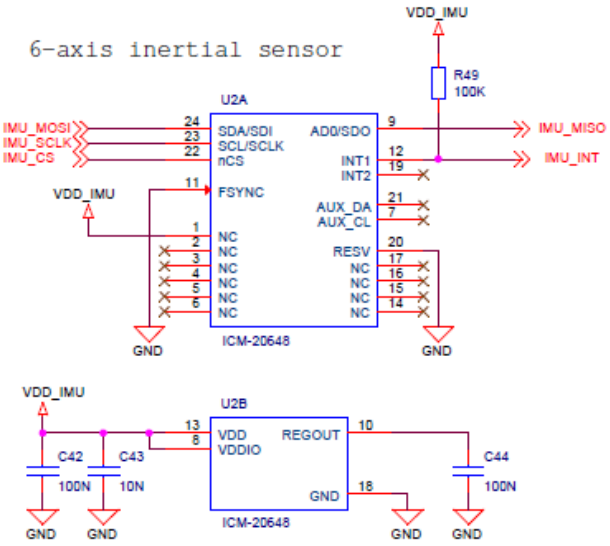

Figura 100. Esquemático del sensor inercial ICM-20648

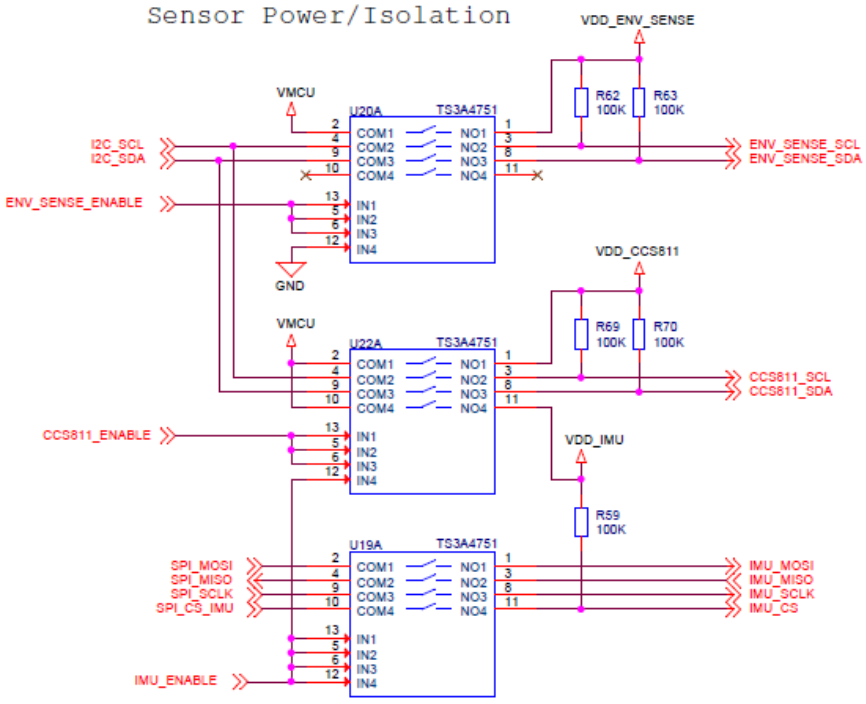

Figura 101. Esquemático del circuito habilitador de los sensores Si11133, CCS811 e ICM-20648

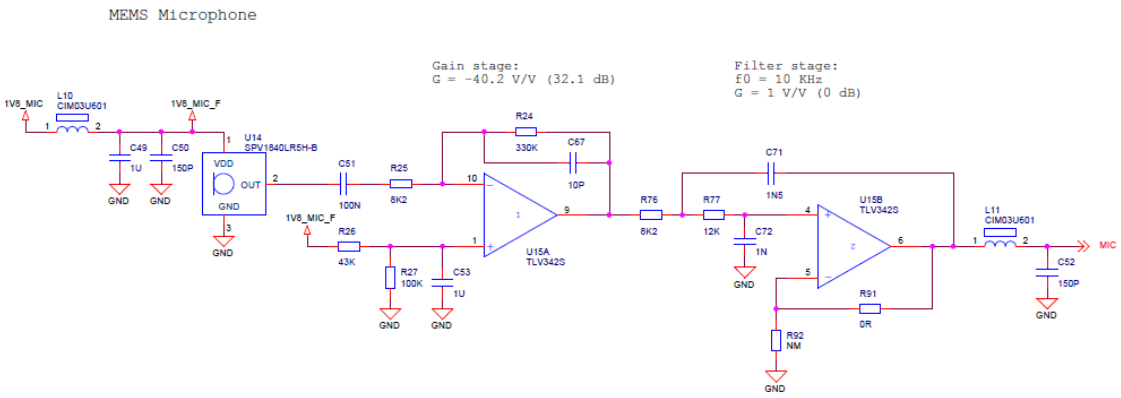

Figura 102. Esquemático del micrófono MEMS

Universidad de Alcalá