Universidad de Alcalá

Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Diseño de una baliza para un ULPS basada en el módulo WiFi EMW3162

Autor: Javier Cebrián Sanz

Tutor: José Manuel Villadangos Carrizo

2018

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Diseño de una baliza para un ULPS basada en el módulo WiFi EMW3162

Autor: Javier Cebrián Sanz

Tutor: José Manuel Villadangos Carrizo

TRIBUNAL:

Presidente: D. Jesús Ureña Ureña

Vocal 1º: D. Álvaro Hernández Alonso

Vocal 2º: D. José Manuel Villadangos Carrizo

FECHA: 18-ENE-2018

AGRADECIMIENTOS

A los que siguen confiando en mí.

RESUMEN

Este proyecto se desarrolla en el ámbito de los sistemas ULPS (Ultrasound Local Positioning System). Se ha diseñado un prototipo de baliza formada por 5 transductores ultrasónicos. Ésta, permite ubicar un objeto en un entorno, a partir de las posiciones absolutas donde se encuentra ubicada la misma. Particularmente, este diseño, permite modificar diversos parámetros de modulación de señal. Estos parámetros son accesibles a través de una interfaz web. Ésta, se ha conseguido gracias a la integración de un módulo wifi EMW3162 en el sistema de balizamiento, y a la implementación de un servidor web en el mismo. Se han usado TDMA (Time Division Multiple Access) como técnica de acceso al medio, BPSK (Binary Phase-Shift Keying) como base de la modulación, la librería mxchipWNet como base de la programación y la arquitectura Cortex-M3 como núcleo del firmware. Este documento, además, expone las posibilidades de funcionamiento que ofrece el módulo wifi mencionado.

SUMMARY

This project is developed in the field of ULPS systems (Ultrasound Local Positioning System). A beacon prototype consisting of 5 ultrasonic transducers has been designed. This allows you to locate an object in an environment, from the absolute positions where it is located. In particular, this design allows to modify various signal modulation parameters. These parameters are accessible through a web interface. This has been achieved due to the integration of an EMW3162 wifi module in the beacon system, and the implementation of a web server in it. TDMA (Time Division Multiple Access) has been used as channel access method, BPSK (Binary Phase-Shift Keying) as the basis of the modulation, the mxchipWNet library as a basis for programming and the Cortex-M3 architecture as the core of the firmware. This document also explains the possibilities of operation offered by the aforementioned wifi module.

PALABRAS CLAVE

ULPS. LPS. Módulo wifi. Servidor web. EMW3162.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ABREVIATURAS

***.bin:** Archivo binario
ACF: Auto-Correlation Function (Función de autocorrelación)
AO: Amplificador Operacional
AP: Acces Point (Punto de acceso wifi)
API: Application Programming Interface (Interfaz de programación de aplicaciones)
BPSK: Binary Phase-Shift Keying (Modulación binaria por desplazamiento de fase)
CCF: Cross Correlation Function (Función de correlación cruzada)
CDMA: Code Division Multiple Access (Acceso múltiple por división en el código)
CSS: Complementary Set of Sequences (Conjunto de secuencias complementarias)
DAC: Digital-to-Analog Converter (Convertidor analógico-digital)
DBPSK: Differential Binary Phase-Shift Keying (Modulación binaria diferencial por desplazamiento de fase)
DHCP: Dinamyc Host Configuration Protocol (Protocolo de configuración dinámica de host)
DQPSK: Differential Quadrature Phase-Shift Keying (Modulación en cuadratura diferencial por desplazamiento de fase)
DTDV: Diferencia de Tiempos De Vuelo
DIP: Dual In-Line Package
FDMA: Frecuency Division Multiple Access (Acceso Múltiple por División de Frecuencia)
FPGA: Field Programmable Gate Array (Matriz de puertas programables por campo)
FW: Firmware
HW: Hardware
GO: Generalized Orthogonal codes (Códigos ortogonales generalizados)
IP: Internet Protocol (Protocolo de internet)
I/O: Input/Output (Entrada/Salida)
JTAG: Joint Test Action Group
LPS: Local Positioning System (Sistema de Posicionamiento Local)
MAC: Media Acces Control (Control de Acceso al Medio)
Mbps: Megabits/segundo
MCU: Micro Controller Unit (Microcontrolador)
OTA: Over The Air
OS: Operative System (Sistema operativo)
PB9: Conector serie de 9 pines
PR: Pseudo-Random sequence (Secuencia pseudo-aleatoria)
PSD: Power Spectral Density (Densidad espectral de potencia)
QPSK: Quadrature Phase-Shift Keying (Modulación en cuadratura por desplazamiento de fase)
RAM: Random Acces Memory (Memoria de acceso aleatorio)
RF: Radio Frequency (Radio Frecuencia)
RTOS: Real-Time Operating System (Sistema operativo en tiempo real)
SDIO: Secure Digital Input Output
SSID: Service Set Identifier (Servicio de identificación)
SSL: Secure Sockets Layer (Capa de puertos seguros)
ST: Station (Equipo conectado a un punto de acceso)
SWD: Serial Wire Debug
TDMA: Time Division Multiple Access (Acceso Múltiple por división de Tiempo)
TDV: Tiempo de Vuelo
TIM2: Timer 2
TIM3: Timer 3
TIM4: Timer 4

TLS: Transport Layer Security (Capa de transporte segura)
UART: Universal Asynchronous Reciever-Transmitter (Transmisor-Receptor asíncrono universal)
UAV: Unmanned Aerial Vehicle (Vehículo aéreo no tripulado)
USB: Universal Serial Bus (Bus Universal Serie)
ULPS: Ultrasound Local Positioning System (Sistema de posicionamiento local por ultrasonidos)
WICED: Wireless Connectivity for Embedded Devices
WLAN: Wireless Local Area Network (Red de Área Local Inalámbrica)
WEP: Wired Equivalent Privacy
WPA: Wifi Protected Access (Acceso Wifi Protegido)
WPA2: Wifi Protected Access (Acceso Wifi Protegido 2ª generación)
WPS: Wifi Protected Setup
XTAL: Oscilador

1.-INTRODUCCIÓN

Este proyecto describe y desarrolla la construcción de un prototipo para implementar el balizamiento de un sistema de posicionamiento local basado en ultrasonidos, ULPS (Ultrasound Local Positioning System). De aquí en adelante, será llamado "baliza" al sistema que aquí se expone. Esta baliza, formada por 5 transductores, emite de forma continua señales ultrasónicas codificadas. Gracias a estas señales, será calcular la posición de un objeto ubicado dentro de la zona de cobertura de la baliza.

Ha sido elegido un módulo wifi (EMW3162) para integrarlo en el diseño. Esto permitirá a la baliza, entre otras cosas, una comunicación vía wifi con un equipo (Tablet, móvil, etc.). Además de esta comunicación, el módulo wifi EMW3162 se encargará de los procesos que tendrían que ver con un sistema ULPS sin comunicación wifi. La frecuencia de la señal modulada será de 40 kHz o de 41.66 kHz (seleccionable), pasando de cualquier forma, del límite auditivo (20 kHz).

Uno de los objetivos, de este proyecto, es realizar un análisis del módulo wifi EMW3162, necesario para conseguir lo anterior comentado.

Y el objetivo principal de este proyecto es crear una interfaz web que, de forma inalámbrica, permita realizar modificaciones de algunos parámetros de emisión de señal del sistema ULPS. Esto se consigue gracias a desarrollar un servidor web que dará acceso a la interfaz web. El módulo wifi se ocupará además de la generación de las señales de los transductores deben emitir.

Para conseguir conseguir los objetivos comentados, en lo que resta de este primer capítulo, se llevará a cabo una breve introducción de los sistemas ULPS para acabar concretando sobre las particularidades del diseño de la baliza. En el capítulo 2 se hablará del desarrollo hardware (HW) acometido. En el capítulo 3 se elabora un análisis del módulo wifi EMW3162: Sus características, ejemplos firmware (FW) desarrollados por el fabricante y posibles configuraciones. A continuación, el capítulo 4, expone la programación que ha sido necesaria y, por último, los capítulos 5, 6 y 7 mostrarán el diagrama de navegación web resultado de la programación, conclusiones y posibles mejoras del prototipo.

Como se ha comentado, se entiende por "baliza", al conjunto de elementos que conforman este diseño (módulo wifi, soportes físicos, transductores, cableado, etc.) y, además esto, se hablará de "objeto" como el elemento del que se quiera calcular su posición.

Quedará excluido de este proyecto el desarrollo de todo lo que no sea la parte de balizamiento y/o emisión de señal del sistema ULPS.

1.1.-Los sistemas de posicionamiento local por ultrasonidos

Se entiende por ULPS a un sistema de posicionamiento que generalmente opera en un área

reducida y tiene tecnología de ultrasonidos. Este tipo de tecnología suele encontrarse en espacios interiores, ya que su rango de cobertura no sobrepasa las decenas de metros.

Para calcular la posición de un objeto se miden tiempos de vuelo (TDV). Este TDV es el tiempo que tarda en viajar la señal desde uno de los transductores de la baliza hasta el receptor (objeto). En función de estos TDV o de la diferencia entre los TDV, llamada diferencia de tiempos de vuelo (DTDV) se puede calcular la posición del objeto.

La caracterización de los ULPS puede ser por: Tipo de posicionamiento, parte que calcula la posición al objeto, sincronismo entre sistema emisor y sistema receptor de señal y técnicas de acceso al medio. Estas características influyen directamente en el desarrollo de un sistema ULPS.

Si se atiende al método de posicionamiento puede ser relativo o absoluto. Un posicionamiento relativo es capaz de obtener la posición de un objeto sin tener puntos de referencias fijos. Un posicionamiento absoluto, en cambio, es capaz de calcular la posición de un objeto respecto una serie de referencias conocidas, estas referencias suelen ser los transductores que tenga la baliza.

Todo sistema ULPS debe procesar la información (señales) con la que trabaje. Si se atiende a la parte que procese esta información, se pueden clasificar los ULPS como sistemas centralizados o descentralizados. Un ULPS centralizado concentra su capacidad de cálculo en la propia baliza. Este sistema captaría una única señal proveniente de un único emisor, que en este caso sería el objeto (este sistema se presenta en la figura 1.1-1, parte derecha). En cambio, en un sistema descentralizado, el objeto, en este caso la parte que va a recibir la señal de la baliza, realiza el cálculo de posicionamiento (figura 1.1-1, parte izquierda).

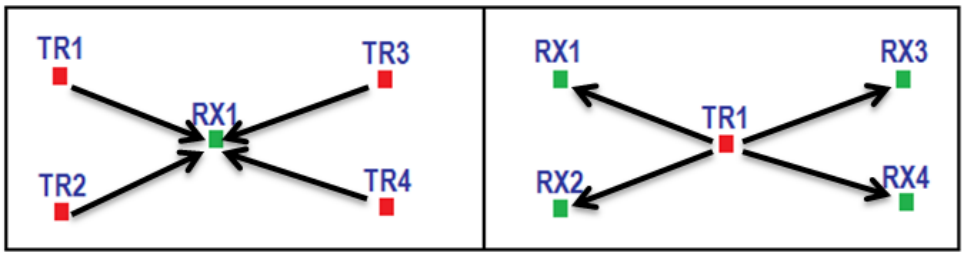


Figura 1.1-1: Izquierda: ULPS descentralizado. Derecha: ULPS centralizado.

En la figura anterior, TRn representa un transductor que transforma la energía eléctrica en ultrasonidos y RXn representa un transductor que transforma los ultrasonidos en energía eléctrica. El sentido de las flechas negras representa el sentido desde donde se emite una señal hacia donde se recibe.

Para calcular la posición de un objeto en función de tiempos de vuelo se puede diferenciar entre sistemas ULPS síncronos y asíncronos. Los ULPS síncronos tienen receptor y emisor en sincronismo, de esta forma se establece un punto de partida para calcular esos espacios de tiempo para calcular la posición. La señal de sincronismo típica es radiofrecuencia. La figura 1.1-2 representa en naranja la señal de sincronismo.

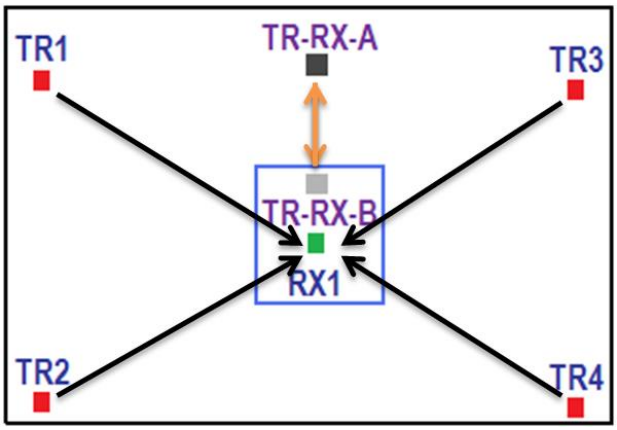


Figura 1.1-2: Ejemplo de sistema ULPS síncrono.

Los ULPS pueden tener transductores emitiendo de forma simultánea o no. Generalmente, la señal que emite cada transductor se llama secuencia. De esta forma se pueden utilizar distintas formas de acceder a la información presente en la señal que emita la baliza.

Existen diversas técnicas de acceso al medio para los sistemas ULPS. Estas técnicas son necesarias para acceder a la señal que el receptor recoge. Aplicado a los ULPS, las más comunes utilizadas son CDMA (Code Division Multiple Access), donde cada secuencia tiene asignado un código que será de gran importancia en caso de emitir simultáneamente a través de varios transductores, FDMA (Frecuency Division Multiple Access) que asigna una banda de frecuencia a cada secuencia de forma que también permite emitir de forma simultánea a varios transductores y TDMA (Time Division Multiple Access), en la que existe un único canal para emitir durante un cierto tiempo cada una de las secuencias. En esta última técnica es necesario demultiplexar el canal para que emita cada transductor su propia secuencia. En [PoCS, 1986] se puede encontrar un análisis de estas técnicas más en profundidad.

La figura 1.1-3 representa la técnica de acceso al medio TDMA de un sistema con un canal transmisor y n canales de usuario. El canal de transmisión contiene toda la información que será demultiplexada y enviada a través de los canales de usuario. A cada canal de usuario, junto con su transmisor (TR1, TR2, TRn), le corresponde un espacio de tiempo en el que tendrá acceso a los datos (D1, D2, Dn) del canal de transmisión. Posteriormente, toda la información, será recibida por el canal de recepción por un receptor (R).

Si se habla de un sistema ULPS con TDMA, y se hace referencia a la figura 1.1-3, a pesar de que los datos en el canal de recepción serán los mismos que en el canal de emisión, los espacios de tiempo entre los datos en ambos canales varían. Esto se debe a los tiempos de vuelo, ya comentados, que tardan las señales en viajar desde un transductor (TR1, TR2, TRn) hasta el receptor (R). En la figura 1.1-3 se representan estas diferencias de tiempos entre datos (D1, D2, Dn) en ambos canales. Estos datos, en un sistema ULPS pueden representar las señales o secuencias concretas emitidas por cada transductor.

También la figura 1.1-3 representa en verde las señales o secuencias que van desde un transductor al receptor.

La técnica TDMA será usada en el diseño de la baliza. Y será presentada en el apartado 1.2.3 junto con otros conceptos de importancia.

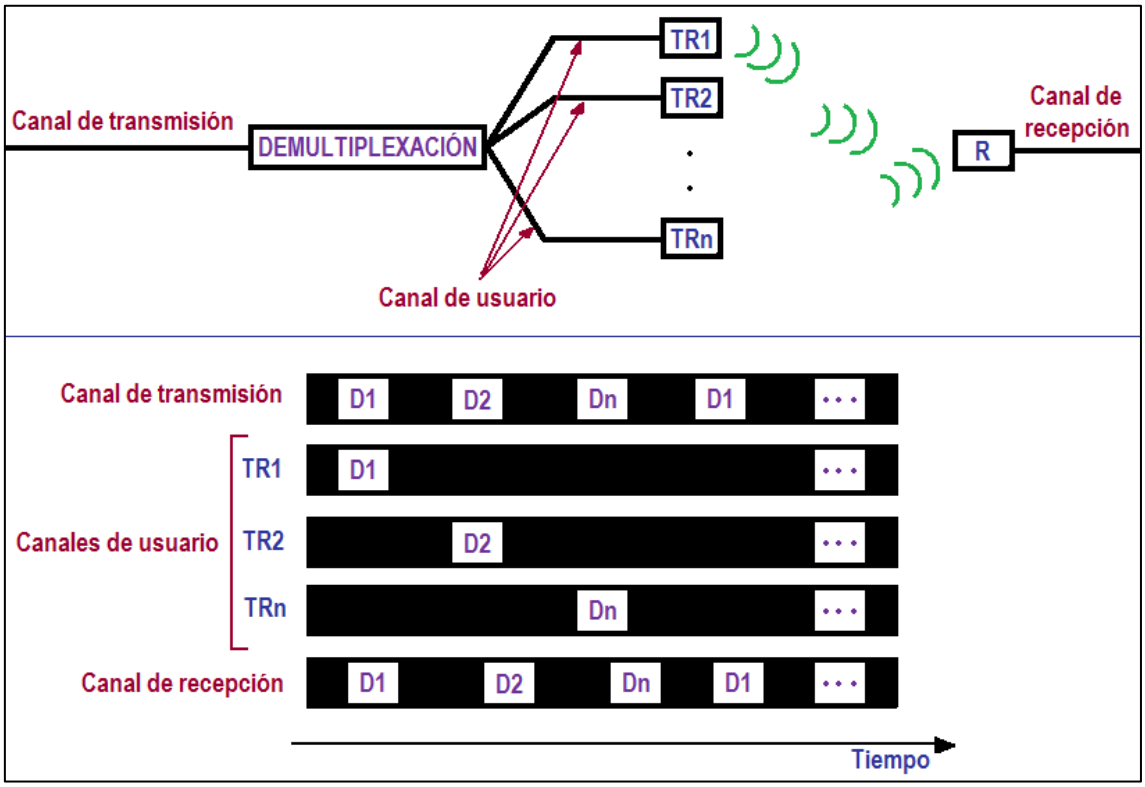


Figura 1.1-3: TDMA para un sistema ULPS.

Para el diseño de la baliza, se ha elegido un sistema de referencias absolutas, asíncrono y descentralizado. Por tanto, el objeto debe recibir las señales que emita la baliza y procesarlas para calcular su posición mediante técnicas de posicionamiento basadas en DTDV. Para calcular esta posición, el receptor debe emplear trilateración hiperbólica. En [J.F. Arango, 2016] se puede encontrar una concisa explicación sobre trilateración hiperbólica. En este trabajo, se explica el número necesario de referencias para calcular el posicionamiento en 3D, así como la ventaja de utilizar este método por no necesitar señal de sincronismo. Para que el receptor pueda cumplir con este método de posicionamiento se utilizan, como ya se ha comentado, 5 transductores en la baliza, que además actuarán como referencias de la misma. En la figura 1.1-4 se muestra una representación del diseño de baliza elegido.

El medio de propagación de la señal será el aire. Otros factores que afectan a la velocidad de propagación de la señal, a parte del medio de propagación son la temperatura y la humedad. Aspectos que deberá tener en cuenta el receptor.

Otros aspectos a tener en cuenta en cuenta en los sistemas ULPS son el HW típico usado, la modulación de señal y los tipos de codificación que permitirán, entre otras cosas, garantizar que la información no interfiera con la de otras señales. Estos aspectos serán tratados a continuación de forma general, y haciendo alusiones al proyecto donde sea necesario.

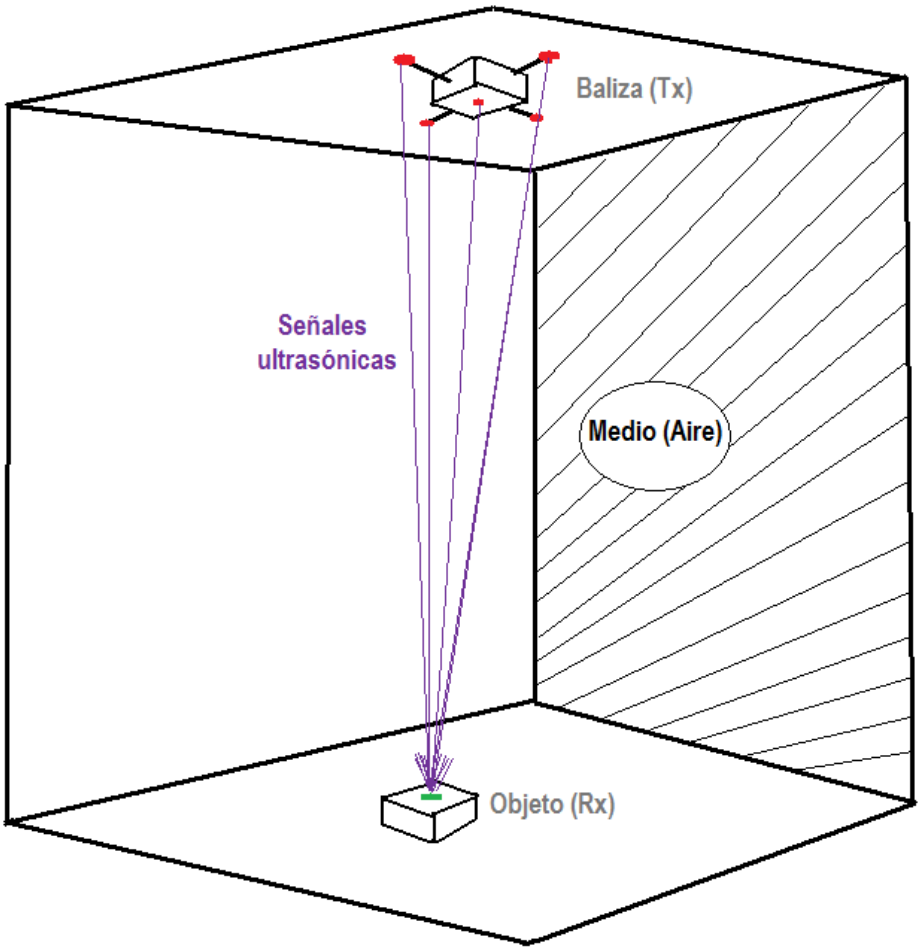


Figura 1.1-4: ULPS descentralizado, con referencias absolutas y asíncrono.

1.1.1.-Hardware

El HW de los sistemas ULPS está basado en, o bien FPGAs o bien microcontroladores. En este diseño, el módulo elegido usa un microcontrolador ARM Cortex-M3 del que más tarde se hablará.

Un ULPS necesita de modulador y demodulador para emitir y recibir la señal respectivamente. Los emisores y receptores de señal del sistema utilizan transductores para convertir la energía eléctrica en energía sonora y al contrario.

El correcto funcionamiento de los transductores exige un acondicionamiento previo de señal antes de pasar por los mismos. El bloque “acondicionamiento/demultiplexación” de la figura 1.1-5, representa este acondicionamiento, además de la demultiplexación, relacionada con la técnica TDMA ya comentada.

La figura 1.1-5, presenta un diagrama de bloques típico de un ULPS.

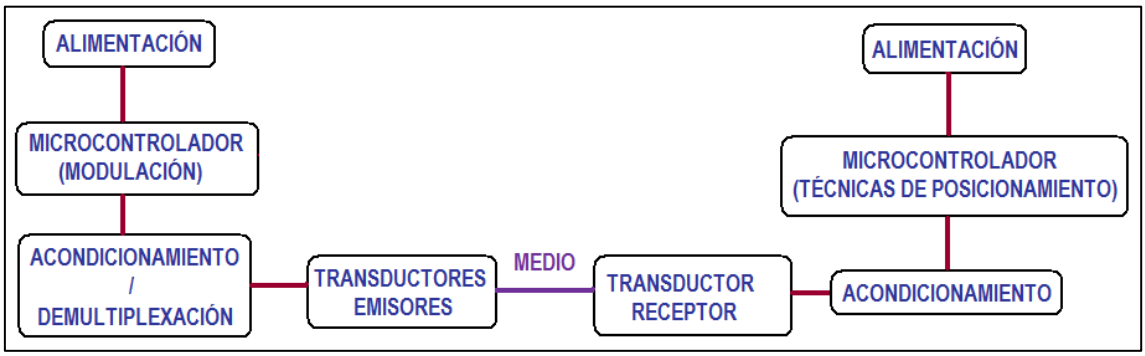


Figura 1.1-5: Esquema de sistema ULPS.

La particularidad importante de este proyecto es que la conexión con un computador o equipo para cambiar la configuración de parámetros de modulación, será inalámbrica, a través de wifi.

1.1.2.-Codificación

Como se ha comentado en el apartado 1.1, los ULPS trabajan con secuencias. Estas secuencias, pueden tener distintos tipos de codificación. Los ULPS pueden utilizar secuencias pseudo-aleatorias (PR), conjuntos de secuencias complementarias (CSS) o códigos ortogonales generalizados (GO) como base de su codificación. Cada tipo de secuencia tiene unas características determinadas que favorecerán o empeorarán aspectos funcionales. En general todos los tipos de secuencias usadas en ULPS, suelen ser binarias.

La calidad de la codificación de una secuencia se puede comprobar con los términos CCF (Cross Correlation Function) y ACF (Auto-Correlation Function). Estos términos permiten comprobar la similitud de una secuencia frente a otra secuencia y frente a sí misma respectivamente. Idealmente la CCF tiende a 0 y ACF debe ser lo más alta posible. Si se cumplen estos estos aspectos para la CCF y ACF la codificación mejora notablemente la detección del instante de llegada de la secuencia al receptor a partir del máximo de la función de correlación entre la señal recibida y el código correspondiente. El subapartado 1.2.3 expone una representación de estos picos de correlación y además relaciona los mismos con la importancia que tienen para el cálculo del posicionamiento del objeto.

Se puede comprobar, en [M.C. Pérez Rubio, 2009], una evaluación de distintos tipos de secuencias para sistemas ultrasónicos (Capítulo 3), así como el análisis de distintos tipos de esquemas de codificación (Capítulo 6) y las conclusiones de la tesis (Capítulo 7).

Usando esta tesis como referencia, se puede decir, que las secuencias Kasami tienen ciertas ventajas por sus ACF y CCF frente a otro tipo de secuencias. Una concisa explicación de la creación de estas secuencias se puede encontrar en [José M. Villadangos, 2012].

El tipo de secuencia elegida para la codificación de señal es Kasami.

1.1.3.-Modulación

Como se ha comentado en el subapartado anterior, los esquemas de codificación para los ULPS se basan en secuencias binarias. La modulación más empleada es BPSK (Binary Phase-Shift Keying), de esta forma, generalmente, un ULPS tiene una modulación binaria para una codificación binaria. No obstante existen trabajos que presentan modulaciones distintas. Un ejemplo con una modulación distinta de BPSK, se puede encontrar en [MMUO, 1999], donde se usa una modulación QPSK (Quadrate Phase-Shift Keying) con determinados tipos de secuencias.

1.1.4.-El medio de propagación

El medio en el que se propagan los ultrasonidos es un aspecto a tener en cuenta para un ULPS. Si se habla de un ULPS cuyo medio es el aire, los ultrasonidos estarán condicionados por las propiedades del mismo aire y por los factores que le afecten; como temperatura, u otros.

Existen ensayos, que demuestran experimentalmente ([VvSt, 2013]), que la velocidad de propagación del sonido en el aire, en función de la temperatura, se ajusta a la siguiente expresión (Vs en m/s: Velocidad del sonido, T en ºC: Temperatura):

$$Vs = 331.4 + 0.61T\ (m/s)$$

Otros factores que influyen en la velocidad de propagación del sonido en el aire son la humedad o la presión atmosférica, los efectos de los mismos no suelen tenerse en cuenta en los ULPS por ser despreciables para distancias cortas (<10m).

Sabiendo la velocidad de propagación del sonido en el aire, se puede calcular la distancia entre una fuente y un receptor de ultrasonidos de la forma que sigue (d: Distancia, Vs: Velocidad del sonido, TDV: Tiempo de vuelo):

$$d = Vs \cdot TDV$$

Esta expresión, pude ser útil para el dispositivo que calcule la posición.

1.1.5.-Interferencias y obstáculos en el medio

El medio a menudo está afectado por armónicos indeseados de distintas frecuencias procedentes de dispositivos electrónicos. Este hecho hace que en el diseño de un ULPS se pueda tener en cuenta el efecto de estos armónicos sobre el mismo. Estas armónicos, o interferencias, amplían el error de las funciones de correlación, influyendo también así en la precisión del posicionamiento. Un ejemplo del efecto de las interferencias que producen los

dispositivos electrónicos en un ULPS se analiza en [HUAV,2017], donde se comprueba el efecto que tienen los rotores de un pequeño UAV sobre la banda de error de la posición de un ULPS.

Otro efecto a tener en cuenta en un ULPS es el multicamino. Éste se produce cuando una copia de la señal original (eco) llega retardada por no llegar de forma directa, o en línea recta, entre emisor y receptor. El receptor del ULPS debe estar preparado para detectar los picos de correlación de mayor amplitud, que serán los que permitan determinar de forma acertada la posición entre una baliza y un objeto.

1.2.-Diseño de la baliza

Este apartado se dedicará a exponer un resumen general de la baliza diseñada. Aquí se explicarán aspectos destacables con los que cumple el diseño y el funcionamiento del mismo. En capítulos posteriores se profundizará en el hardware (HW), el módulo EMW3162 y la programación del mismo.

Concretamente la posición que calcule el sistema que posicione el objeto, será el receptor que el mismo porte o un receptor únicamente.

La baliza diseñada emite, a través de los transductores, las señales necesaria para que el receptor, las reciba. La señal está compuesta de 5 secuencias. Estas secuencias, se emiten de forma consecutiva por cada uno de los 5 transductores. Estas 5 secuencias son las secuencias Kasami, moduladas en BPSK y emitidas de forma cíclica (5,1,2,3,4,5,1,2,…) con una frecuencia de 40 kHz o 41.66 kHz.

Los transductores son las referencias que el receptor deberá de usar. Se ha elegido un diseño asíncrono, por lo que no son necesarias señales de sincronismo. Como consecuencia, el receptor deberá extraer las medidas de las DTDV entre el instante de llegada de una de las secuencias (que actuará como referencia) con cada una de las restantes.

Por otra parte, para permitir modificar los parámetros de modulación se ha creado una interfaz. Esta interfaz es un servidor web implementado en el EMW3162.

Para conseguir un canal de comunicaciones entre un equipo y el EMW3162 se usa un rúter que actúa como punto de acceso (AP) para los dos anteriores. De esta manera, tanto el equipo (en el que se quiera visualizar la interfaz web), como el EMW3162, se conectan a este rúter. Y únicamente teniendo acceso a la WLAN que proporcione el rúter, se puede acceder a la interfaz. Ésta, se puede visualizar en un dispositivo con un navegador (como Firefox o Chrome) y un SO (como Android o Windows) compatibles con el servidor web de la baliza. El EMW3162, hace que el sistema sea muy versátil pues permite interactuar con la baliza a través de un Smartphone.

Sólo en el caso de querer actualizar el FW actual del servidor web, será necesaria una conexión por cable (SWD).

La figura 1.2-1 representa el conjunto de dispositivos de la baliza y los dispositivos implicados en la modificación de la modulación. También en esta figura aparecen las siglas AP (Access Point) y ST (Station) en referencia a punto de acceso y equipo conectado al mismo.

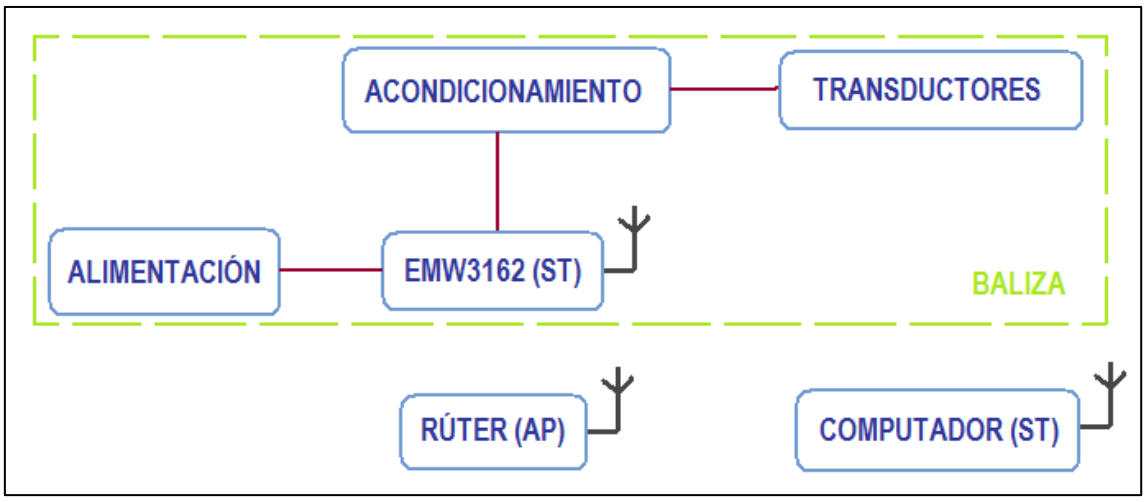


Figura 1.2-1: Baliza para ULPS con conexión inalámbrica.

Por otro lado, la baliza dispone de dos leds. Un led de color verde (**led de comunicación**) y un led de color rojo (**led de comprobación de conexión**). El led de comunicación emite un parpadeo cuando se interactúa con el servidor web. El led de comprobación de la conexión, en caso de lucir continuadamente, indica que hay una conexión establecida entre la baliza y un rúter.

La electrónica de la baliza está en el interior de una caja de plástico que permite el paso de la señal de radio wifi. El que los equipos involucrados en la configuración de parámetros de modulación se conecten a través de un rúter, permite también, acceder a la interfaz web a través de Ethernet. La figura 1.2-2 representa la baliza con una conexión a través de cable de Ethernet y otra Wireless.

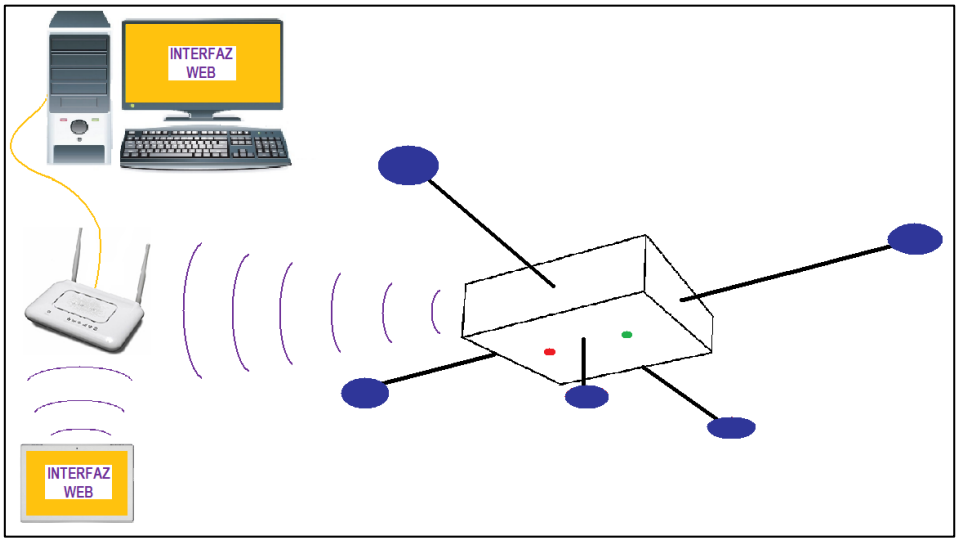


Figura 1.2-2: La baliza y sus posibles tipos de conexión.

La figura 1.2-2 tiene representado en azul los transductores y en verde y rojo los leds de comunicación y comprobación de la conexión respectivamente.

1.2.1.-Modulación para la baliza

La codificación de las secuencias Kasami es binaria y se ha elegido modulación BPSK para la emisión de las mismas. La frecuencia de modulación, como se ha comentado anteriormente, será de 40 o 41.66 kHz. Esta señal, sale en primer lugar por uno de los DAC (convertidor analógico-digital) del EMW3162.

La emisión de secuencias, como se ha comentado, no será simultánea y se usa TDMA como técnica de acceso al medio. No obstante aunque no sea necesaria, se ha dotado al diseño de codificación por los aspectos mencionados en el subapartado 1.1.2.

La modulación estará relacionada con los términos de “ciclo” y “símbolo”.

Se llamará “ciclo” al patrón más pequeño que se pueda representar en función de una codificación Kasami. En BPSK, un ciclo se puede representar como aparece en la figura 1.2-3 Los valores “1” y “-1” representan un código de una secuencia Kasami. De esta manera, el ciclo toma una forma o la inversa respecto al eje de tiempos en función de estos valores.

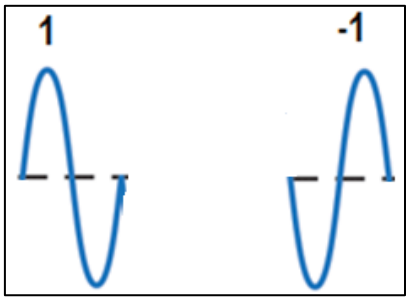


Figura 1.2-3: Posibles ciclos. (No discretizados).

Además, un símbolo puede tomar el valor de 1 o varios ciclos. La figura 1.2-3 podría representar también el valor de 1 ciclo/símbolo. Pero en caso de hablar de N ciclos/símbolo, se repetiría el patrón de un ciclo, N ciclos. Finalmente, esos N ciclos/símbolo, representarían un nuevo patrón que se podría codificar de nuevo con dos valores binarios.

Como ejemplo, un valor de una secuencia en función de dos valores binarios (“1” o “-1”) para 2 ciclos/símbolo, sería como aparece en la figura 1.2-4.

La baliza permitirá escoger de 1 ciclo/símbolo a 4 ciclos/símbolos con determinadas condiciones (se comentarán en el subapartado 1.2.2).

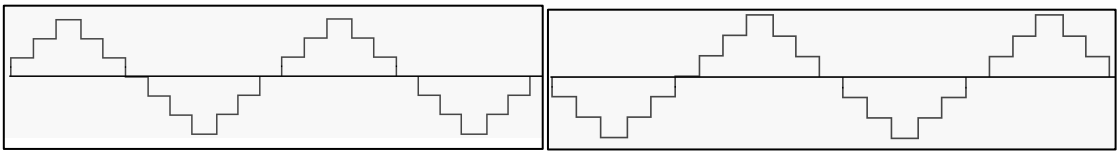


Figura 1.2-4: Posibles símbolos con 2 ciclos/símbolo. ("1" a la izquierda, "-1" a la derecha).

Esta figura 1.2-4 aparece con los símbolos discretizados tal como se mostrarán en la práctica.

El número de muestras/ciclo será una constante de valor 12 para cualquier combinación de parámetros de modulación.

1.2.2.-Parámetros de modulación

Los parámetros de modulación, además de ser configurables a través de la interfaz web, también son configurables en tiempo de ejecución del EMW3162. Los parámetros de modulación y sus respectivas definiciones son las que siguen:

- **Frecuencia portadora**: El sistema permitirá una frecuencia portadora de 40 kHz o de 41.66 kHz. Lo que equivale a 25us o 24us de espacio de tiempo entre cada muestra respectivamente. Al emplear 12 muestras por ciclo, la frecuencia de muestreo sería de 480 kHz o 500 kHz.

- Selección de **número de bits** de las secuencias Kasami: Se puede elegir entre 255 bits y 1023 bits. Este número de bits es el que tengan todas las secuencias Kasami, es decir, o todas las secuencias emitidas tendrán 255 bits o bien todas tendrán 1023 bits. Esto es debido a la limitación de la memoria.

- Número de **ciclos/símbolo**: Esta definición ha sido comentada en el subapartado 1.2.1. Se podrán seleccionar valores de 1 a 4. Si se elige 1023 de número de bits, se podrán seleccionar los valores 1 o 2, si se escoge 255, se podrán seleccionar valores de 1 a 4.

- **Periodo de emisión**: Determina el espacio de tiempo que se quiere que exista entre el principio de la secuencia emitida de un transductor hasta el principio de la secuencia emitida del siguiente transductor. Este parámetro puede tomar valores entre 14ms y 120ms.

- Asignación del número de **secuencia Kasami**: Existen 5 transductores. Si se elige un número de bits de 255 se podrán elegir 16 posibles secuencias. Si se elige un número de bits de 1023 se podrán elegir entre 32 secuencias distintas.

En total hay 5 parámetros con posibilidad de modificarse a través de la interfaz web.

Estos parámetros se graban en memoria cada vez que uno de ellos es modificado. De esta forma, con un fallo de alimentación o un reinicio del EMW3162 no se pierden los parámetros últimos con los que la baliza estuviera trabajando.

1.2.3.-Secuencias y tiempos de vuelo

La señal de salida del DAC está formada por 5 secuencias Kasami. Estas secuencias se repiten de forma continua y con el mismo periodo de emisión hasta que haya una modificación de parámetros. Después de la salida del DAC la información se demultiplexa de tal forma que se asigna un espacio de tiempo a cada secuencia. Este espacio de tiempo asignado a cada secuencia está relacionado con el apartado 1.1 y la figura 1.1-3 donde se comentaba la técnica de acceso al medio TDMA, donde cada usuario tiene asignado un tiempo fijo. En este caso, el usuario es el transductor y el tiempo fijo es el periodo de emisión.

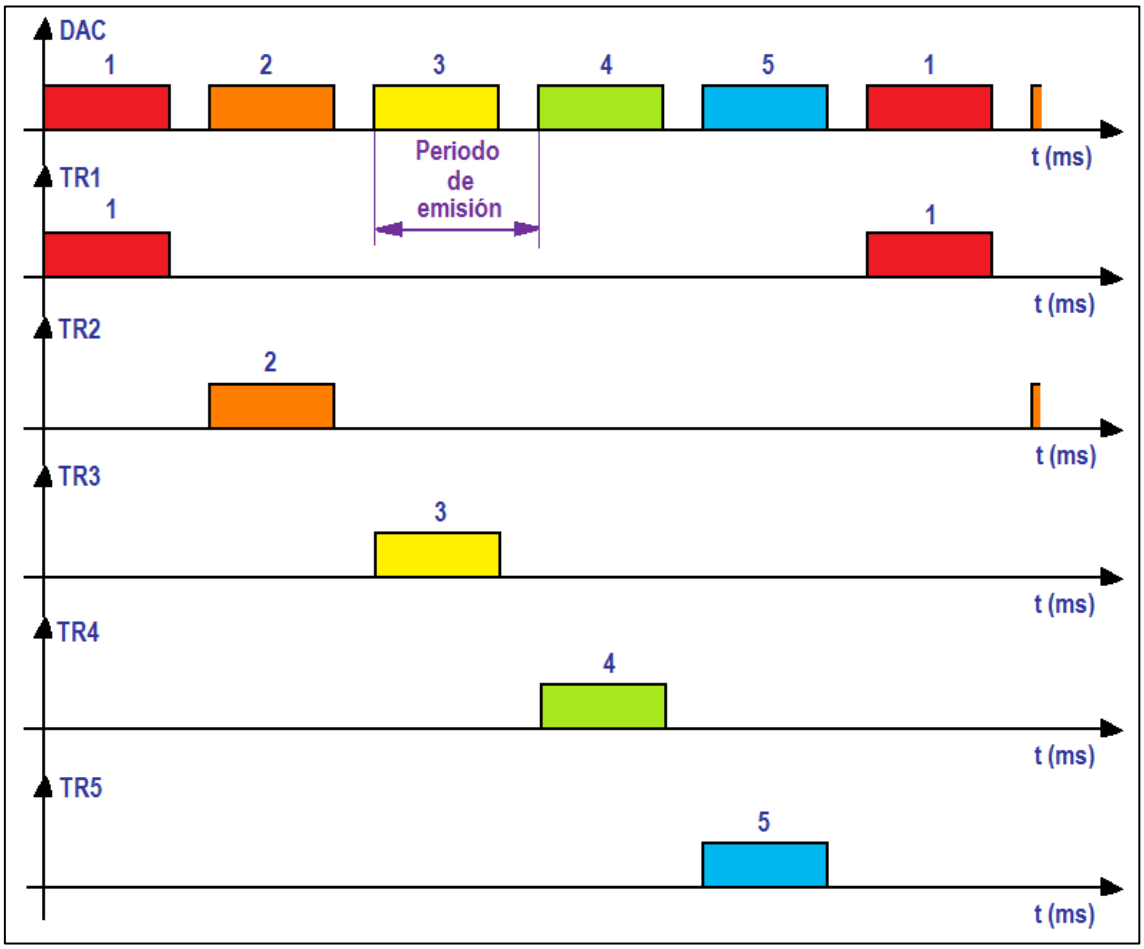


Figura 1.2-5: Secuencias de salida en el DAC y en los transductores.

El receptor deberá realizar las correlaciones oportunas y calcular su posición en el entorno de la baliza.

Cada una de las secuencias que aparecen en la figura 1.2-5 está formada por la codificación Kasami pertinente, creada, de la forma que se ha visto en el apartado 1.2.1

Como se ha dicho la frecuencia de la señal es de 40 o 41.67 kHz, es decir, que su periodo será de 25us o de 24us respectivamente.

1.2.4.-Imágenes de la baliza

A continuación, las figuras 1.2-7 y 1.2-8 muestran dos imágenes de la baliza funcionando ubicada en el techo. Se pueden observar los transductores, la batería y los led comentados.

Figura 1.2-6: Captura inferior de la baliza.

Figura 1.2-7: Captura lateral inferior de la baliza.

1.2.5.-Imágenes de la interfaz web

El desarrollo de la programación del EMW3162 para la interfaz web, que se expondrá en el capítulo 4, dará como resultado dos páginas web fundamentales (web principal y web del transductor) que permitirán la modificación paramétrica de la modulación. La figura 1.2-8 muestra los resultados de estas dos webs. En el lado izquierdo de la figura se muestra la web principal con los 4 parámetros comunes que afectarán a las secuencias que emitan los 5 transductores; en el lado derecho, se muestra la captura de la interfaz web que permite modificar el número de secuencia Kasami de uno de los transductores (esta interfaz web, será igual para todos los transductores y es accesible a través de los hipervínculos de la web principal).

Figura 1.2-8: Web principal (izquierda) y web del transductor (derecha).

2.-HARDWARE DE LA BALIZA

La electrónica creada para la baliza se ha montado sobre una protoboard con componentes en formato DIP (Dual In-Line Package). La figura 2-1 presenta el conexionado de la electrónica.

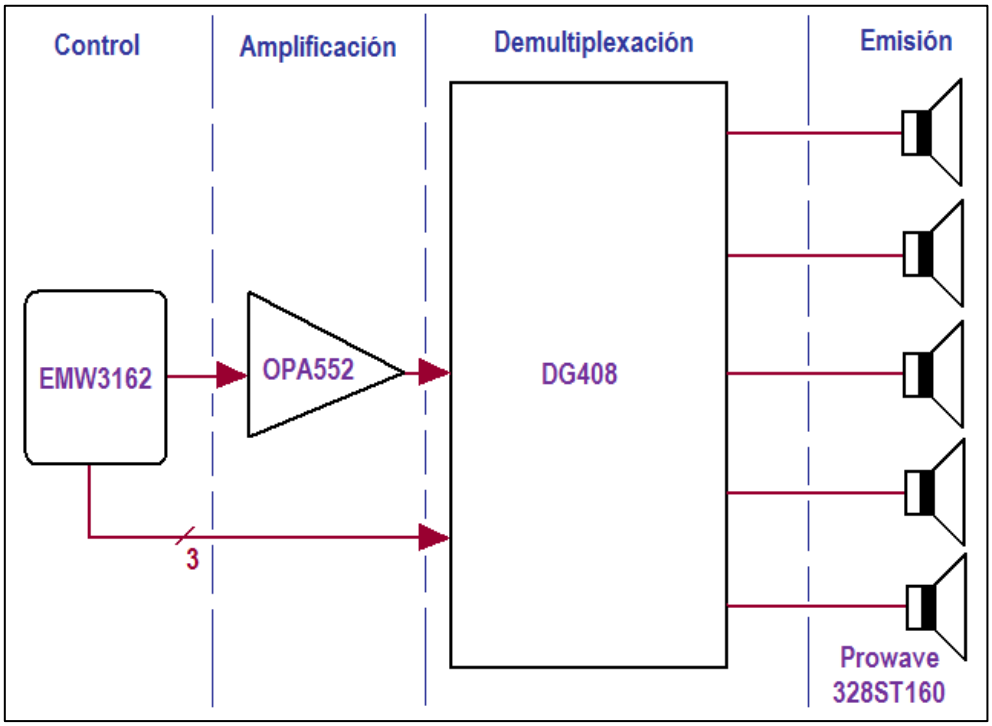


Figura 2-1: Breve esquema electrónico de la baliza.

La electrónica va dentro de la caja de la baliza. La imagen de la electrónica de la baliza se puede ver en la figura 2-2.

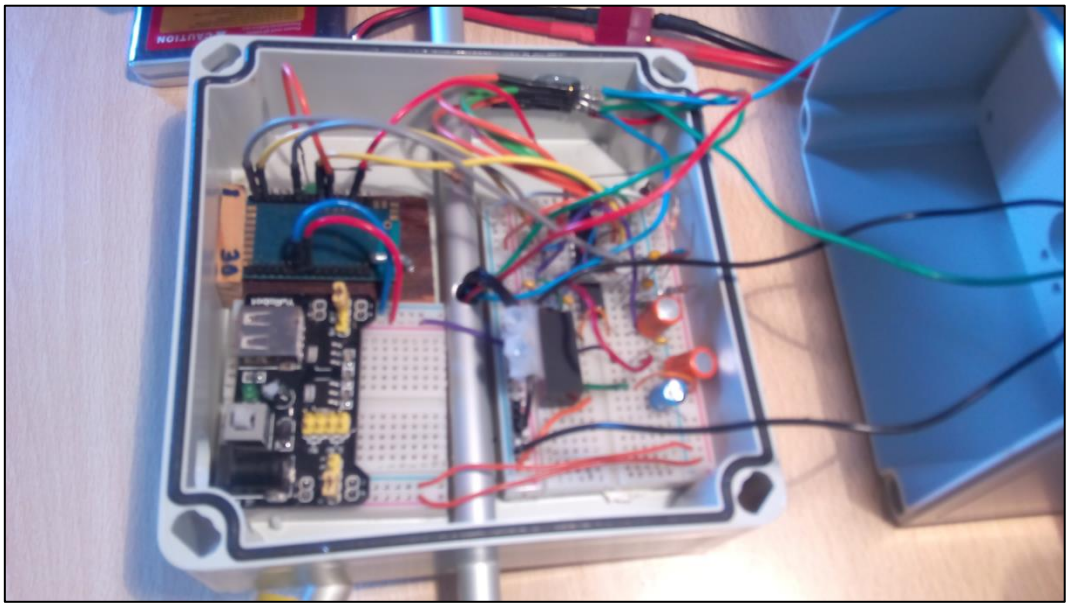


Figura 2-2. Imagen del interior de la baliza.

Como se comentará más adelante en el capítulo 3, el EMW3162 está formado por dos microcontroladores (uCs). El microcontrolador (uC) que se presenta en la figura 2-1 es uno de

ellos. Este uC está conectado a un amplificador para elevar la tensión de salida del DAC y al multiplexor para seleccionar el transductor que deba emitir. El multiplexor, en función de la selección que realice el uC asignará un espacio de tiempo para uno de los canales conectados a su transductor.

A continuación de este mismo capítulo se explicarán más en profundidad los aspectos concernientes al HW del diseño: Alimentación, acondicionamiento de señal y transductores. Finalmente se mostrará el esquema electrónico general del diseño. Este esquema se encuentra en el apartado 2.6.

También se pueden ver los esquemas del módulo EMW3162 que aporta su fabricante, MXCHIP, en el anexo B.

2.1.-Alimentación

La alimentación partirá de una batería de 7.4V. Esta tensión, es la tensión de entrada que va conectada a una fuente de alimentación MB102 Breadboard. Este componente permite generar dos tensiones de valores 3.3V y 5V.

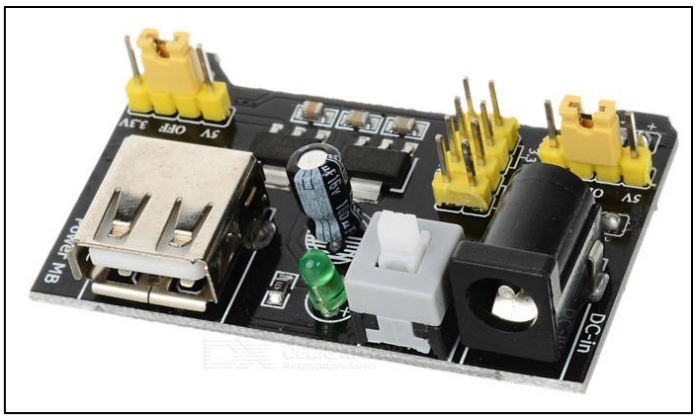


Figura 2.1-1: MB102 Breadboard.

Tabla 2.1-1: Características de la fuente MB102 Breadboard. [MB102].

MB102 Breadboard	
Voltaje de entrada	6.5-12.0V DC
Voltaje de salida 1	3.3V/5V
Voltaje de salida 2	3.3V/5V
Corriente máxima de salida	<700 mA

Se ha adquirido un DC/DC de TRACO POWER, modelo TMR 3-0523E que convierte tensiones de 4.5V-9.0V a +15V y -15V. Estas tensiones se conectan al multiplexor y al amplificador para que los transductores alcancen valores de tensión superiores a la salida del DAC del uC.

Las tensiones de +15V y -15V que otorga el DC/DC son utilizadas en entradas de baja impedancia para el multiplexor y el amplificador operacional (AO).

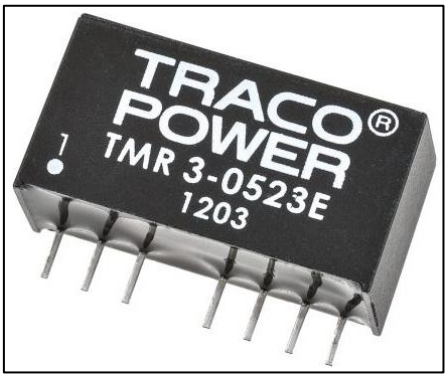


Figura 2.1-2: Conversor 5-15V de TRACO POWER.

Tabla 2.1-2: Características del conversor DC/DC 5-15V. [TP, 2009].

TMR 3 – 0523E (TRACO POWER)	
Voltaje de entrada	4.5-9.0 V DC
Corriente máxima de entrada 1	779 mA a plena carga
Corriente máxima de entrada 2	60 mA sin carga
Voltaje de salida	+15V y -15V
Corriente de salida	+100 mA a -100 mA

2.2.-El EMW3162

El módulo wifi EMW3162 consta de dos DAC. Estos convertidores trabajan desde 0V a 3.3V. Por la salida del DAC1 del EMW3162 se emitirán las secuencias.

En el anexo B se pueden ver los planos del EMW3162.

En el capítulo 3 se analizará el EMW3162 en detalle.

2.3.-Acondicionamiento y demultiplexación

Para asignar un espacio de tiempo a cada secuencia se demultiplexa la señal emitida por el DAC1. Previamente, se amplifica la salida de este DAC para conseguir valores superiores a 3.3V. Para ello, se ha utilizado el AO OPA552, con relación salida/entrada igual a 10.

Tabla 2.3-1: Características del AO OPA552. [ti, 2016].

OPA552	
Rango máximo de V+ a V-	60V (V+: +30V, V-: -30V)
Rango mínimo de V+ a V-	8V (V+: +4V, V-: -4V)
Ancho de banda	12 MHz
SR	24V/us

El SR (Slew Rate) del OPA552 permite amplificar una señal de 41.66 kHz fielmente. Las dos salidas del convertidor DC/DC irán conectadas a este componente en los terminales V+ y V-.

Para demultiplexar las señales, se ha elegido el multiplexor analógico 408DG.

Tabla 2.3-2: Características del multiplexor DG408. [intersil, 2006].

DG408	
Rango de V+ a V- máximo	44.0V (V+: +19V, V-: -25V)
Resistencia de ON	100 Ω
tr (tiempo de subida)	250ns máximo
tf (tiempo de bajada)	250ns máximo
ton (tiempo de ON)	225ns máximo
toff (tiempo de OFF)	150ns máximo

Los tiempos que requiere este multiplexor son un orden inferior que el periodo de la señal de salida del DAC.

2.4.-El modelo del transductor

Para modular en BPSK es necesario tener un transductor de fase lineal en un ancho de banda que permita cubrir la densidad espectral de la frecuencia de la señal con la que se trabaje. En la modulación BPSK, la densidad espectral se concentra en torno a la frecuencia de emisión de la propia señal. La figura 2.4-1 muestra una imagen de la densidad espectral de potencia (PSD) en BPSK frente a la frecuencia, en este caso con frecuencia de trabajo de 0 Hz. De la misma forma se tendrá una distribución espectral similar en los 40 o 41.66 kHz.

Debido a esta distribución espectral de potencia se ha escogido un modelo de transductor que tenga una fase lineal en torno a los 40 kHz. De esta forma, la señal emitida por los transductores será más fiel a la señal modulada que se escoja, contribuyendo así a que las distintas componentes espectrales no presenten retardos en el sistema de recepción que pudieran traducirse además, en cálculos de posicionamiento erróneos.

Otra característica que se ha tenido en cuenta es el nivel de presión sonora que el transductor tuviera en la frecuencia de emisión. Tener un alto nivel de presión sonora implica tener mayor área de cobertura.

En [D. de Diego López, 2013] se presenta un exhaustivo análisis de varios modelos de transductores presentes en el mercado de precio reducido. Este autor realiza análisis experimentales con los transductores y escoge el transductor Prowave 328ST160 como el que aúna mejores relaciones entre características en torno a los 40 kHz. Las figuras 2.4-2 y 2.4-3 representan las respuestas experimentales en fase y de nivel de presión sonora del transductor Prowave 328ST160.

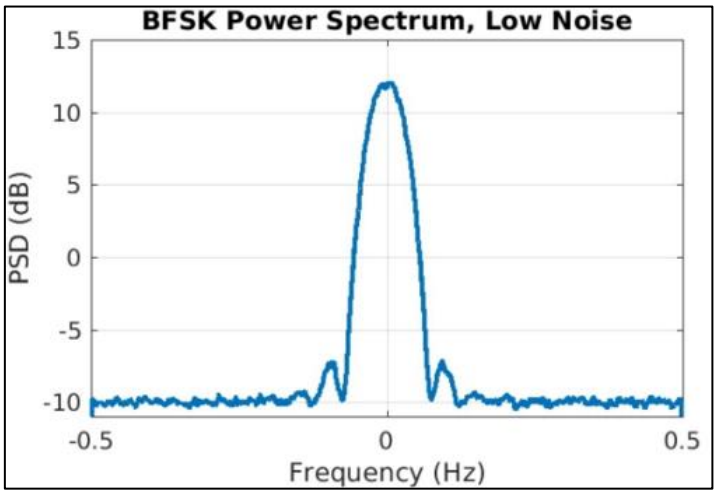


Figura 2.4-1: Densidad espectral de la modulación BPSK.

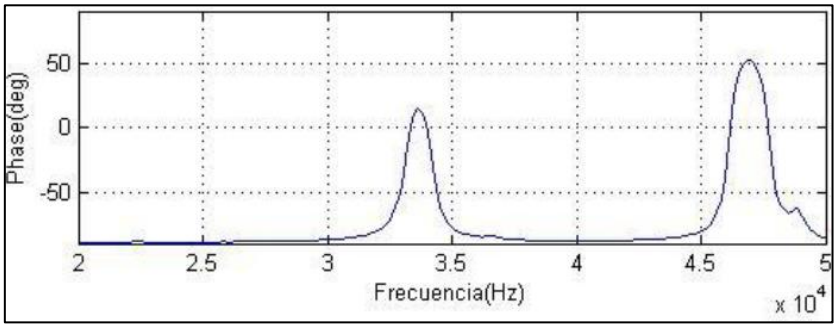


Figura 2.4-2: Respuesta en fase del Prowave 328ST160. [D. de Diego López, 2013].

En [D. de Diego López, 2013], el autor llega a la conclusión de que el transductor Prowave 328ST160, aunque no se haya diseñado específicamente para operar en torno a 40 kHz, puesto que su frecuencia central es más baja (32 kHz), los parámetros experimentales demuestran que es más adecuado que el resto de transductores por tener una respuesta en fase lineal en torno a 40 kHz y un nivel de presión sonora suficiente para tener una distancia de cobertura adecuada en la baliza. Esta distancia cobertura en la que se debe encontrar el objeto para calcular su posición será de menos de 10m.

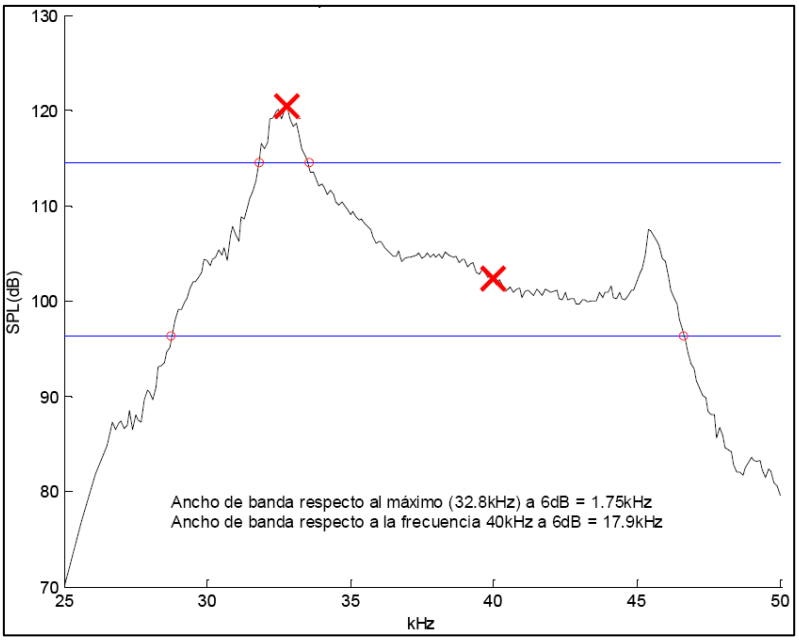


Figura 2.4-3: Nivel de presión sonora del Prowave 328ST160. [D. de Diego López, 2013].

Figura 2.4-4: Transductores Prowave 328ST160.

Tabla 2.4-1: Características del transductor Prowave 328ST160. [ProWa, 2005].

Prowave 328ST160	
Frecuencia central	32.8 ± 1.0 kHz
Ancho de banda	2.0 kHz
Nivel de presión sonora	115 dB mínimo

2.5.-Aportaciones de señal no deseadas

El conversor DC/DC introduce un rizado no deseado. Para corregir este mismo, se han puesto un condensador electrolítico de 10uF y un condensador de cerámica en paralelo a las líneas de

alimentación de +15V y -15V. Con estos condensadores se reducirá el rizado por una parte y se atenderá a la recomendación del fabricante el OPA551 mostrada en la figura 2.5-1.

En la salida del DAC se han conectado un condensador en serie y una resistencia y un condensador en paralelo para eliminar la componente continua a la salida del DAC.

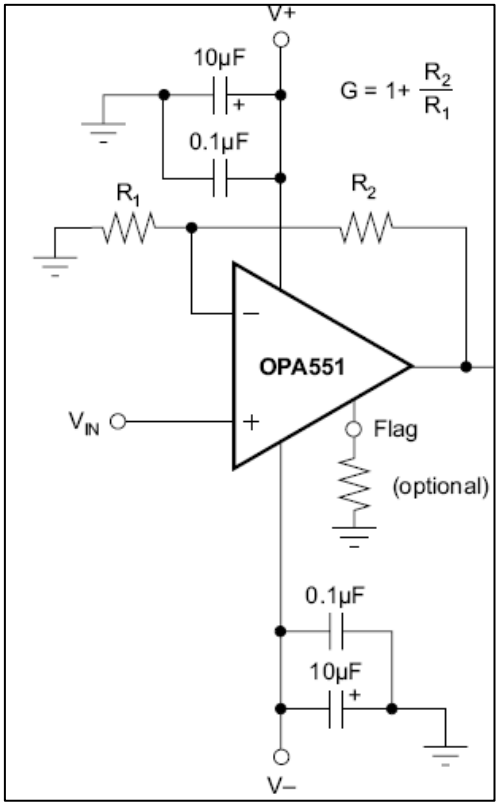
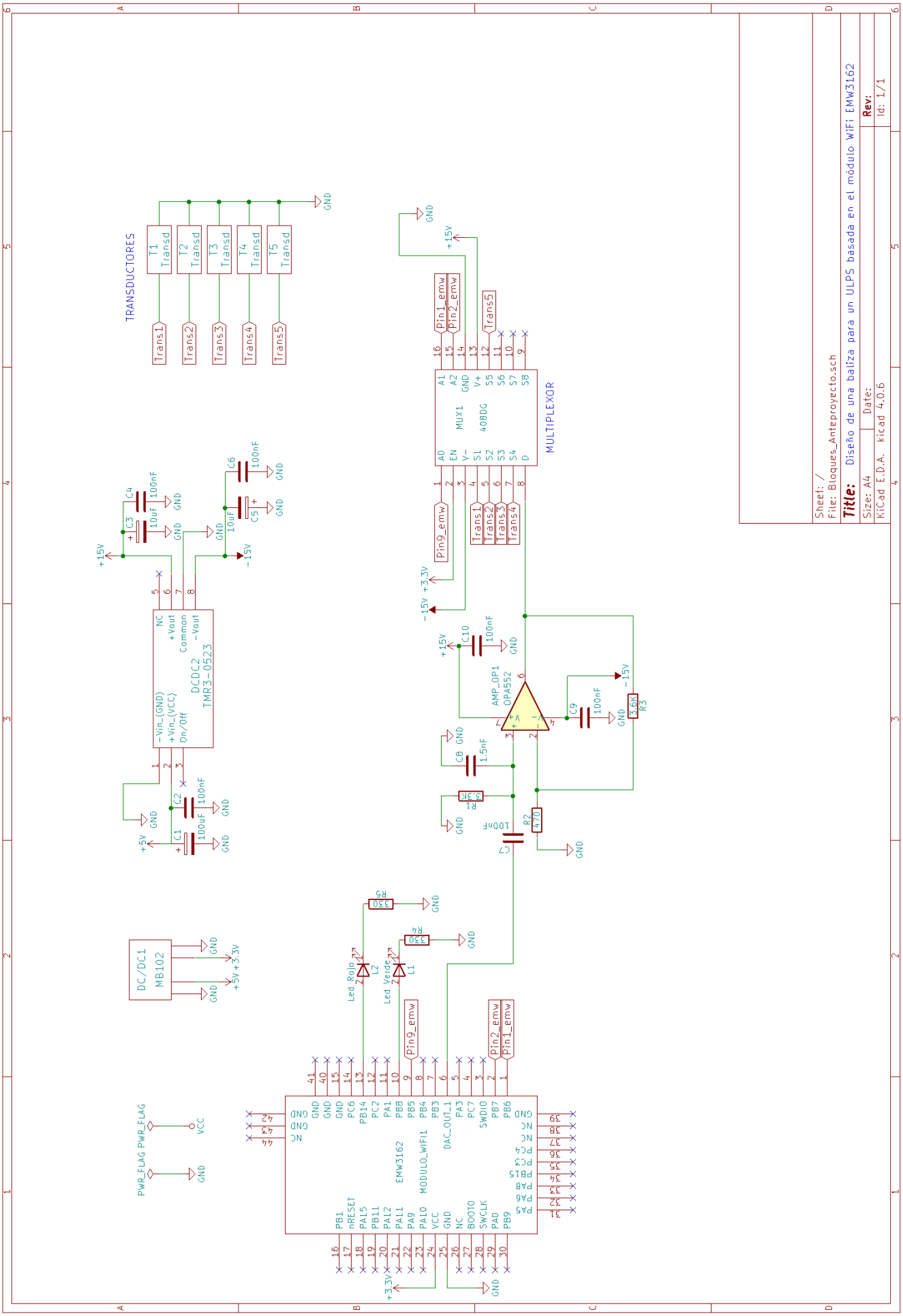


Figura 2.5-1: Montaje típico de la serie OPA55x. [ti, 2016].

2.6.-Esquema del diseño

En este apartado se presenta un esquema electrónico del circuito. En él aparecen los componentes electrónicos comentados exceptuando el adaptador de red, que como se ha comentado será el encargado de suministrar energía a la fuente de alimentación MB102.

El valor de 5V y 3.3V se obtienen a partir de la fuente de alimentación MB102.


TRANSDUCTORES
T1 Transd
T2 Transd
T3 Transd
T4 Transd
T5 Transd
Trans1
Trans2
Trans3
Trans4
Trans5
GND
MULTIPLEXOR
MUX1
408DG
A0
EN
V-
S1
S2
S3
S4
D
A1
A2
GND
V+
S5
S6
S7
S8
Pin1_emw
Pin2_emw
Pin9_emw
+15V
-15V
+3.3V
DCDC2
TMR3-0523
-Vin_(GND)
+Vin_(VCC)
On/Off
NC
+Vout
Common
-Vout
C1 100uF
C2 100nF
C3 10uF
C4 100nF
C5 10uF
C6 100nF
AMP_OP1
OPA552
C7 100nF
C8 1.5nF
C9 100nF
C10 100nF
R1 5K
R2 470
R3 3.6K
DC/DC1
MB102
+5V
+3.3V
Led Rojo
Led Verde
L1
L2
R30
R31
R32
MODULO_WIFI1
EMW3162
PWR_FLAG
VCC
Sheet: /
File: Bloques_Anteproyecto.sch
Title: Diseño de una baliza para un ULPS basada en el módulo WiFi EMW3162
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 4.0.6
Id: 1/1

3.-EL MÓDULO WIFI EMW3162

Este capítulo está dedicado al dispositivo principal de la baliza, el módulo wifi EMW3162. Se expondrá cómo se distribuye su memoria flash, los modos de funcionamiento, qué FW necesita tener instalado para hacer posibles las comunicaciones, la librería MXCHIP, los ejemplos que aporta el fabricante, las tarjetas de desarrollo y las funcionalidades que ofrece el FW de fábrica.

Figura 3-1: Módulo EMW3162 (seeedstudio.com).

Se puede comprobar el esquemático del EMW3162 en el anexo B.

Nota: En este capítulo, al hablar de "parámetros" habrá que diferenciar los parámetros que se usan para modificar la modulación de los parámetros que configuran los estados del EMW3162, los parámetros de la WLAN.

3.1.-Características principales

En este apartado se verá qué componentes forman el EMW3162, sus características eléctricas y otras, las zonas de memoria que asigna el fabricante por defecto para el módulo y cuáles son las librerías que admite el mismo.

El módulo wifi consta de dos microcontroladores y una interfaz SDIO como vía de comunicación entre ellos. Un microcontrolador se ocupa exclusivamente de la comunicación wifi y no se tiene acceso a sus periféricos. Otro microcontrolador permite el acceso a sus periféricos y modificar sus espacios de memoria. Un esquema HW a groso modo, pero que proporciona una idea general del módulo es el mostrado en la figura 3.1-1.

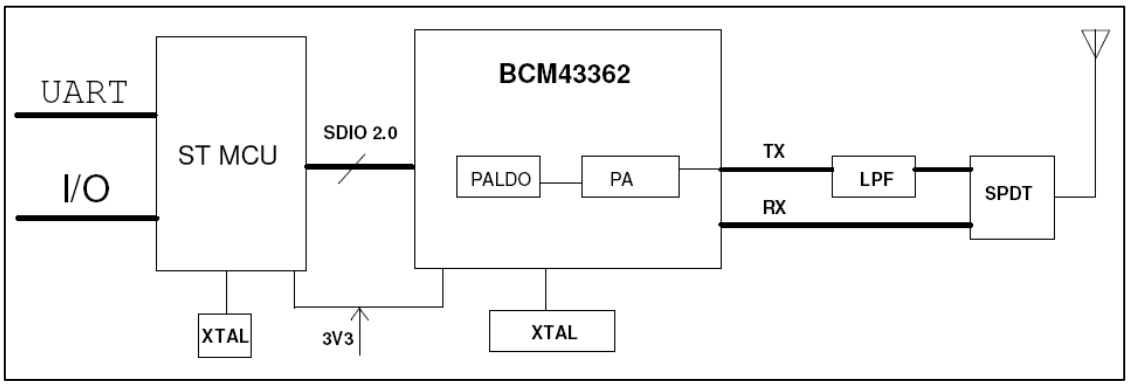


Figura 3.1-1: Breve esquema del EMW3162.

En la figura 3.1-1 también se aprecian las interfaces entrada salía (I/O) y UART. Esto se debe a que el FW de fábrica utiliza dos leds para indicaciones y la UART como uno de los medios de comunicación con el módulo.

Concretamente los microcontroladores que contiene el módulo wifi son:

* (Parte izquierda de la figura 3.1-1) STM32F205. Este microcontrolador tiene la función de albergar el firmware (bootloader, drivers necesarios para la comunicación wifi) y permitir la programación por parte del usuario. Consta de una memoria flash, y esta misma es la memoria flash del EMW3162. Este microcontrolador tiene un procesador Cortex-M3 y consta de un cristal externo independiente (XTAL).

* (Parte derecha de la figura 3.1-1) BCM43362. Éste, es el intermediario entre el STM32F205, y el/los equipos involucrados en la comunicación wifi. Es un Cortex-M3 con unidad de encriptación desarrollada por Cypress.

A continuación, se expone un resumen de la información relevante que aporta el fabricante y más ligada está a este proyecto:

- **3.3V** de alimentación.
- **120MHz** de frecuencia de trabajo de MCU.
- Bajo consumo (**320mA máximo** de consumo).
- **24mA** de consumo **promedio** con frecuencia de procesador de 120Mhz a 25 ºC, con el wifi conectado y en envío de datos.
- **1Mb de memoria Flash, 128Kb de RAM**.
- Interfaz **SWD**.
- 2 UART.
- **2 DAC**.
- Tolerancia de protocolo **OTA** (Over The Air).
- Funcionamiento en modo **AP, ST**.
- Encriptación soportable: WEP, WPA, WPA2.
- Estándares wifi soportables: 802.11b, 802.11g y 802.11n con ratios desde 11Mbps para el estándar b y hasta por encima de 72Mbps para el estándar n.
- Tipos de modulaciones soportadas: DBPSK, DQPSK, **BPSK**, QPSK y otras.

El diagrama de bloques que aporta el fabricante es el siguiente:

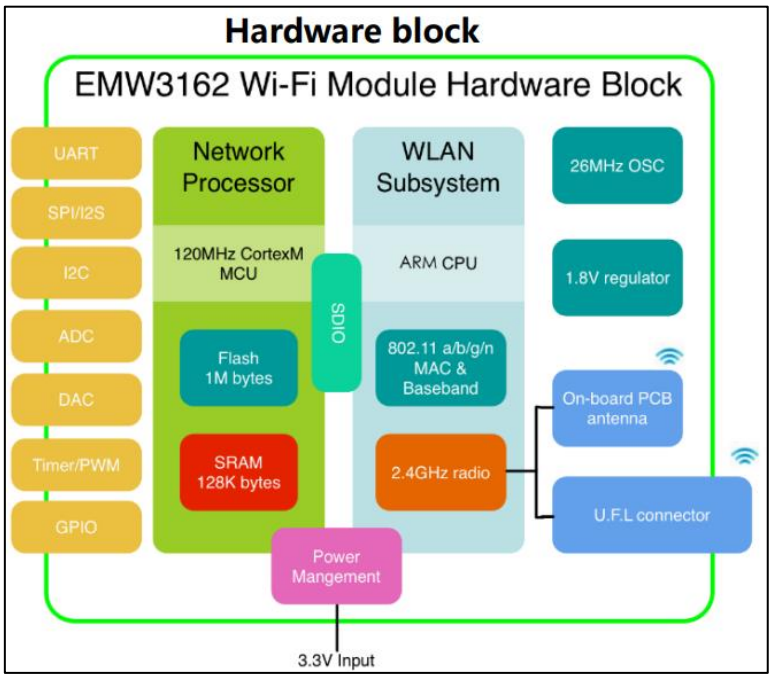


Figura 3.1-2: Esquema de bloques del módulo EMW3162.

En la figura 3.1-2 se muestran los periféricos disponibles para el usuario. Esta figura muestra representa el microcontrolador STM32F205 (“Network Processor”, parte izquierda), la zona de comunicaciones wifi conectada con el BCM43362 (“WLAN Subsystem”, parte derecha), la alimentación compartida para ambos y la interfaz SDIO que separa ambos microcontroladores.

3.2.-Firmware de fábrica

El fabricante organiza de forma específica la memoria flash. Esta memoria, como se ha comentado, es la propia memoria flash del STM32F205, y de fábrica, está organizada de la forma muestra la tabla 3.2-1.

Tabla 3.2-1: Distribución de la memoria del EMW3162.

Start	End	Type	Size (bytes)	Content
0x08000000	0x08003FFF	B	16k	Bootloader
0x08004000	0x0800BFFF	P	32k	OTA info，user para.
0x0800C000	0x08060000	A	336k	User application
0x08060000	0x080C0000	–	384k	OTA temporary storage
0x080C0000	0x080FFFFF	D	256k	RF Driver

La tabla 3.2-1 muestra el tamaño y posición de los espacios o sectores de memoria flash asignados al EMW3162. Pero estas zonas de memoria, pudieran ser modificadas en función de las necesidades del diseño. Las zonas asignadas por MXCHIP para el EMW3162 son:

- Bootloader: Viene instalado de fábrica, el cual permite desplazar la tabla de vectores del Cortex-M3 hacia la posición de la zona de usuario ("User application"), además de permitir ciertas funciones, como actualizaciones u otra información, vía UART.

- OTA info: Es una zona de memoria que permite que se grabe información de ciertos ejemplos que aporta el fabricante (por ejemplo, el guardado del usuario y la clave de una red).

- User application: Viene asignada para las aplicaciones de usuario. En esta zona, de fábrica, viene un FW instalado que más tarde será comentado.

- OTA temporary storage: Esta zona de memoria sirve para el almacenamiento temporal del FW que será instalado en un sector. Está ligada al protocolo OTA (Over The Air). Previamente a actualizar un sector, se grabará el FW que se instalará en el mismo en esta zona de memoria y posteriormente se instalará en la zona de memoria correspondiente al sector elegido. El protocolo OTA permite la actualización de uno de los sectores de forma inalámbrica. Este protocolo, se expone en el apartado 3.7.

- RF (Radiofrecuencia) drivers: Espacio de memoria para los drivers que controlan la comunicación entre el STM32F205 y el BCM43362.

Por defecto, el fabricante, establece ciertos valores que permiten funcionar al EMW3162 de forma determinada. Los parámetros de fábrica del módulo (WLAN, UART, etc.) son los que muestra la figura 3.2-1.

Default settings :

WLAN settings : SSID: "MXCHIP_XXXX" (XXXX=the last 2 bytes of module's MAC address) , Soft AP mode

UART settings : 115200/8/n/1

IP Address: 192.168.1.1/255.255.255.0, DHCP server enabled. Bonjour service enabled.

TCP Server mode, Port : 8080

Figura 3.2-1: Valores por defecto del EMW3162.

El FW de fábrica se llama Standard AT y puede aparecer en formato binario con este nombre o bien con un nombre de 11 cifras separadas las últimas 3 por un punto (por ejemplo: 31620303.006.bin), mostrando el número de la versión en las últimas 3 cifras.

3.3.-Librerías

Hay 3 librerías que pueden ser instaladas en el microcontrolador STM32F205 del EMW3162 y sólo una que presenta el fabricante que se puede usar en el BCM43362. Las librerías o FW compatibles con el módulo wifi son:

- mxchipWNET DTU: Es el FW Standard AT comentado en el punto 3.2 que viene instalado de fábrica.

- mxchipWNet Library: Esta librería, es aportada por el fabricante para desarrollar aplicaciones a nivel de usuario. El anexo A muestra más información de esta librería.

- mxchipWNet Library Plus: Librería similar a la anterior que, además permite correr un RTOS (Real-Time Operating System).

- WICED (Wireless Connectivity for Embedded Devices) Firmware development kit: Este FW está integrado en el BCM43362 para permitir al módulo actuar bajo los estándares IEEE 802.11 y, en general, darle al módulo el carácter de WLAN.

La tabla 3.3-1 muestra las funciones relevantes de cada librería.

Tabla 3.3-1: Librerías compatibles con el módulo EMW3162.

Firmware/Library	Function
mxchipWNet™ - DTU	Predefined firmware: UART/Wi-Fi conversion
mxchipWNet™ Library	Software library used to develop custom firmware
mxchipWNet™ Library Plus	Software library based on RTOS
WICED™ Firmware development kit	WICED™ source codes with TCP/IP, Wi-Fi MAC RTOS and GCC tool chain

3.4.-Tarjetas de desarrollo para el EMW3162

En este apartado se expondrán las tarjetas de desarrollo compatibles con el módulo wifi, así como las principales particularidades de cada una.

Todas las tarjetas de desarrollo que se verán a continuación constan de 3 botones que ofrecen ciertas funciones y dos interruptores que permiten entrar en diferentes modos de

funcionamiento del módulo. Los botones e interruptores que tienen todas las tarjetas de desarrollo presentadas son los que siguen:

- WPS: Botón para configuración con Wifi Protection Security.

- EASYLINK: Botón que permite interactuar al módulo con ciertas aplicaciones móviles. Para poder usar este botón es necesario tener una aplicación en un SO que permita soportar la misma e implementar la programación necesaria en el EMW3162 para hacer posible esta interactuación.

- RESET: Inicializa el módulo.

- Interruptores BOOT y STATUS: Permiten entrar en distintos modos de funcionamiento del EMW3162.

A continuación se nombrarán cada una de las tarjetas de desarrollo y se explicarán sus particularidades de cada una.

1. EMB 380 S2

Esta tarjeta tiene una conexión con microusb, una conexión DB9 como interfaz serie y un puerto para JTAG o SWD de 20 pines. Incluye dos botones, sleep y standby:

- SLEEP: Permite al módulo entrar en modo de bajo consumo.

- STANDBY: Con este botón se puede acceder al modo de más bajo consumo del módulo, lo que se llama deepsleep mode. Este modo está basado en el modo deepsleep del Cortex M3.

Como particularidad para esta tarjeta, el fabricante le ha incluido la posibilidad de alimentar al módulo con una pila.

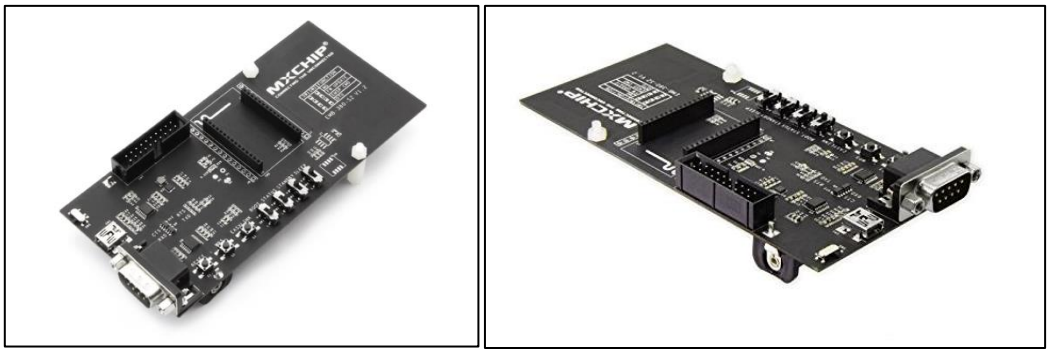


Figura 3.4-1: Tarjeta de desarrollo EMB 380 S2 (seeedstudio.com).

Esta tarjeta de desarrollo es compatible con otros módulos como EMW3280 y cualquier módulo de la serie EMW316x.

2. EMB WICED S

Esta tarjeta tiene a través de microUSB la alimentación y una interfaz UART.

Incluye también, los botones de sleep y standby, cuya función es la misma que en la anterior tarjeta. También, al igual que la anterior tarjeta, permite alimentar al módulo con una pila.

Figura 3.4-2: Tarjeta de desarrollo EMB WICED S (seeedstudio.com).

3. EMW3162 WIFI SHIELD

Esta tarjeta incluye interfaz serie y conexión SWD con 3 pines.

Existe bastante información sobre esta tarjeta. Tiene indicaciones de los pines de los que consta el PCB por detrás de la misma y también es más compacta que las anteriores.

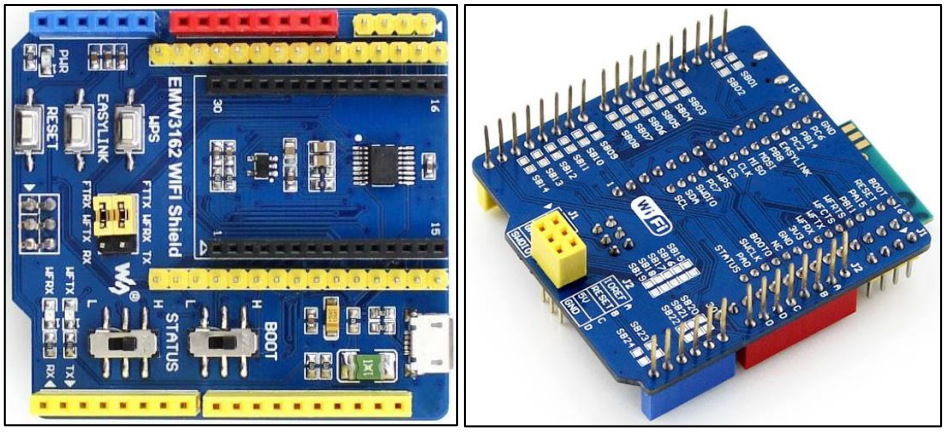


Figura 3.4-3: Tarjeta de desarrollo EMW3162 WIFI SHIELD (waveshare.com).

3.5.-Modos de funcionamiento y actualizaciones

El EMW3162 tiene varios modos de funcionamiento. Entre otras funciones, estos modos, permiten hacer comprobaciones, actualizaciones, cambios de parámetros (de red) y otras operaciones.

En este apartado, se analizan los modos de funcionamiento del EMW3162 y algunos programas que pueden funcionar con alguno de los mismos.

El módulo tiene 4 modos de trabajo a los que se puede acceder a través de los interruptores BOOT y STATUS de la tarjeta de desarrollo. A continuación se muestra una lista de modos y los niveles que deben tener estos interruptores para acceder a cada uno de los mismos.

Tabla 3.5-1: Modos de funcionamiento del EMW3162.

BOOT	STATUS	Working Mode
0	0	Test Mode
0	1 (Default)	Firmware Update Mode
1 (Default)	0	EMSP command Mode
1 (Default)	1 (Default)	Direct Transmission Mode

En caso de no tener una tarjeta de desarrollo, los pines para acceder a uno u otro modo de funcionamiento son los que se muestran en la tabla 3.5-2.

Tabla 3.5-2: Pines para selección de modo de funcionamiento del EMW3162.

Module	MXCHIP BOOT pin*	ST BOOT pin*
EMW3161	PIN 36	PIN 30
EMW3162	PIN 16	PIN 27

Tal como muestra la tabla 3.6-2, para el EMW3162 el pin BOOT, es el 16, y el pin ST (status), es el 27.

Los modos de funcionamiento de EMW3162 son los siguientes:

- “Normal mode”: Este modo permite que el módulo funcione de la forma que se ha configurado. Es el modo habitual de trabajo.

- “Firmware update mode”: Si se usa este modo, se pueden realizar actualizaciones de determinados sectores de memoria (como se ha visto, la tabla 3.2-1 muestra la distribución de memoria del EMW3162).

- “EMSP command mode”: Permite cambiar o leer ciertos parámetros del EMW3162 (AP o ST, obtención de MAC (Media Access Control), parámetros de la UART, etc.) a través de una serie de comandos. El módulo trabajará de la forma que se ha configurado una vez se vuelva al modo “normal mode”. Este modo de funcionamiento es sólo posible si se tiene instalado el FW de fábrica del EMW3162.

- “Test mode”: En este modo, la WLAN se conecta de acuerdo con la configuración realizada en otro modo. De esta forma, recibe datos de la UART, que a través de un protocolo TCP o UDP los envía a través de la misma WLAN automáticamente. También realiza la operación contraria en caso de recibir datos a través de la WLAN. Este modo puede ser una forma de comprobar el correcto funcionamiento de la configuración de la WLAN que se ha realizado en otro modo.

Para actualizar la memoria flash, existen distintas formas. Se puede actualizar en modo "Firmware update mode" si el bootloader está instalado, en modo "EMSP command mode" (con el FW de fábrica instalado) a través de comandos y en "normal mode" a través de la interfaz SWD.

A continuación se expondrán las formas de actualizar a través de los 3 modos posibles. Las indicaciones al principio de este apartado, así como las figuras 3.5-1 y 3.5-2, indican la forma de entrar en cada modo de funcionamiento.

Actualizaciones en modo "Firmware update mode"

En este modo de funcionamiento se puede realizar la actualización de un sector. Permite crear una interfaz gracias al puerto serie, que puede verse a través de, por ejemplo, el Hyperterminal de Windows. Esta interfaz se crea gracias al bootloader. Siempre que el bootloader esté instalado en la zona de memoria que aparece en la tabla 3.2-1 se creará esta interfaz. A continuación se muestra el proceso de actualización de uno de los sectores.

1. Se conecta un terminal serie de la siguiente forma:

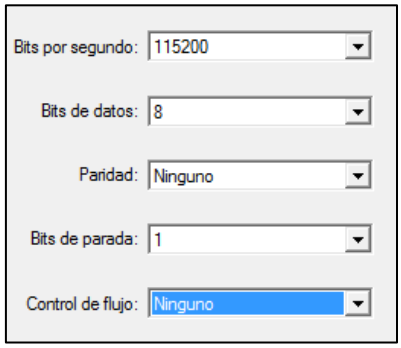


Figura 3.5-1: Configuración de los parámetros de la UART en Hyperterminal.

Los valores que aparecen en la figura 3.5-1 introducidos para configurar el puerto serie, se corresponden con los parámetros por defecto que vienen de fábrica que se pueden ver en la figura 3.2-1.

2. Se pone el módulo en modo "Firmware update mode" y se resetea. A continuación aparece la siguiente interfaz:

```
MICO bootloader for EMW3162, v2.1, HARDWARE_REVISION: 3162
0:BOOTUPDATE,1:FWUPDATE,2:DRIVERUPDAT,3:PARAUPDATE,4:FLASHUPDATE,5:MEMORYMAP,6:B
OOT,7:REBOOT
```

Figura 3.5-2: Interfaz serie en modo "Firmware update mode".

3. Se pulsa el número correspondiente a la tarea a realizar. En este caso, pulsamos 2.

```
MXCHIP> 1

Updating application...
Waiting for the file to be sent ... (press 'a' to abort)
CCCCCC
```

Figura 3.5-3: Espera para recibir un archivo *.bin.

Mientras que están saliendo letras “C”, el módulo está a la espera de recibir un archivo en formato binario (*.bin).

4. Se realiza el envío del archivo binario (el fabricante indica que se debe realizar el envío con protocolo Ymodem):

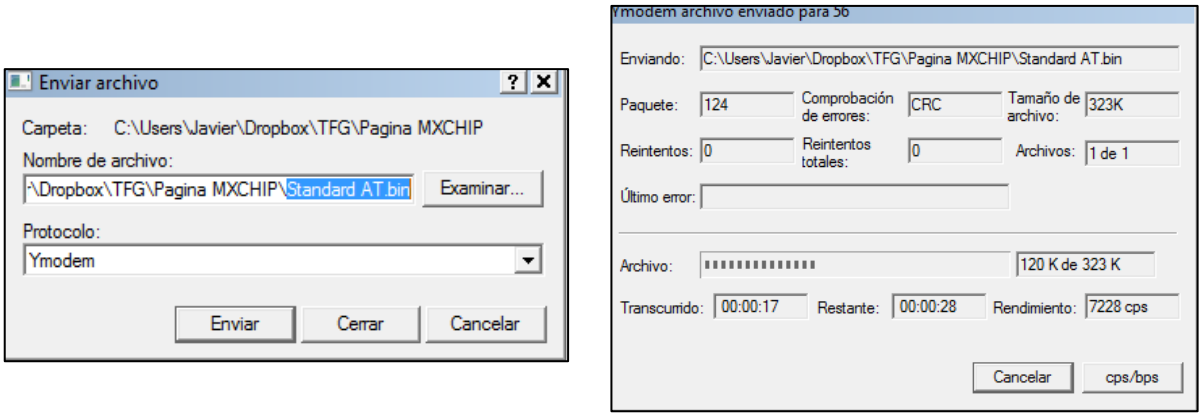


Figura 3.5-4: Selección y proceso de envío de archivo binario a través de UART.

5. Por último, después de estas operaciones, se muestra un mensaje satisfactorio a través de la UART y se tendrá actualizado el sector escogido.

```
Successfully!

Name: Standard AT.bin
Size: 330030     Bytes
```

Figura 3.5-5: Mensaje de actualización satisfactorio.

Actualizaciones en modo “EMSP command mode”

Se pueden realizar ciertos cambios en el funcionamiento del EMW3162 con el FW que viene de fábrica.

Este modo permite modificar parámetros de red u otros del EMW3162 (UART, TCP/UDP, etc.) a través de comandos. Estos parámetros tienen que ver exclusivamente con los estados que puede tener el EMW3162. Obtener la versión de FW, cambiar entre protocolo TCP/UDP, así como comprobar la intensidad de señal de redes del entorno, son unas de las operaciones que permite realizar.

Para que el módulo funcione en este modo, MXCHIP ha creado un programa llamado EMW Tool Box. Este programa tiene una interfaz gráfica en la que se pueden realizar las modificaciones de estos parámetros que se comentan. El programa traduce a comandos las acciones realizadas en la interfaz gráfica.

La figura 3.5-6 muestra una captura del programa EMW Tool Box, donde se pueden diferenciar entre la zona de comandos que se han traducido (derecha) desde la interfaz gráfica (izquierda).

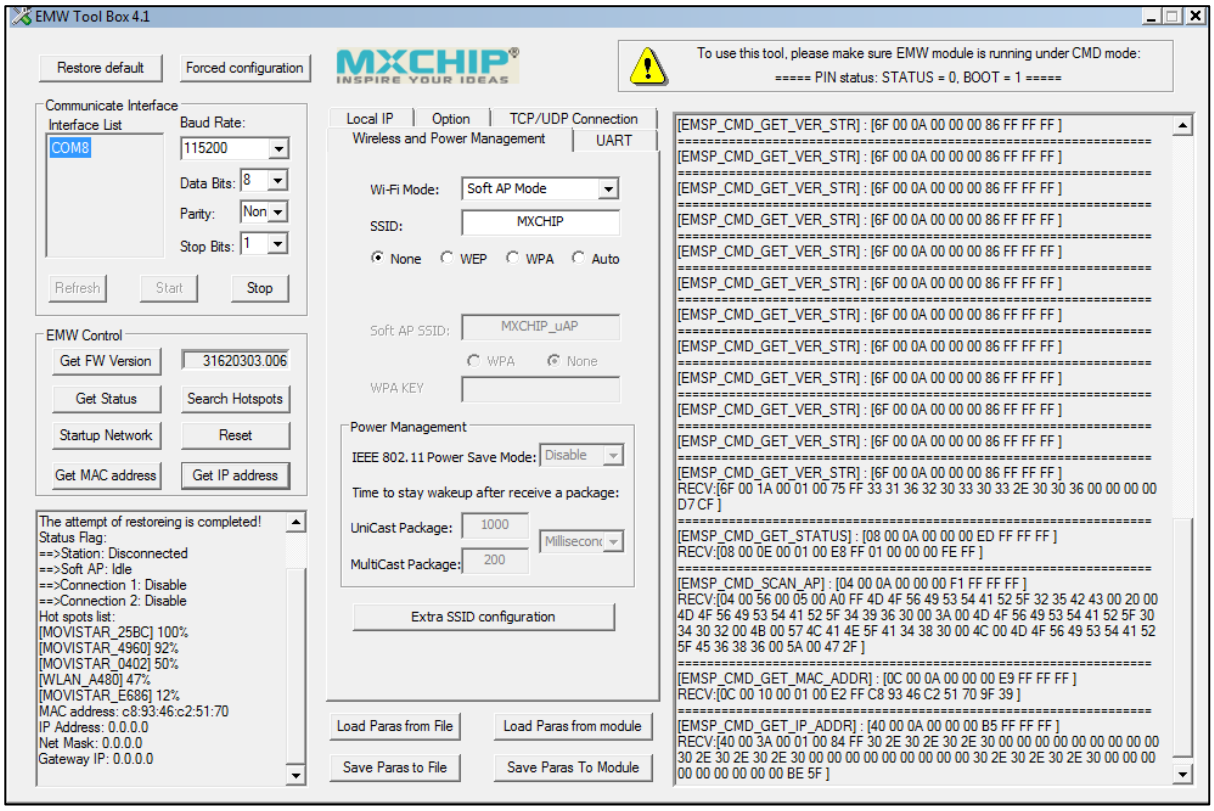


Figura 3.5-6: Programa EMW Tool Box.

EMW Tool Box también tiene una pequeña ventana (se puede apreciar abajo a la izquierda en la figura 3.5-6) en la que se muestran las acciones que se han ido realizando.

Actualizaciones en modo "normal mode"

Generalmente se trabajará con el EMW3162 en este modo de funcionamiento. Este modo, permite actualizar a través de la interfaz SWD que dispone el EMW3162 y otros programas.

Uno de los dispositivos que se pueden usar en este modo es el J-Link de SEGGER, como enlace y su interfaz JTAG, con el programa SEGGER J-Flash, y la tarjeta desarrollo EMB-380-S2.

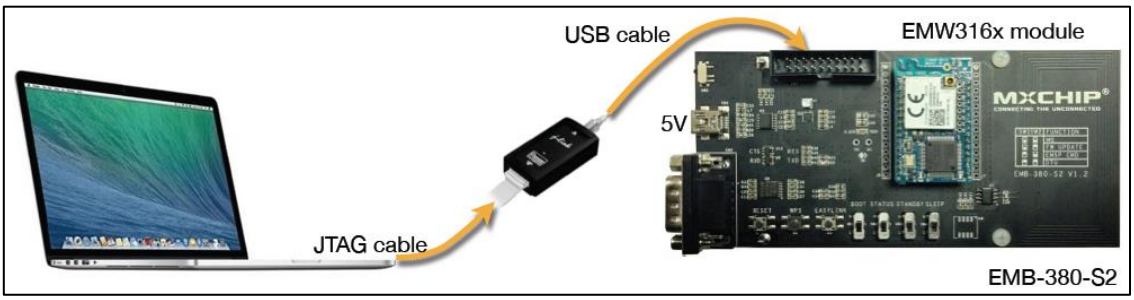


Figura 3.5-7: Interfaz JTAG con tarjeta de desarrollo EMB-380-S2.

Se dará una breve explicación de las herramientas usadas para la programación en el capítulo 4.

3.6.-Comunicación serie-wifi

Con el FW de fábrica, Standard AT, y ahora en "normal mode" se puede tener una comunicación serie-wifi. Ésta, se hace posible con programas que permitan la transferencia serie entre un equipo y el módulo y un cliente TCP. Se ha puesto el nombre de serie-wifi, pero la transferencia de información va en ambos sentidos, en la que dos equipos implicados en la comunicación comparten un único canal y la comunicación va en ambos sentidos.

Para realizar esta comunicación son necesarios dos equipos. En uno será necesario una entrada USB para la comunicación con el puerto COM elegido y en otro una tarjeta de red para comunicarse con el EMW3162 sin cables.

Los pasos para establecer una conexión serie-wifi con los valores de fábrica expuestos en la figura 3.2-1 son los siguientes:

1. Conectarse a la red wifi del módulo con un equipo (EQUIPO2 en este caso) y abrir el programa para la comunicación entre el equipo y la WLAN (Hyperterminal en este caso). Establecer un cliente TCP, poner el puerto (8080 para TCP) y la IP por defecto que lleva el módulo (192.168.1.1). Esta configuración se expone en la figura 3.6-1

2. Configurar el otro equipo (EQUIPO 1 en este caso) con los valores de fábrica para la UART. Se puede ver esta misma configuración necesaria en la figura 3.5-1.

3. Aceptadas las configuraciones en ambos equipos, la conexión serie-wifi está establecida. Ya se pueden enviar y recibir caracteres entre ambos equipos. Las figuras 3.6-2 y 3.6-3 muestran los mensajes que han sido recibidos desde el equipo del otro lado de la comunicación.

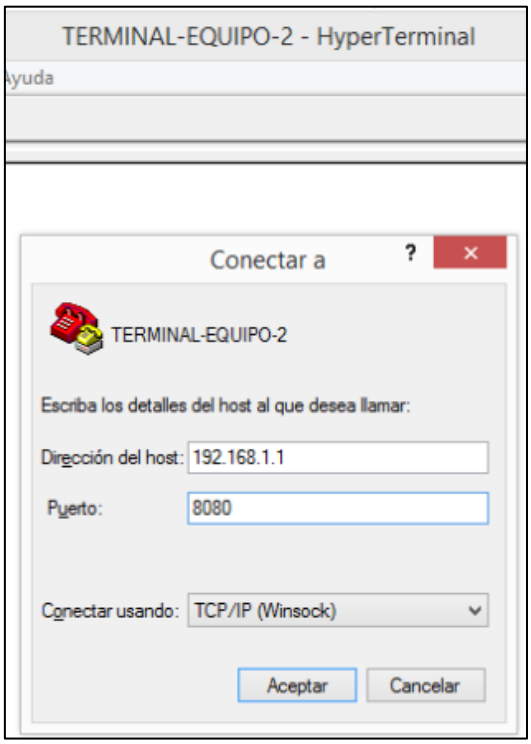


Figura 3.6-1: Terminal con protocolo TCP (EQUIPO 2).

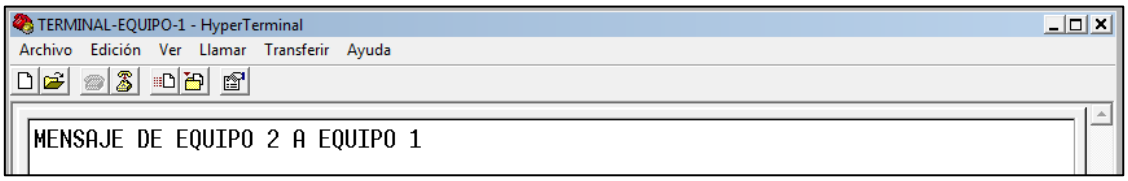


Figura 3.6-2: Transferencia wifi-serie.

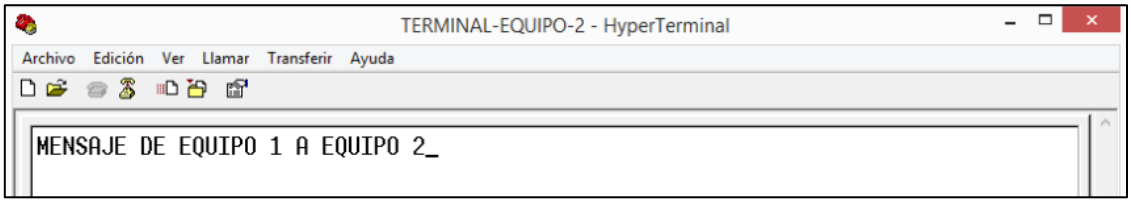


Figura 3.6-3: Transferencia serie-wifi.

Otro programa que aparece en la guía del EMW3162 que permite este tipo de comunicación es TCP232. Con este tipo de programas, únicamente con un módulo EMW3162 y con un PC con tarjeta de red con wifi y un puerto USB, se puede comprobar la conexión serie-wifi.

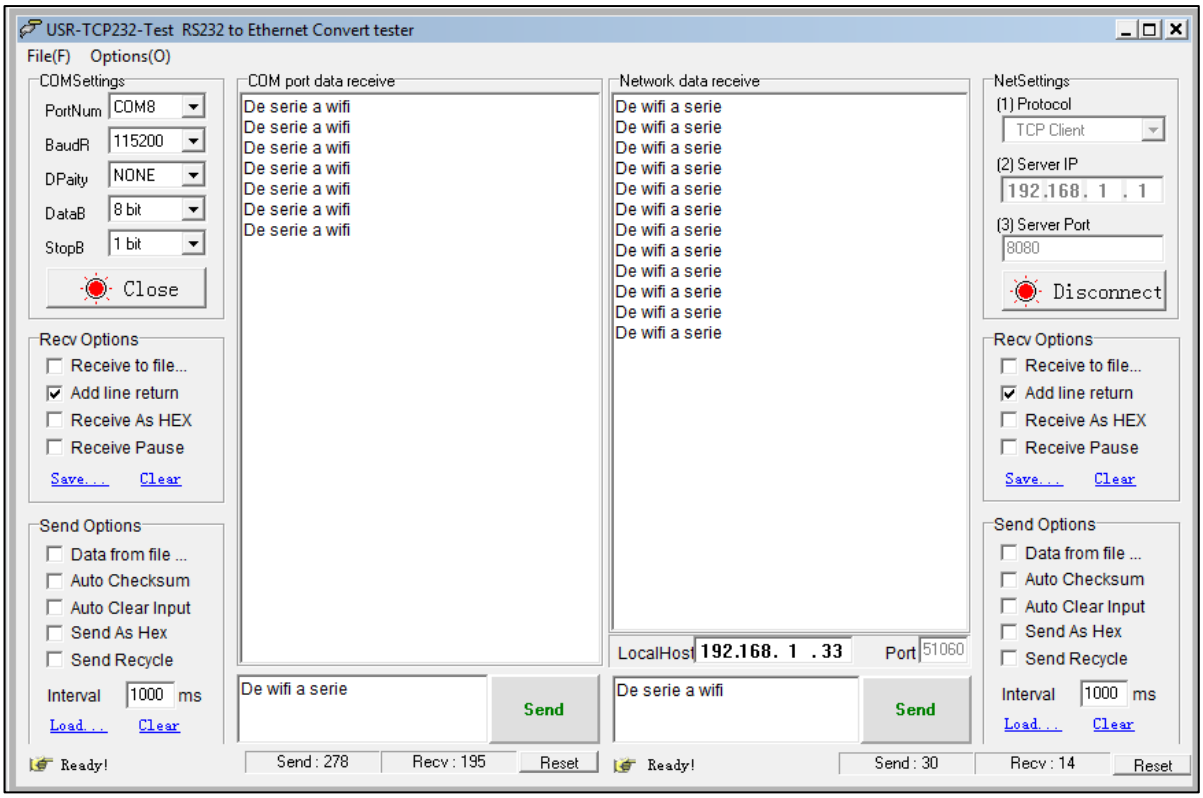


Figura 3.6-4: Comunicación serie-wifi con TCP232.

3.7.-Protocolo OTA

OTA (Over The Air) es un procedimiento concreto que tolera y permite la actualización de FW sin cables. De esta manera, es borrado el FW anteriormente instalado en memoria y el nuevo ocupa su lugar.

El EMW3162, acepta este protocolo y sigue el procedimiento que muestra la figura 3.7-1.

Este procedimiento sirve para actualizar cualquier parte de la memoria que se quiera tal como se muestra en la figura 3.7-2.

Es necesario dejar libre el espacio de memoria “OTA temporary storage” para usar OTA; ya que el FW a actualizar, primeramente se graba en este espacio de memoria para después grabarse en el sector elegido. El bootloader está programado para borrar el sector a actualizar previamente a la instalación del FW.

No será necesario no ocupar el espacio de memoria “OTA temporary storage” si no se va a usar este protocolo.

Este protocolo es usado en el ejemplo que aparece en el subapartado 3.8.4 Webserver.

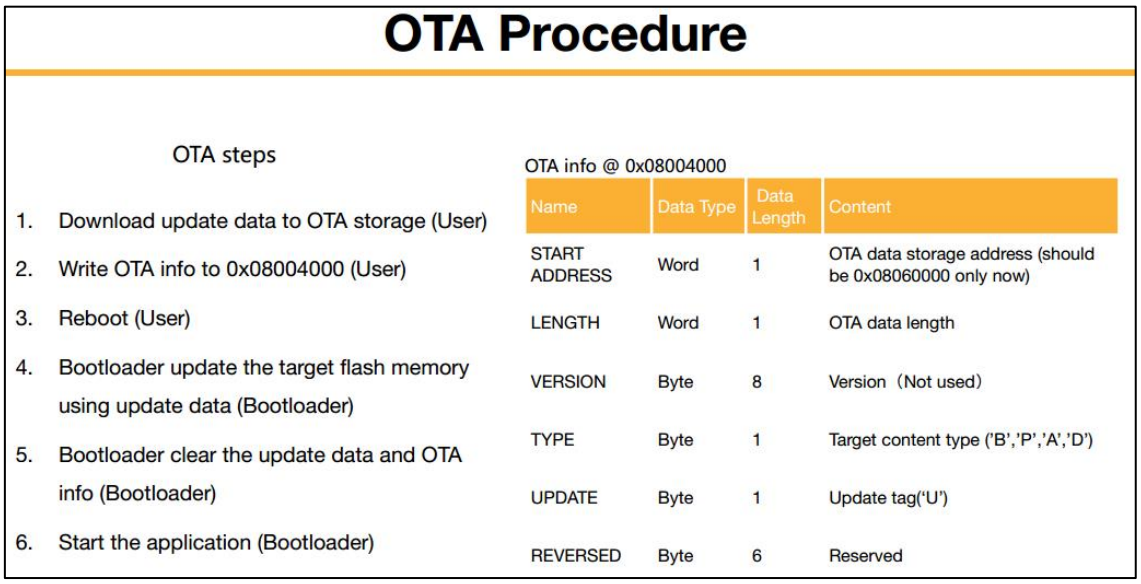


Figura 3.7-1: Procedimiento de actualización vía OTA.

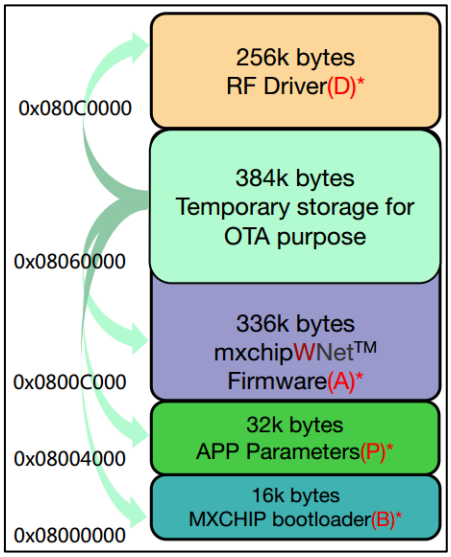


Figura 3.7-2: Procedimiento de posibles actualizaciones vía OTA.

3.8.-Ejemplos proporcionados por el fabricante

En este apartado se van a analizar proyectos realizados por el fabricante que sirven como ejemplos para comprobar ciertas posibilidades que permite el EWM3162. Se hará especial hincapié en el proyecto Webserver OTA, ya que ha servido como base, junto con la librería mxchipWNet para realizar la programación necesaria para la baliza.

Los análisis de los ejemplos (código o proyectos) que se podrán observar a continuación, se realizarán con el bootloader y los drivers RF instalados en los espacio de memoria asignados

por el fabricante. Además se instalará el propio código en el espacio de memoria asignado para las aplicaciones de usuario. Se pueden comprobar estos espacios de memoria en la tabla 3.2-1.

Algunos análisis se realizarán a través de Hyperterminal, que permitirá leer la respuesta que ofrece el EMW3162 a través del puerto serie.

Se debe tener en cuenta que el fabricante indica que la respuesta del EMW3162, a través del puerto serie, puede ser distinta en función de la versión del ejemplo proporcionado. No obstante, la función de cada ejemplo es la misma en las distintas versiones.

3.8.1.-Wi-Fi Link

Este ejemplo nos permite identificar puntos de acceso wifi y sus respectivas intensidades de señal. La figura 3.8-1 muestra una captura de la interfaz de este ejemplo.

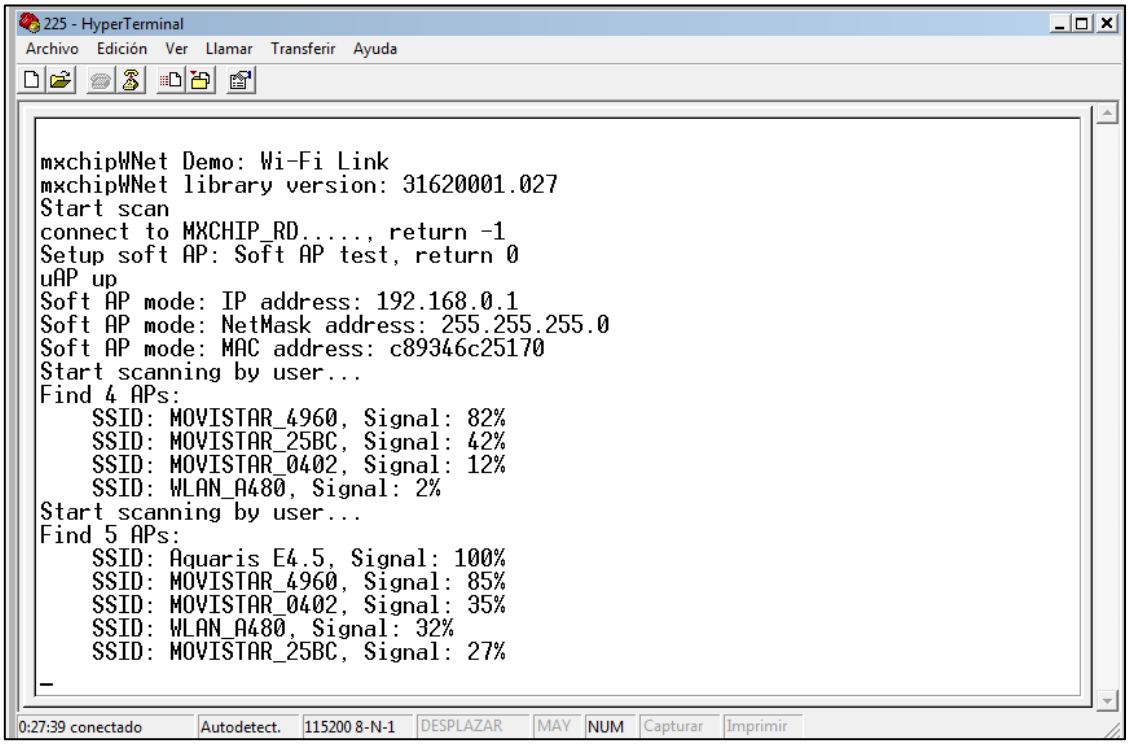


Figura 3.8-1: Interfaz del firmware Wi-Fi Link.

En este caso, en primer lugar, el EMW3162 da información sobre la versión de la librería. También da el valor de la MAC del módulo concreto con el que se trabaja.

Es necesario pulsar el botón WPS para que comience a escanear. Pasados unos segundos nos da los nombres de las redes que detecta y la intensidad de cada una de ellas.

Se puede ver, en la figura 3.8-1, que en este caso, se realizan dos pruebas, y en la segunda, se incluye una red más que en la primera.

3.8.2.-TCP UDP Echo

Con este ejemplo se puede acceder al tamaño del contenido de una web. Para ello el módulo wifi EMW3162 se conecta como ST a la red del rúter. Para acceder a la red del rúter, en el propio proyecto TCP UDP Echo, habrá que cambiar, el AP_NAME (SSID) y la contraseña de acceso (AP_PASSWORD) a la red antes de instalar en el módulo este FW. Si se han escrito correctamente estos valores, el EMW3162 se conectará al rúter.

```
#define AP_NAME             "MOVISTAR_25BC"
#define AP_PASSWORD         "9UPA9FU7UA7EC7CVJU
#define WEB_SERVER          "www.baidu.com"
```

Figura 3.8-2: Modificación de parámetros de red para conectarse a un AP (rúter).

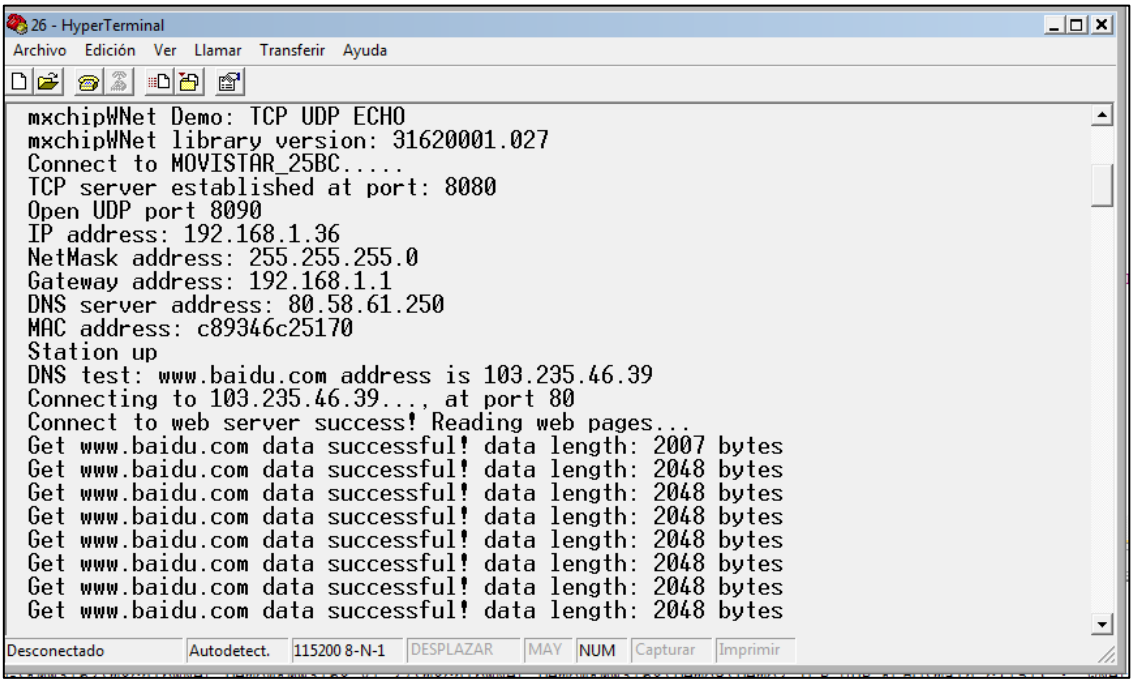


Figura 3.8-3: Interfaz del firmware TCP UDP Echo.

En la figura 3.8-3 se puede ver la respuesta a través de Hyperterminal. El programa envía cierta información de la red y por último se comprueba el tamaño del servidor web introducido. En la figura 3.8-2 se puede ver, a parte del nombre y clave de red, el nombre de la web de la que se recoge información de su tamaño (“www.baidu.com” en este caso).

3.8.3.-WPS Easylink

Este FW demuestra que se puede acceder a una red con el módulo en ST a través del método de autenticación WPS (Wifi Protected Setup).

Ejecutando el módulo con este FW instalado, la interfaz que que aparece es la mostrada en la

figura 3.8-4. Se ve en primer lugar el cuadro con las opciones que nos envía la programación.

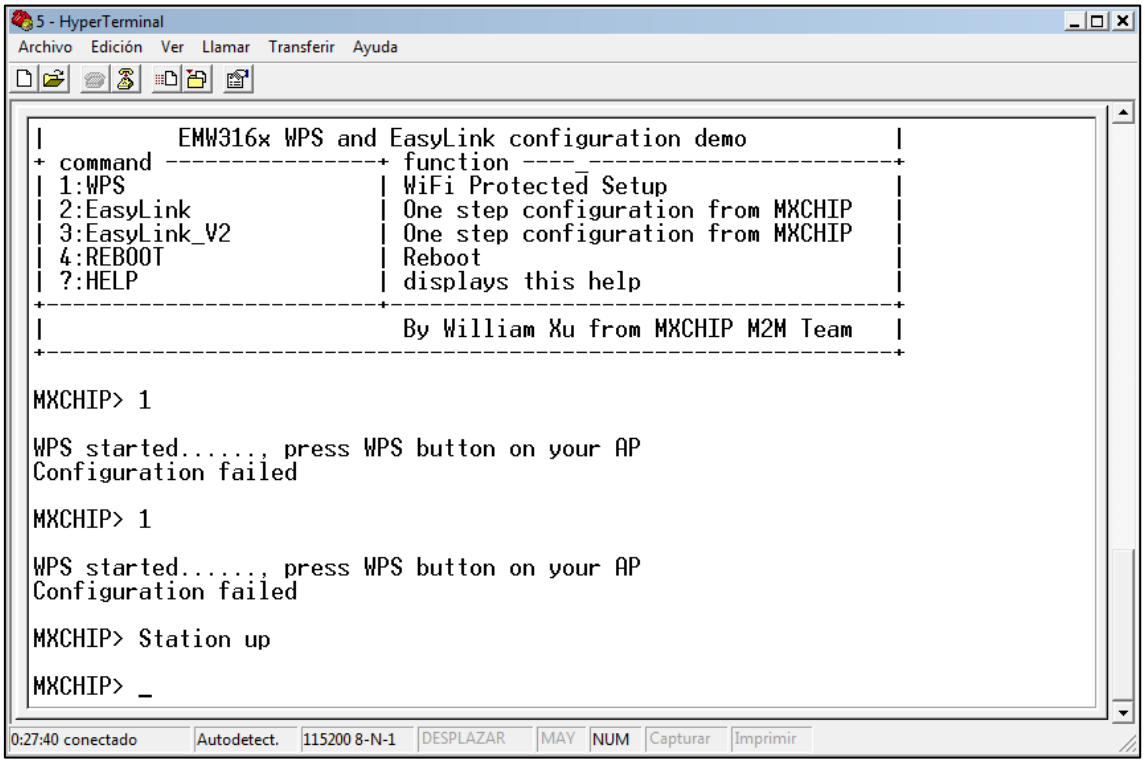


Figura 3.8-4: Interfaz del firmware WPS Easylink.

Se puede comprobar, en la figura 3.8-4, que se han realizado 2 intentos fallidos y un tercer intento correcto. Al usar WPS, al igual que como se realizaría con un móvil u otro equipo hay que pulsar el botón WPS del rúter (en este caso se pulsa el botón WPS del rúter cuando aparece "…. press WPS button on your AP").

Las opciones EasyLink y EasyLink_V2 que muestran la interfaz serie de la figura 3.8-4 se pueden usar si se tiene una aplicación compatible (para Android o iOS) que permita interactuar con el módulo, pero esto queda fuera de los objetivos del proyecto.

3.8.4.-Webserver OTA

El ejemplo Webserver OTA es un servidor con una interfaz web. Éste, permite cambiar parámetros de red (nombre y clave de red), así como actualizar el FW de la zona de usuario de forma inalámbrica.

Con este FW, el EMW3162 funciona como AP. De esta forma se puede acceder a la red que proporciona el módulo a través de un equipo compatible. El navegador mostrará la interfaz que se muestra en la figura 3.8-6.

Cuando se trata de acceder al servidor web, en primer lugar, se muestra un acceso a credenciales de la forma que muestra la figura 3.8-5. Este acceso, por defecto, tiene como

nombre de usuario y contraseña “admin”.

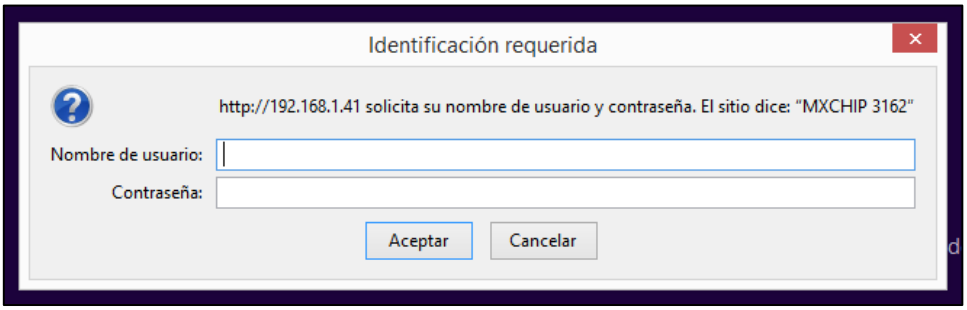


Figura 3.8-5: Acceso a credenciales.

Una vez introducidos estos valores, se muestra la web que muestra la figura 3.8-6.

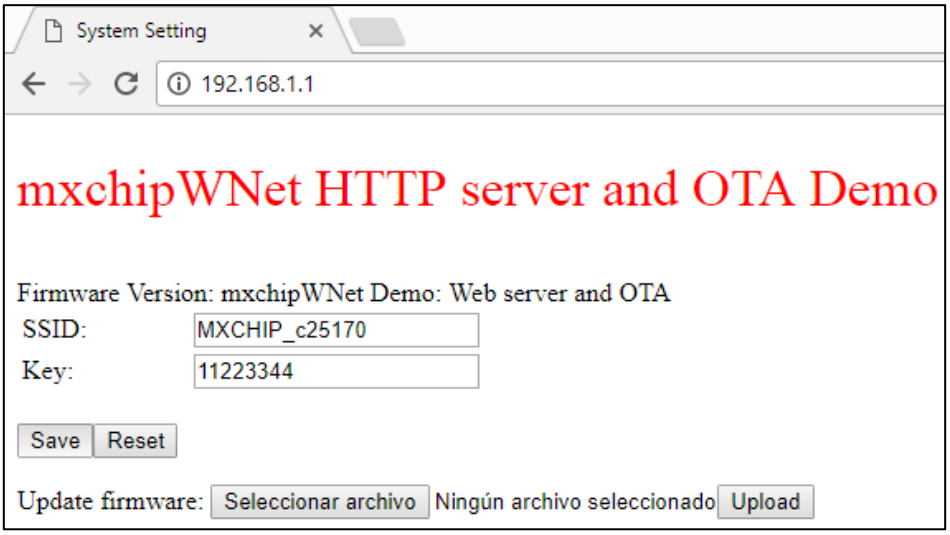


Figura 3.8-6: Interfaz web del firmware Webserver OTA.

El poder modificar parámetros de forma inalámbrica ha sido un aspecto muy importante para el diseño de la baliza, ya que el código se ha podido adaptar para recoger otros parámetros, como los parámetros de modulación para la baliza. La adaptación de este ejemplo, Webserver OTA, junto con el resto de la programación que ha sido necesaria, se mostrará en el capítulo 4.

4.-PROGRAMACIÓN DEL EMW3162

En primer lugar, en el capítulo de introducción, se definieron los parámetros de modulación que se iban a poder modificar a través de la interfaz web (frecuencia, número de bits, ciclos/símbolo, periodo de emisión, secuencia Kasami), en total había 5 parámetros. Se puede comprobar en el subapartado 1.2.2.

En segundo lugar, la librería a usar para desarrollar esta programación es la librería mxchipWNet. Ésta, aportará ciertas funciones que permiten modificar el comportamiento del EMW3162. El anexo A recoge más información de la librería mxchipWNet.

En tercer lugar, en el subapartado 3.8.4, se explicó brevemente la importancia que tenía el ejemplo Webserver OTA para este proyecto. Este FW se ha usado como base para la programación del EMW3162 que se integra la baliza, concretamente del microcontrolador STM32F205, que es el que se puede programar por un usuario del módulo.

Teniendo en cuenta estos tres importantes aspectos, este capítulo se dedica a explicar la programación que se ha llevado a cabo para conseguir los requisitos, que se explicaron, que debía tener la baliza (apartado 1.2).

La figura 4-1 representa las secuencias a la salida del DAC. También presenta el momento de interrupción del timer 3 (TIM3), que se corresponde con el inicio de cada secuencia, y el espacio entre secuencias (L).

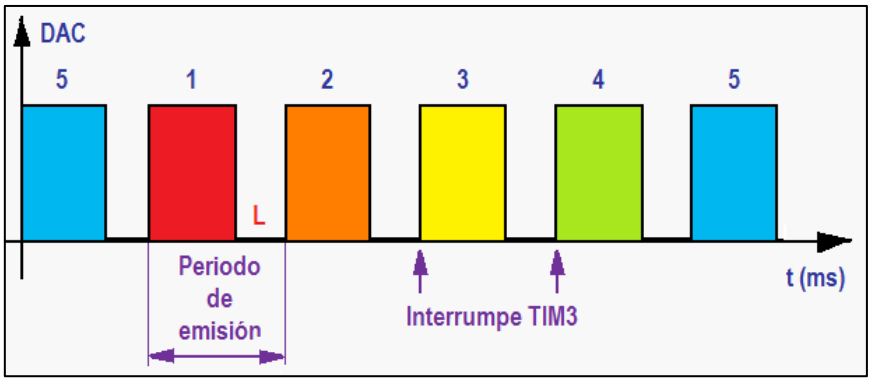


Figura 4-1: Timers involucrados en la generación de secuencias.

Si se hiciera zoom sobre una de estas secuencias, se vería una forma similar a la que aparece en la figura 4-2. Cada escalón es una muestra que presenta el DAC, a su salida, y se mantiene a lo largo de un tiempo. Este tiempo, el que tarda en salir cada muestra, es controlado por la interrupción del TIM2. El TIM2 muestrea o bien a 480 kHz, generando la señal modulada a la salida del DAC de 40 kHz o bien a 500 kHz, con la señal modulada de 41.66 kHz.

Tener 40 kHz o 41.66 kHz implica que se puedan tener señales moduladas de 25us (1/ 40e3) o 24us (1/41.66e3) respectivamente.

La figura 4-2 representa una señal modulada con 25us de periodo. Además indica el instante de interrupción del TIM2.

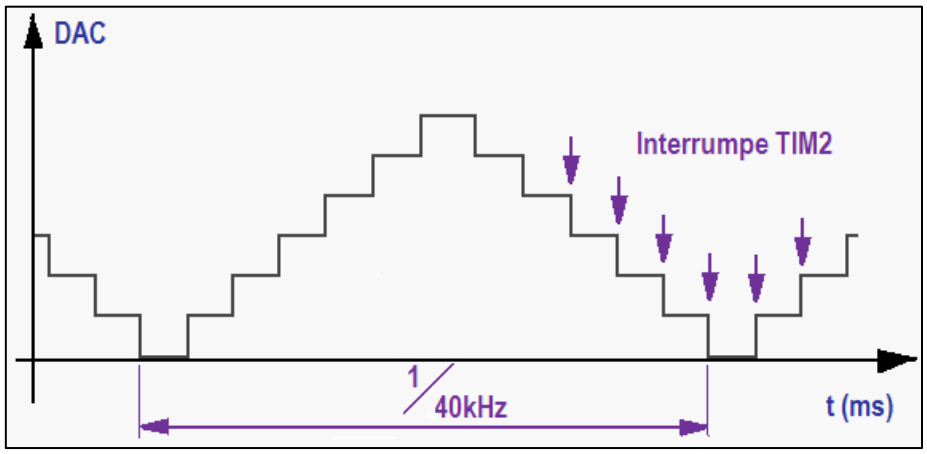


Figura 4-2: Representación de señal e instante de interrupción del TIM2.

Nota: La figura 4-2 muestra un escalonamiento muy estricto. En la práctica, la subida o bajada de un escalón a otro se aprecia más suavizada debido a los tiempos de propagación de la señal a través de la electrónica.

La estructura general del software que permite soportar todos los requerimientos del diseño de la baliza es la que aparece en la figura 4-3, donde se muestra el proyecto Webserver_BPSK con 5 subcarpetas. Las subcarpetas startup, mxchipWNet y STM32_HAL_LIB, contienen el archivo de inicialización del STM32F205, el archivo de la librería mxchipWNet y la configuración del NVIC y drivers del mismo respectivamente.

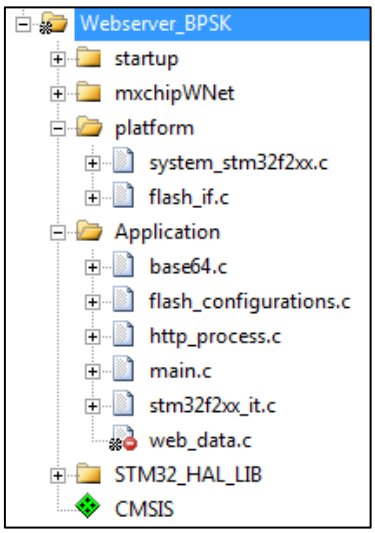


Figura 4-3: Estructura general del software.

Las subcarpetas restantes “platform” y “Application”, que también se pueden ver en la figura 4-3, recogen ciertos archivos que serán comentados a continuación para tener esta idea general de cómo está estructurado el software del STM32F205:

Carpeta platform

- system_stm32f2xx.c: Contiene funciones de inicialización del microcontrolador para la configuración del PLL y las variables que definen la frecuencia principal de reloj como "SystemCoreClock".

- flash_if.c: Proporciona funciones relacionadas con operaciones con la memoria flash del STM32F205.

Carpeta Application

- "base64.c": Contiene funciones de codificación y decodificación binaria para ciertas comunicaciones de red.

- "flash_configurations.c": Este archivo contiene las funciones creadas para grabar y leer la memoria flash.

- "http_process.c": Este archivo es un soporte fundamental para el servidor web. Permite que el servidor responda y funcione acorde al diseño web.

- "main.c": Aquí se configuran los estados de la WLAN y se controlan los procesos de comunicación de la misma. También se inicializan los periféricos usados y las prioridades de los que sean necesarios. Además se definen aquí los parámetros globales de los que consta el diseño y se define la función encargada de generar la modulación.

- "stm32f2xx_it.c": Recoge las funciones de las rutinas principales.

- "web_data.c": Este archivo contiene el código en html que es necesario para la visualización de la web en un dispositivo. Conforma el diseño web.

4.1.-Entorno de programación

El entorno de trabajo en el que se ha desarrollado la programación es keil uVision, versión 5.17.0.0.

Como se ha comentado anteriormente, existen dos microcontroladores en el EMW3162, y el único que se puede programar es el STM32F205. Este microcontrolador tiene una interfaz SWD accesible en el propio EMW3162. Para actualizar la programación se ha adquirido el ST-LINK V2:

Figura 4.1-1: ST-LINK V2 [1075, 2016].

Otro programa que ha resultado de utilidad ha sido el STM32 ST-LINK Utility, con el que se han podido realizar operaciones de actualización, borrado y comprobaciones de memoria.

4.2.-Función "Genera_Kasami_Modulada()"

Existen dos arrays que almacenan las secuencias Kasami y están escritos en memoria. Uno de los arrays, tiene almacenados, para secuencias Kasami de 255 bits, 16 secuencias. El otro, para secuencias de 1023 bits, tiene almacenadas 32 secuencias. La programación trabajará con 255 bits o 1023 dependiendo del número de bits que se escojan en la interfaz web.

"Genera_Kasami_Modulada()" se ejecuta después de emitirse una secuencia, en el espacio de tiempo entre secuencias (representado con la letra "L" en la figura 4-1). Esta función permitirá generar las muestras para la modulación, que posteriormente presentará el DAC a su salida, y almacenar las mismas en formato digital en el array "Kasami_Modulada[]". El array "Kasami_Modulada[]" es de tipo global.

De la figura 4.2-1, se pueden relacionar los símbolos "Simbolo_A[]" y "Simbolo_B[]" con el apartado 1.2.1, donde se explicaba la definición de "ciclo" y de "símbolo".

El array "Kasami_Modulada[]" tiene un número de elementos máximo que puede almacenar. Este número máximo, se consigue escogiendo un número de bits de 1023, 12 muestras/ciclo y 2 ciclos/símbolo. Si se multiplican estos tres valores, se comprueba que el número máximos de elementos almacenables es de 24552.

Las muestras que se almacenen digitalmente en el array "Kasami_Modulada[]" serán espaciadas en el tiempo en función de la interrupción del TIM2, pero siempre, generando una señal modulada de 40 o 41.66 kHz, dependiendo del valor elegido.

```
void Genera_Kasami_Modulada(char n_fila_kasami, int n_bits_kasami,
  char muestras_ciclo, char ciclos_simbolo){
  int a,b,j,i,indice=0;

  int8_t Simbolo_A[12*4];                  // Maximo numero de elementos que puede tomar Simbolo
  int8_t Simbolo_B[12*4];                  // Maximo numero de elementos que puede tomar Simbolo

  for(a=0;a<muestras_ciclo*ciclos_simbolo;a++)Simbolo_A[a]=(255*(1+sin(2*Pi*a/muestras_ciclo))/2);
  for(b=0;b<muestras_ciclo*ciclos_simbolo;b++)Simbolo_B[b]=(255*(1-sin(2*Pi*b/muestras_ciclo))/2);

  if(n_bits_kasami==255){
    for(i=0;i<n_bits_kasami;i++){
      if(Kasami255[n_fila_kasami][i]==+1)
        for(j=0;j<muestras_ciclo*ciclos_simbolo;j++)
      Kasami_Modulada[indice++]=Simbolo_A[j];
      else
        for(j=0;j<muestras_ciclo*ciclos_simbolo;j++)
      Kasami_Modulada[indice++]=Simbolo_B[j];
    }
  }
  else if (n_bits_kasami==1023){
    for(i=0;i<n_bits_kasami;i++){
      if(Kasami1023[n_fila_kasami][i]==+1)
        for(j=0;j<muestras_ciclo*ciclos_simbolo;j++)
      Kasami_Modulada[indice++]=Simbolo_A[j];
      else
        for(j=0;j<muestras_ciclo*ciclos_simbolo;j++)
      Kasami_Modulada[indice++]=Simbolo_B[j];
    }
  }
}
```

Figura 4.2-1: Función “Genera_Kasami_Modulada()”.

Como “Genera_Kasami_Modulada()” se ejecuta en el espacio de tiempo entre secuencias, el espacio de tiempo entre las mismas debe ser superior al tiempo de ejecución de la propia función “Genera_Kasami_Modulada()”. Si esto se cumple se garantiza que ha dado tiempo a generarse la siguiente secuencia correctamente.

Para garantizar que existe un espacio de tiempo entre secuencias suficiente para que se ejecute “Genera_Kasami_Modulada()” se ha comprobado cuánto tarda en ejecutarse la misma, teniendo en cuenta los parámetros que influyen en el tiempo que tarda en ejecutarse. Estos parámetros son: El número de ciclos/símbolo y el número de bits. (Se puede comprobar en la figura 4.2-1, que estos dos parámetros de modulación influyen en el tiempo de ejecución de la función, a través de los bucles “for”).

Esta medición de tiempos, se ha realizado con el reloj del entorno de programación keil uVision. Los resultados aparecen en la tabla 4.2-1.

Tabla 4.2-1: Tiempos de ejecución de “Genera_Kasami_Modulada()”.

Ciclos/símbolo	Número de bits	Tiempo de ejecución (ms)
1	255	1.79540
2	255	3.76202
3	255	5.78392
4	255	7.77385
1	1023	3.11317
2	1023	6.25186

Sabiendo estos tiempos de ejecución anteriores y cuánto dura un ciclo en función de la frecuencia seleccionada, se puede establecer un mínimo de duración para el periodo de emisión que englobe los tiempos de emisión y de generación de las secuencias. De esta forma se tiene:

- Tiempo de emisión de la señal = Duración de ciclo * Número de bits * ciclos/símbolo

- Tiempo de ejecución de Genera_Kasami_Modulada (Tabla 4.2-1).

La suma de estos dos anteriores tiempos será el tiempo mínimo necesario que necesita la programación para la correcta emisión y generación de frecuencias. La figura 4.7-4 muestra, subrayado en azul, varios condicionales que establecen un mínimo de tiempo para garantizar este correcto funcionamiento. Se ha redondeado en exceso para que con estos 4 condicionales se respeten los tiempos al introducir un periodo de emisión.

Nota: Cada prueba expuesta en la tabla 4.2-1 se ha realizado 5 veces. En cada caso se ha escogido el valor más alto.

4.3.-Configuración del PLL

En el anexo B, se puede apreciar que el microcontrolador STM32F205 funciona con un cristal de 26 MHz. Para este proyecto, se ha querido aprovechar el máximo valor que recomienda el fabricante para el reloj principal (120 MHz).

La figura 4.3-1, muestra la configuración elegida para el PLL para el STM32F205 con un cristal externo de 26 MHz. También aparece el valor de la frecuencia del reloj principal (SYSCLK). En rojo se remarcan la frecuencia del reloj principal y la frecuencia de refresco de los timers y periféricos del bus APB1.

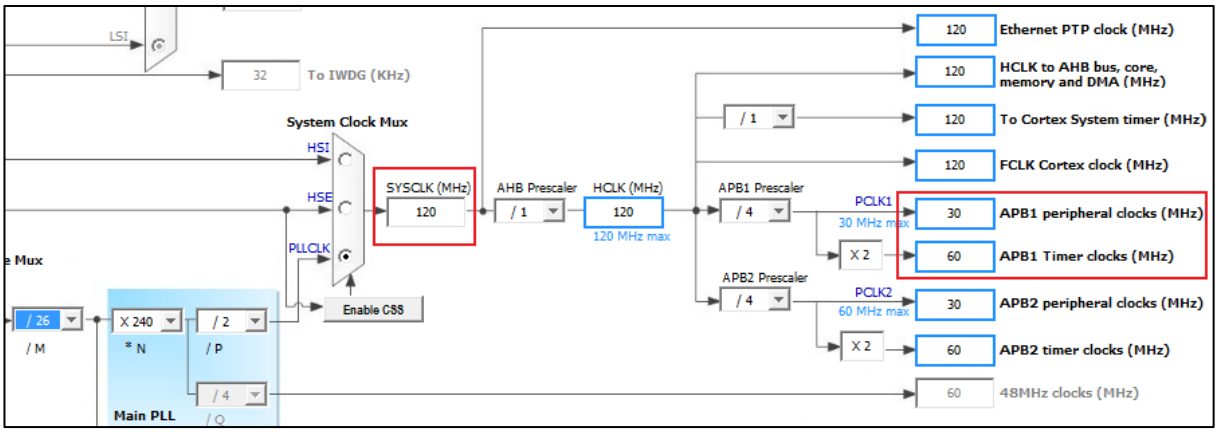


Figura 4.3-1: Configuración del PLL (STM32CubeMX).

4.4.-Prioridades de los periféricos

Ha de darse una preferencia a determinados periféricos que intervendrán en la ejecución del programa. El TIM3 está íntimamente relacionado con el periodo de emisión. Para que este periodo sea fiel al valor introducido en la interfaz web se pone esta interrupción con la mayor prioridad.

Hay 3 prioridades asignadas en el FW Webserver OTA necesarias para ciertas funciones del módulo (por ejemplo, la interfaz SDIO para la comunicación entre microcontroladores). Por lo que éstas no se han tocado. Por ello, el TIM2 tiene prioridad 4 y por último, el TIM4, que se encarga del led de comunicaciones prioridad 5.

La figura 4.4-1, muestra la prioridad otorgada a cada timer. No se han otorgado niveles de subprioridad.

```
void Prioridades(void){
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_4);            //0 niveles de subprioridad

  NVIC_Prioridades.NVIC_IRQChannel = 28;                     //TIM2
  NVIC_Prioridades.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Prioridades.NVIC_IRQChannelPreemptionPriority = 4;
  NVIC_Prioridades.NVIC_IRQChannelSubPriority = 0;
  NVIC_Init(&NVIC_Prioridades);

  NVIC_Prioridades.NVIC_IRQChannel = 29;                     //TIM3
  NVIC_Prioridades.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Prioridades.NVIC_IRQChannelPreemptionPriority = 0;    //Más prioritaria
  NVIC_Prioridades.NVIC_IRQChannelSubPriority = 0;
  NVIC_Init(&NVIC_Prioridades);

  NVIC_Prioridades.NVIC_IRQChannel = 30;                     //TIM4
  NVIC_Prioridades.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Prioridades.NVIC_IRQChannelPreemptionPriority = 5;
  NVIC_Prioridades.NVIC_IRQChannelSubPriority = 0;
  NVIC_Init(&NVIC_Prioridades);
}
```

Figura 4.4-1: Prioridad de los timers.

4.5.-Rutina del Timer 2

La señal modulada a la salida del DAC es siempre de 40 o 41.66 kHz. El TIM2 permite que así lo sea, ya que se configura en función de la frecuencia elegida para lograrlo.

Para conseguir una de estas frecuencias, cada vez que se modifica el número de frecuencia, se modifica también la frecuencia de interrupción del TIM2, que puede ser, como se ha comentado anteriormente, de 480 kHz para conseguir una señal modulada de 40 kHz o de 500 kHz para conseguir una señal de 41.66 kHz. Esta configuración de la frecuencia del TIM2 se puede ver en la figura 4.5-2, donde la función “TIM_SetAutorreload()” modifica la frecuencia de interrupción (480 o 500 kHz) en función de si el sistema opera con los 40 o los 41.66 kHz (41 en la figura para discriminar).

```
void TIM2_IRQHandler (void){
  TIM2->SR &= ~(TIM_SR_UIF);                                        //Flag del maneja

  Static_Kasami_Modulada = Kasami_Modulada[Contador_Timer2++];      //Incremento del

  DAC_SetChannel1Data(DAC_Align_8b_R, Static_Kasami_Modulada);      //Valor de salida

  if (Contador_Timer2 >= (Static_Cuentas_Tim2) && (Static_Flag_Timer == 0)){    //N
    DAC_SetChannel1Data(DAC_Align_8b_R, 0x00);                      //Valor de salida
    Static_Flag_Timer = 1;
    Contador_Timer2=0;                                              //Contador_Timer2
    Genera_Kasami_Modulada(Static_Sec_Ka, Static_Nbits, 12, Static_Ciclos_Simbolo);
    Bit0_mux = (0x00000001 & (Static_Canal_Mux-1));                 //Se adquiere cad
    Bit1_mux = (0x00000002 & (Static_Canal_Mux-1));
    Bit2_mux = (0x00000004 & (Static_Canal_Mux-1));
    GPIOB->ODR &= 0xFFFFFF1F;                                       //1º se resetean
    GPIOB->ODR |= ((Bit0_mux<<5)|(Bit1_mux<<5)|(Bit2_mux<<5));      //2º se ponen en
    TIM_Cmd(TIM2, DISABLE);                                         //Deshabilitación
  }
}
```

Figura 4.5-1 : Rutina del Timer 2.

```
if(Flag_Actualizacion_Param == 0){                                  //Gu
    Flag_Actualizacion_Param = 1;
    config_param.ciclos_simbolo     = Ciclos_Simbolo_Html;
    config_param.frecuencia         = Frecuencia_Html;
    config_param.numero_bits        = Nbits_Html;
    config_param.periodo_emision    = Periodo_Emision_Html;
    config_param.secuencia_transductor1 = Sec_Trans_1;
    config_param.secuencia_transductor2 = Sec_Trans_2;
    config_param.secuencia_transductor3 = Sec_Trans_3;
    config_param.secuencia_transductor4 = Sec_Trans_4;
    config_param.secuencia_transductor5 = Sec_Trans_5;
    Guarda_Config(&config_param);                                   //
    Static_Ciclos_Simbolo  = Ciclos_Simbolo_Html;                   //
    Static_Frecuencia  = Frecuencia_Html;
    Static_Periodo_Emision = Periodo_Emision_Html;
    Static_Nbits = Nbits_Html;
    if(Static_Frecuencia == 40){
      Tim2_TimeBaseInit.TIM_Prescaler = 0x0000;
      Tim2_TimeBaseInit.TIM_CounterMode = TIM_CounterMode_Down;
      Tim2_TimeBaseInit.TIM_Period = 60e6/480e3-1;
      Tim2_TimeBaseInit.TIM_ClockDivision = TIM_CKD_DIV1;
      Tim2_TimeBaseInit.TIM_RepetitionCounter = 0x0000;
      TIM_TimeBaseInit(TIM2, &Tim2_TimeBaseInit);
    }
    else if (Static_Frecuencia == 41){
      Tim2_TimeBaseInit.TIM_Prescaler = 0x0000;
      Tim2_TimeBaseInit.TIM_CounterMode = TIM_CounterMode_Down;
      Tim2_TimeBaseInit.TIM_Period = 60e6/500e3-1;
      Tim2_TimeBaseInit.TIM_ClockDivision = TIM_CKD_DIV1;
      Tim2_TimeBaseInit.TIM_RepetitionCounter = 0x0000;
      TIM_TimeBaseInit(TIM2, &Tim2_TimeBaseInit);
    }
    TIM_SetAutoreload(TIM3,(Static_Periodo_Emision-1));
    TIM_SetCounter(TIM2, 0);
  }
```

Figura 4.5-2: condicional para actualización de parámetros (Timer 3).

La variable “Contador_Timer_2” se incrementa a cada paso por esta rutina y el valor actual del

elemento “Kasami_Modulada[]” se lleva a la salida del DAC gracias a la función “DAC_SetChannel1Data()”. La variable “Contador_Timer_2” se resetea cuando alcanza el valor de “Static_Cuentas_Tim2”. Esta variable, “Static_Cuentas_Tim2”, se actualiza cada vez que se realiza la rutina del TIM3 (se puede comprobar en la figura 4.6-1). De esta forma se consigue extraer, del array “Kasami_Modulada[]”, un número de muestras igual al número de muestras almacenado en este mismo array a través de la función “Genera_Kasami_Modulada()”.

Por último, en el “if()”, se genera la siguiente secuencia para el siguiente transductor y se programan los tres bits para la información de la demultiplexación.

Nota: La figura 4.5-2 forma parte de la rutina del TIM3.

4.6.-Rutina del Timer 3

La rutina del TIM3 está relacionada con el periodo de emisión. Éste parámetro se puede modificar en la interfaz web. El periodo de emisión está programado para trabajar en múltiplos de 1ms.

La figura 4.6-1 muestra la rutina del TIM3. En la primera parte de la rutina se actualizan todos los parámetros de modulación y, se guardan en memoria flash los mismos, en caso de haber realizado una actualización en el servidor web (esto se puede ver en la figura 4.5-2, donde aparece el condicional obviado de la figura 4.6-1).

En la rutina del TIM3, también se configura, como se ha comentado, la frecuencia de interrupción del TIM2. Además, se configura la frecuencia de interrupción del TIM3.

En la figura 4.6-1, se puede comprobar además, que se establece el número de muestras totales que deben salir por el DAC, a través de la variable “Static_Cuentas_Tim2”, (como se comentó en el apartado 4.5). A continuación el programa realiza una elección del transductor siguiente (representado como “Static_Numero_Baliza”) y, a través de otra variable (“Static_Canal_Mux”), se activan las salidas para el canal, que corresponda, del multiplexor. Finalmente se activa la cuenta del TIM2 de nuevo para que se pueda continuar con la emisión de las muestras de la secuencia siguiente y se realiza un chequeo del led de comprobación de la conexión.

```
void TIM3_IRQHandler (void){  //Establecido con prescaler para múltip
  TIM_TimeBaseInitTypeDef Tim2_TimeBaseInit;                    //Cre
  Static_Flag_Timer = 0;                                        //Fla
  TIM3->SR &= ~(TIM_SR_UIF);                                    //Fla

  if(Flag_Actualizacion_Param == 0){                          //Guard
    TIM_Cmd(TIM2, ENABLE);                  //Habilitación de la fuer

    Static_Cuentas_Tim2 = Static_Nbits*12*Static_Ciclos_Simbolo;  //A
    switch (Static_Numero_Transductor){               //Se carga una
      case 1:Static_Sec_Ka = Sec_Trans_2;break;
      case 2:Static_Sec_Ka = Sec_Trans_3;break;
      case 3:Static_Sec_Ka = Sec_Trans_4;break;
      case 4:Static_Sec_Ka = Sec_Trans_5;break;
      case 5:Static_Sec_Ka = Sec_Trans_1;break;
    }
    Static_Canal_Mux++;                             //Static_Canal_Mux
    Static_Numero_Transductor++;                    //Transductor sigui
    if (Static_Canal_Mux >= 6){                     //Reinicio de varia
      Static_Numero_Transductor = 1;
      Static_Canal_Mux=1;
    }
    Static_Led_Conexion = GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_1);
    if(Static_Led_Conexion)                                      //Led rojo
      GPIO_WriteBit(GPIOB, GPIO_Pin_14, Bit_RESET);     //PB14 = pin
    else
      GPIO_WriteBit(GPIOB, GPIO_Pin_14, Bit_SET);       //PB14 = pin

}
```

Figura 4.6-1: Rutina del Timer 3.

4.7.-El servidor web y su entorno

Como se ha comentado, el FW Webserver OTA ha sido importante para la programación del EMW3162 de la baliza. Anteriormente, en el subapartado 3.8.4, se vieron archivos importantes de este FW. En este apartado, se profundizará sobre algunos de ellos, comentando funciones importantes y las modificaciones necesarias en algunas de las mismas para el diseño del servidor web.

Para el entendimiento correcto de este apartado, es importante tener en cuenta el diseño web que se ha implementado. Este diseño web aparece en la figura 1.2-8.

4.7.1.-Archivos “http_process”

Estos archivos están formados por “http_process.c” y “http_process.h”. Son los archivos más importantes del FW Webserver OTA. Se ocupan de que las páginas web respondan acorde a su contexto. Existen bastantes funciones en estos archivos cuya extensión es, en algunas, bastante amplia. Primero se comentarán las funciones más importantes no modificadas, para luego explicar las funciones más importantes que han sido modificadas o creadas, escapando del análisis otras funciones, consideradas para este proyecto, de menor importancia.

Las funciones más importantes del archivo “http_process.c” no modificadas son:

- **"http_tick()"**: En esta función están programados los sockets del servidor web.

- **"auth_init()"**: Esta función permite la creación del mensaje de acceso a credenciales que aparece previo a la web principal del FW Webserver OTA y que reclama nombre de usuario y contraseña (este mensaje es el que presenta la figura 3.8-5).

- **"PostParse()"**: Realiza una comparación sintáctica del código html que se elija con el formulario POST correspondiente. Esta función es muy importante, ya que permite comprobar qué valores se han introducido en la interfaz web.

Por otro lado, las funciones que han sido modificadas son:

- **"HandleHttpClient()"**: Esta función es muy extensa, pero a groso modo, su función principal es recoger la solicitud de un cliente. Cuando existe una solicitud, se compara en primer lugar, el tipo de petición que se ha escogido: GET o POST. Después cada petición se comprueba la web que ha sido solicitada. La figura 4.7-1 muestra la función completa, las figuras 4.7-2 y 4.7-3 muestran los tipos de formularios, GET y tipo POST respectivamente.

La figura 4.7-1 muestra la discriminación entre el tipo de formulario GET y POST. Dependiendo de la web a través de la que se realiza una solicitud se ejecutará un tipo u otro. Una petición GET se ejecuta cuando el cliente o bien trata de acceder a la web principal ("/system.htm" en la figura 4.7-2) o cuando se pulsa uno de los hipervínculos de la web principal (TRANS X en referencia al número de transductor, que se puede observar en la figura 1.2-8) para acceder al número de secuencia Kasami de un transductor. Si se ha pulsado uno de los botones "OK" presentados en la figura 1.2-8, se habrá realizado una petición tipo POST y se entrará en una de las funciones que permiten realizar un análisis sintáctico para obtener los valores de los parámetros de modulación a través de "get_settings_param()" o "get_settings_param_1()" (se puede observar su acceso en la figura 4.7-3.

- **"get_settings_param()"**: En el FW Webserver OTA, esta función servía para recoger el nombre de usuario y clave que se introdujeran, estos valores se recogían gracias a realizar un análisis sintáctico de determinados caracteres en html. Por ello se ha adaptado esta función, para que realice un análisis sintáctico (a través de la función "PostParse()") pero de los parámetros número de bits ("NB": en referencia número de bits, o bien 255 o bien 1023), frecuencia ("FR"), ciclos/símbolo ("CS") y periodo de emisión ("PS"). La figura 4.7-4 muestra la función.

También en la función "HandleHttpClient()", existen las limitaciones comentadas para respetar los tiempos de emisión y generación de las secuencias necesarios que se mostraban en la tabla 4.2-1. En caso de ser correctos los parámetros de modulación (NB: Número de bits, FR: Frecuencia, CS: Ciclos/símbolo y PE: periodo de emisión), se pasan los valores auxiliares a las variables globales (Frecuencia_Html, Ciclos_Simbolo_Html, etc.).

El último condicional, de esta función, configura los códigos de las secuencias emitidas a valores por defecto en caso de haber trabajado previamente con 1023 bits y una secuencia de valor superior a 16, y haber metido una secuencia Kasami de 255 bits, ya que éstas no tienen valores superiores a 16 (como se comentó en el subapartado 1.2.2).

```
150 static void HandleHttpClient(int index){
151   int NumOfBytes;
152   httpToken_struct httpToken ={0,0,0};
153   char *p_auth;
154
155   msleep(200);                                                    //sleep 200
156   NumOfBytes = recv(index, httpRequest, HTTP_DATA_MAX_LEN, 0);
157   if (NumOfBytes < 0) {
158     return;
159   } else if (NumOfBytes == 0) {
160     return;
161   }
162
163   httpRequest[NumOfBytes] = '\0';
164   if(!HTTPParse(httpRequest,&httpToken))
165     goto EXIT;;                                                   //http requ
166
167     p_auth = strstr(httpToken.pToken2, auth_str);                 //Autentica
168   if (p_auth == NULL) {                                           // un-authr
169         send_http_data(index, (char*)authrized, strlen(authrized));
170         goto EXIT;
171   } else {
172         p_auth += strlen(auth_str);
173         if (*p_auth != 0x0d) {
174             send_http_data(index, (char*)authrized, strlen(authrized));
175             goto EXIT;
176         }
177     }
178
179   if(!strcmp(httpToken.pToken1, "GET"))
180   {
210   else if(!strcmp(httpToken.pToken1, "POST"))
211   {
223 EXIT:
224
225   return;
226 }
```

Figura 4.7-1: Función "HandleHttpClient()".

- **"get_settings_param_1 ()"**: Realiza la misma operación que "get_settings_param()" pero con el valor de un número de secuencia Kasami de un transductor.

- **"send_system_page ()"**: Esta función permite almacenar el bloque de memoria de la página web principal. Además, se ha añadido un "if()" que permite iniciar la cuenta del TIM4 para que se active la rutina del led de comunicaciones (explicado en el apartado 4.10). La figura 4.7-5 muestra esta función, con el condicional añadido en azul.

- **"send_system_page_2 ()"**: Esta función permite almacenar el bloque de memoria de la web del transductor. Sólo es necesario esta única función para los 5 transductores, ya que se puede reutilizar con cada una de ellos.

```
if(!strcmp(httpToken.pToken1, "GET"))
{
  if(!strncmp(httpToken.pToken2, "/system.htm", strlen("/system.htm"))){
    send_system_page(index);
  }//Web principal
  if(!strncmp(httpToken.pToken2, "/transductor1.htm", strlen("/transductor1.htm")))  {
    Static_Numero_Transductor_Process=1;
    send_system_page_2(index);
  }//Web Transductor 1
  if(!strncmp(httpToken.pToken2, "/transductor2.htm", strlen("/transductor2.htm")))  {
    Static_Numero_Transductor_Process=2;
    send_system_page_2(index);
  }//Web Transductor 2
  if(!strncmp(httpToken.pToken2, "/transductor3.htm", strlen("/transductor3.htm")))  {
    Static_Numero_Transductor_Process=3;
    send_system_page_2(index);
  }//Web Transductor 3
  if(!strncmp(httpToken.pToken2, "/transductor4.htm", strlen("/transductor4.htm")))  {
    Static_Numero_Transductor_Process=4;
    send_system_page_2(index);
  }//Web Transductor 4
  if(!strncmp(httpToken.pToken2, "/transductor5.htm", strlen("/transductor5.htm")))  {
    Static_Numero_Transductor_Process=5;
    send_system_page_2(index);
  }//Web Transductor 5
  else if(!strncmp(httpToken.pToken2, "/ ", 2))  {
    send_system_page(index);
  } else {
    send_http_data(index, (char *)not_found, strlen(not_found));
  }//No se encuentra ningún dominio de los anteriores
}
```

Figura 4.7-2: Solicitud tipo GET.

```
else if(!strcmp(httpToken.pToken1, "POST"))
{
  NVIC_ClearPendingIRQ(TIM2_IRQn);                              //Quitar, en caso de TIM2 esté pe
  TIM2->CR1 &= ~(0x1);                                          //Deshabilitacion de cuenta

  if(!strncmp(httpToken.pToken2, "/paramconfig.htm", strlen("/paramconfig.htm")))  {
    get_settings_param(index, httpToken.pToken2, NumOfBytes - (httpToken.pToken2-httpRequest));
  }
  if(!strncmp(httpToken.pToken2, "/paramconfig1.htm", strlen("/paramconfig1.htm")))  {
    get_settings_param_1(index, httpToken.pToken2, NumOfBytes - (httpToken.pToken2-httpRequest));
  }
}
```

Figura 4.7-3: Solicitud tipo POST.

- **“save_reset_Response()”**: Permite lanzar mensajes correctos o erróneos a través de la interfaz web en función de los parámetros de modulación que han sido introducidos. La figura 4.7-6 muestra esta función con los tipos de respuesta posibles para las diferentes configuraciones (seguir las indicaciones que aparecen en la web, configuración correcta, conflicto con el periodo de emisión, etc). Todas estas posibles respuestas en la interfaz web se pueden ver en el capítulo 5 en el diagrama de navegación y enlazan con la explicación del subapartado 4.7.2 donde se explica el código html de las mismas.

```
static void get_settings_param(int index, char *postdata, int len){
  //get_param(index, httpToken.pToken2, NumOfBytes - (httpToken.pToken2-httpRequest));
  char* pToken1, *pValue;
  char ciclos_simbolo_aux, frecuencia_aux;
  int periodo_emision_aux;
  int nbits_aux;

  pToken1  = postdata;

  PostParse(&pToken1,"FR",&pValue);                                    //frecuencia
  frecuencia_aux = atoi (pValue);
  PostParse(&pToken1,"NB",&pValue);                                    //kasami 255 o 1023
  nbits_aux = atoi (pValue);
  PostParse(&pToken1,"CS",&pValue);                                    //Ciclos/simbolo
  ciclos_simbolo_aux = atoi (pValue);
  PostParse(&pToken1,"PE",&pValue);                                    //Periodo de emisión
  periodo_emision_aux = atoi(pValue);

  if ((nbits_aux == 0x3FF) && (ciclos_simbolo_aux > 2)){
    save_reset_Response(FALSE, index);
    return;
  }
  else if ((nbits_aux == 0xFF) && (frecuencia_aux == 40) && ((((25*ciclos_simbolo_aux*nbits_aux)/1000) +
    (2*ciclos_simbolo_aux)) >= (periodo_emision_aux))){     //25us*ciclo_simbolo*nbits>=periodo_emisión (ms)
  else if ((nbits_aux == 0xFF) && (frecuencia_aux == 41) && ((((24*ciclos_simbolo_aux*nbits_aux)/1000) +
    (2*ciclos_simbolo_aux)) >= (periodo_emision_aux))){     //24us*ciclo_simbolo*nbits>=periodo_emisión (ms)
  else if ((nbits_aux == 0x3FF) && (frecuencia_aux == 40) && ((((25*ciclos_simbolo_aux*nbits_aux)/1000) +
    (4*ciclos_simbolo_aux)) >= (periodo_emision_aux))){     //25us*ciclo_simbolo*nbits>=periodo_emisión (ms)
  else if ((nbits_aux == 0x3FF) && (frecuencia_aux == 41) && ((((24*ciclos_simbolo_aux*nbits_aux)/1000) +
    (4*ciclos_simbolo_aux)) >= (periodo_emision_aux))){     //24us*ciclo_simbolo*nbits>=periodo_emisión (ms)
  else if(strstr(pToken1, "botonokey")){
}
```

Figura 4.7-4: Función “get_settings_param”.

```
static void send_system_page(int index){
  char *body;
  u32 NumOfBytes;

#define FORMAT_POST_STR sprintf(body, systemPage, APP_INFO,\
            configParas.wifi_ssid, configParas.wifi_key);

  memset(httpRequest,0,HTTP_DATA_MAX_LEN);//Permite iniciar bloque
  //2 valor para iniciar cada byte del bloque,3 numero de bytes del
  body = httpRequest;
  FORMAT_POST_STR;
  sprintf(httpRequest, headerPage, strlen(body));    // recalute the
  body = httpRequest+strlen(httpRequest);            //Posicion + ta
  FORMAT_POST_STR;
  NumOfBytes = strlen(httpRequest);

  send_http_data(index, httpRequest, NumOfBytes);

  if (Flag_Led==0){
          TIM4->CNT = 0;
          TIM4->CR1  |= 0x1;
          Flag_Led =1;
  }
}
```

Figura 4.7-5: Función “send_system_page()” y flag para led de comunicaciones..

```
void save_reset_Response(u8 result, int index){

  if(result == FALSE)
  {
    sprintf(httpRequest, HTTPSaveResponse,
    strlen(SaveResponseError), SaveResponseError);
  }
  else if(result == 3)
  {
    sprintf(httpRequest, HTTPSaveResponse,
    strlen(RespuestaParametros), RespuestaParametros);
  }
    else if(result == 4)
  {
    sprintf(httpRequest, HTTPSaveResponse,
    strlen(RespuestaIncorrectaPeriodo), RespuestaIncorrectaPeriodo);
  }
    else if(result == 5)
  {
    sprintf(httpRequest, HTTPSaveResponse,
    strlen(RespuestaIncorrectaParentesis), RespuestaIncorrectaParentesis);
  }

  send_http_data(index, httpRequest, strlen(httpRequest));
}
```

Figura 4.7-6: Función "save_reset_Response()".

4.7.2.-Archivo "web_data.c"

Este archivo, "web_data.c", contiene todo el código en html, que servirá para mostrar al usuario la web que corresponda en cada momento.

En este archivo existen varias cadenas de caracteres, como "authrized[]" o "not_found[]". En lo que compete a este proyecto, se comentará el código html modificado o creado, profundizando en las dos webs principales (mostradas en el capítulo 1, figura 1.2-8).

Las siguientes 4 cadenas de caracteres permiten visualizar mensajes en la interfaz web cuando sea oportuno. Este aspecto enlaza con lo comentado sobre la figura 4.7-6 del anterior apartado. También se pueden visualizar estos mensajes, como se ha comentado en la figura 5-1.

- **"SaveResponseError[]":** Permite emitir un mensaje de error cuando se intenta trabajar con más de 2 ciclos/símbolo y 1023 bits.

- **"RespuestaParametros[]"**: Permite emitir un mensaje satisfactorio si la configuración de parámetros introducida es correcta. Este código html ha sido creado.

- **"RespuestaIncorrectaPeriodo[]"**: Permite emitir un mensaje insatisfactorio en caso de haber introducido un periodo de emisión que no respeta los espacios de tiempo necesarios para la generación y ejecución de una secuencia. Este código ha sido creado.

- **"RespuestaIncorrectaParentesis[]"**: Permite mostrar un mensaje de error cuando se trata de utilizar un número de secuencia Kasami mayor de 16 cuando se trabaja con 255 bits.

- **"systemPage[]"**: El código de esta cadena de caracteres ha sido modificado casi en su

totalidad. Esta cadena, permite mostrar la web principal. Está formada por los siguientes elementos destacables, que se pueden ver en la figura 4.7-8 de arriba abajo: Color de fondo elegido, creación de hipervínculos las webs de los transductores, formulario tipo POST para los parámetros de modulación frecuencia, número de bits, ciclos/símbolo y periodo de emisión. Por último se puede ver "botonokey", que es el nombre en html del botón "OK" de la web principal, que permite que se recojan los parámetros de modulación en las variables globales (se puede ver al final de la figura 4.7-4), donde se realizaba el análisis sintáctico necesario para recoger los parámetros de modulación.

- **"Transductor1htm[]"**: En esta cadena de caracteres, se crea otro formulario tipo POST y se imponen ciertos límites para introducción de algunos parámetros. Sólo es necesaria esta cadena de caracteres para la web de todos los transductores, ya que al ser valores independientes, se puede reutilizar la web, con el mismo análisis sintáctico de "KS" (en referencia a Kasami y número de secuencia) para cualquier transductor. Se puede observar esta cadena de caracteres en la figura 4.7-7.

Al igual que la cadena de caracteres del diseño de la web principal, tiene un valor "botonokey" para el botón "OK" de la página web del transductor.

Esta función está relacionada con la función "send_system_page_2 ()", que almacenaba el bloque de memoria de la web del transductor.

```
const char Transductor1htm[]={
"<html>\r\n\
<head>\r\n\
<head>\r\n\
<style type=\"text/css\">\r\n\
body{background-color:orange}\r\n\
width=\"480px\"\r\n\
</style>\r\n\
</head>\r\n\
<title>Configurador de número de secuencias</title>\r\n\
</head>\r\n\
<body>\r\n\
<p><font size=\"5\" color=\"blue\"><b>Número de secuencia</b></font><p>\r\n\
<form action=\"paramconfig1.htm\" method=\"post\">\r\n\
<p align=\"left\"><font size=\"3\"><b>Secuencia Kasami</b><br>(De 1 a 16 para 255 bits. \r\n\
De 1 a 32 para 1023 bits)<br/></font><p>\r\n\
<p align=\"left\"><font size=\"3\"><Input type=\"number\" min=\"1\" max=\"32\" name=\"KS\" value = \r\n\
\"%c\"/></font><p>\r\n\
<p align=\"left\"><font size=\"5\">           \r\n\
 <INPUT type=\"submit\" name=\"botonokey\" value=\"OK\"></font></p>\r\n\
</FORM>\r\n\
</body>\r\n\
</html>"};
```

Parámetro de modulación

Botón OK de la web de los transductores

Figura 4.7-7: Código html de la web del transductor.

```
const char systemPage[] = {
"<html><head><title>System Setting</title>\r\n\
<head>\r\n\
<style type=\"text/css\">\r\n\
body{background-color:orange}\r\n\
</style>\r\n\
</head>\r\n\
<body>\r\n\
<p align=\"left\"><font size=\"3\" </font></p>\r\n\
<p align=\"left\"><font size=\"6\" color=\"red\"><b>SISTEMA DE BALIZAMIENTO</b></font></p>\r\n\
<hr><hr>\r\n\
<p align=\"left\"><font size=\"5\" color=\"blue\"><b>PARÁMETROS INDEPENDIENTES </b></font></p>\r\n\
<p align=\"left\"><font size=\"3\" </font></p>\r\n\
<h4><a  href=\"/transductor1.htm\">TRANS 1</a>  \r\n\
<a href=\"/transductor2.htm\">TRANS 2</a>  \r\n\
<a href=\"/transductor3.htm\">TRANS 3</a>  \r\n\
<a href=\"/transductor4.htm\">TRANS 4</a>  \r\n\
<a href=\"/transductor5.htm\">TRANS 5</a></h4>\r\n\
<hr>\r\n\
<hr>\r\n\
<p align=\"left\"><font size=\"5\" color=\"blue\"><b>PARÁMETROS COMUNES</b></font></p>\r\n\
<h5><p align=\"left\"><b>12 muestras/ciclo</b></p>\r\n\
<form action=\"paramconfig.htm\" method=\"post\">\r\n\
<p align=\"left\"><b>Selección de frecuencia de muestreo</b></p>\r\n\
<select name=\"FR\">\r\n\
<option selected value=\"40\">40 kHz</option>\r\n\
<option value=\"41\">41.66 kHz</option>\r\n\
</select>\r\n\
<p align=\"left\"><b>Selección de número de bits de la secuencia
<select name=\"NB\">\r\n\
<option selected value=\"255\">Kasami 255 bit</option>\r\n\
<option value=\"1023\">Kasami 1023 bit</option>\r\n\
</select></h5>\r\n\
<h5><p align=\"left\"><b>Ciclos/Simbolo </b><br>(De 1 a 4 para 255 bits, \r\n\
1 o 2 para 1023 bits)<br><Input type=\"number\" name=\"CS\" min=\"1\" max=\"4\" value = \"%c\"/></p>\r\n\
<p align=\"left\"><b>Periodo de emisión (ms) </b><br>(De 9 a 120) <br><Input \r\n\
type=\"number\" min=\"9\" max=\"120\" name=\"PE\" value = \"%c\"/></p>\r\n\
<p align=\"left\">            \r\n\
            \r\n\
<INPUT type=\"submit\" name=\"botonokey\" value=\"OK\"></p></h5>\r\n\
</FORM>\r\n\
<hr><hr>\r\n\
</body></html>\r\n"
};
```

Color de fondo

Hypervínculos para la web de los transductores

Siglas o referencias para recogida de parámetros de modulación

Botón OK de la web principal

Figura 4.7-8: Código html de la web principal.

4.8.-Escritura de parámetros en memoria

Aunque el EMW3162 tenga distribuida la memoria en 5 sectores, tal como indicaba la figura 3.2-1, en realidad, como se puede apreciar en la figura 4.8-1, el microcontrolador STM32F205 tiene más sectores.

En el FW Webserver OTA, usado como base para la programación, se puede comprobar, tal como aparece en la figura 4.8-2, que el sector 2 del STM32F205 no está definido. Por ello se crea un espacio de memoria para la guardar los parámetros de modulación. Esta zona ha llamado "CONFIG_FLASH_SIZE" y se puede ver en la figura 4.8-2. Creando esta zona de memoria y almacenando los valores que se modifiquen, en caso de reseteo del módulo, fallo de alimentación, etc., se podrá iniciar el sistema con la última configuración de parámetros de modulación.

Block	Name	Block base addresses	Size
Main memory	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbyte
	Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbyte
	Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbyte
	Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbyte
	Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbyte
	Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbyte
	Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbyte
	. . .	. . .	. . .
	Sector 11	0x080E 0000 - 0x080F FFFF	128 Kbyte

Figura 4.8-1: Mapeo de la memoria flash del STM32F205. [0033, 2011].

También se crea una estructura en el archivo "flash_configurations.h". Esta estructura es “config_param_t{}”. En ella se almacena de forma ordenada y en función del tipo primitivo los parámetros de señal en memoria flash. También se han creado funciones para la lectura y escritura en memoria (“Guarda_Config()” y “Lee_Config_Param()”). "Guarda_Config()", se usa para guardar los parámetros de modulación en flash cada vez que hay una modificación de uno de ellos. "Lee_Config()", lee los parámetros que "Guarda_Config()" guardó en flash. Esta estructura y funciones aparecen en la figura 4.8-3.

En caso de estar la zona de memoria flash, del sector creado, borrada, los parámetros de modulación se inicializan como se ve en la figura 4.8-4.

“Sec_Trans_x” hace referencia a cada uno de los transductores y “Nbits_Html” hace referencia al número de bits, o bien 1023 bits o bien 255 bits.

Nota: Se han obviado poner las definiciones de “Guarda_Config()” y “Lee_Config_Param()”.

Nota 2: Frecuencia_Html toma los valores 40 o 41, pero únicamente para realizar discriminaciones entre 40 kHz o 41 kHz como la que se veía en la figura 4.5-2.

```
#define BOOT_START_ADDRESS          (uint32_t)0x08000000
#define BOOT_END_ADDRESS            (uint32_t)0x08003FFF
#define BOOT_FLASH_SIZE             (BOOT_END_ADDRESS - BOOT_START_ADDRESS + 1)

#define DRIVER_START_ADDRESS        (uint32_t)0x080C0000
#define DRIVER_END_ADDRESS          (uint32_t)0x080FFFFF
#define DRIVER_FLASH_SIZE           (DRIVER_END_ADDRESS - DRIVER_START_ADDRESS +

#define PARA_START_ADDRESS          (uint32_t)0x08004000
#define PARA_END_ADDRESS            (uint32_t)0x08007FFF

#define PARA_FLASH_SIZE             (PARA_END_ADDRESS - PARA_START_ADDRESS + 1)

#define CONFIG_START_ADDRESS        (uint32_t)0x08008000
#define CONFIG_END_ADDRESS          (uint32_t)0x0800BFFF

#define CONFIG_FLASH_SIZE           (PARA_END_ADDRESS - PARA_START_ADDRESS + 1)
```

Figura 4.8-2: Sector creado en "flash_if.h".

```
typedef struct _config_param{          //Estructu
  int numero_bits;
  int frecuencia;
  int ciclos_simbolo;
  int periodo_emision;
  char secuencia_transductor1;
  char secuencia_transductor2;
  char secuencia_transductor3;
  char secuencia_transductor4;
  char secuencia_transductor5;
}config_param_t;

void readConfig(config_t *pConfig);
void restoreConfig(void);
void updateConfig(config_t *pConfig);
void Guarda_Config (config_param_t *pConfig);
void Lee_Config_Param (config_param_t *pConfig);
```

Figura 4.8-3: Estructura para el guardado de los parámetros de modulación.

```
/* Parámetros de modulación por defecto-----------------------------------------------
char Sec_Trans_1=0, Sec_Trans_2=1, Sec_Trans_3=2, Sec_Trans_4=3, Sec_Trans_5=4;
int  Nbits_Html              = 255;                  //kasami 255
int  Frecuencia_Html         = 41;                   // 41.66 kHz
int  Ciclos_Simbolo_Html     = 2;                    // 2ciclos/símbolo en el inicio
int  Periodo_Emision_Html    = 20;
```

Figura 4.8-4: Inicialización de parámetros de modulación por defecto.

4.9.-Configuración de la red del EMW3162

Una característica de diseño interesante es que se pueda acceder al sistema conectándose a una red a través de un rúter. De esta forma, si tenemos acceso al rúter, ni si quiera sería necesario acercarse a la baliza para realizar modificaciones (se podrían realizar por Ethernet). En este caso, como se comentaba en el apartado 1.2, ha sido necesario configurar el módulo como estación para que se conectara a un rúter, que actúa como punto de acceso.

La estructura de tipo "network_InitTypeDef_st" presente en la librería mxchip, permite la configuración de la WLAN. Por ello se modifican los parámetros de red, de la misma, necesarios para configurar una IP estática, SSID y clave de rúter determinados, módulo en modo estación, etc.

La figura 4.9-1 muestra los cambios pertinentes de la estructura "network_InitTypeDef_st".

En este caso se ha elegido como IP estática la "192.168.253.41". Como se puede comprobar en la figura 1.2-8 es necesario introducir esta IP para acceder al servidor web.

```
memset(&wNetConfig, 0x0, sizeof(network_InitTypeDef_st));
wNetConfig.wifi_mode = Station;
strcpy(wNetConfig.wifi_ssid, AP_NAME);
strcpy(wNetConfig.wifi_key, AP_PASSWORD);
sprintf(wNetConfig.local_ip_addr, "192.168.253.41");
sprintf(wNetConfig.net_mask, "255.255.255.0");
sprintf(wNetConfig.gateway_ip_addr, "192.168.253.254");
sprintf(wNetConfig.address_pool_start, "192.168.253.100");
sprintf(wNetConfig.address_pool_end, "192.168.253.200");
wNetConfig.dhcpMode = DHCP_Disable;
wNetConfig.wifi_retry_interval = 1000;
StartNetwork(&wNetConfig);
```

Figura 4.9-1: Modificaciones para las configuraciones de red del EMW3162.

4.10.-Led de comunicaciones

En caso de tener varias balizas u otros elementos que obstaculicen la visualización, se crea una rutina en el TIM4 que permite que parpadee un led verde (durante 4 segundos, a razón de 3 veces por segundo) cuando se pulse un hipervínculo en la web. Una ver se llega al valor "Static_Contador_Led" se para la cuenta y se asegura un nivel bajo en el bit/pin conectado al led. Se puede visualizar la programación de esta rutina en la figura 4.10-1.

```
void TIM4_IRQHandler (void){//PB8 (Pi
  TIM4->SR &= ~(TIM_SR_UIF);

  GPIO_ToggleBits(GPIOB, GPIO_Pin_8);

  if((Static_Contador_Led++) >= 24){
    TIM4->CR1 &= ~(0x1);
    GPIOB->ODR &= 0xFFFFFEFF;
    Static_Contador_Led=0;
    Flag_Led=0;
  }
}
```

Figura 4.10-1: Rutina para el led de comunicaciones.

4.11.-Led de comprobación de la conexión

En el FW Webserver OTA hay programado un pin, el pin PB1, para que luzca continuamente cuando se conecte como estación a un rúter. De esta forma, para programar el led de comprobación de la conexión, se realiza una lectura de este pin y en función de si el mismo se encuentra a un nivel u otro, se pone este mismo valor para el pin PB14, que está conectado al propio led de comprobación de la conexión. En las últimas líneas de la figura 4.6-1, donde se mostraba la rutina del TIM3 se recoge la programación de esta salida.

5.-DIAGRAMA DE NAVEGACIÓN WEB

Este capítulo expone la navegación web como resultado de la programación descrita en el capítulo anterior.

En primer lugar, antes de entrar a la navegación web, se ha dejado el acceso a credenciales, que viene por defecto en el FW Webserver OTA (mostrado en la figura 3.8-4).

Como se ha comentado antes, el diseño web, tiene los parámetros acotados para que no se den respuestas incoherentes de señales. Todas estas limitaciones pueden ofrecer un mensaje negativo y/o un mensaje informativo, pueden llevarnos a otra web o lanzar un mensaje sin acceder a otra.

Por otro lado, cuando se realiza una correcta configuración, el sistema también ofrece un mensaje positivo.

El diagrama del diseño web se muestra en la figura 5-1. En esta figura, se puede apreciar que en el diseño de las webs se ha diferenciado entre parámetros comunes (que afectan a todas las señales emitidas: Número de bits, muestras/ciclo, ciclos/símbolo y periodo de emisión) y parámetros independientes (número de secuencia Kasami a emitir por cada transductor).

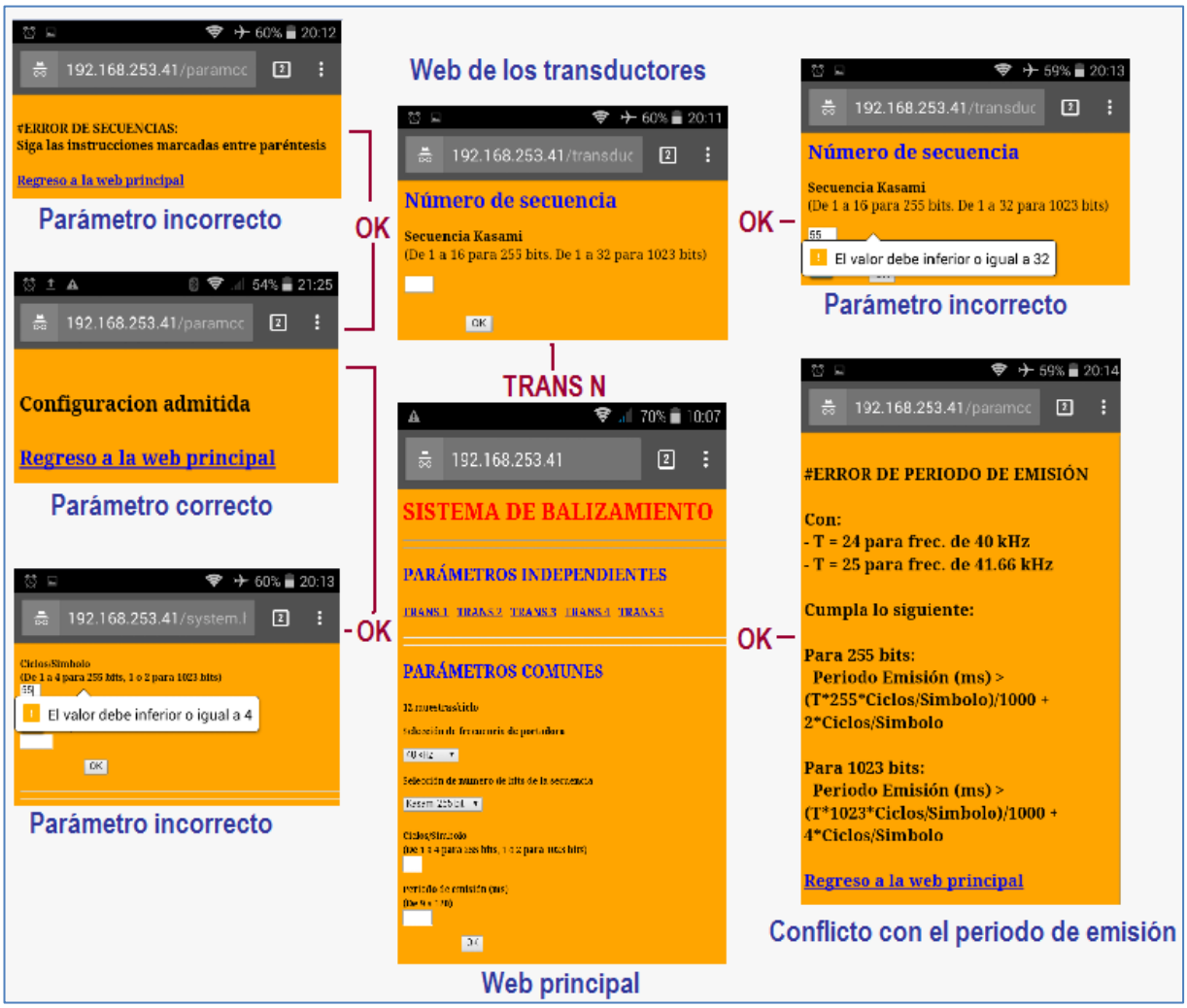


Figura 5-1: Diagrama de navegación web.

6.-CONCLUSIONES

Se ha conseguido finalmente una vía de comunicaciones que pueda ser operable a través de un móvil u otro dispositivo compatible con tecnología wifi. Esto va a permitir más comodidad a la hora de programar la parametrización de la baliza.

Las correlaciones del receptor demuestran que la baliza es altamente precisa en cuanto a tiempo de muestreo, obteniendo picos de correlación destacables con configuraciones de parámetros de modulación de cualquier tipo. Se puede comprobar en la figura 6-1 que con 255 bit y 1 ciclo/símbolo es suficiente para conseguir buenos resultados, mejorando al incrementar el número de ciclos/símbolo o el número de bits (figura 6-2).

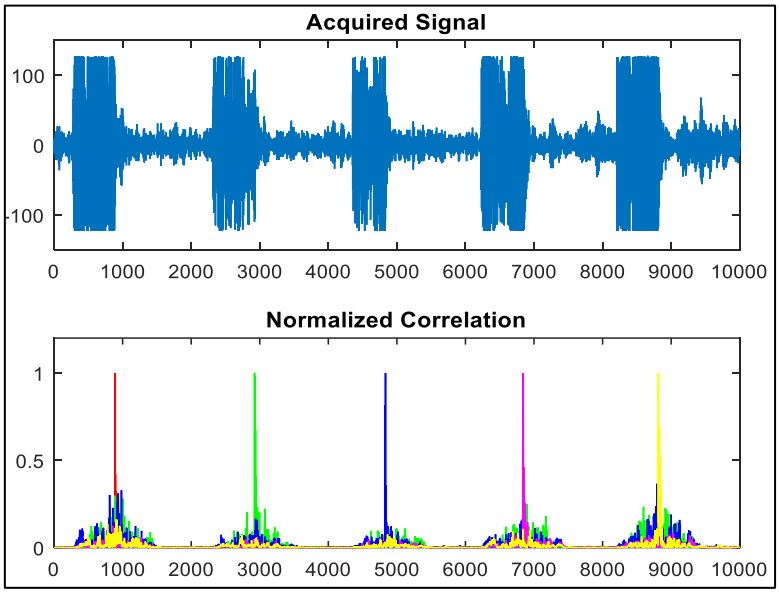


Figura 6-1: Correlaciones con 255 bit y 1 ciclo/símbolo (20ms de periodo de emisión).

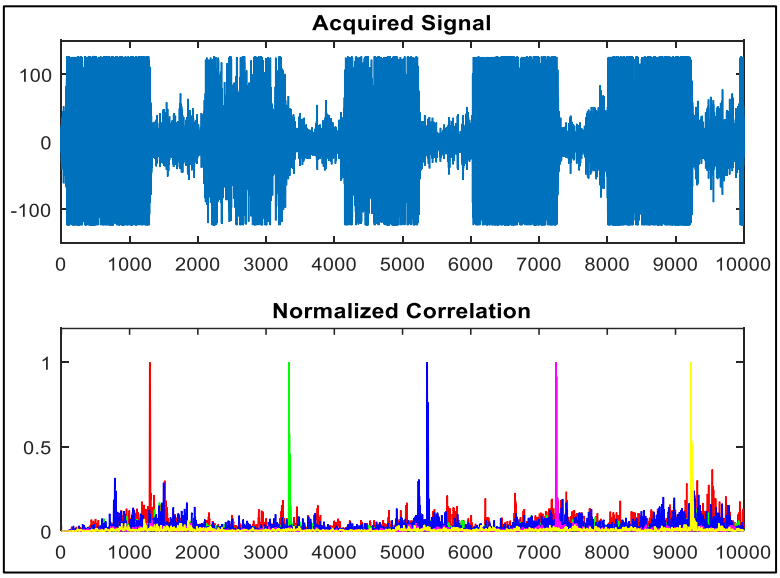


Figura 6-2: Correlaciones con 255 bit y 2 ciclos/símbolo (20ms de periodo de emisión).

La figura 6-3 muestra los resultados obtenidos para 1023 bits.

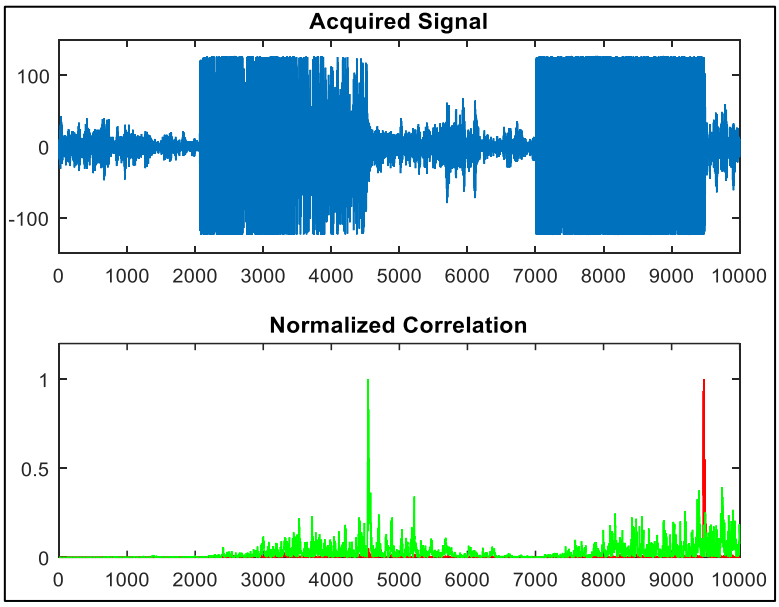


Figura 6-3: Correlaciones con 1023 bit y 1 ciclo/símbolo (50ms de periodo de emisión).

Nota: Todas las pruebas se realizan con una frecuencia de portadora de 41.66 kHz. Se han obviado recoger más capturas con las que fuera necesarias un mayor periodo de repetición por tener el receptor una ventana de captura de 100ms.

En los ejemplos que otorga el fabricante se han comprobado las posibilidades que el módulo wifi EMW3162 puede ofrecer. Como se ha demostrado en el apartado 3.6, el EMW3162 puede actuar como puente de comunicaciones entre dos equipos, modificar sus estados para proveer o para conectarse a una red y realizar otras operaciones. Luego lo que se podría destacar de este módulo, a parte de la precisión comentada, es la versatilidad que ofrece.

Por último, además de la comodidad de la programación y las altas prestaciones de la baliza, este proyecto, demuestra que se puede realizar proyectos de este tipo con poca inversión en materiales (se puede comprobar en el capítulo 8).

7.-POSIBLES MEJORAS

Al tratarse de un prototipo, se pueden realizar algunas mejoras. Las posibles y más importantes mejoras para esta baliza son:

1. Diseño PCB

Como se ha comentado, todos los componentes van montados en una protoboard. Un aspecto que pudiera ser importante es realizar un montaje SMD que permitiera hacer un diseño de la electrónica más compacto y eficiente.

2. Diseño web

Otra posible mejora es realizar un diseño web distinto, haciendo mejorables los elementos existentes que aparecen en el navegador: Poner imágenes, botones distintos, ampliar el número de páginas web o realizar más explicaciones para hacer más intuitivo el diseño.

8.-PRESUPUESTO

Cada una de las tablas de este capítulo exponen costes con IVA incluido.

Coste de licencias

Tabla 8-1: Coste de licencias.

Concepto	Precio (€)	Periodo de amortización (año)	Tiempo de uso (mes)	Coste (€)
Windows Vista Home Premium	120	10	12	12
keil uVision 5.17	2300	3	12	766.67
STM32 ST-Link Utility	0	-	12	0.00
STM32CubeMX	0	-	12	0.00
Matlab R2014A	3000	3	1	83.34
HyperTerminal Applet	0	-	3	0.00
EMWToolBox	0	-	3	0.00
BitScope DSO	0	-	12	0.00
TCP232	0	-	3	0.00
Total	**-**	**-**	**-**	**850.01**

Coste de equipos

Tabla 8-2: Coste de equipos.

Concepto	Precio (€)	Periodo de amortización (año)	Tiempo de uso (mes)	Coste (€)
PC hp a6513.es + periféricos	900	10	12	90
Tektronix TDS210	500	5	2	8.34
BitScope Micro	160	2	6	40.00
Lupa Bresser 2x/4x	25.18	1	3	6.27
Kit de soldadura INTEY	25.99	1	3	6.50
Total	**-**	**-**	**-**	**151.11**

Coste de material

Tabla 8-3: Coste de materiales.

Dispositivo o producto	Precio/unidad (€)	Número de unidades	Precio (€)
EMW3162	11.50	1	11.50
Wifi Shield	17.60	1	17.60
ST-Link V2	18.78	1	18.78
Multiplexor	1.86	1	2.86
DC/DC Traco Power	14.20	1	14.20
AO OPA 552	4.20	1	4.20

Transductor	1.50	5	7.50
MB102 Breadboard	12.80	1	12.80
Elementos pasivos	-	-	6.00
Batería 7.4V	19.99	1	9.99
Varillas de aluminio	1.10	4	4.40
Caja de plástico	17.50	1	17.50
Cables y conectores	-	-	12.00
Piezas impresión 3D	-	-	10.00
Protoboard	1.82	2	3.64
Adhesivo Loctite	5.95	1	5.95
Barra de pegamento termofusible	0.70	2	1.40
Estaño Fixpoint 51062	9.26	1	9.26
LA-CO Pasta de soldadura	12.15	1	12.15
Total	**-**	**-**	**181.73**

Coste de personal

Este presupuesto incluye la mecanografía, diseño, desarrollo y montaje, de la documentación y de la aplicación, en el supuesto de contratación de un ingeniero en electrónica y automática industrial y en el supuesto de que este mismo realizara todas las anteriores labores.

Tabla 8-4: Coste de personal.

Concepto	€/h	Número de horas	Coste (€)
Ingeniero en Electrónica y Automática Industrial	65	650	42250
Total	**-**	**-**	**42250**

Presupuesto total

Tabla 8-5: Coste final de ejecución.

Concepto	Coste (€)
Coste de licencias	850.01
Coste de equipos	151.11
Coste de material	181.73
Coste de personal	42250.00
Coste total	**43432.85**

El coste total estimado, de ejecución de este proyecto, asciende a **cuarenta y tres mil cuatro cientos treinta y dos con ochenta y cinco euros**.

ANEXO A.-LIBRERÍA MXCHIPWNET

Esta librería, es el soporte principal del EMW3162. Todas las funciones de la misma aparecen ocultas y tan solo se pueden apreciar las referencias a las funciones, así como las estructuras definidas en la misma.

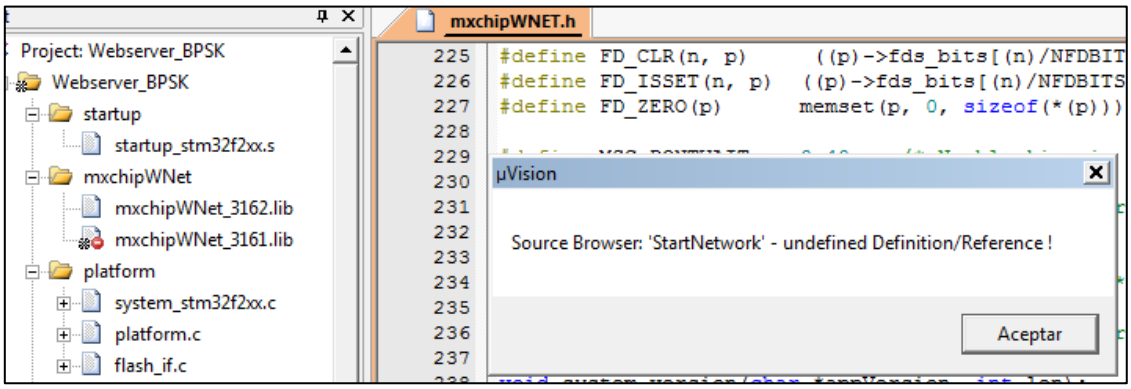


Figura A-1: Acceso no permitido en la librería mxchipWNet.

Existen muchas funciones y estructuras que pudieran ser comentadas, pero que escapan a las competencias de este proyecto y otras que se dan por válidas por funcionar correctamente y no necesitar modificaciones. Por ello, se comentarán las funciones consideradas más importantes y ligadas al proyecto.

La información a continuación se obtiene de la documentación creada por MXCHIP. En esta documentación se separan en bloques las funcionalidades de la librería, aunque todas formen parte de la misma.

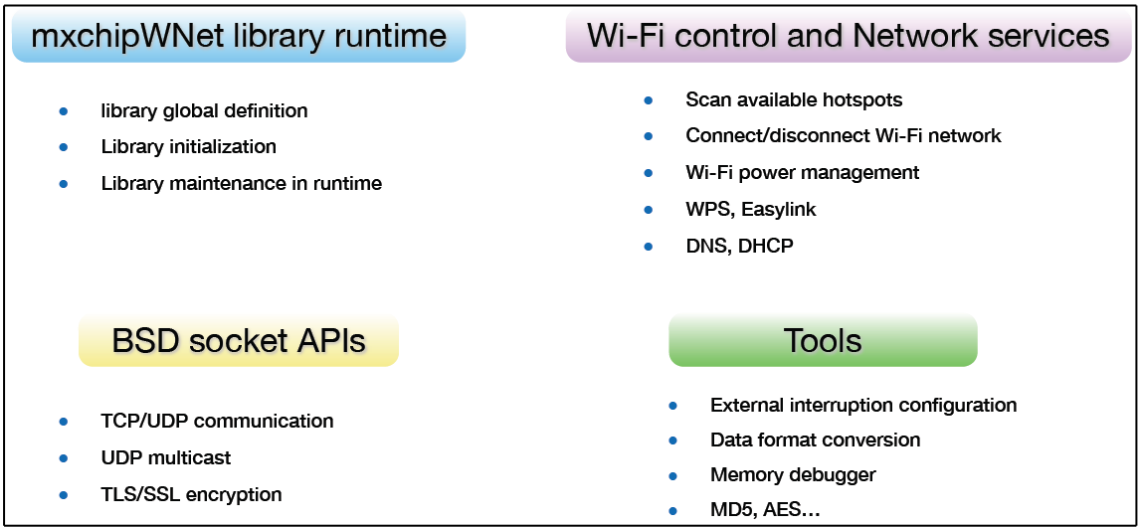


Figura A-2: Funcionalidades de la librería mxchipWNet ordenada por bloques.

Estructuras a tener en cuenta de la librería mxchipWNet:

- “network_InitTypeDef_st”: Esta estructura almacena la configuración de red que vamos a iniciar. Se puede elegir el modo ST o AP, el nombre y clave de la WLAN del EMW3162 y otras funciones.

Parameter Name	Note	Parameter Name	Note
Wi-Fi mode	Soft AP or Station	**Gateway IP address**	Use in static address
SSID	Name of Wi-Fi network	**DNS server IP address**	Use in static address
KEY	Encryption type is self-adaption	**DHCP mode**	DHCP server/client
Local IP address	Use in static address mode	**Address pool start**	
Net mask	Use in static address mode	**Address pool end**	
		Retry interval	Default: 100ms

Figura A-3: Parámetros de red de la estructura de tipo “network_InitTypeDef_st”.

- “netpara_st”: Esta estructura almacena los parámetros IP del módulo.

A continuación se expondrán las funciones importantes que atañen a este proyecto:

- “mxchipInit()”: Inicializa la librería mxchipWNet.

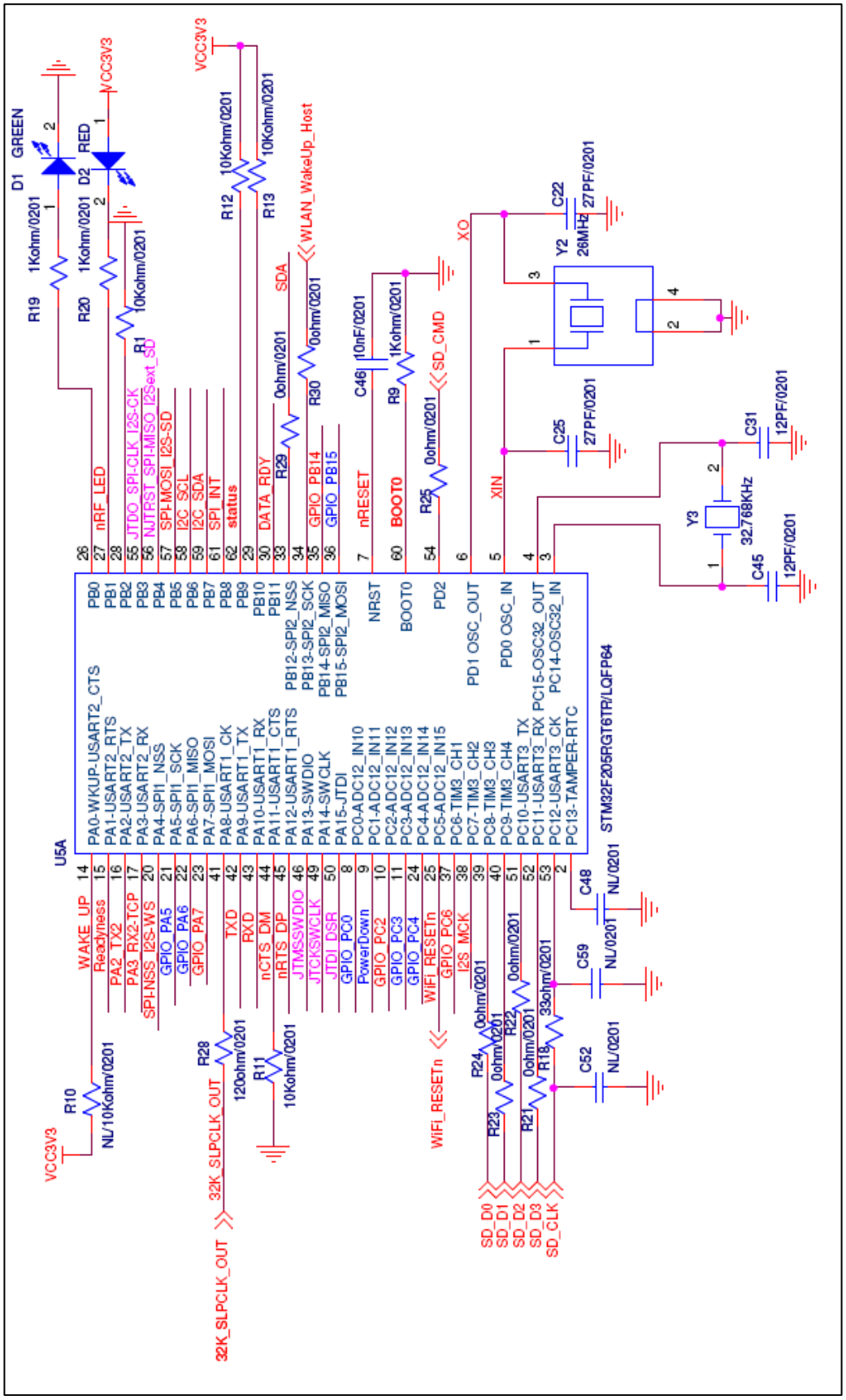
- “mxchipTick()”: Controla procesos de pila TCP/IP y los drivers wifi de la aplicación.

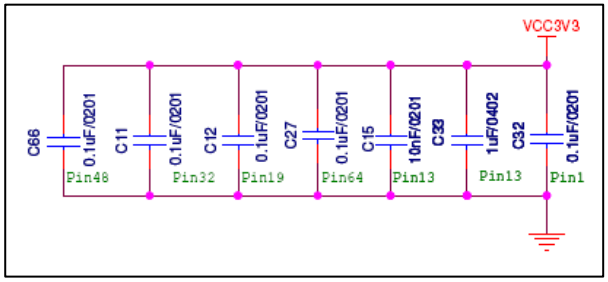
- “StartNetwork()”: Inicia una conexión de red en función de una estructura de tipo “network_InitTypeDef_st*”.

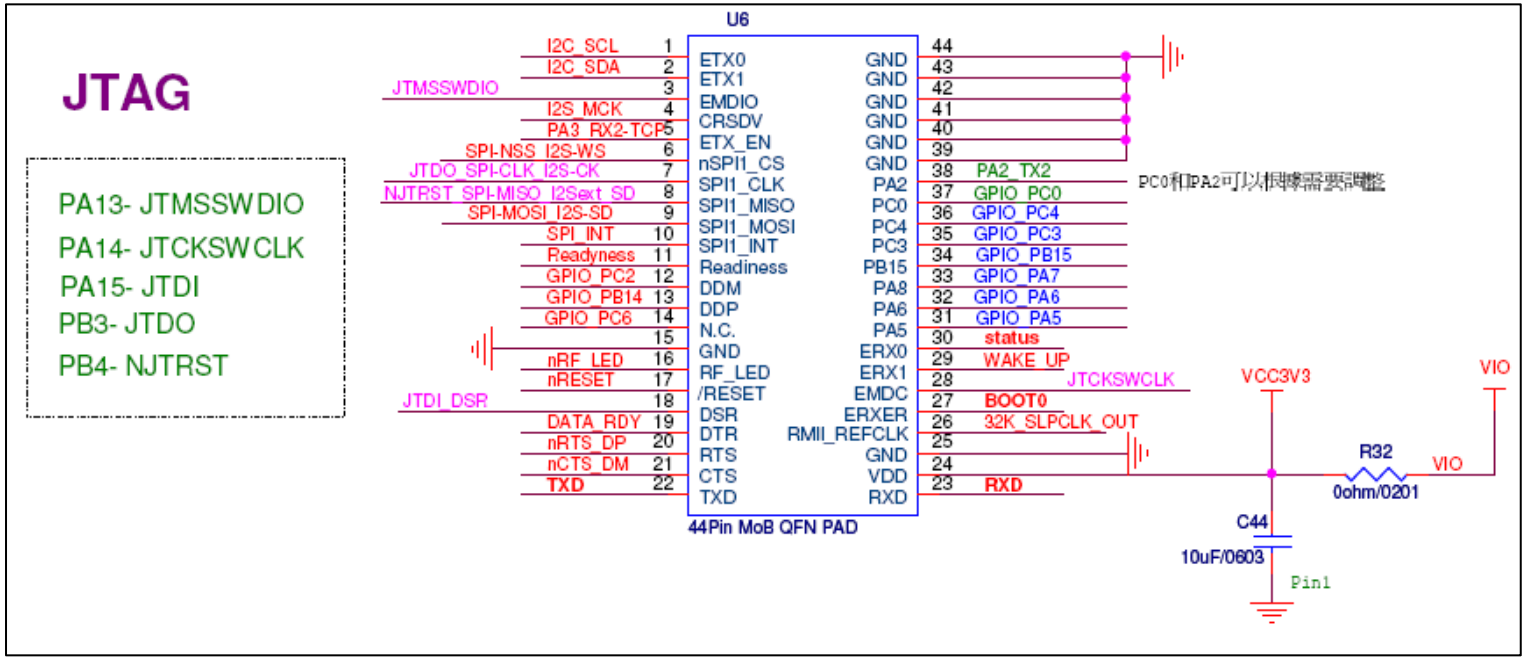
- “getNetPara(net_para_st * pnetpara, WiFi_Interface iface)”: Se usa para recoger los valores IP grabados en el módulo tales como la MAC, en función de una estructura de tipo “net_para_st”.

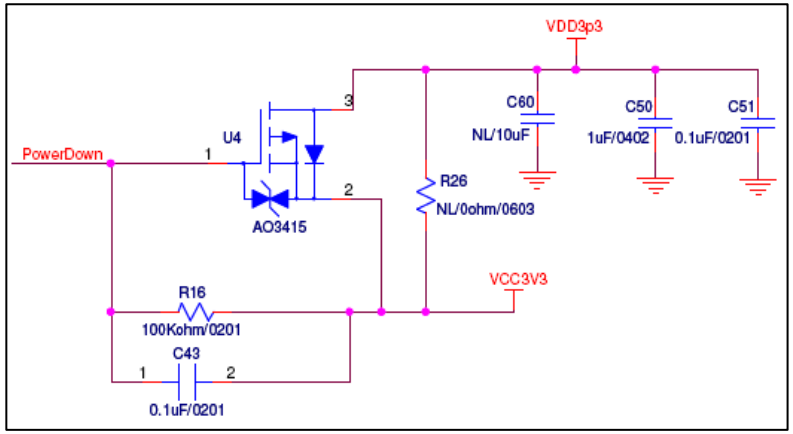
Hay otras funciones que se ocupan de la capa de abstracción (funciones, procedimientos y métodos de comunicaciones) del módulo wifi a través de la WLAN como la API (Aplication Programming Interface) Berkeley sockets. Otras funciones permiten el encriptado de comunicaciones con protocolos TLS/SSL. Y otras funciones que permiten manejar ciertas herramientas para la depuración de código que no serán comentadas por escapar de las competencias del proyecto.

ANEXO B.-PLANOS DEL EMW3162

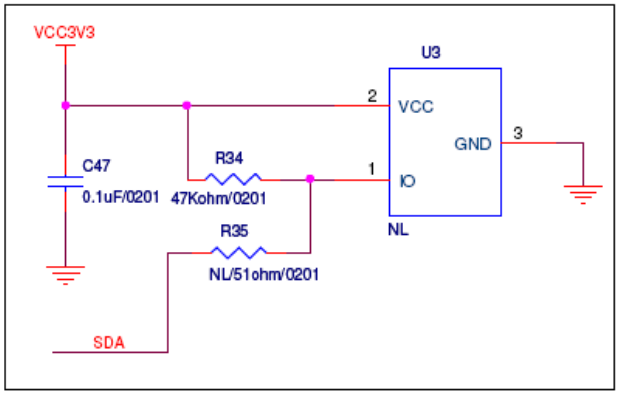


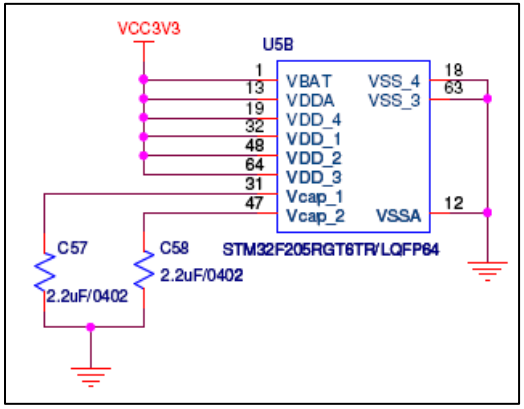




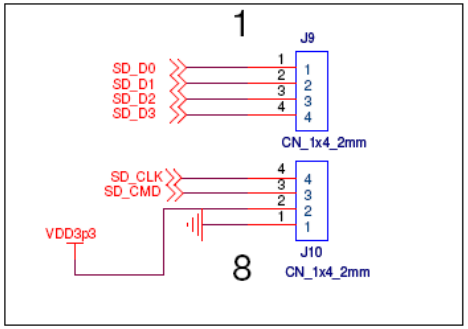


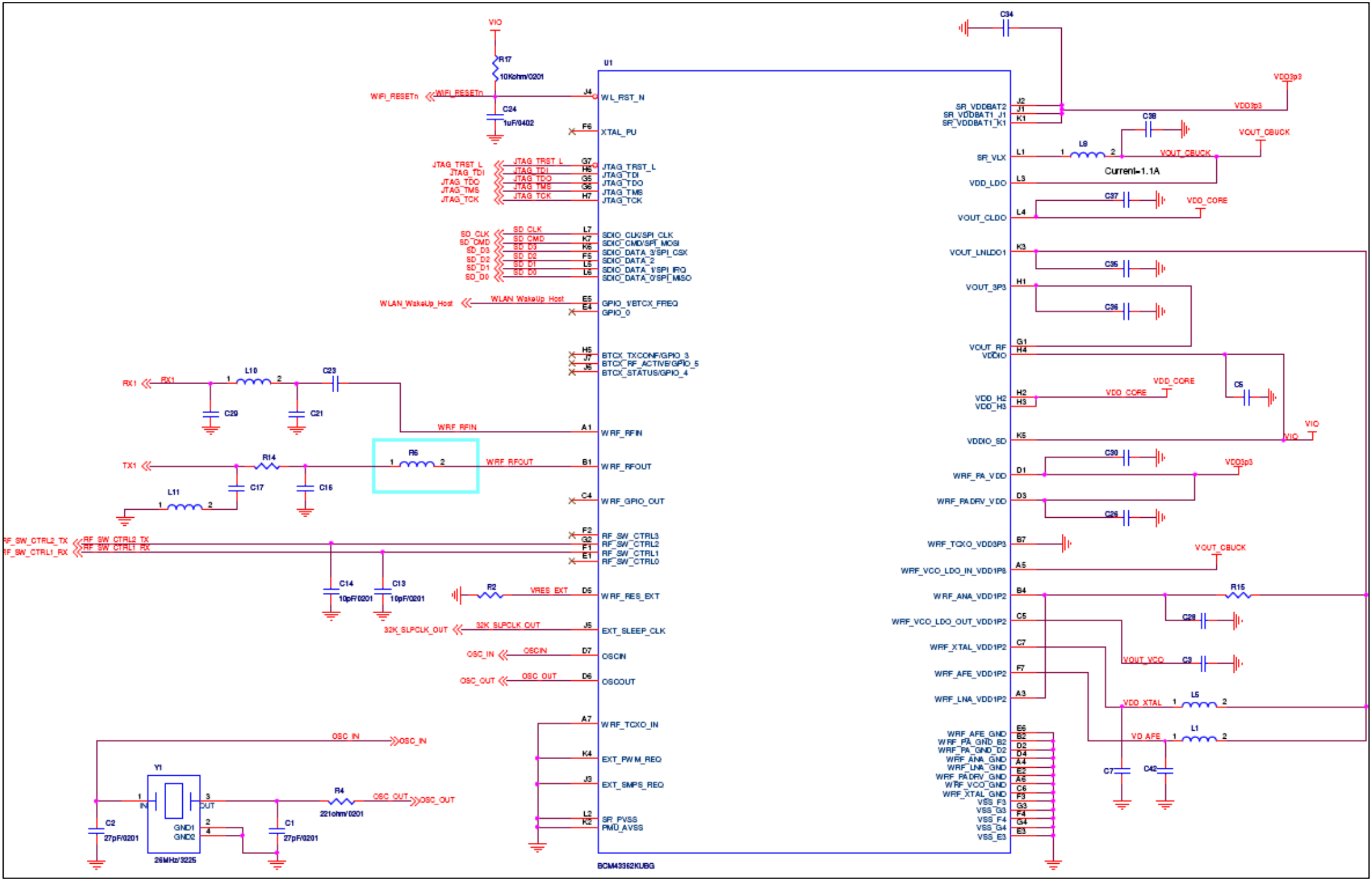
Reserve

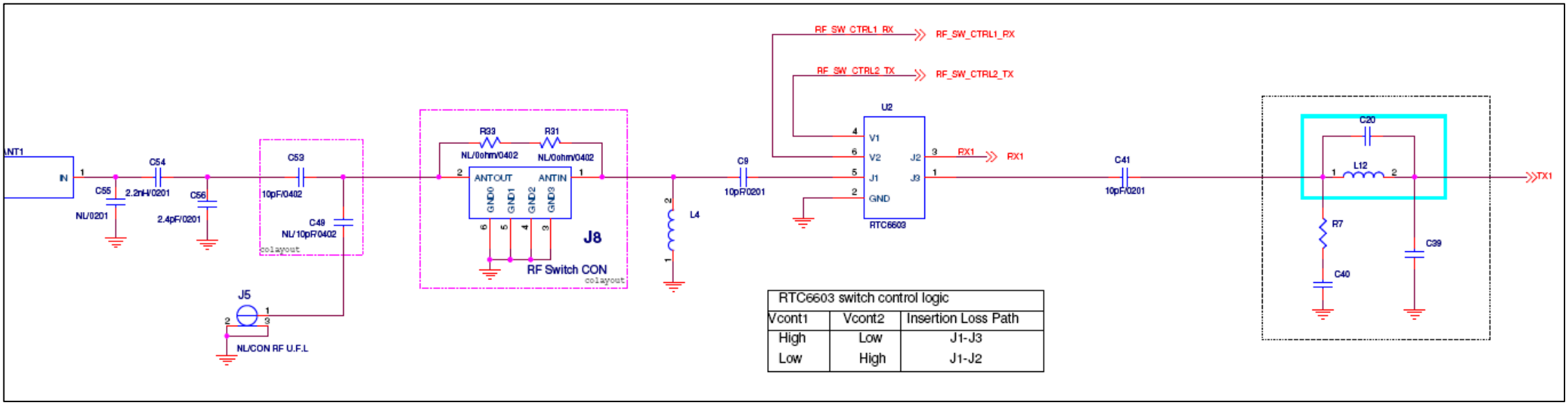




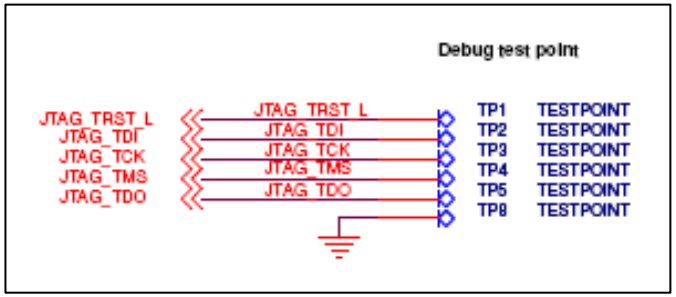
SDIO Test Connector




VIO
R17
10Kohm/0201
WIFI_RESETn
C24
1uF/0402
U1
WL_RST_N
XTAL_PU
JTAG_TRST_L
JTAG_TDI
JTAG_TDO
JTAG_TMS
JTAG_TCK
SD_CLK
SD_CMD
SD_D3
SD_D2
SD_D1
SD_D0
WLAN_WakeUp_Host
GPIO_VBTCX_FREQ
GPIO_0
BTCX_TXCONF/GPIO_3
BTCX_RF_ACTIVE/GPIO_5
BTCX_STATUS/GPIO_4
RX1
L10
C23
C29
C21
WRF_RFIN
TX1
R14
R6
WRF_RFOUT
C17
C16
L11
WRF_GPIO_OUT
RF_SW_CTRL2_TX
RF_SW_CTRL1_RX
RF_SW_CTRL3
RF_SW_CTRL2
RF_SW_CTRL1
RF_SW_CTRL0
C14
10pF/0201
C13
10pF/0201
R2
VRES_EXT
WRF_RES_EXT
32K_SLPCLK_OUT
EXT_SLEEP_CLK
OSC_IN
OSCIN
OSC_OUT
OSCOUT
WRF_TCXO_IN
EXT_PWM_REQ
EXT_SMPS_REQ
SR_PVSS
PMU_AVSS
Y1
R4
221ohm/0201
C2
27pF/0201
C1
27pF/0201
26MHz/3225
BCM43362KUBG
C34
SR_VDDBAT2
SR_VDDBAT1_J1
SR_VDDBAT1_K1
VDD3p3
C38
VOUT_CBUCK
SR_VLX
L8
Current=1.1A
VDD_LDO
C37
VDD_CORE
VOUT_CLDO
VOUT_LNLDO1
C35
VOUT_3P3
C36
VOUT_RF
VDDIO
VDD_H2
VDD_H3
C5
VIO
VDDIO_SD
C30
WRF_PA_VDD
WRF_PADRV_VDD
C26
WRF_TCXO_VDD3P3
WRF_VCO_LDO_IN_VDD1P8
WRF_ANA_VDD1P2
R15
C28
WRF_VCO_LDO_OUT_VDD1P2
WRF_XTAL_VDD1P2
VOUT_VCO
C3
WRF_AFE_VDD1P2
WRF_LNA_VDD1P2
VDD_XTAL
L5
VD_AFE
L1
C7
C42
WRF_AFE_GND
WRF_PA_GND_B2
WRF_PA_GND_D2
WRF_ANA_GND
WRF_LNA_GND
WRF_PADRV_GND
WRF_VCO_GND
WRF_XTAL_GND
VSS_F3
VSS_G3
VSS_F4
VSS_G4
VSS_E3



RTC6603 switch control logic		
Vcont1	Vcont2	Insertion Loss Path
High	Low	J1-J3
Low	High	J1-J2



BIBLIOGRAFÍA

[M.C. Pérez, 2009]
M.C. Pérez. "*Generación y correlación eficiente de códigos binarios derivados de conjuntos de secuencias complementarias para sistemas ultrasónicos*". Universidad de Alcalá (2009).

[José M. Villadangos, 2012]
José M. Villadangos. "*Contribución al diseño de un LPS ultrasónico de amplia cobertura empleando técnicas de codificación*". Universidad de Alcalá (2012).

[D. de Diego López, 2013]
D. de Diego López. "*Diseño, desarrollo y despliegue de un sistema de posicionamiento local basado en ultrasonidos*". Universidad de Alcalá (2013).

[MMUO, 1999]
V. Díaz, J. Ureña, J.J.García, Á. Hernandez, M. Mazo. "*Multi-mode ultrasonic operation using Golay complementary sequences and QPSK modulation*". Universidad de Alcalá (1999).

[J. F. Arango, 2016]
J. F. Arango Vargas. "*Diseño de un módulo receptor para un LPS ultrasónico*". Universidad de Alcalá (2016).

[HUAV, 2017]
F.J. Álvarez, J. Esteban, José M. Villadangos "*High accuracy APS for Unmanned Aerial Vehicles*". TELSIKS. (2017).

[PoCS, 1986]
H.Taub, D.L.Schilling. "*Principles of Communication Systems*". McGraw-Hill International Editions (1986).

[MB102]
http://www.ece.usu.edu/ece_store/spec/breadboard_datatsheet1.pdf

[intersil, 2006]
https://www.intersil.com/content/dam/Intersil/documents/dg40/dg408-09.pdf. (2006)

[ti, 2016]
http://www.ti.com/lit/ds/symlink/opa551.pdf. (2016)

[TP, 2009]
http://assets.tracopower.com/20170906155421/TMR3/documents/03_Application%20Note.pdf. (2009)

[ProWa, 2005]
http://www.prowave.com.tw/english/products/ut/open-type/328s160.htm. (2005).

[VvSt, 2013]
http://www.sc.ehu.es/sbweb/fisica_/ondas/acustica/sonido/sonido1.html. (2013)

[1075, 2016]
ST. "*UM1075 User Manual*". (2016).

[0033, 2011]
ST. "*RM0033 Reference Manual*". (2011).

J.O. Roa, A.R. Jiménez, F. Seco, C. Prieto, J.L. Ealo, F. Ramos y J. Guevara. "*Distribución optimizada de balizas para sistemas de localización fundamentados en Trilateración Esférica e Hiperbólica*". MAEB (2007).

J.M. Huidobro, P. Pastor. "*Infraestructuras comunes de Telecomunicaciones*". Creaciones Copyright (2011).

F. Daura. "*El mito del condensador de 100 nF*". CEMDAL (2013).

MXCHIP. "*EMW3162 Datasheet*". (2013).

J. Yiu. "*The definitive guide to the ARM Cortex-M3*". (2010).

ARM. "*Cortex-M3 Technical Reference Manual*". (2005).

REFERENCIAS

github.com

mxchip.com

cypress.com

st.com

waveshare.com

elingesor.com

electronicdesign.com

tracopower.com

octopart.com

dx.com

vandyke.com

elbibliote.com

stackoverflow.com

energysistem.com

rfcafe.com

intersil.com

prowave.com

sc.ehu.es

equaphon-university.net

tjinguytech.com

seeedstudio.com

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá