WEMAT (Wifi Embedded Morse Automatic Translator)

Ingeniería Técnica en Telecomunicación, especialidad Telemática (ITTT)

Estudiante
Samuel Melero Cuesta

Consultor
Jordi Bécares Ferrés

14 de Enero de 2014

Dedicatoria y agradecimientos

A mi hija, Claudia, que con dos añitos y medio me ha enseñado todo un mundo nuevo.

A mi mujer, Mary, por su gran ayuda y paciencia durante toda la carrera.

Y también, agradecer al consultor Jordi Bécares por los consejos y ánimos recibidos. Han sido determinantes para el buen transcurso del proyecto.

Muchas gracias.

Resumen

El proyecto que se describe a continuación, enmarcado dentro del área de sistemas empotrados, trata del desarrollo de un traductor entre código Morse audible y texto.
Para llevar a cabo esta tarea, el dispositivo hace uso de un micrófono y un pequeño altavoz. La comunicación con el usuario se consigue mediante una aplicación Java y conexión Wifi.

El funcionamiento es simple, aunque no su implementación. Consta de codificador y decodificador de código Morse y de conexión Wifi y USB. Mediante la conexión USB se puede configurar la conexión inalámbrica del dispositivo. Una vez configurado, se conecta a la aplicación de usuario mediante Wifi. Desde ésta se pueden configurar diversos parámetros como el tono y volumen de los pitidos del codificador Morse y el ajuste de velocidad, tanto del codificador como del decodificador Morse, en palabras por minuto.
Una vez en este punto, el dispositivo recoge el sonido en código Morse, lo decodifica y lo manda como texto a la aplicación de usuario. Por otro lado, desde dicha aplicación el usuario envía texto al dispositivo el cual codificará en Morse y lo hará sonar por el altavoz.

Para hacer más versátil el sistema, éste incluye parte de *código Morse extendido*[1] además del estándar, el cual incluye caracteres como "ñ" o "ç" y algunos símbolos de numeración no existentes en código Morse internacional. De esta manera el dispositivo soporta más idiomas incluyendo el Catalán y el Español.

[1] Caracteres no existentes en código Morse estándar.

Índice de contenido

Índice de figuras

1. Introducción

Se propone un TFC para realizar traducción texto a Morse y viceversa mediante un dispositivo empotrado. La idea detrás del sistema es construir un traductor que sea capaz, por un lado, de recibir código Morse y transmitirlo como texto plano a un ordenador y, por otro, recibir texto plano del ordenador y convertirlo en código Morse audible todo ello a través de una aplicación Java con interfaz sencilla y amigable.

El microcontrolador seleccionado permite que el traductor sea portátil y de bajo consumo, características que lo hacen versátil. Ofrece un dispositivo dedicado que libera al ordenador y lo capacita para realizar más tareas y, a su vez, el sistema se conecta sin necesidad de cables con las grandes posibilidades y comodidad del Wifi.

Para llevar a cabo la idea se hacen necesarios conocimientos en lenguaje C y sistemas empotrados en lo referente al funcionamiento del dispositivo empotrado, teoría de circuitos para el desarrollo de la parte electrónica analógica, telemática (comunicación por red), programación orientada a objetos para el desarrollo de la aplicación de usuario y gestión de proyectos para hacer viable la consecución del TFC.

Toda la información necesaria ha sido obtenida de las distintas asignaturas que componen la carrera y complementada mediante búsquedas específicas en la red.

1.1 Justificación

La idea surgió como una necesidad en la traducción de código Morse recibido de estaciones radio lejanas que el estudiante, como radioaficionado, escuchaba frecuentemente. Asimismo, también se hacía necesario la traducción desde texto para la transmisión del mensaje. Además, se encontraba la dificultad de la velocidad de código que empleaban los radioaficionados más experimentados, realmente alta para usuarios con poca práctica.

La razón principal de utilizar código Morse en las comunicaciones es su mayor robustez frente al ruido. Cuando el nivel de señal recibido es muy pequeño, las comunicaciones basadas en modulación por voz se hacen ininteligibles. Dada la *naturaleza digital*[2] del código Morse, permite salvar este problema al simplificar la recepción en pitidos que son fácilmente identificables, proporcionando comunicación a largas distancias utilizando menos energía para la transmisión.

El sistema desarrollado en este proyecto añade la posibilidad de utilizar *ultra fast Morse*[3], con una velocidad asombrosa que hace imposible su interpretación por parte de una persona pero que permite la transmisión y recepción de texto de manera muy rápida.

Aunque existen soluciones a este problema actualmente, los pocos dispositivos portátiles creados para este fin no suelen incluir la traducción bidireccional, es decir, sólo soportan la traducción del código a texto o de texto a código pero no ambos. Además de no incluir ajustes tales como la frecuencia de la señal y la velocidad en palabras por minuto ni comunicación sin hilos. Añadir que todas las implementaciones, incluidas las que hacen uso de los *smartphones*, son propietarias y no permiten su acceso a detalles sobre la implementación del sistema.

Añadir que el dispositivo empotrado empleado ofrece flexibilidad y autonomía al proporcionar conexión Wifi y bajo consumo.

[2] El código Morse se fundamenta en dos estados.
[3] Término descriptivo no estándar.

1.2 Descripción del proyecto

El objetivo principal del proyecto es la traducción de sonido a texto y texto a sonido, dado que el código Morse *se suele utilizar* [4] en forma de señales de audio.

El corazón del proyecto es un microcontrolador, el NXP LPC1769, que se encarga de procesar la información para traducirla entre un formato y otro. Para la recepción de las señales de audio, se ha optado por un pequeño circuito de recepción y acondicionamiento, que provee al microcontrolador de una señal con dos niveles de tensión. Para la conversión inversa, se ha utilizado un pequeño *buzzer*[5] controlado por el LPC1769. Al hacerse mediante una señal PWM[6], se ofrece la posibilidad de ajustar la frecuencia a la que suena.

El control de la traducción es realizada a través de la aplicación de usuario codificada en lenguaje Java y la comunicación entre aplicación y dispositivo empotrado se lleva a cabo a través de Wifi. Java es multiplataforma lo que hace a la aplicación de usuario capaz de ser utilizada en cualquier sistema operativo que lo soporte.

El hecho que sea el sistema empotrado el que se encarga de la traducción utilizando texto en código ASCII hace viable su control mediante cualquier otra aplicación creada, por ejemplo, para dispositivos móviles. Por lo tanto, el producto obtenido es fácilmente ampliable ofreciendo mayores posibilidades.

[4] El código Morse es utilizado bajo varios tipos de señal, como puede ser la luz.
[5] Transductor electromagnético utilizado como pequeño altavoz
[6] Siglas en Inglés de modulación por ancho de impulso.

1.3 Objetivos del TFC

El TFC tiene como objetivos los siguientes:

- Ofrecer interfaz de configuración de conexión, mediante USB.
- Dotar al sistema de conexión inalámbrica (Wifi).
- Implementar entrada y acondicionamiento de audio.
- Traducir código Morse (audio) a texto (ASCII).
- Generar señales de audio en código Morse y controlar su tono y volumen.
- Traducir texto (ASCII) a código Morse (audio).
- Ofrecer interfaz de usuario (Java).
- Implementar control de velocidad envío/recepción en palabras por minuto (*wpm*[7]).
- Control de errores en la conexión Wifi.

[7] En Inglés, words per minute

1.4 Enfoque y método seguido

El proyecto se ha dividido en fases, cada una con unas tareas determinadas.
En cada tarea se ha seguido el mismo método:

- Documentación, incluyendo búsquedas en la red si es necesario.
- Desglose de cada tarea en partes específicas y comprobar cada una. Si existen problemas son fáciles de solucionar gracias a la división de la tarea.
- Implementar cada tarea y probarla con un pequeño test específico.

En cuanto al sistema empotrado, se ha aplicado modularidad, empezando con los drivers genéricos (como conexión USB y Wifi), para seguir avanzando en periféricos específicos con sus drivers, librerías de mayor nivel y por último la aplicación final, la cual descansa sobre las distintas librerías y drivers creados y comprobados por lo que su realización resulta más directa y sin mayores problemas.
Dada la modularidad empleada la modificación de cada módulo se hace mucho más sencilla y concreta, sin tener que modificar el resto.

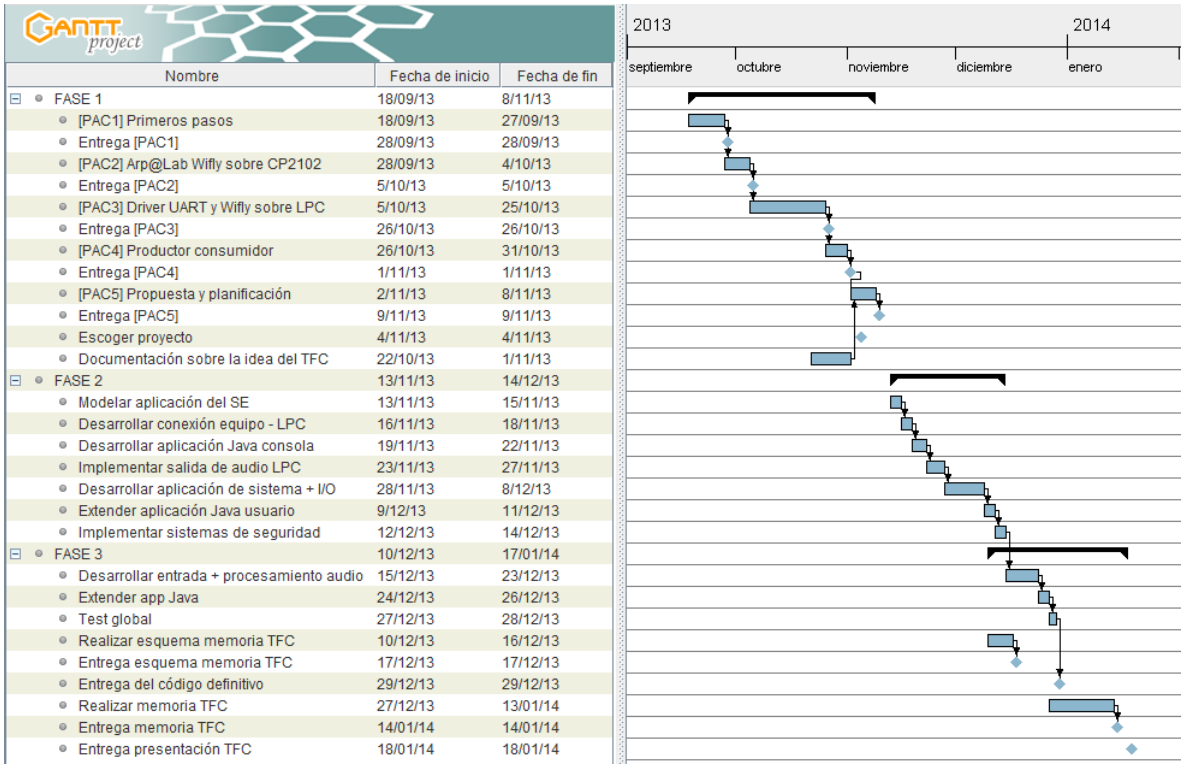
1.5 Planificación del proyecto

El proyecto se divide en tres fases:

- La primera fase ha consistido en documentación sobre el sistema empotrado a emplear, desarrollo inicial de los drivers de comunicación con el dispositivo y su prueba mediante pequeñas aplicaciones específicas.

- La segunda fase incluye el desarrollo del sistema, comenzando con los periféricos específicos, sus librerías y la ampliación de los drivers de comunicación cuando ha sido necesario, todo ello comprobado mediante pequeños test. Un punto importante en esta fase ha sido la implementación del sistema de traducción Morse, con el codificador y decodificador. También incluye el desarrollo de la aplicación Java de usuario y su prueba con el sistema, además de incluir la comunicación para las distintas opciones de configuración desde la aplicación de usuario.

- La tercera y última fase se ocupa de la implementación del tratamiento de fallos de comunicación y la realización de la memoria y presentación. En un principio esta fase albergaba la ampliación de la aplicación de usuario y otras mejoras.



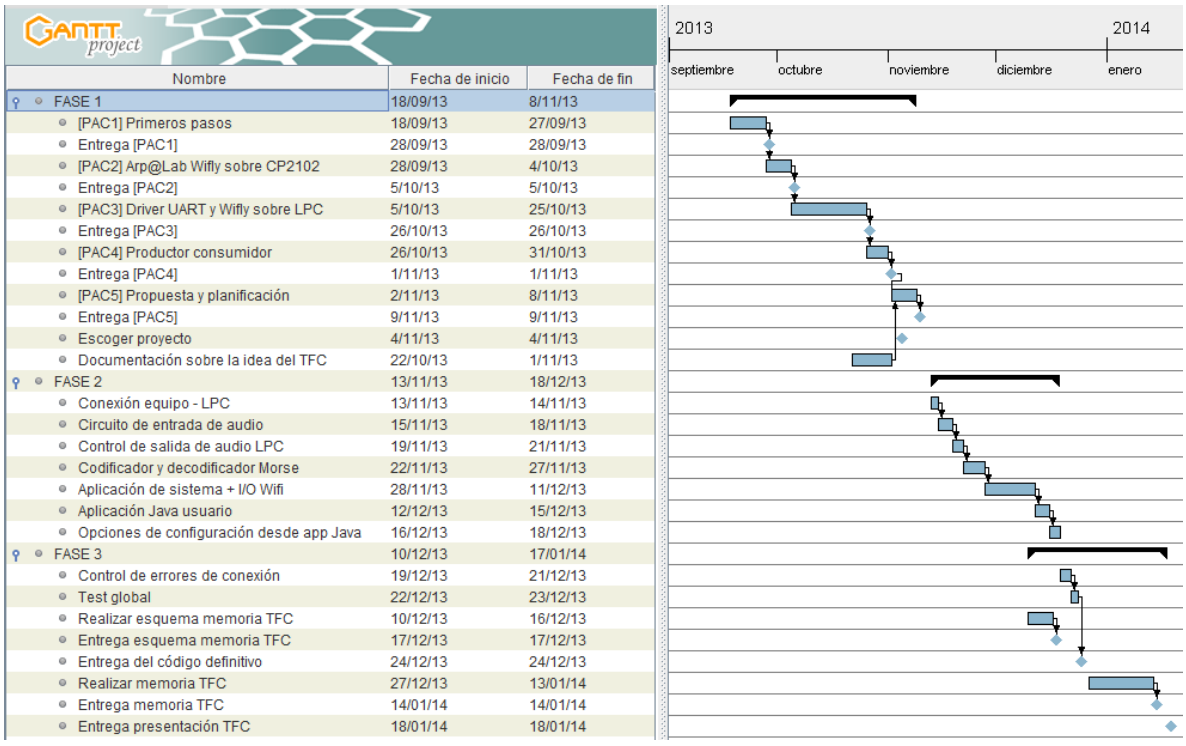
Nombre	Fecha de inicio	Fecha de fin
FASE 1	18/09/13	8/11/13
[PAC1] Primeros pasos	18/09/13	27/09/13
Entrega [PAC1]	28/09/13	28/09/13
[PAC2] Arp@Lab Wifly sobre CP2102	28/09/13	4/10/13
Entrega [PAC2]	5/10/13	5/10/13
[PAC3] Driver UART y Wifly sobre LPC	5/10/13	25/10/13
Entrega [PAC3]	26/10/13	26/10/13
[PAC4] Productor consumidor	26/10/13	31/10/13
Entrega [PAC4]	1/11/13	1/11/13
[PAC5] Propuesta y planificación	2/11/13	8/11/13
Entrega [PAC5]	9/11/13	9/11/13
Escoger proyecto	4/11/13	4/11/13
Documentación sobre la idea del TFC	22/10/13	1/11/13
FASE 2	13/11/13	14/12/13
Modelar aplicación del SE	13/11/13	15/11/13
Desarrollar conexión equipo - LPC	16/11/13	18/11/13
Desarrollar aplicación Java consola	19/11/13	22/11/13
Implementar salida de audio LPC	23/11/13	27/11/13
Desarrollar aplicación de sistema + I/O	28/11/13	8/12/13
Extender aplicación Java usuario	9/12/13	11/12/13
Implementar sistemas de seguridad	12/12/13	14/12/13
FASE 3	10/12/13	17/01/14
Desarrollar entrada + procesamiento audio	15/12/13	23/12/13
Extender app Java	24/12/13	26/12/13
Test global	27/12/13	28/12/13
Realizar esquema memoria TFC	10/12/13	16/12/13
Entrega esquema memoria TFC	17/12/13	17/12/13
Entrega del código definitivo	29/12/13	29/12/13
Realizar memoria TFC	27/12/13	13/01/14
Entrega memoria TFC	14/01/14	14/01/14
Entrega presentación TFC	18/01/14	18/01/14

Figura : Planificación inicial

Después, en la ejecución del proyecto, han surgido problemas que han derivado en algunos cambios importantes de la planificación inicial, todos ellos en la segunda fase y principio de la tercera. Dichos cambios no han truncado la correcta ejecución del proyecto y se refieren a la implementación del circuito de entrada de audio y la aplicación Java, la cual se implementó posteriormente.

En un principio, se pensó en implementar la entrada de código Morse con un pulsador, dado que la opción de implementar el circuito de entrada y acondicionamiento de audio podría complicar la correcta finalización del proyecto. Las pruebas con pulsador, pareciendo la solución más asequible, dieron problemas debidos a los *rebotes*[8] del mismo por el ruido generado en sus contactos al accionarlo y soltarlo. Además, existía la problemática de hacer las pruebas sin saber código Morse, lo que empeora todavía más los resultados. Por lo tanto se le dio prioridad al circuito de entrada y acondicionamiento de audio que resultó en éxito inmediato, ya que no adolecía de los problemas del pulsador y las pruebas podían ser ejecutadas mediante código Morse generado con un ordenador, mucho más conveniente.

El resto de diferencias se deben al cambio en el tiempo empleado en algunas tareas que parecían de menor consideración resultando en una planificación más ajustada a la ejecución real del proyecto:



Nombre	Fecha de inicio	Fecha de fin
FASE 1	18/09/13	8/11/13
[PAC1] Primeros pasos	18/09/13	27/09/13
Entrega [PAC1]	28/09/13	28/09/13
[PAC2] Arp@Lab Wifly sobre CP2102	28/09/13	4/10/13
Entrega [PAC2]	5/10/13	5/10/13
[PAC3] Driver UART y Wifly sobre LPC	5/10/13	25/10/13
Entrega [PAC3]	26/10/13	26/10/13
[PAC4] Productor consumidor	26/10/13	31/10/13
Entrega [PAC4]	1/11/13	1/11/13
[PAC5] Propuesta y planificación	2/11/13	8/11/13
Entrega [PAC5]	9/11/13	9/11/13
Escoger proyecto	4/11/13	4/11/13
Documentación sobre la idea del TFC	22/10/13	1/11/13
FASE 2	13/11/13	18/12/13
Conexión equipo - LPC	13/11/13	14/11/13
Circuito de entrada de audio	15/11/13	18/11/13
Control de salida de audio LPC	19/11/13	21/11/13
Codificador y decodificador Morse	22/11/13	27/11/13
Aplicación de sistema + I/O Wifi	28/11/13	11/12/13
Aplicación Java usuario	12/12/13	15/12/13
Opciones de configuración desde app Java	16/12/13	18/12/13
FASE 3	10/12/13	17/01/14
Control de errores de conexión	19/12/13	21/12/13
Test global	22/12/13	23/12/13
Realizar esquema memoria TFC	10/12/13	16/12/13
Entrega esquema memoria TFC	17/12/13	17/12/13
Entrega del código definitivo	24/12/13	24/12/13
Realizar memoria TFC	27/12/13	13/01/14
Entrega memoria TFC	14/01/14	14/01/14
Entrega presentación TFC	18/01/14	18/01/14

Figura : Planificación final

8 En Inglés, switch bouncing

1.6 Recursos empleados

Referidos al hardware:

- LPCXpresso con microcontrolador NXP LPC1769.
- Módulo Wifly Roving Networks RN-171 con antena incorporada y adaptador XBEE a DIP.
- Módulo convertidor USB - UART basado en CP2102 de Silicon Labs.
- Transductor electromagnético 2V, 10 mA, 1KHz
- Micrófono electret 1,5 - 10V , 30 Hz - 16 KHz, , sensibilidad 64 dB
- Router Wifi B/G/N protegido mediante WPA2 - AES.
- PC 4GHz - 8 Core 64 bits, 16 GB RAM
- Fuente de alimentación de laboratorio 3 - 15V, 4A
- Protoboard 82x55 mm, 400 contactos + cableado
- Componentes electrónicos específicos para el circuito de entrada de audio.

Referidos al software:

- Windows 7 64 bit
- IDE LPCXpresso 5.2.6_2137 + Librerías CMSISv2p00
- Sistema operativo en tiempo real FreeRTOS + Librerías
- IDE Java NetBeans 7.4
- LTSpice IV para el diseño del circuito electrónico.

1.7 Productos obtenidos

Finalizada la ejecución del proyecto obtenemos dos productos:

- Sistema empotrado, formado por la placa LPCXpresso con microcontrolador NXP LPC1769 y los periféricos: Wifly RN-171, CP2102, transductor electromagnético y circuito de entrada de audio con micrófono electret.

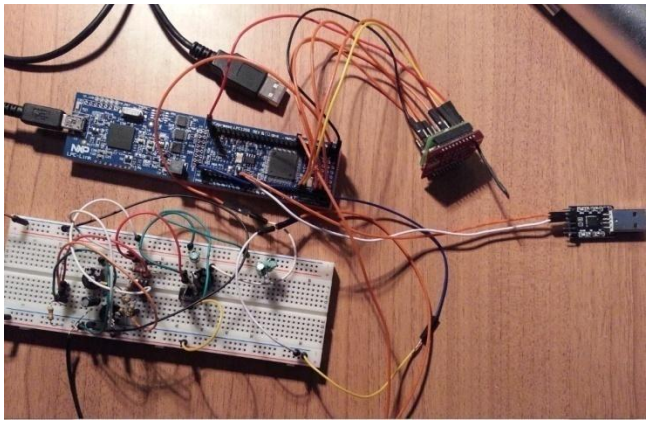


Figura : Sistema empotrado con sus periféricos

- Interfaz gráfica de usuario Java para la comunicación con el sistema, además de proporcionar los ajustes de tono, volumen, velocidad en palabras por minuto y monitorización[9].

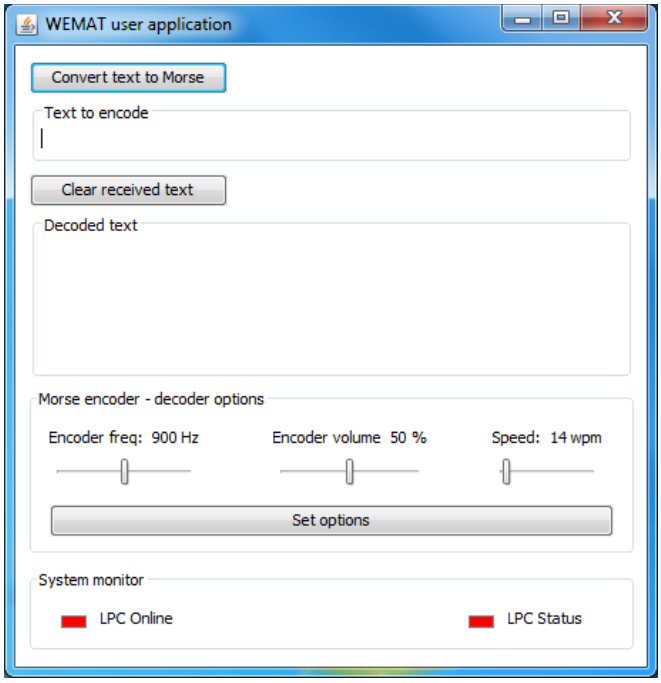


Figura : Interfaz de usuario

[9] Estado de la conexión y del sistema.

1.8 Breve descripción del resto de capítulos de la memoria

Hasta aquí se ha llevado a cabo la introducción del proyecto, así como su enfoque y planificación.

En los próximos apartados se detalla:

- ***Estado del arte y estudio de mercado***, el cual nos da una idea de la situación actual respecto a los sistemas empotrados y el mercado respecto a los productos obtenidos.

- ***Descripción funcional***, donde se detallan de manera esquemática el diseño empleado y el funcionamiento del sistema y la interfaz de usuario, objetos del TFC.

- ***Descripción detallada***, capítulo dedicado a la explicación del sistema en detalle, tanto del hardware como del software.

- ***Viabilidad técnica***, donde se muestran los puntos fuertes y débiles del sistema.

- ***Valoración económica***, capítulo del presupuesto del TFC.

- ***Conclusiones***, lista de objetivos cumplidos y sin cumplir, además de mostrar soluciones a problemas encontrados en la viabilidad técnica y una autoevaluación objetiva sobre el trabajo en el TFC.

- ***Glosario*** de términos que necesitan de aclaración. Asimismo, el autor ha proporcionado notas al pie de página para una consulta rápida.

- ***Bibliografía*** utilizada para la realización del proyecto con sus enlaces si están disponibles online.

- ***Anexos***, incluyendo los manuales de usuario tanto del sistema como la interfaz gráfica.

2. Antecedentes

En el pasado, los sistemas que incluían procesadores se utilizaban mayormente para cómputo. Estos equipos eran muy voluminosos, costosos, con un gran consumo de energía y difícil manejo. Además, el mantenimiento era muy costoso y sus capacidades de ampliación eran mínimas.

Con el transcurso de los años, estos sistemas fueron ganando en todos los aspectos anteriormente citados, a excepción de su tamaño el cual ha venido reduciéndose hasta el extremo. Debido a su mejor eficiencia y portabilidad, surgieron los *sistemas empotrados* como solución a una amplia gama de problemas en los cuales es necesario un dispositivo de control, portátil, que sea rápido en su ejecución y con un consumo muy bajo. Al contrario de los equipos de propósito general, como los ordenadores personales, estos dispositivos ofrecen una respuesta muy rápida a los distintos problemas para los cuales fueron desarrollados. Por lo tanto, se trata de sistemas muy eficientes creados para un propósito específico.

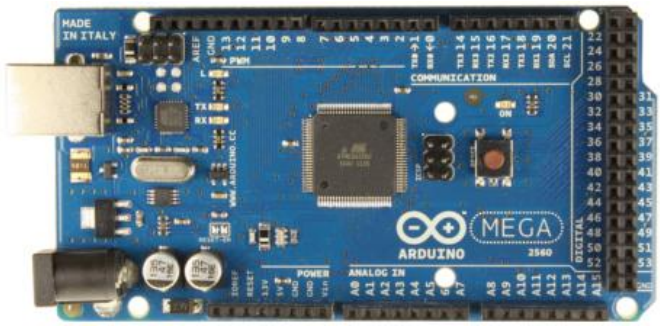
Hoy día estos sistemas han ganado mucho en popularidad y siguen evolucionando. De hecho, han evolucionado hasta tal punto que rivalizan en capacidad de cómputo con ordenadores personales potentes y permiten una gran variedad de soluciones para todo tipo de problemas en todos los campos, como la industria, la pequeña empresa y el hogar.

2.1 Estado del arte

Los sistemas empotrados han crecido mucho en popularidad, tanto que es fácil encontrarlos en cualquier sitio. También ha crecido su popularidad entre personas que desarrollan *gadgets* con ellos en casa. Ahora son muy accesibles en cuanto a su programación y contienen un número elevado de conexiones para todo tipo de periféricos.

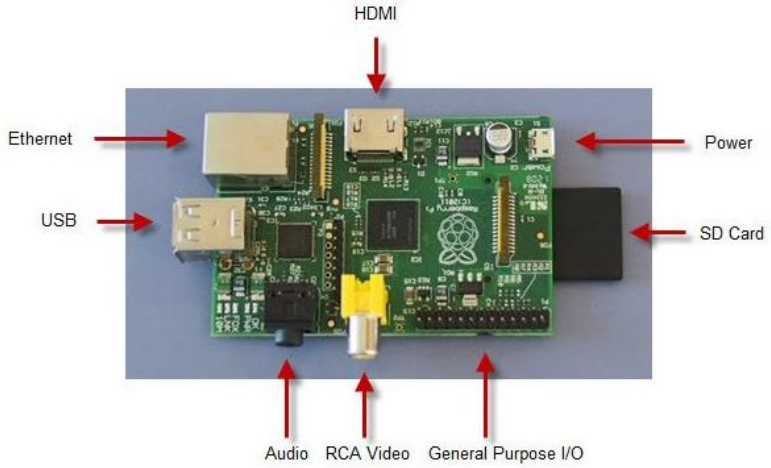
Llegados a este punto, nos centraremos en mostrar las características de dispositivos similares al utilizado en el proyecto, el LPC1769.
Un sistema que se ha hecho enormemente popular es Arduino, así como el emergente Raspberry. Mostraremos las características de un modelo concreto por cada uno y así poder comparar nuestro dispositivo.

Arduino Mega 2560:



Microcontrolador: ATmega 2560
Alimentación: 7 - 12v
Pines I/O digitales: 54, de los cuales 15 proveen de salida PWM
Entradas analógicas: 16
Flash: 256 KB, 8 KB utilizados por el bootloader
SRAM: 8 KB
EEPROM: 4 KB
CPU Clock: 16 MHz
Precio: 40€

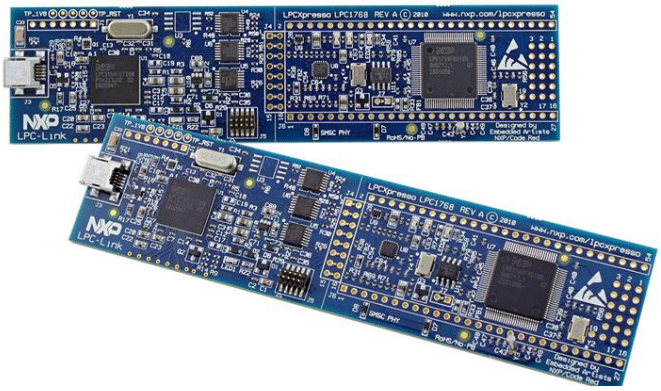
Raspberry pi:



Microcontrolador: ARM1176JZF-S
Alimentación: USB a 5v
Pines GPIO: 26

Periféricos: Audio, RCA Video, USB, Ethernet, HDMI, SD Card
Flash: SD Card
RAM: 512 MB
CPU Clock: 700 MHz (overclock a 1GHz)
Precio: 20€

- NXP LPCXpresso con LPC1769:



Microcontrolador: ARM Cortex-M3 LPC1769
Alimentación: USB a 5v
Pines GPIO: 70
Periféricos: Ethernet MAC, USB, 8 canales GPDMA, 4 UART, 2 CAN, 2 SSP, SPI, 3 I^2C, 2 I^2S, 8 canales ADC de 12 bit, DAC de 10 bit, PWM, bajo consumo RTC
Flash: 512 KB
RAM: 64 KB
CPU Clock: 120 MHz
Precio: 30€

Comparando los tres sistemas, el que menos prestaciones ofrece es el Arduino Mega 2560, además es más caro. Su ventaja radica en su entorno de desarrollo el cual abstrae a muy alto nivel del hardware, aunque no es una ventaja que no se pueda obtener con los otros dos sistemas.

El Raspberry PI se decanta como un dispositivo barato, aunque observando sus características se llega a la conclusión que se trata de un dispositivo más cercano al tipo de propósito general.

El LPC1769, el dispositivo utilizado en el proyecto, es el que más ofrece como sistema empotrado, además de buen precio. La cantidad de periféricos disponibles, junto con una buena velocidad de reloj, lo hace un dispositivo idóneo para su uso como sistema empotrado en cualquier entorno.

Seguidamente pasamos a detallar distintas tecnologías de conexión sin hilos que podemos utilizar junto al sistema empotrado empleado en el proyecto:

Especificaciones	Bluetooth	ZigBee	WiFi
Rate	1 Mbps	250 Kpbs	54 Mbps
Alcance	10-100m	10-100m	100-300m
Seguridad	Baja	Media	Alta
Potencia máx	10 dBm	-25 dBm	20 dBm
Canales	79	16	14
Latencia	200 ms	300 ms	150 ms
Presencia	Alta	Alta	Alta

De las tres tecnologías presentadas, la que mayores prestaciones nos aporta es el WiFi.
ZigBee está orientado a redes de sensores sin hilos, lo que lo hace idóneo para aplicaciones que utilicen sensores repartidos por el recinto, pero no para la aplicación que se presenta en este proyecto.

Bluetooth puede servir para nuestros propósitos, aunque su punto débil respecto a WiFi se da en seguridad. Si utilizamos nuestro dispositivo en un área altamente congestionada la seguridad en la comunicación se hace imprescindible. Como ventaja presenta un gran número de canales.

WiFi se presenta como la tecnología idónea para el producto presentado en este TFC. Además, ofrece mayor facilidad para crear conexiones TCP, utilizadas por el dispositivo presentado.

2.2 Estudio de mercado

El mercado hacia el que va dirigido el producto obtenido en este TFC se engloba en amateurs y radioaficionados, dado que son el público general en utilizar el código Morse. Su uso militar también está extendido, utilizando Morse incluso con focos de luz. El porqué del uso de la codificación Morse viene dado por su facilidad de implementación con casi cualquier método que pueda conmutar entre dos estados. Además, se encuentra su mayor fiabilidad frente al ruido comparado con otras tecnologías más modernas, aunque no rivaliza con las digitales.

Tras una búsqueda en la red se han encontrado un par de dispositivos portátiles capaces de decodificar código Morse, aunque sólo uno de ellos puede hacer el paso inverso: La codificación a Morse y su emisión en sonido.

Figura : CW Reader a la izquierda, MFJ461 a la derecha

Las características de los dispositivos son las siguientes:

Característica	CW Reader Decoder Keyer	MFJ461 CW
Velocidad máx	60 wpm	99 wpm
Ajuste wpm auto	SI	SI
Micrófono	NO	SI
Conexión sin hilos	NO	NO
Conexión con ordenador	NO	SI (RS-232)
Aplicación usuario	NO	NO
Codificación Morse	SI	NO

Si los comparamos con el sistema presentado en el TFC, vemos que ambos lo superan en cuanto a la capacidad de ajuste automático de velocidad de escucha. Sin embargo, sólo el MFJ461 CW lo supera en velocidad de decodificación Morse. En cuanto a sistema portátil, ambos dispositivos comparten la característica de pequeño tamaño.

El sistema presentado, sin embargo, tiene facilidad para incorporar las dos primeras características. Incluso, modificando la aplicación Java se le puede exigir una velocidad de decodificación máxima de 100wpm o incluso más, el límite de 80wpm se impuso sin ninguna razón en particular. En el apartado conclusiones de esta memoria se detalla la implementación del ajuste automático de velocidad, totalmente viable en el dispositivo objeto de este proyecto. E incluso, dada la potencia de la CPU en el LPC1769, se pueden implementar mejoras en el software, introduciendo análisis en frecuencia entre otros. Precisamente, el sistema presentado es totalmente ampliable.

Los dispositivos presentados en la comparativa carecen de aplicación de usuario y de opciones de configuración de volumen o tono de sonido en el CW Reader. Tampoco incorporan capacidad de conexión sin hilos y uno de ellos ni siquiera se puede conectar a otro equipo. La conexión con un ordenador puede ampliar las posibilidades del sistema si los dos equipos, ordenador y sistema empotrado, se reparten las características. Así, desde el sistema empotrado podríamos mandar datos al ordenador y que éste haga un análisis en frecuencia, o viceversa. Las posibilidades son enormes.

El público al que están destinados los dos equipos es el mismo, el radioamateur.

3. Descripción funcional

En el presente capítulo se describe funcionalmente el sistema. Éste debe ser capaz de recibir y transmitir mensajes de audio en código Morse. Por esta razón, el sistema incorpora un pequeño micrófono para la escucha y un pequeño altavoz para generar los mensajes en audio. Si estamos escuchando sonido en código Morse el dispositivo lo recibe, lo traduce a texto y lo envía. Si deseamos traducir texto a código Morse el dispositivo lo recibe, lo codifica y lo emite a través del altavoz. Para el envío y la recepción de texto el sistema utiliza conexión sin hilos (Wifi).

La interacción con el usuario se lleva a cabo con un ordenador capaz de conectar por Wifi y una aplicación Java instalada que, mediante interfaz gráfica, proporciona lo necesario para la traducción de mensajes, configuración e información del estado del sistema y la conexión.
Por último, el dispositivo utiliza conexión USB para permitir la configuración inicial de la comunicación Wifi.

3.1 WEMAT (Wifi Embedded Morse Automatic Translator)

a) Diagrama de bloques del sistema

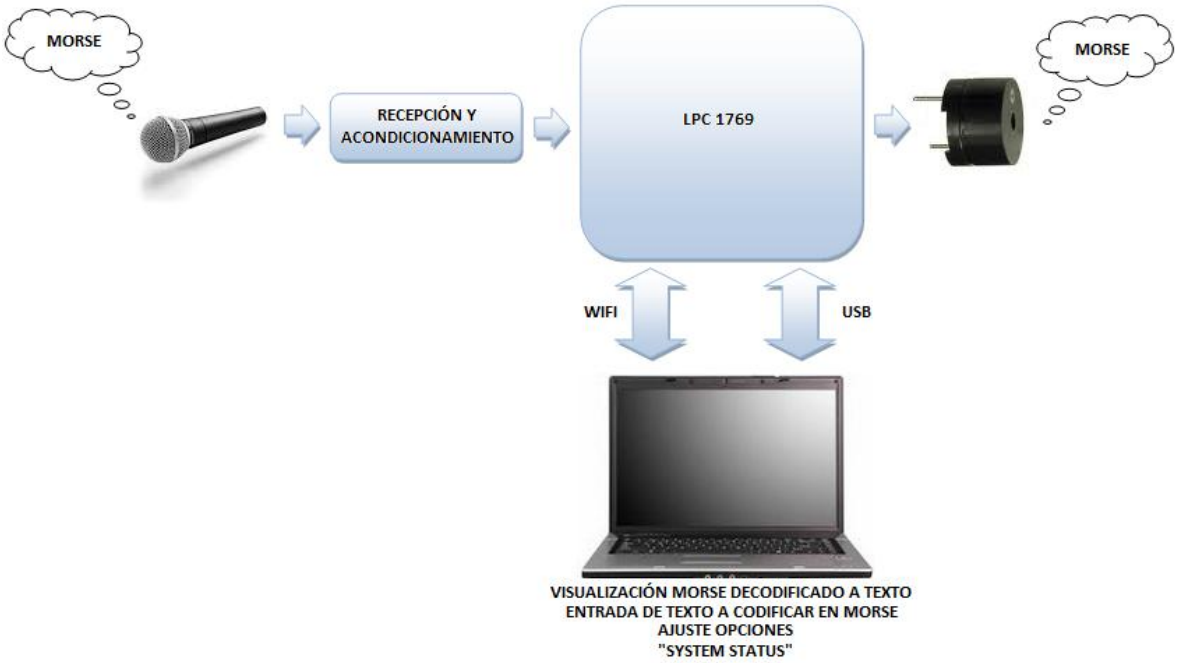


Figura : Diagrama de bloques del sistema

b) Descripción de la red

El sistema utiliza el módulo Wifly RN-171 que proporciona conexión sin hilos a la red para la comunicación del dispositivo con el ordenador que alberga la interfaz de usuario. Se encuentra capacitado para trabajar con redes Wifi b/g hasta 54 Mbit/sec (configurado a 24 Mbit/sec) y autenticación segura WEP-128, WPA-PSK y WPA2-PSK. También soporta auto-conexión, característica que se ha utilizado en este TFC.

Cuando el sistema inicia requiere los parámetros de la red a la que conectar como SSID, contraseña si es necesaria y la dirección IP del equipo destino. Una vez configurado, busca la red e intenta la conexión.
Si el dispositivo ha conseguido conectarse a la red Wifi, intenta crear una conexión TCP a través del puerto 6001 actuando como cliente. La rapidez de los datos que se intercambian las distintas partes del sistema no son una prioridad, pero sí que lo son el orden de llegada y el control de errores en los paquetes. Por esta razón se ha seleccionado el protocolo TCP, la capa de transporte se ocupa de la fiabilidad de los datos a la vez que los entrega ordenados.
La aplicación Java de usuario actúa de servidor y está configurada para aceptar la conexión entrante del sistema empotrado. Una vez establecida la conexión, el sistema queda preparado para enviar y recibir texto[10], opciones de configuración e información sobre el estado de la conexión y del dispositivo.

c) Interacción de los distintos objetos del sistema

Podemos dividir la interacción completa del sistema en cuatro acciones bien diferenciadas:

- ***Inicialización del sistema y la red***: Cuando el dispositivo inicia, busca conectarse pidiendo los datos de la red Wifi. Aquí, el sistema proporciona una serie de preguntas y muestra los parámetros introducidos antes de conectarse para su comprobación por parte del usuario. Una opción para introducir el SSID al cual conectar es mostrar la lista de redes WLAN disponibles. Con esta opción sólo hace falta proporcionar el número de la lista en vez de introducir el nombre de SSID. Dicha opción no se ha implementado ya que este sistema puede no mostrar todos los accesos WLAN disponibles, bien porque el punto de acceso no lo muestra por cuestiones de seguridad o bien por falta de señal, entre otros. En las pruebas se comprobó dicha opción y el SSID no apareció en algunas de ellas aún cuando la intensidad de la señal no era un problema. Por esta razón se decidió que el sistema pidiera el nombre directamente.

[10] En siguientes apartados se detalla el formato y la codificación del texto.

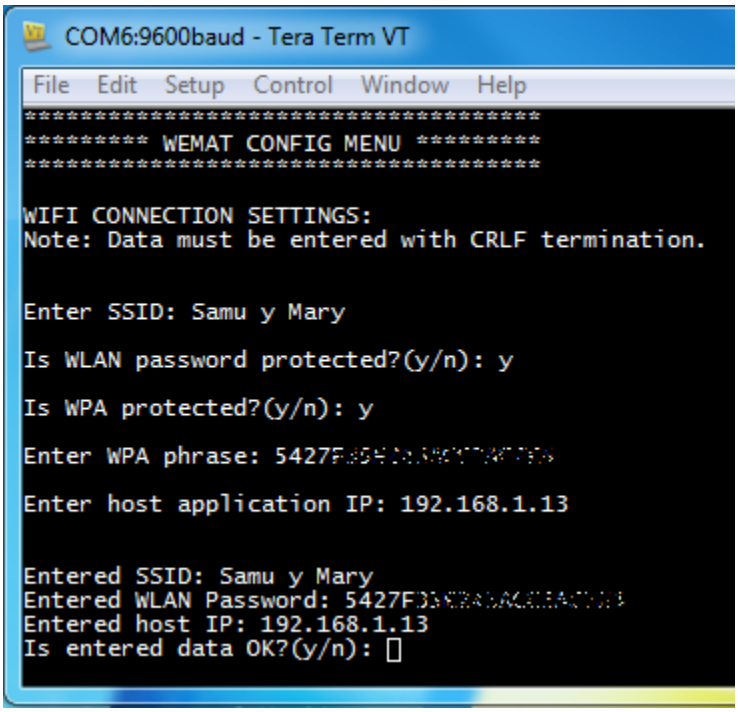


Figura : Detalle del menú de configuración inicial de red

En este punto, y una vez establecida la comunicación Wifi, el dispositivo y la aplicación Java en el ordenador intentan conectarse. Si se ha establecido la conexión el dispositivo manda datos de configuración a la aplicación de usuario para actualizar los controles de ésta a los valores que tiene el sistema por defecto, que son el tono, volumen y la velocidad a la que el sistema escucha y emite (en palabras por minuto).

- ***Recepción de código Morse***: El sistema incluye un circuito que transforma la señal de audio recibida en impulsos que el dispositivo puede tratar. Dicho circuito recibe la señal con un micrófono electret, la filtra para eliminar sonidos que no corresponden con la banda de frecuencias necesaria, la amplifica y convierte en impulsos con un voltaje adecuado. La salida de dicho circuito se conecta con un pin del dispositivo el cual, mediante interrupciones en flancos de subida y bajada, detecta y mide su tiempo de actividad. Si la señal corresponde con código Morse, el microcontrolador la decodifica a texto y la envía a la aplicación de usuario mediante la conexión TCP realizada a través de Wifi (puerto 6001).

- ***Transmisión de código Morse***: Si el usuario, a través de la aplicación Java en su equipo, envía texto para codificar en Morse, el dispositivo la codifica y la envía al buzzer mediante una señal PWM de duración determinada. Dicha señal excita el transductor, el cual

convierte en sonido, produciéndose audio en código Morse. El texto se envía a través de la conexión TCP establecida con el sistema empotrado mediante Wifi.

- ***Envío de datos de configuración al dispositivo***: Mientras el dispositivo no se encuentra ocupado, el usuario puede enviar la nueva configuración desde la aplicación Java. El dispositivo recibe los datos y los aplica de inmediato. Los parámetros configurables disponibles en la aplicación de usuario son tono, volumen y velocidad de la traducción (en palabras por minuto).

3.2 Aplicación de usuario

a) Diagrama de bloques de la aplicación de usuario

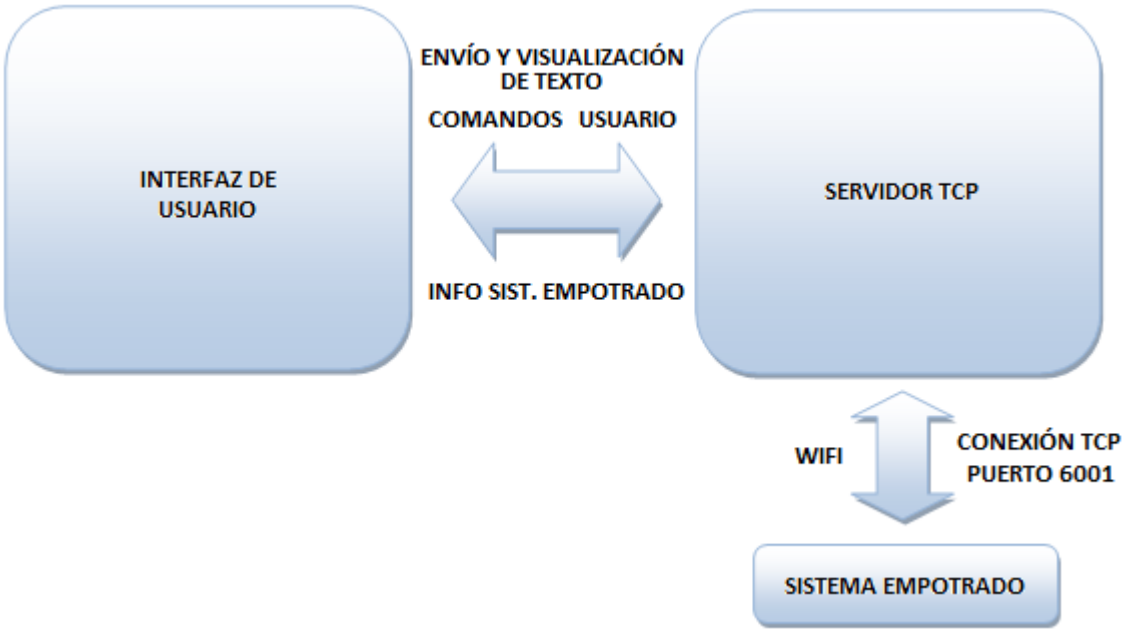


Figura : Diagrama de bloques de la aplicación de usuario

b) Interacción de los distintos objetos de la aplicación de usuario

La aplicación de usuario se divide en dos partes bien diferenciadas, que son la interfaz gráfica de usuario y un pequeño servidor TCP que escucha a través del puerto 6001 del ordenador.
Cuando el usuario inicia la aplicación, se configura la interfaz de usuario y el servidor TCP queda a la escucha para permitir la conexión desde el sistema empotrado. Una vez conectado, interactúa con la interfaz de usuario en tres casos:

- ***Actualización del estado de la conexión***: Cuando el servidor TCP acepta la conexión, actualiza el estado de la misma en la interfaz de usuario.

- ***Recepción de texto y/o configuración desde el sistema empotrado***: El servidor TCP actualiza la interfaz de usuario con los datos que recibe del sistema empotrado, tanto texto como configuración y estado.

- ***Envío de texto y/o configuración desde la interfaz de usuario***: El usuario, al interactuar con la interfaz gráfica de la aplicación, envía texto y comandos al objeto servidor TCP. Éste último, a su vez, envía dichos datos al sistema empotrado a través de la conexión establecida previamente.

3.3 Sistema empotrado (LPC 1769)

a) Diagrama de bloques

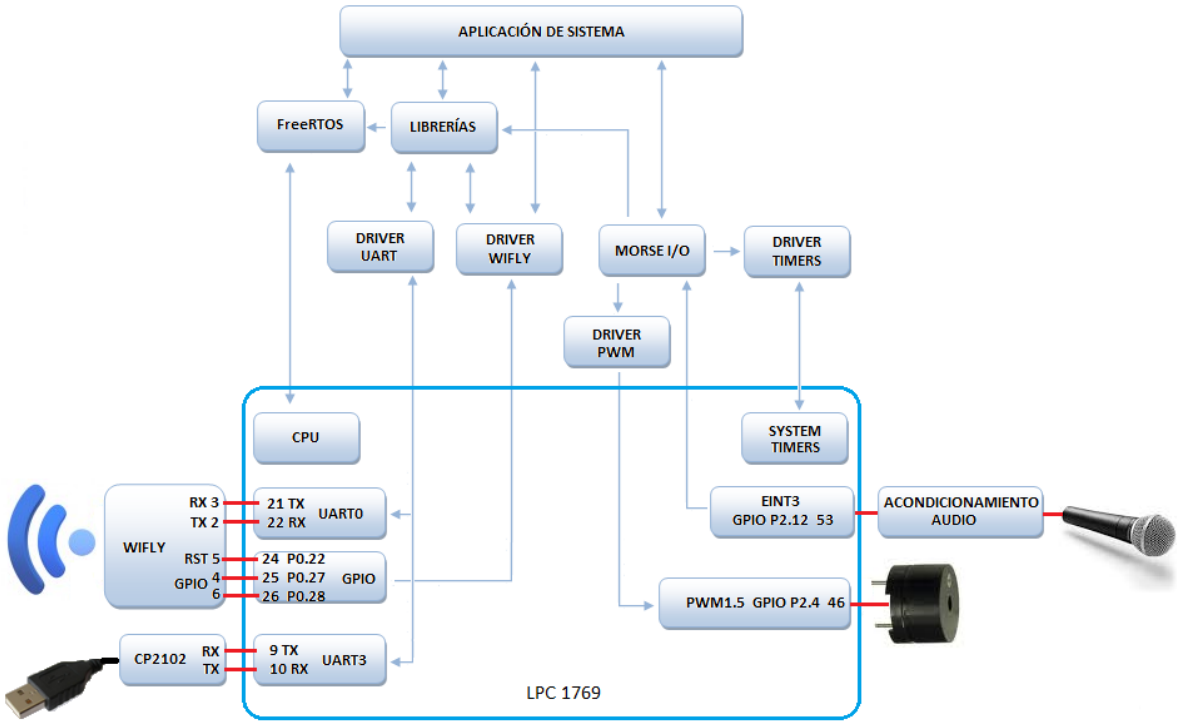


Figura : Diagrama de bloques del sistema empotrado

b) Diseño y funcionamiento de la aplicación del sistema empotrado

En el diseño seguido para el desarrollo de la aplicación del sistema empotrado destaca la *modularidad*. Se basa en dividir el sistema en distintas *capas*[11], desde los drivers a bajo nivel hasta la aplicación principal en el más alto, aportando las siguientes ventajas:

- ***Mayor facilidad para desarrollar la aplicación final***, la cual descansa sobre el resto de capas, que aportan abstracción sobre código de bajo nivel que resulta mucho más difícil de interpretar y se encuentra fuertemente ligado al hardware.

- ***Proporciona aislamiento de problemas*** que puedan surgir. Los errores se pueden solventar fácilmente identificando el área del problema y su correspondiente módulo.

- ***Aumenta considerablemente la ampliación de funcionalidad***, simplemente añadiendo y modificando los módulos correspondientes.

A continuación se detalla el funcionamiento dividido en bloques y enlazando todos los elementos que intervienen en cada uno de ellos:

- ***Captación y decodificación de código Morse***: La primera etapa la forman el micrófono y el circuito de acondicionamiento de la señal de audio. Aquí la señal se filtra y se convierte en impulsos de duración variable que recoge el sistema empotrado por el pin 53. En dicho pin la señal produce interrupciones que opera el driver *I/O Morse* el cual descarta ruido, decodifica a texto y lo manda a la aplicación de sistema para su envío posterior por Wifi.

- ***Emisión de sonido en código Morse***: En este caso la aplicación de sistema envía texto (procedente de la conexión Wifi) al driver *I/O Morse* el cual lo codifica y genera señal PWM de duración variable a través del *driver PWM*, utilizando el canal 5 por el pin 46 del sistema empotrado. En dicho pin se conecta un pequeño transductor electromagnético que emite los sonidos en código Morse. La frecuencia de la señal PWM se corresponde con el tono del sonido emitido y el *duty cycle*[12] de dicha señal determina el volumen al que se escucha. El driver I/O Morse controla estos parámetros a petición de la aplicación de sistema.

- ***Envío y recepción de texto por Wifi***: Los drivers UART y Wifly, junto a las librerías de texto a UART y sincronización con semáforos, proporcionan el control del módulo Wifly

[11] El término "capas" se refiere a los distintos niveles que abstraen del hardware y facilitan el aislamiento y la comprensión del código.
[12] Fracción de tiempo de un ciclo de señal PWM en el cual se encuentra en estado alto.

RN-171 que se encarga de la conexión sin hilos del sistema empotrado. La conexión del módulo se realiza a través del puerto 0 de UART, pines 21 y 22 del sistema empotrado. La aplicación de sistema procesa los datos recibidos a través de UART observando si existe *cabecera*[13] en los mismos. La cabecera sirve de control para discriminar los mensajes de texto de comandos de configuración y mensajes de información procedentes del módulo Wifly. Igualmente, la aplicación de sistema manda los datos al módulo con una cabecera que indica a la aplicación de usuario si se trata de texto u otro tipo de información.

- ***Control del estado de la conexión***: El módulo Wifly puede informar del estado de la conexión a través de comandos disponibles en el *driver Wifly*. No obstante, cuando el módulo se encuentra ocupado puede tardar en responder o, incluso, los comandos enviados pueden hacer caer la conexión. Para evitar esto, en el proyecto se ha optado por utilizar dos de los pines GPIO que incorpora el módulo Wifly y que informan sobre el estado de la conexión por hardware. Dichos pines son el 4 y 6 del módulo y se conectan al sistema empotrado a través de los pines 25 y 26 del mismo. El driver Wifly implementa la configuración y lectura de éstos a través del GPIO puerto 0 presente en el dispositivo.

- ***El módulo Wifly incorpora un pin de reinicio*** que utiliza la aplicación de sistema a través del *driver Wifly* y que nos permite asegurar que el dispositivo se encuentre disponible y admitir la configuración inicial. El acceso a dicho pin se realiza a través del GPIO puerto 0 pin 24.

- ***Conexión USB para configuración y debug***: El módulo CP2102 proporciona un puente de USB a UART y se conecta a través de los pines 9 y 10 del sistema empotrado. Dichos pines dan acceso al puerto 3 del UART del dispositivo. Su función es la configuración inicial del sistema y proporcionar información sobre las acciones realizadas por el mismo. La aplicación del sistema accede a través de la librería texto a UART con acceso controlado por semáforos, la cual a su vez controla el driver UART en el puerto adecuado.

- ***El sistema operativo en tiempo real***, *FreeRTOS*[14], proporciona todo lo necesario para el control de semáforos, tareas y CPU a la aplicación de sistema y permite un uso eficiente del sistema empotrado.

[13] El formato de los datos se detalla en el capítulo 4.

[14] Sistema operativo en tiempo real, libre, utilizado en el proyecto.

4. Descripción detallada

En el presente capítulo se realiza la descripción detallada del sistema, la aplicación de usuario y el sistema empotrado. En el capítulo anterior se describían el diseño y las partes del sistema de manera funcional, ahora se profundizará en aspectos más técnicos, tanto en hardware como en software, que permitirá aclarar conceptos que han podido quedar difusos o faltos de explicación.

4.1 WEMAT (Wifi Embedded Morse Automatic Translator)

a) Diagrama de bloques detallado del sistema

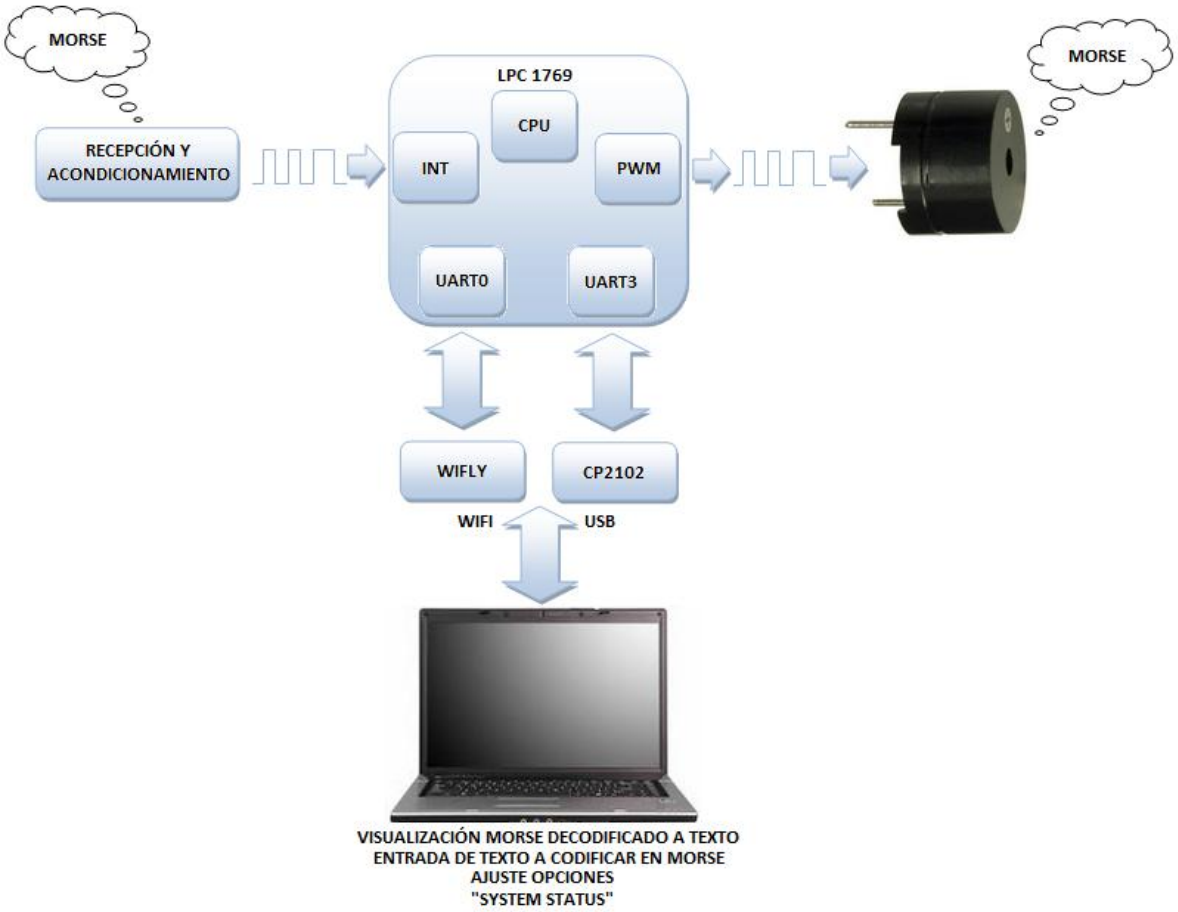


Figura : Diagrama de bloques detallado del sistema

b) Diagrama de ejecución del sistema

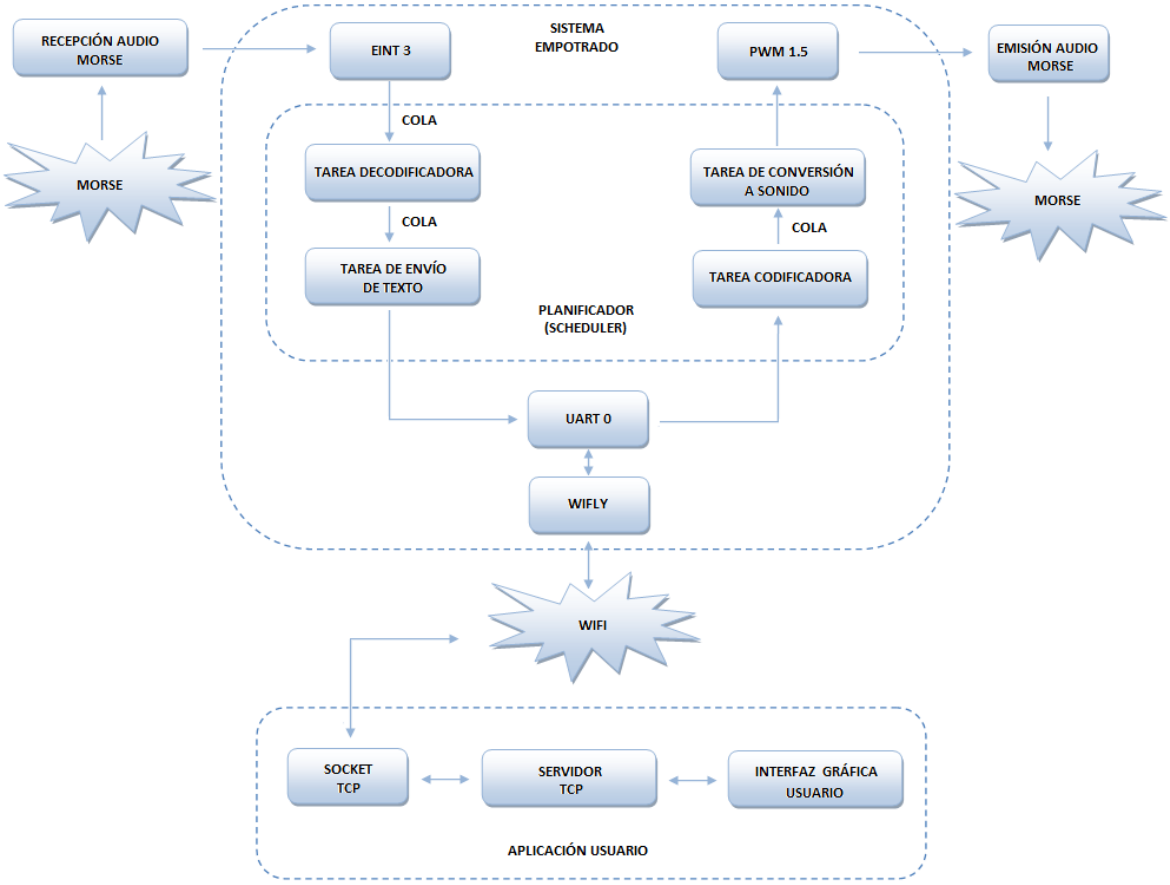


Figura : Diagrama de ejecución del sistema

c) Detalles técnicos del sistema

En este sub apartado se detalla el sistema desde la entrada de audio hasta su salida por altavoz, incluyendo los tipos de datos y/o señales en cada etapa.

- Circuito de entrada y acondicionamiento de audio

El sistema está preparado para recibir código Morse en señales de audio. Aunque el LPC1769 viene preparado con ADCs[15] que pueden tratar este tipo de señales, el procesamiento se hace costoso y la velocidad de ejecución puede verse afectada, por no hablar del nivel de complejidad en los algoritmos que tendrían que incluirse.

Por esta razón, se ha optado por la realización de un circuito electrónico que realice esta tarea y liberar al sistema empotrado de realizarla por software.

[15] Siglas en Inglés de convertidores analógico-digital.

El esquema electrónico de dicho circuito se detalla a continuación:

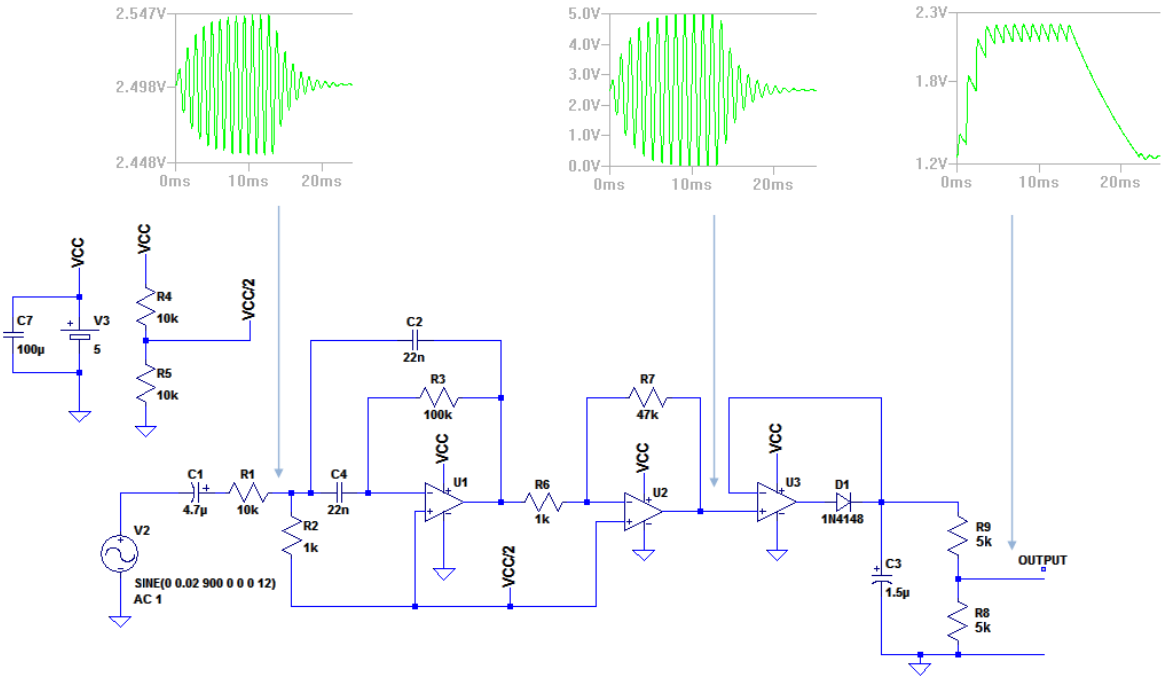


Figura : Esquema de la etapa de entrada y acondicionamiento de audio

En la figura se pueden observar las formas de onda en cada etapa del circuito. Se ha diseñado en base a una tensión de alimentación de 5V, de esta forma se ofrece la posibilidad de alimentarlo a partir del USB, como el LPC1769, sin necesidad de otra fuente de alimentación. A continuación se detalla cada una de ellas:

- Etapa de entrada y filtrado:

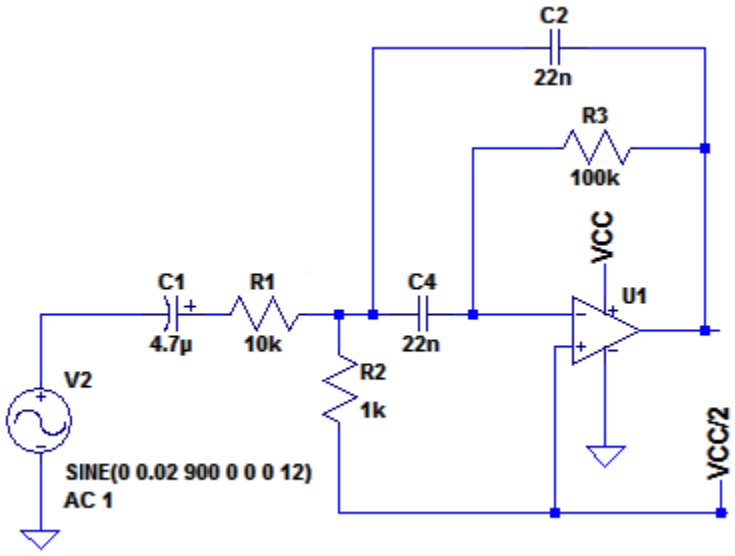


Figura : Circuito de entrada y filtro MFB con TL071

Esta etapa la forman el micrófono y un filtro pasa banda con topología de realimentación múltiple (*MFB*[16]). El Condensador electrolítico C1 desacopla la DC que alimenta al micrófono electret. El filtro pasa banda, configurado como en el esquema, atenúa todas las frecuencias que no entren en el rango estándar de código Morse (600Hz - 1KHz). De esta forma se descartan ruidos y sonidos que no interesan. En el esquema electrónico, el micrófono se ha simulado con una fuente senoidal de 900Hz y 0.01V de amplitud.

El filtro pasa banda MFB hace uso de un único amplificador operacional y resulta sencillo de implementar. Las ecuaciones de diseño utilizadas para su cálculo son las siguientes:

$$R3 = \frac{Q}{\pi f C}$$

$$R1 = \frac{R3}{-2 \cdot A}$$

$$R2 = \frac{-A \cdot R1}{2Q^2 + A}$$

Y eligiendo los valores:

$$C4 = C2 = 22\ nF$$

$$Q = 5$$

$$f = 800\ Hz$$

$$A = -5$$

Resultan los valores del circuito. El valor de la frecuencia central se puede ajustar con R2, por lo tanto en el ensamblaje del circuito se ha sustituido por un potenciómetro de 5K que permita un ajuste fino. El valor de la ganancia, A, es negativo debido a que el circuito actúa como inversor. Este hecho no tiene importancia debido a que la polaridad de la señal no es importante. La ganancia del circuito no se ha seleccionado alta por razones de estabilidad del circuito.

- Etapa de amplificación:

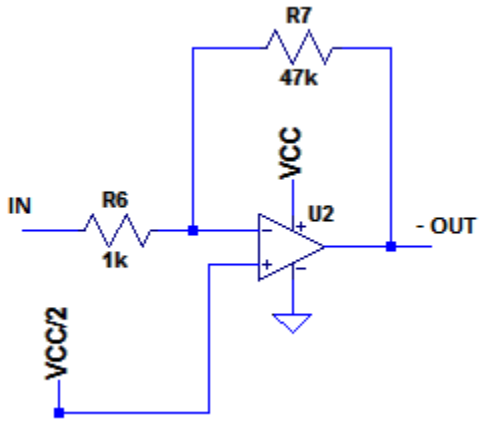


Figura : Amplificador inversor con TL071

[16] Siglas en Inglés de *Multiple Feedback*.

La siguiente etapa es un simple amplificador inversor, con una ganancia de:

$$A = -\frac{R7}{R6}$$

La señal vuelve a invertirse, con lo que a la salida de esta etapa tiene la misma polaridad que a la entrada, aunque como ya se ha indicado este hecho no tiene importancia en el resultado.

Con la máxima ganancia de la etapa de filtrado más la actual tendremos una señal $5 \cdot 47 = 235$ veces la amplitud de la entrada. Ya que la ganancia es considerable y las frecuencias implicadas rondan el KHz, los amplificadores operacionales seleccionados son de propósito general pero con un slew rate mayor y menor figura de ruido que, por ejemplo, el conocido LM741. Sus características lo hacen ideal para circuitos de audio y su precio es bajo.

El modelo seleccionado para ambas etapas es el TL072, que son dos TL071 en el mismo encapsulado (DIN de 8 patillas).

- Detección de envolvente:

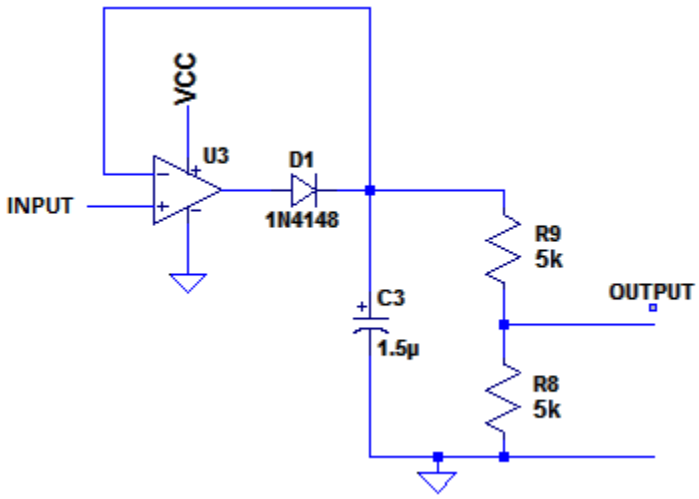


Figura : Detector de envolvente con LM741

La última etapa de compone de un detector de envolvente con amplificador operacional. La clave aquí es el diodo D1, el cual rectifica la señal de entrada a valores positivos. Sus características deben cumplir, principalmente:

- Que sea un diodo de pequeña señal.
- Que sea lo suficientemente rápido para rectificar señales de frecuencia mayor que 1 KHz.

El diodo seleccionado es el conocido 1N4148, un diodo de pequeña señal capaz de rectificar señales con frecuencias de 100 MHz. Además, es barato y muy fácil de localizar.

En esta etapa se ha incluido un amplificador operacional que compensa la caída de tensión en el diodo rectificador. Debido a la poca amplificación que proporciona, se ha utilizado un operacional de propósito general, LM741.

La función del condensador C3 es la de eliminar el rizado de la señal rectificada, proporcionando impulsos cuadrados que siguen la amplitud de las señales en código Morse captadas por el micrófono. El divisor de tensión formado por R8 y R9 permiten el ajuste a un rango de tensión adecuado. Dado el offset introducido por los amplificadores operacionales, es necesario ajustar su valor para que, en ausencia de señal, la salida se encuentre por debajo de 1.7V. Esta tensión es el umbral de los pines GPIO del LPC1769, por debajo de ésta se toma como nivel bajo. Al contrario, por encima de 1.7V se toma como nivel alto.

En el ensamblaje del circuito, se ha sustituido el divisor de tensión formado por R8 y R9 por un potenciómetro de 10K y permitiendo ajustar la tensión a la que detecta el LPC1769 y así ajustamos la sensibilidad.

Para el cálculo del valor de C3 se tiene en cuenta que forma un circuito RC con las resistencias R8 y R9 y, además, el valor de tensión a la que se tomará como nivel bajo en el sistema empotrado.

Teniendo en cuenta la ecuación de descarga del circuito RC, tenemos que:

$$V_C(t) = Ke^{-\frac{t}{RC}}$$

Y teniendo en cuenta los valores de tensión en la salida:

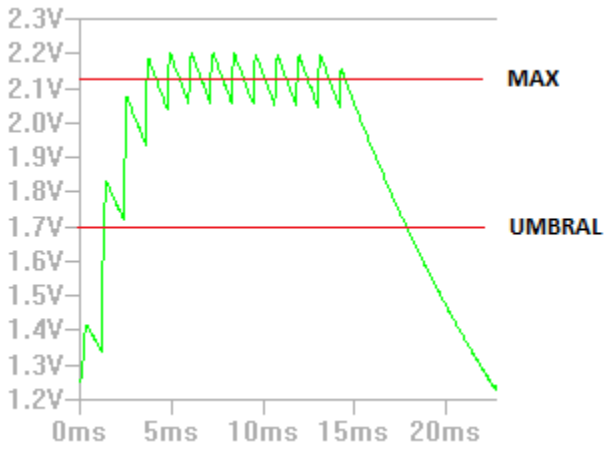


Figura : Señal de salida teórica calculada con LTSpice

Teniendo en cuenta que el sistema empotrado tiene que trabajar con un máximo de 80 palabras por minuto, tomando como referencia estándar la palabra *PARIS*[17], tenemos que el tiempo máximo de descarga del circuito RC a 1.7V es:

$$t(ms) = \frac{1200}{80} = 15\ ms$$

La máxima tensión que alcanza el circuito es de 2.2V y el umbral para que el LPC1769 lo tome como nivel bajo es 1.7V, es decir, el 77% del voltaje máximo. Con estos datos, el valor máximo que puede tomar el condensador C3 es:

$$C = \frac{-t}{ln(0.77) \cdot R} = \frac{-0.015}{-0.26 \cdot 10000} = 5.76\ \mu F$$

[17] Más adelante se explica esta referencia de tiempo en código Morse.

Para asegurar, se ha seleccionado un valor de 1.5uF con tres condensadores de 4.7uF en serie que se encontraban disponibles en la gaveta de componentes. Con un condensador de 4.7uF se hicieron pruebas y, a la máxima velocidad de 80 palabras por minuto, a veces se encontraban errores en la decodificación, debido a que en la práctica los valores de tensión máxima a la salida pueden llegar a ser mayores[18], provocando que el tiempo de descarga sea también mayor. Con 1.5uF las pruebas nunca han dado errores debidos a este problema.

El circuito ensamblado se detalla a continuación:

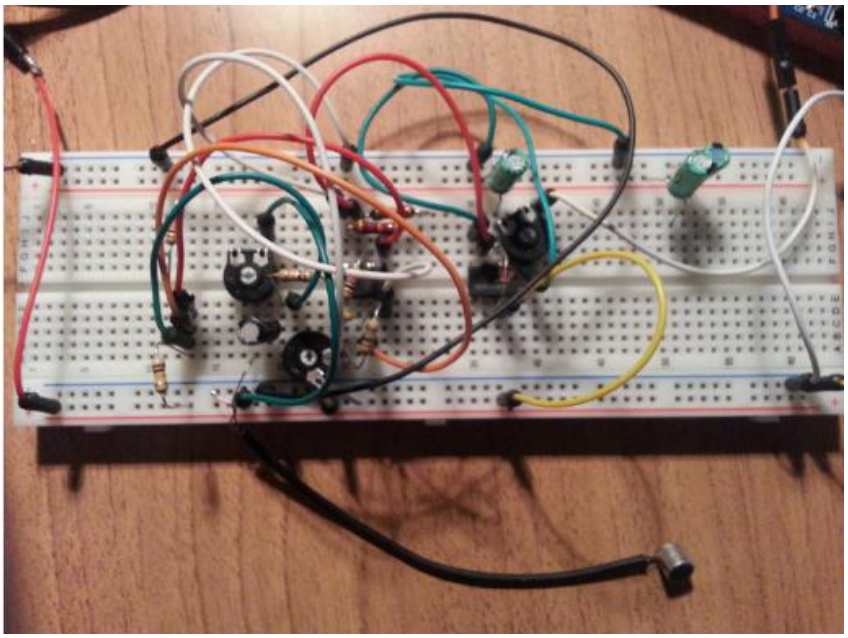


Figura : Detalle del circuito de audio ensamblado

- Entrada de señal en el sistema empotrado y uso de interrupciones

En esta etapa, el sistema empotrado detecta los flancos de subida y de bajada de la señal entregada por el circuito de entrada de audio por medio de interrupciones, con un umbral de 1.7V, y hace uso de uno de sus timers para medir tiempos entre interrupciones.

En el momento que se genera una interrupción por flanco de subida, el dispositivo pone a 0 el timer de sistema utilizado. Cuando, al contrario, se genera una interrupción por flanco de bajada, el dispositivo mide el tiempo que ha pasado y lo almacena como *tiempo de señal*. En este momento viene una pausa la cual, cuando termina, se genera otra interrupción por flanco de subida y se mide el tiempo de pausa como:

$$t_{pausa} = t_{total} - t_{señal}$$

[18] En la práctica, los valores de tensión de alimentación y las tensiones entregadas por el micrófono electret pueden ser ligeramente mayores.

Las medidas de tiempos se pueden entender fácilmente mediante el siguiente gráfico:

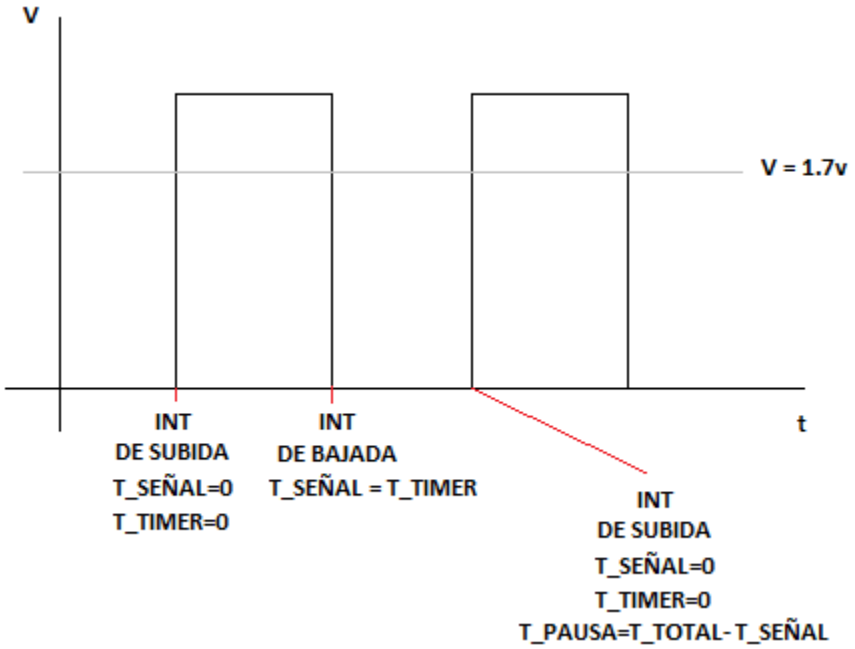


Figura : Medidas de tiempo mediante interrupciones por flanco

En la primera interrupción que se produce, el tiempo de pausa resulta muy largo. El sistema ignora los tiempos de pausa mayores de 1 segundo y se toma como el primer símbolo Morse a tener en cuenta.

Estos cálculos se hacen mediante el driver *I/O Morse* (morse.c en el código fuente) y determina si es un punto, una raya y si pertenece a un nuevo carácter.

En código Morse, los tiempos se toman con la siguiente tabla estándar:

- El tiempo del símbolo "punto" es la unidad de tiempo de referencia.
- El tiempo del símbolo "raya" es 3 x tiempo punto.
- El tiempo entre símbolos es 1 x tiempo punto.
- El tiempo de pausa entre caracteres es 3 x tiempo punto.
- El tiempo de pausa entre palabras es de 7 x tiempo punto.

Para saber si el tiempo de señal corresponde a un *punto o una raya*[19], el sistema utiliza la medida estándar de cantidad de palabras *PARIS*[20] por minuto. Es la mayormente adoptada, y utiliza la palabra PARIS como unidad de medida. La razón es que esta palabra, en código Morse, tiene una longitud exacta, en tiempo, de 50 puntos. Entonces, sabiendo el número de palabras que se envían en un minuto, se puede saber el tiempo por símbolo "punto" (que es el tiempo de referencia, el más corto). La relación se queda como:

$$t_{punto} = \frac{60s}{num\_palabras \cdot 50}$$

[19] En Morse es común utilizar *dot* para punto y *dash* para raya.

[20] PARIS se corresponde en código Morse con .--. .- .-. .. ...

Esto, referido a la palabra PARIS. Una fórmula directa de esta relación, que es la utilizada en el proyecto, es:

$$t_{punto} = \frac{1200}{palabras\ \ por\ minuto\ (wpm)}$$

El resultado viene dado en milisegundos. De esta forma, sabiendo la velocidad en palabras por minuto, el sistema calcula el tiempo de referencia y decodifica los símbolos con la tabla de tiempos estándar anterior.

Las interrupciones van poniendo en cola los símbolos. Cuando se tiene un carácter, se decodifica y se envía a la cola de envío por Wifi. Si la pausa entre caracteres corresponde a un espacio en blanco, se pone en cola el carácter + espacio en blanco.

Añadir que el sistema toma como pausa entre palabras a toda pausa de tiempo igual o mayor que 4 puntos, y el tiempo tomado como punto tiene un rango de un 15% para permitir código cuyos tiempos no sean "exactos".

- De Morse a texto y de texto a Morse:

Tras investigar diversos métodos se seleccionó como probables dos de ellos:

- Correspondencia directa en un array por medio de números binarios.
- Búsqueda dicotómica, también partiendo de números binarios.

La búsqueda dicotómica complicaba el algoritmo y hacía uso de operaciones matemáticas, como exponenciación. Para un sistema empotrado, puede resultar un inconveniente serio en la velocidad de ejecución.

Por lo tanto, se optó por la primera opción.

La solución se basa en dos pasos:Primero, se hace corresponder al punto con un "1" y a la raya con un "0". Después, se toma una variable entera de 8 bits con el valor numérico 1. Por cada símbolo, basta con hacer un *shift de bits* hacia la izquierda. Si el símbolo es un "1", se le suma 1 a la variable después del corrimiento de bits hacia la izquierda (se queda un "1"). Si es un "0" tan sólo es necesario el corrimiento de bits.

El proceso puede observarse en el siguiente gráfico:

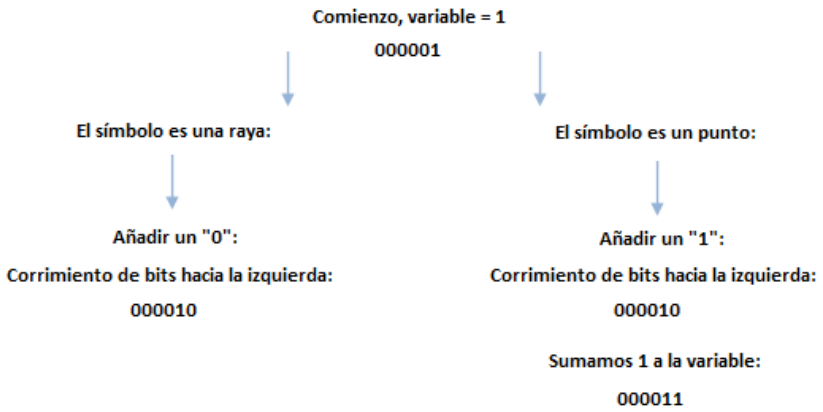


Figura : Primer paso en la decodificación de código Morse

Como se ha podido observar, la variable entera comienza con el valor 1, esto es así para que cada carácter tenga un número binario diferente.

Una vez tenemos todos los símbolos que corresponden a un carácter, el código binario resultante es unívoco y podemos saber cual es por relación directa. Los caracteres se buscan en un array en el cual la posición de cada carácter corresponde con el número entero resultante del binario anterior, es decir, el número binario resultante da un valor decimal a la variable entera. Todo lo que hay que hacer es extraer el carácter con dicha posición en el array.

El número de caracteres de código Morse estándar resultante en el array no es largo:

"##TEMNAIOGKDWRUS##QZYCXBJP#L#FVH";

Para la codificación, se hace el proceso inverso. Se busca el caracter en el array y, una vez encontrado, el número de posición resultante corresponde con sus símbolos en código Morse una vez se pasa a binario.

El problema viene cuando queremos incluir caracteres no estándar y los correspondientes a numeración, los espacios no válidos en el array se incrementan y, en ese caso, sí que resulta en un array más largo. La solución adoptada ha sido añadir el código extendido en otro array, con lo que se reduce mucho su tamaño. A la vez, disminuye el tiempo de búsqueda de caracteres ya que su número ASCII correspondiente nos indica si hay que buscarlo en un array o en el otro, simplemente con la relación:

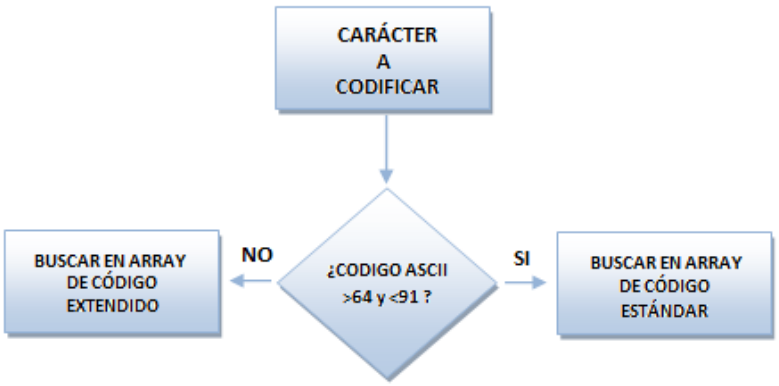


Figura : Selección de array en el que realizar la búsqueda

Como la posición en el segundo array no coincide con el número binario correspondiente, cada elemento de dicho array proporciona información del carácter ASCII y de su correspondencia en código binario, con lo que cada elemento ocupa 16 bits en vez de 8. Este hecho no ha supuesto problemas de memoria para el dispositivo.

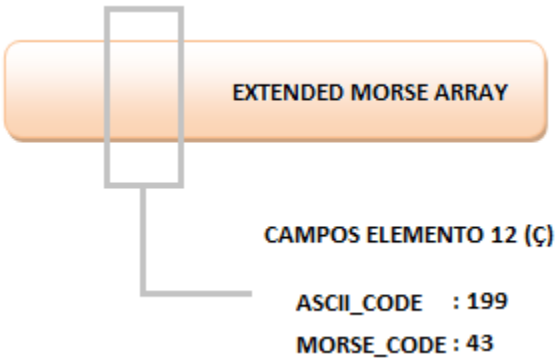


Figura : Estructura de un elemento del array de Morse extendido

Este método ha resultado en una ejecución muy rápida y ágil. El utilizar corrimientos de bits y correspondencias directas en array, la decodificación resulta muy rápida. En la codificación, aún resultando más costosa, no se han apreciado lags y la velocidad de ejecución resulta igualmente muy rápida.

- Conversión de código Morse a audio

En esta etapa, una vez codificados los caracteres de texto ASCII en código Morse, se utiliza el driver PWM para generar una señal que está activa y en pausa dependiendo de los tiempos estándar del código Morse y los datos de frecuencia, volumen y palabras por minuto deseados.

Para conseguir que la señal tenga la frecuencia deseada, basta con configurar los registros PWM con los tiempos resultantes de comparar esta frecuencia con la frecuencia de reloj PWM, que en nuestro caso es:

$$PWM\_CLK = \frac{CPU\_CLK}{8}$$

En el primer registro, MR0, se introduce el resultado de dividir la frecuencia del reloj PWM con la frecuencia a la que queremos que funcione. El segundo registro, responsable del tiempo de duración del ciclo, se le introduce el valor anterior multiplicado por el porcentaje de duty cicle, el cual disminuye con el porcentaje de volumen deseado en la siguiente relación:

$$duty\ cicle = porcentaje\ \ volumen\ \cdot 0.08$$

En la cual, el porcentaje de volumen se refiere a:

$$porcentaje\ \ volumen\ = \%volumen\ \cdot 0.01$$

A mayor volumen, mayor tiempo se encuentra activa la señal PWM en cada ciclo, resultando en un mayor volumen emitido por el transductor electromagnético. El transductor simplemente se conecta entre la salida 5 del PWM correspondiente a la patilla 46 del dispositivo y masa.

La señal PWM tiene un dc_offset por contener sólo ciclos positivos. Si se desea eliminar dicha DC basta con conectar entre la salida y el transductor un condensador electrolítico de desacoplo. En la práctica no ha hecho falta incluir dicho condensador.

- Módulo Wifly y CP2102:

El módulo empleado para la comunicación vía Wifi es el Wifly RN-171 de Rovign Networks.

Figura : Módulo Wifly empleado en el proyecto

La configuración utilizada en el Wifly incluye un comando para utilizar sus GPIOs 4 y 6 como test de la conexión:

- GPIO 4: Este pin se encuentra a nivel alto (3.3V) si la conexión con el punto de acceso Wifi es correcta.
- GPIO 6: Este pin se encuentra a nivel alto (3.3V) si la conexión TCP con la aplicación es correcta.

Otro punto importante que incluye la configuración que el sistema hace sobre este módulo, es la capacidad de auto conexión. Si la conexión TCP se pierde y vuelve, Wifly se conecta automáticamente.

Otra característica interesante que se ha utilizado es el RESET del módulo. Para hacerlo, basta con pasar el pin 3 del Wifly de estado alto a bajo en un tiempo mayor que 160 μs.

El módulo CP2102 es el encargado de conectar el puerto 3 UART del sistema empotrado con un puerto USB del equipo:

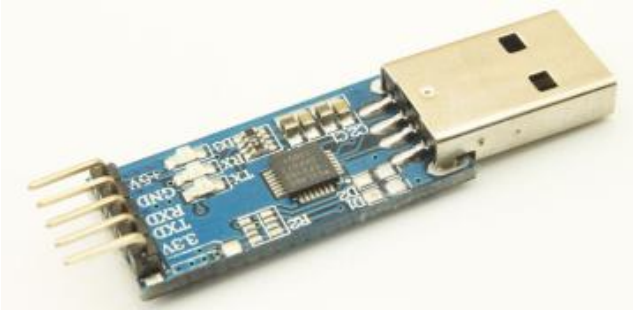


Figura : Módulo CP2102 utilizado en el proyecto

Dicho módulo no necesita configuración y tan sólo hay que conectarlo convenientemente a los pines del puerto 3 UART en el sistema empotrado. No necesita alimentación, la proporciona el puerto USB al que se conecta. Su uso en el proyecto viene indicado por la configuración inicial del sistema y los mensajes de estado de conexión que el sistema empotrado indica cuando existe un problema.

- Tipos de paquetes de datos:

La aplicación de usuario y el sistema empotrado se intercambian datos de distinta función. Además, el LPC1769 debe distinguir los mensajes del módulo Wifly, ya que el puerto UART utilizado es el mismo.

Con el fin de distinguir la información que transportan los datos, en cada paquete se incluye una cabecera distintiva de 16 bits seguida de los datos (payload) que se detalla a continuación:

- ***Texto enviado desde la aplicación de usuario***: La cabecera se compone de los caracteres "*!". Si el dispositivo recibe datos precedidos por esta cabecera, lo trata como cadena de caracteres a codificar en Morse.

Figura : Formato texto a codificar en Morse

- ***Configuración enviada desde la aplicación***: La cabecera contendrá los caracteres "!*". En este caso, además, el payload consiste en datos numéricos separados por el carácter "#" los cuales separa el dispositivo antes de aplicar la nueva configuración.

Figura : Formato de datos de configuración enviados desde aplicación usuario

- ***Configuración enviada desde el sistema empotrado***: La cabecera contendrá los caracteres "!#" seguida de los datos de configuración separados por el carácter "#".

Figura : Formato de datos de configuración enviados desde LPC1769

- ***Estado del sistema empotrado***: Este tipo de mensajes contiene la cabecera "¡#" seguida del estado del dispositivo, el cual consiste en una cadena de caracteres. Actualmente, sólo envía "OK" en el payload, indicando que la operación de codificación Morse ha terminado correctamente. La aplicación de usuario responde a este mensaje habilitando los controles de nuevo, indicando que el sistema se encuentra disponible.

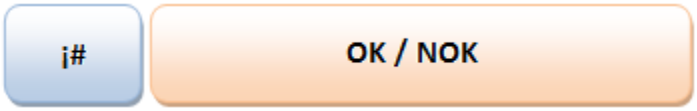


Figura : Formato de datos de estado del sistema empotrado

4.2 Detalles de la aplicación de usuario

a) Diagrama de flujo de la aplicación de usuario

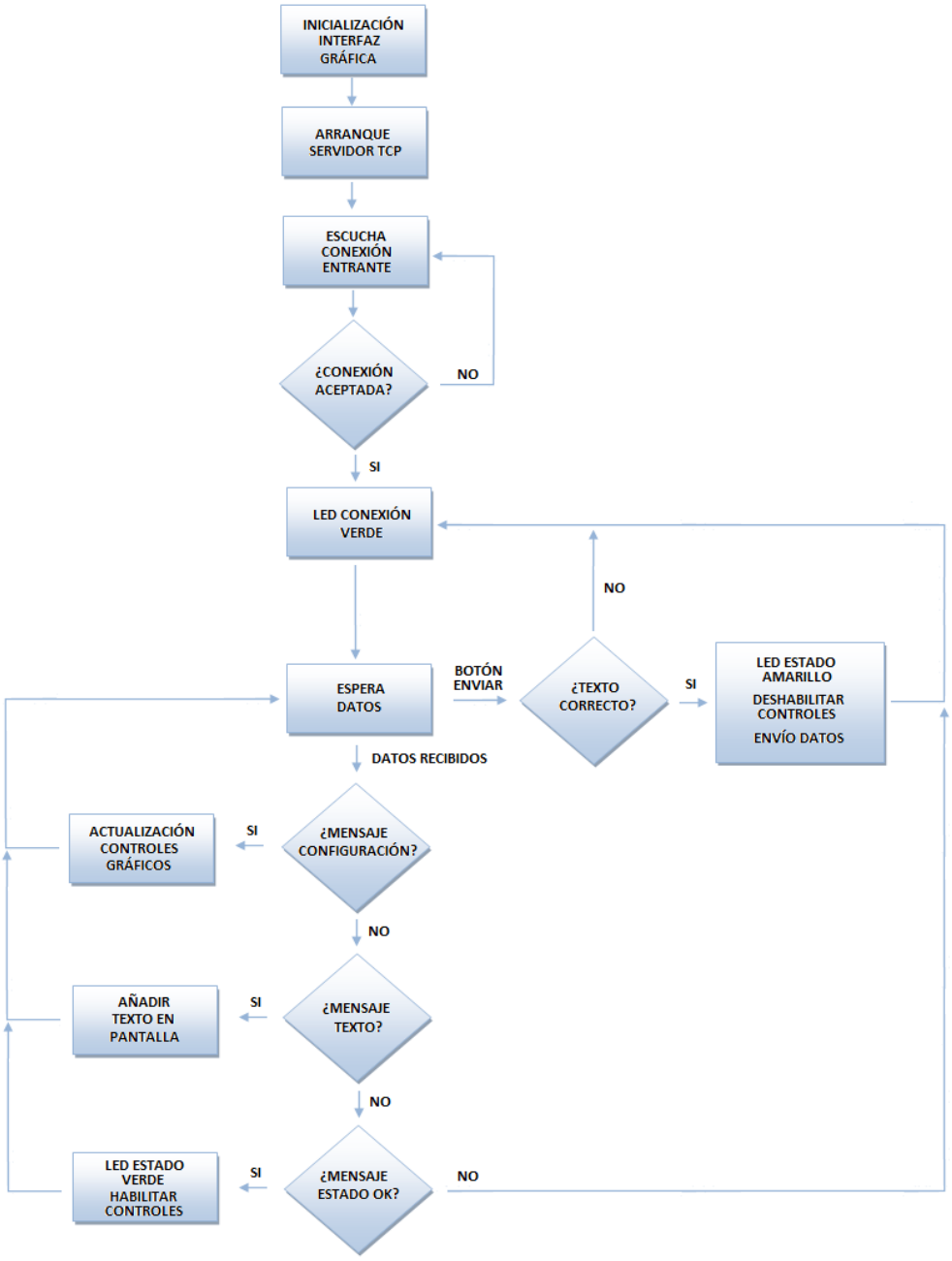


Figura : Diagrama de flujo interfaz usuario

Como se puede observar, la aplicación de usuario no es compleja y consiste, básicamente, en un servidor TCP que actualiza la interfaz gráfica dependiendo de los datos recibidos.

En lo que respecta tanto a la escucha de conexión entrante como a la espera de datos, se llevan a cabo mediante bloqueo de ejecución. Por lo tanto, mientras se produce la espera no consume recursos.

b) Interfaz gráfica y funcionamiento

Cuando la aplicación de usuario inicia, el servidor TCP se pone a la escucha de conexión entrante por el puerto 6001. Una vez conectado, el led de conexión "LPC Online" se pone en verde indicando que la conexión ha sido establecida correctamente. A su vez, el led "LPC Status" se pone en verde indicando que el sistema empotrado se encuentra disponible y en funcionamiento.

Los controles de tono (frecuencia), volumen y velocidad se pueden ajustar en cualquier momento menos cuando el sistema empotrado se encuentra ocupado codificando texto a sonido en código Morse. En este caso, los ajustes vuelven a estar habilitados cuando el dispositivo informa que se encuentra disponible de nuevo. Los ajustes de dichos controles no serán efectivos hasta que se pulse el botón "Set options" mediante el cual la aplicación envía los ajustes al sistema empotrado.

El envío de texto al dispositivo para codificar en código Morse audible por el transductor no se produce en caso que no exista texto en la caja "Text to encode".

Por otro lado, si la aplicación recibe texto decodificado del dispositivo lo mostrará inmediatamente en la caja "Decoded text".

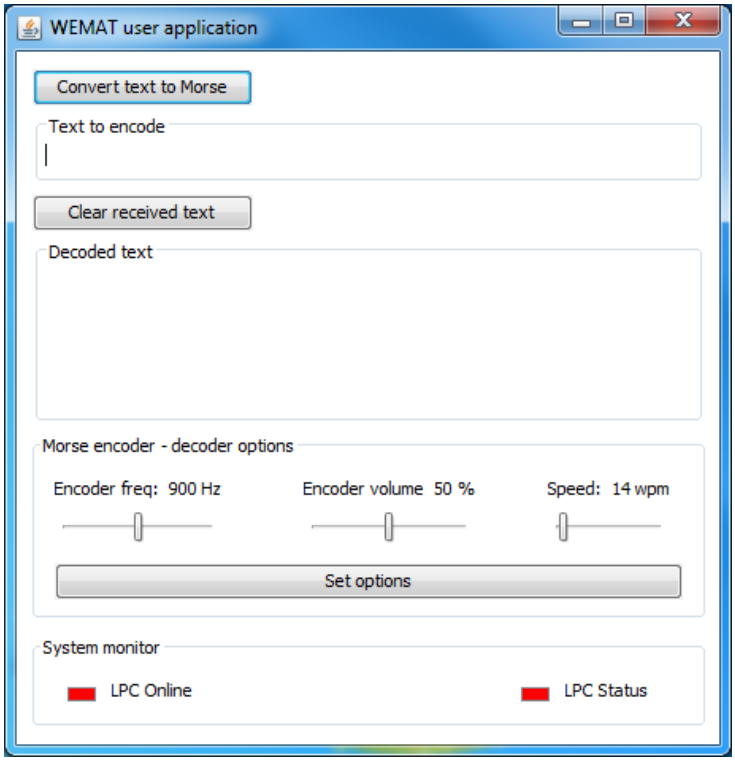


Figura : Detalle de la interfaz de usuario y los controles detallados

La aplicación ha sido compilada y probada en Windows 7 mediante la versión jdk 1.7.0_40. En teoría no debería dar problemas en otro sistema operativo que soporte Java.

Así mismo, y dados los pocos requerimientos que emplea la aplicación de usuario, el sistema queda abierto a su control mediante otro dispositivo cuya aplicación cumpla los formatos en los que trabaja el sistema empotrado que aquí se presenta, como puede ser un Smartphone.

4.3 Aplicación en el LPC 1769

a) Diagrama de flujo de la aplicación en el sistema empotrado

En el presente apartado vamos a ir recorriendo la aplicación del sistema empotrado por partes.

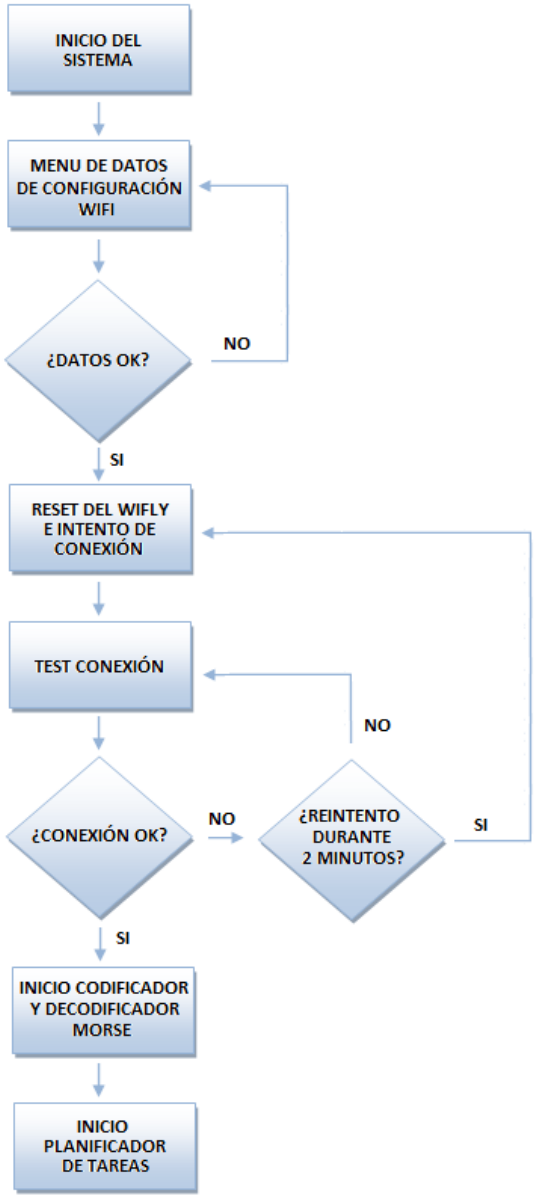


Figura : Diagrama de flujo del inicio de la aplicación en el LPC1769

Al inicio de la aplicación, el LPC1769 inicializa los puertos 0 y 3 del UART. Posteriormente, y utilizando el UART 3, pide los datos de conexión Wifi. Una vez introducidos, el sistema los muestra y pregunta si son correctos. Si lo son, hace un reset al módulo Wifly, lo configura e intenta conectarse y, si no lo consigue, realiza comprobación del estado de la conexión durante dos minutos. El módulo Wifly intenta la conexión por sí mismo pero si el sistema observa que no se ha producido pasados los dos minutos, hace reset al Wifly, lo configura de nuevo e intenta la conexión. Todos los accesos al UART se hacen controlados mediante semáforos (librería *sync.c*) y así evitar el acceso múltiple por parte de las tareas del sistema.

Si la conexión se ha establecido, el dispositivo configura el codificador y el decodificador Morse. En este punto, se realiza la configuración de la interrupción INT3 necesaria para recibir código Morse, timers del sistema y el canal 5 del PWM en el LPC1769. Además, configura las tareas y las colas que utilizan para sincronizarse.

El siguiente paso que realiza el sistema es el arranque del planificador de tareas de FreeRTOS, el cual inicia la aplicación propiamente dicha.

A continuación se describe la tarea decodificadora:

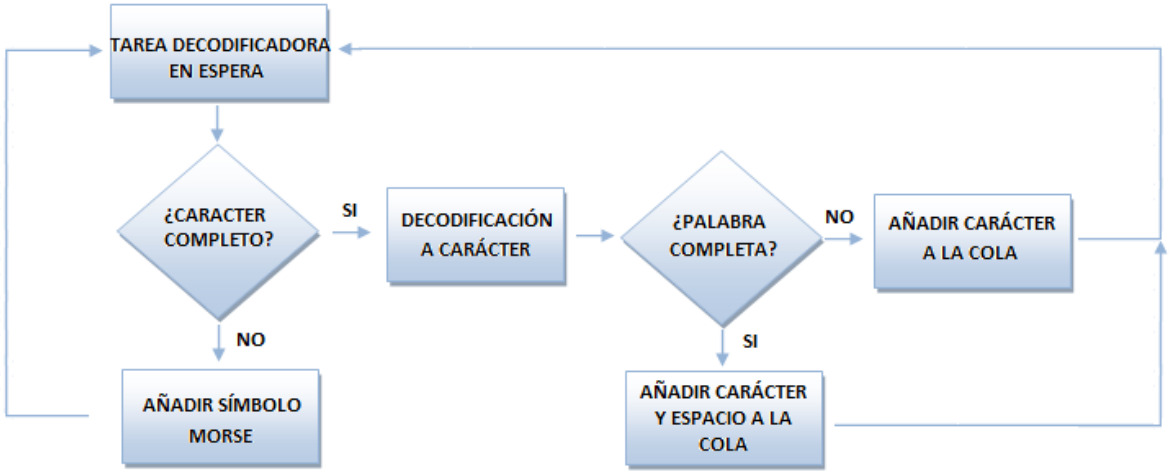


Figura : Diagrama de flujo de la tarea decodificadora

La tarea espera la llegada de símbolos Morse desde la interrupción EINT3.

En el momento que llegan, comprueba si existe un carácter completo y, de ser así, lo decodifica y lo manda a la cola de la tarea de envío de texto a la aplicación de usuario. Si, además, el carácter cierra una palabra, se envían a la cola el carácter un espacio en blanco.

En caso contrario, añade el símbolo recibido al buffer de bits Morse y vuelve a quedarse a la espera.

A continuación se detalla la tarea de envío de texto a la aplicación, la cual recoge los caracteres que deposita la tarea explicada anteriormente.

Figura : Diagrama de flujo de la tarea de envío de texto

Esta tarea manda los caracteres que recibe de la tarea decodificadora.

A continuación, se detalla la tarea receptora de texto mandado por la aplicación de usuario.

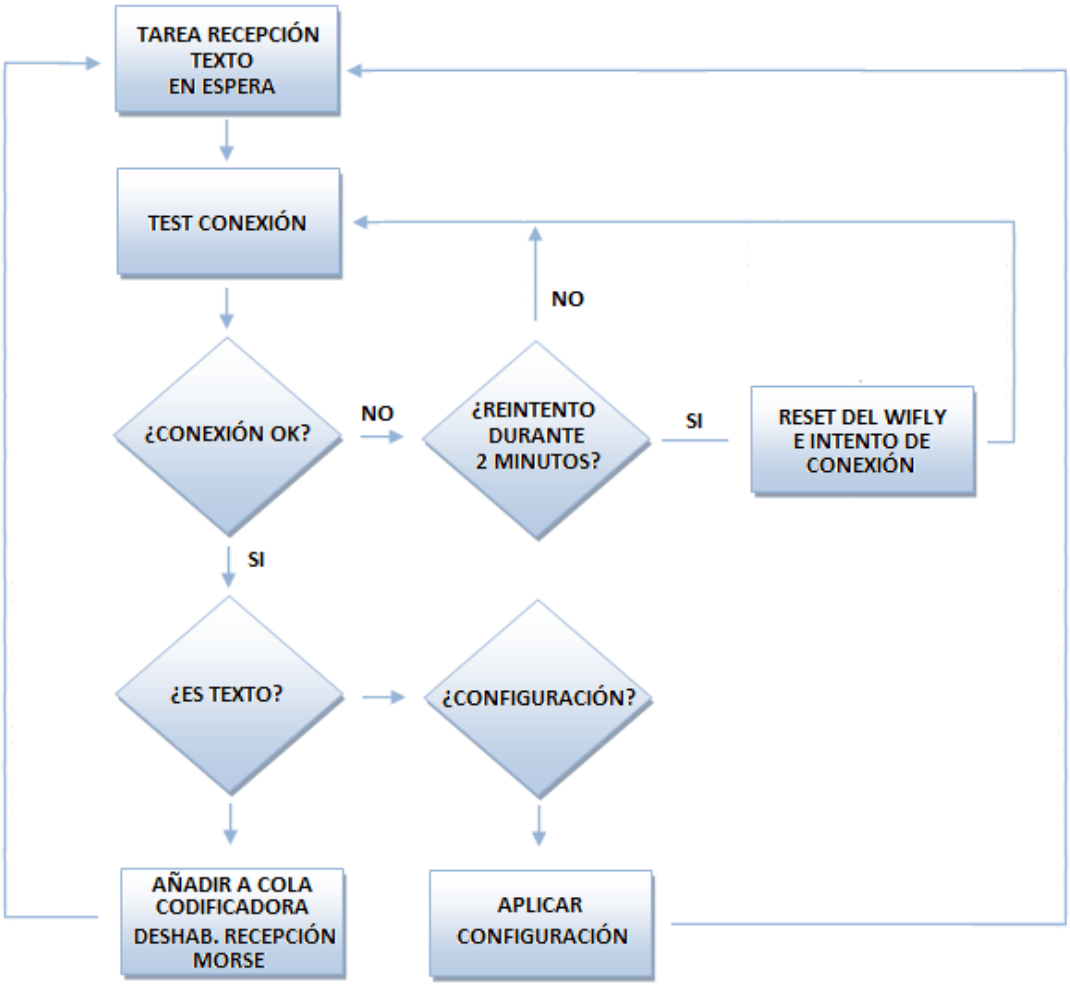


Figura : Diagrama de flujo de la tarea receptora de datos

La tarea que nos ocupa se encarga de recibir el texto enviado por la aplicación de usuario, cuyo destino es ser codificado en Morse y enviado al transductor para convertirlo en sonido.

Lo primero que realiza la tarea es una comprobación del estado de la conexión. Si no está correcta, el sistema intenta recuperarla tras esperar dos minutos. Si está correcta, comprueba la cabecera del paquete de datos:

- Si es texto, añade los caracteres a la cola de la tarea codificadora.
- Si son datos de configuración, los aplica.

Si no es ninguna de las opciones anteriores los datos pertenecen al módulo Wifly y la tarea los ignora. Normalmente, el módulo Wifly envía datos en formato texto cuando ha surgido un problema con la conexión que ya se habrá solucionado por la comprobación que la tarea realiza antes de tratar los datos.

Por último, se describe la tarea codificadora.

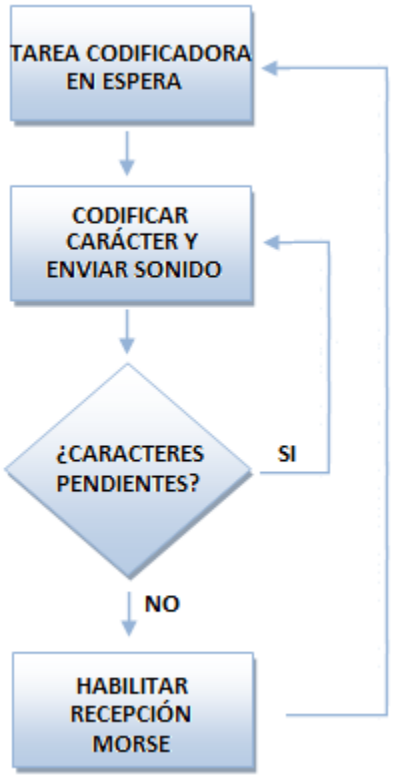


Figura : Diagrama de flujo de la tarea codificadora Morse

La tarea codificadora recibe los caracteres en formato texto (ASCII) de la tarea anterior (receptora de texto). Su misión es codificar los caracteres en símbolos Morse y, acto seguido, generar la señal PWM que permite su audición. La señal resultante se aplica al transductor electromagnético que vibra a su frecuencia y con intensidad correspondiente al ajuste del duty cycle realizado por el sistema empotrado.

5. Viabilidad técnica

El sistema desarrollado no es perfecto. Como tal tiene sus puntos fuertes pero también adolece de limitaciones y falta de desarrollo, ésta última por falta de tiempo.

Después de realizar las pertinentes pruebas, el sistema es perfectamente viable en los siguientes puntos:

- **Portabilidad**. El sistema desarrollado es capaz de comunicarse mediante Wifi, tecnología de conexión sin hilos ampliamente desarrollada y disponible casi en cualquier sitio. Esto le aporta flexibilidad, portabilidad y comodidad al no tener que estar conectado continuamente al equipo que hace de interfaz al usuario.
- **Uso de estándar ASCII**. Si bien es cierto que la codificación Unicode ofrece soporte para cualquier idioma y símbolos, también lo es que no todas las aplicaciones lo soportan. Además, su extensión es truncada al tratarse de un dispositivo que traduce código Morse, ya que se trata de una codificación antigua que no soporta entera, ni siquiera, la codificación ASCII. Por esta razón, tratar de desarrollar una aplicación que haga de interfaz con el sistema empotrado se hace extremadamente sencilla y soportada por cualquier lenguaje de programación.
- **Alto rechazo al ruido**. El sistema desarrollado realiza discriminación de sonidos que no corresponden con código Morse tanto en hardware como en software. De esta forma, se mantiene inalterable ante ruidos como clicks, pops y de fondo e incluso rechaza el sonido de una conversación normal cerca del micrófono, especialmente las voces agudas al estar más elevadas en frecuencia.
- **Bajo consumo**. Tanto el circuito de audio como el LPC1769 tienen un consumo bajo. Este punto proporciona buena autonomía al sistema.
- **Alta velocidad de decodificación de mensajes Morse**. Aunque se ha creado para este fin, es asombroso observar que es capaz de decodificar mensajes a velocidades muy altas que los hacen ininteligibles para una persona, incluso bien entrenada. El corazón del sistema, el LPC1769, contiene una CPU muy potente que se mueve cómodamente a velocidades de 100MHz.

Como ya se ha indicado, el sistema no es perfecto y adolece de las siguientes limitaciones [21]:

- **Ambiente ruidoso**. El código Morse se puede utilizar en distintos tipos de señales. Cuando se utiliza audio, recae en bandas en las cuales entran sonidos casi de cualquier tipo. Desde motores, maquinaria industrial, la voz humana, etc. Este hecho ocasiona que el sistema se encuentre abierto a interferencias sonoras. Las pruebas se han realizado con ruido ambiente controlado. Si existe alto ruido ambiental su funcionamiento no está asegurado.
- **Es dependiente de la conexión sin hilos**. Aun siendo una ventaja, la conexión sin hilos también adolece de interferencias y errores. El sistema desarrollado depende totalmente de este sistema. Si cae o nos encontramos en un punto con múltiples colisiones por no existir canales libre, el sistema no puede ser utilizado.
- **Necesidad de ajuste fino de la amplitud de la señal de audio.** Si la salida del circuito no se ajusta adecuadamente, el sistema realiza mal la decodificación. En las pruebas se ha ajustado tan sólo una vez y ha funcionado correctamente para distintos niveles de audio. Pero un usuario podría encontrar dificultad en el ajuste.

[21] En el apartado conclusiones se intenta dar solución a los problemas aquí detallados.

6. Valoración económica

A continuación se detalla el coste del proyecto. La instalación, así como el mantenimiento no parecen viables en el producto desarrollado.

Producto	Unidades	P. unitario	Importe
LPCXpresso 1769	1	30€	30€
Wifly RN-171	1	38€	38€
CP2102	1	5€	5€
Cableado	14	2,6€	36,4€
Protoboard 82x55 mm, 400 contactos	1	4,8€	4,8€
Resistencias 1/4W	7	0,16€	1,12€
Condensadores 4,7 uF	3	0,9€	2,7€
Potenciómetros 5K	3	0,34€	1,02€
TL072 op amp	1	0,2€	0,2€
LM741 op amp	1	0,4€	0,4€
Transductor electromagnético 1KHz	1	0,36€	0,36€
Micrófono electret	1	0,43€	0,43€
Horas desarrollo proyecto	110	12€	1320€
		TOTAL	**1440,43€**

7. Conclusiones

En este apartado se revisan los objetivos cumplidos como los que no, así como propuesta de mejoras y autoevaluación objetiva.

7.1 Conclusiones

En general, el proyecto ha sido una experiencia satisfactoria y enriquecedora, aunque no han faltado los momentos de stress y falta de tiempo. El aprendizaje ha sido elevado y se ha añadido parte de investigación, que lo ha hecho todavía más interesante. El mundo de los sistemas empotrados es fascinante y me ha dado la sensación de estar haciendo *pura ingeniería*, toca muchos campos distintos de la carrera.

En cuanto a la programación del LPC1769, en un principio parecía que no sabía lenguaje C. El estar constantemente tan cerca de la capa hardware me ha permitido entender muchos conceptos que antes eran obtusos, aunque no sin un gran esfuerzo.

En la parte hardware, me ha permitido recordar conceptos un tanto perdidos en la memoria y, además, reforzarlo. Me ha permitido descubrir un mundo de nuevas y fascinantes posibilidades que no voy a tardar en aprovechar en el tiempo libre. En el mundo laboral ya me ha ayudado.

En cuanto a los objetivos del proyecto se han cumplido todos, menos algunas mejoras que han quedado pendientes por falta de tiempo:

- ✓ Ofrecer interfaz de configuración de conexión, mediante USB.
- ✓ Dotar al sistema de conexión inalámbrica (Wifi).
- ✓ Implementar entrada y acondicionamiento de audio.
- ✓ Traducir código Morse (audio) a texto (ASCII).
- ✓ Generar señales de audio en código Morse y controlar su tono y volumen.
- ✓ Traducir texto (ASCII) a código Morse (audio).
- ✓ Ofrecer interfaz de usuario (Java).
- ✓ Implementar control de velocidad envío/recepción en palabras por minuto.
- ✓ Control de errores en la conexión Wifi. Este objetivo, aunque conseguido, necesita mejorar.

Puntos extra no realizados:

- Implementación del driver watchdog, para el control sobre eventuales caídas del software o el mismo hardware.
- Me hubiera gustado implementar mayor número de capas, como un driver GPIO.

7.2 Propuesta de mejoras

En lo referente al nivel de ruido ambiental elevado, se puede implementar la conexión directa de la entrada de audio al equipo que recibe el código Morse mediante un jack de audio. Normalmente, los equipos utilizados para la recepción Morse son radiotransceptores, los cuales incorporan una salida de audio además del altavoz. De esta forma, el sistema se hace inmune a cualquier sonido a cualquier nivel, se trate de ruido o no. La desventaja es tener que depender de cableado, cuando el dispositivo ha sido pensado para no depender de él.

La dependencia de una conexión Wifi puede solucionarse utilizando un puerto USB con el CP2102, modificando además la aplicación de sistema para esta nueva situación. De nuevo, no es el objetivo del proyecto tener que depender de cableado para la comunicación.

En cuanto a la necesidad de ajuste fino de la amplitud de la salida del circuito de audio, podría solucionarse con un control automático de ganancia. Mediante un feedback de la salida y mediante el gobierno de un transistor, se puede conseguir un ajuste automático.

En la aplicación Java ha sido imposible detectar el abandono repentino de la conexión por parte del sistema empotrado. El socket sigue estando conectado, aunque el dispositivo ya no lo esté por ejemplo, por una caída del sistema. Se ha probado, incluso, en comprobar la conexión fantasma haciendo una escritura por el stream del socket, pero tampoco dio resultado. Este punto necesita más investigación.

Utilizar detección de la señal de audio por medio de software y ADC. Esta opción aportaría la posibilidad de realizar procesamiento en frecuencia de la señal muestreada y, por lo tanto, mayor control. La parte negativa sería la mayor complejidad del software y, además, el sistema empotrado perdería velocidad de procesamiento del audio entrante resultando en una respuesta más lenta y poco satisfactoria. Tampoco mejoraría el problema del ruido ambiental excesivo, ya que las frecuencias utilizadas son muy comunes para este tipo de ruidos.

Otro punto extra trata de la intensificación de los mensajes entre sistema empotrado y aplicación de usuario. Si la aplicación Java se cierra y se vuelve a abrir, los controles puede que no reflejen el valor actual de los mismos en el sistema empotrado. Para sincronizar, hay que volver a enviar la configuración al sistema empotrado.

Por último, existe la posibilidad que el sistema se ajuste automáticamente a la velocidad del código Morse recibido. Por falta de tiempo no se ha podido indagar en esta opción, pero su funcionamiento radica en el tiempo que muestran las interrupciones de detección. El sistema necesitaría un tiempo de entrenamiento, tras el cual simplemente calcularía la velocidad con el tiempo recibido más pequeño, es decir, el *punto* (dot). Esta mejora, aunque no comprobada, es perfectamente viable.

7.3 Autoevaluación

He observado que, a veces, daba prioridad a puntos del proyecto no imprescindibles. Este hecho ha restado tiempo que podría haber mejorado los resultados, sin duda. Aspectos de la organización me han fallado, tengo que mejorar en este punto.

Por otro lado, el resultado del proyecto es muy satisfactorio y, ciertamente, funciona mejor que lo que se podía pensar en un principio. Me ha ampliado muchísimo la visión de los sistemas empotrados, así como el trabajo a bajo nivel. Me ha demostrado que la limitación no existe, sólo la que uno mismo se impone.

8. Glosario

- ***wpm***: Siglas en Inglés de palabras por minuto. En el proyecto, se ha tomado como referencia la palabra estándar PARIS.

- ***buzzer***: Pequeño transductor que convierte las señales eléctricas en sonido. Existen dos tipos, los electromagnéticos que actúan como un pequeño altavoz y los piezoeléctricos que, al aplicarles tensión, resuenan a una frecuencia predeterminada. Los electromagnéticos reproducen la frecuencia de la señal que los alimenta dentro de un rango de frecuencias.

- ***PWM***: Señal cuadrada cuyo ancho de impulso es ajustable.

- ***Switch bouncing***: Rebotes que produce un pulsador al ser accionado. Dichos rebotes hacen reaccionar la CPU como si fueran pulsaciones voluntarias.

- ***Duty cycle***: Fracción de tiempo de un ciclo de señal PWM en el cual se encuentra en estado alto.

- ***FreeRTOS***: Sistema operativo libre en tiempo real utilizado en el proyecto. Es el encargado de la planificación de CPU, así como de la sincronización multitarea.

- ***ADC***: Siglas en Inglés de convertidor analógico - digital, encargado de muestrear la señal a su entrada en valores discretos aptos para su uso en sistemas digitales.

- ***PARIS***: Palabra utilizada como referencia de medidas de velocidad en código Morse. Contiene 50 símbolos, cifra que facilita el cálculo del tiempo mínimo utilizado, el símbolo *punto*.

9. Bibliografía

- Apuntes de la asignatura de Fundamentos Tecnológicos II.
- Apuntes de la asignatura Protocolos y Aplicaciones de Internet.
- Wiki del área de TFC Sistemas Empotrados.

- Nibbles and Bits: Página donde se detalla interesante información para decodificar código Morse con Arduino.

- FreeRTOS: Página oficial del sistema operativo en tiempo real utilizado en el proyecto.

- NXP: Página oficial del fabricante del dispositivo LPC1769, donde se encuentra material de descarga sobre el microcontrolador y la placa.

- Active Filter Design: Documento utilizado para la realización del filtro activo pasa banda, de Texas Instruments.

- Foro NXP: Foro de consultas sobre las placas y microcontroladores del fabricante.

- Code Red: Desarrollador del IDE LPCXpresso. Incluye interesantes documentos sobre el IDE utilizado en el proyecto.

- Módulo Wifly: Página del fabricante del módulo Wifly (ahora *MicroChip*) utilizado para crear conexiones a través de WiFi.

- Code Red_Getting access to target chip, consulta realizada para solucionar un problema puntual con LPC1769.

- Datasheet TL071, TL072 de Texas Instruments.

- Datasheet LM741 de National Semiconductors.

10. Anexos

En este último capítulo se incluyen los manuales de usuario para configurar el terminal y la configuración inicial del sistema empotrado, la interfaz de usuario Java y la compilación del proyecto en el IDE LPCXpresso.

10.1 Configuración del terminal utilizado para la configuración del sistema

El terminal utilizado en el proyecto y que se describe a continuación es el recomendado por el fabricante del módulo Wifly, Rovign Networks (ahora *MicroChip*). Se trata del programa TeraTerm 4.8 y comenzaremos por su configuración:

- El primer paso es instalar el programa. En el proyecto se ha utilizado la versión portable incluida en un fichero ZIP. Una vez instalado/descomprimido se procede a la configuración del puerto serie. Primero, conecte el módulo CP2102[22] a un puerto USB disponible en el equipo. Ahora, en el menú, vaya a *Setup --> Serial Port*. Los valores deben coincidir con los siguientes:

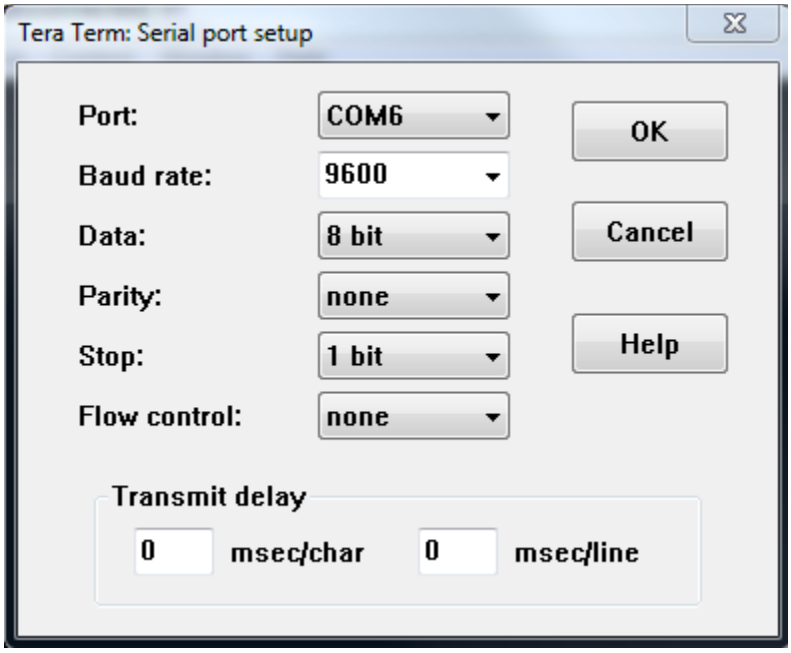


Figura : Detalle de configuración del puerto serie en TeraTerm

El valor de la casilla Port debe adecuarse al puerto USB al cual se conecta el módulo CP2102 en el equipo. Una vez configurados los valores, pulse OK.

- El segundo paso es configurar el terminal. Para ello, en el menú, vaya a Setup --> Terminal y modifique los valores para que aparezcan como la figura siguiente:

[22] Los drivers del CP2102 deben estar instalados en el equipo. Disponibles en el sitio de Silicon Labs.

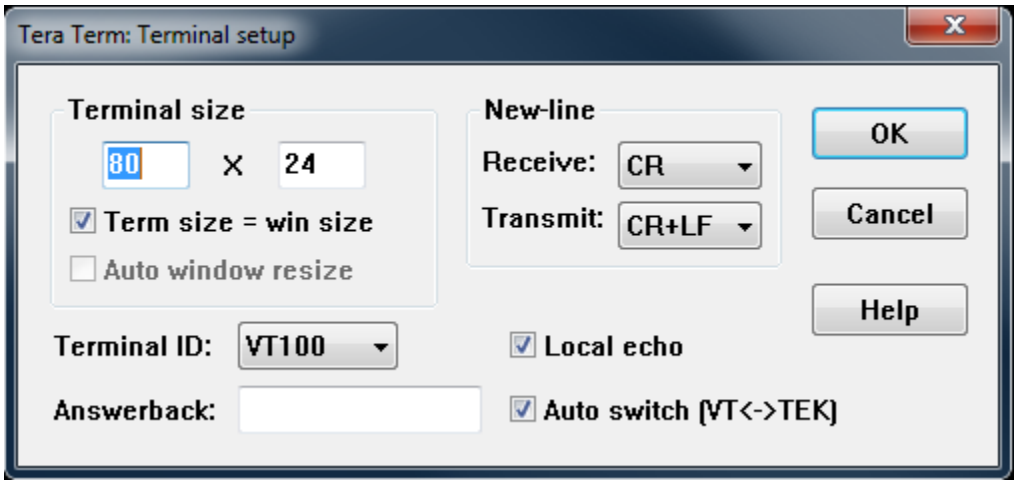


Figura : Detalle de configuración del terminal

Los controles que deben coincidir con la figura son el valor de receive, transmit y local echo. Pulse OK cuando haya terminado.

- El tercer paso es realizar la conexión. No importa si el sistema empotrado no está ejecutándose todavía, el terminal se conecta con el módulo CP2102. Para realizar la conexión, vaya al menú y seleccione File --> New Connection. En la ventana emergente, seleccione serial y busque en el desplegable el módulo CP2102:

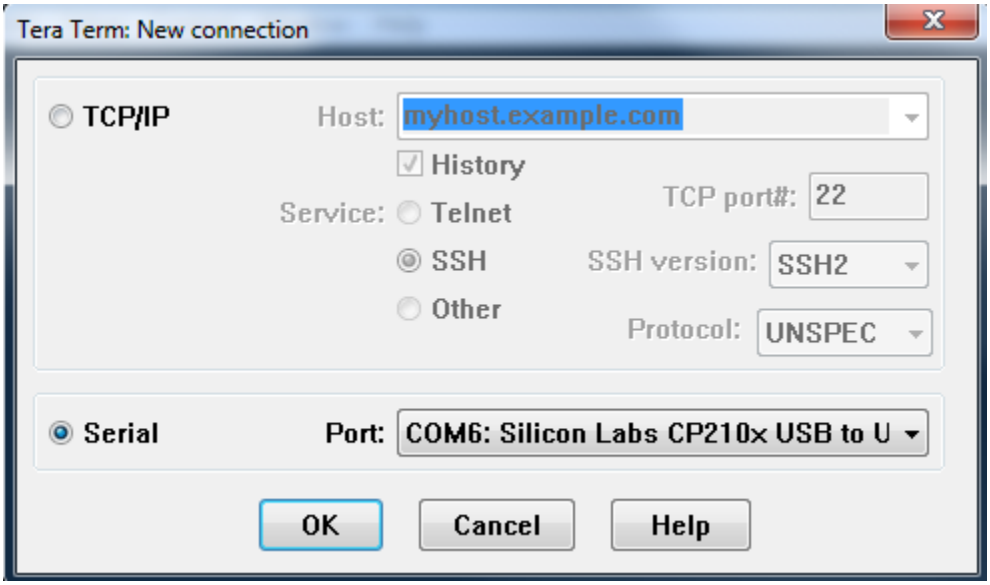


Figura : Realización de la conexión con el CP2102

Sólo queda pulsar OK y tendremos realizada la conexión.

10.2 Configuración inicial del sistema

- Con el Terminal conectado al CP2102, encienda el sistema empotrado. La siguiente figura muestra lo que debe aparecer en el terminal:

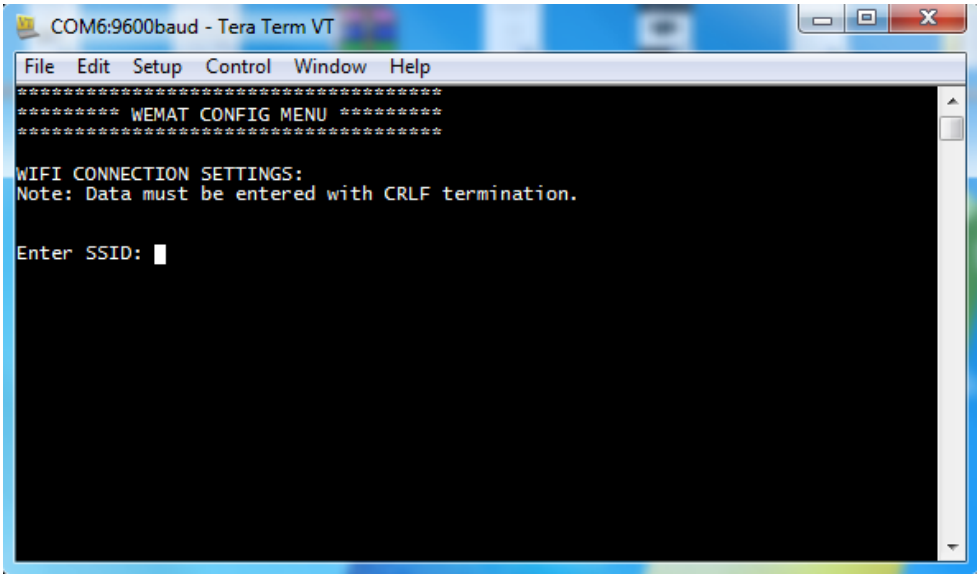


Figura : Detalle de inicio del sistema

- La configuración es muy sencilla, simplemente siga los pasos que le indicará el sistema. Al final, el sistema empotrado muestra la información introducida para su comprobación. Si todo está OK, teclee "y" y el sistema iniciará:

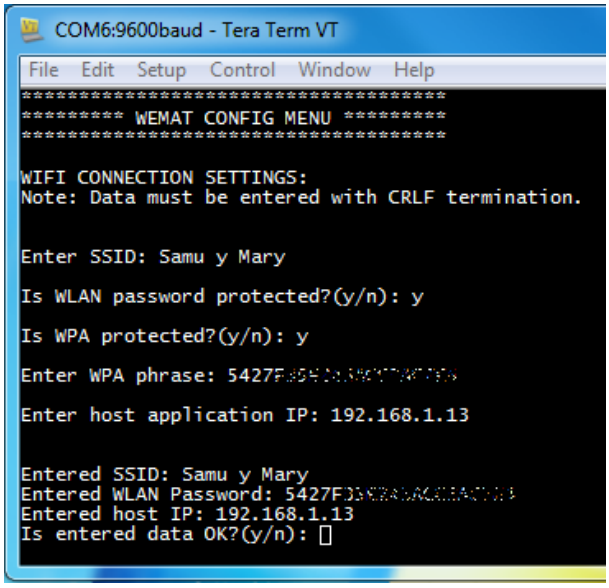


Figura : Configuración de la conexión en el sistema empotrado

Nota: La clave de seguridad, si es necesario introducirla, se puede copiar y pegar en la pantalla del terminal con el botón derecho del ratón.

10.3 Manual de la interfaz de usuario

La interfaz de usuario es muy sencilla y no requiere instalación. En Windows, ejecute el archivo MorseTranslator.jar y la aplicación saldrá en pantalla:

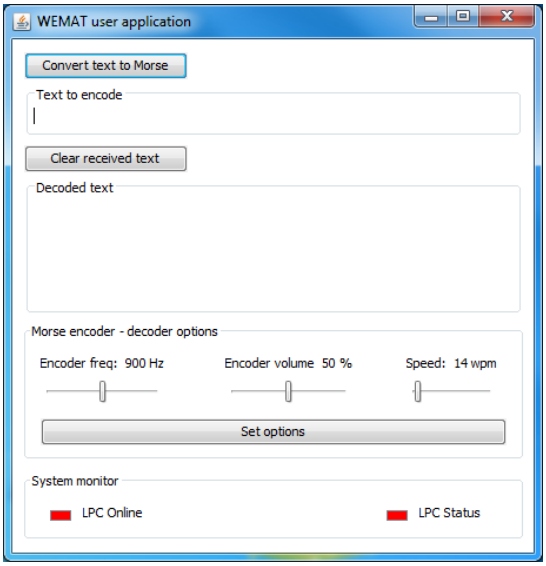


Figura : Inicio de la aplicación de usuario

- La aplicación Java conectará automáticamente con el sistema empotrado cuando éste intente la conexión. Los leds de estado del sistema cambiarán a verde:

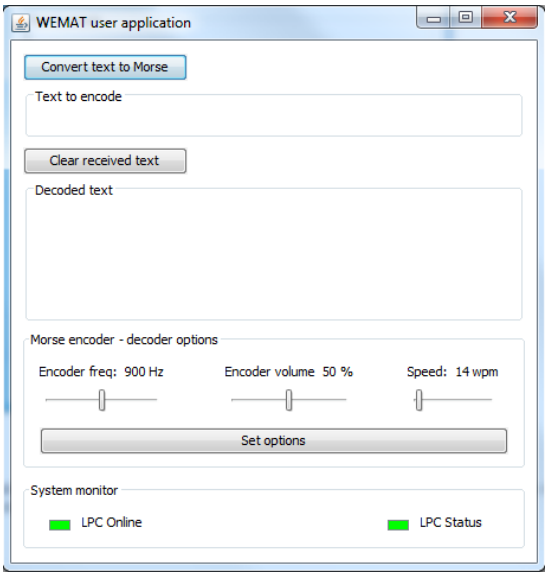


Figura : Detalle de la aplicación de usuario conectada con el sistema empotrado

- A partir de este punto, ya puede enviar texto a codificar en código Morse y recibir la decodificación del sonido Morse que capte del dispositivo. Para enviar el texto, tan sólo debe escribirlo en la caja titulada *Text to encode*, en la parte superior de la interfaz, y pulsar el botón *Convert text to Morse*. La aplicación esperará la finalización de codificación en el sistema empotrado (led LPC Status en amarillo):

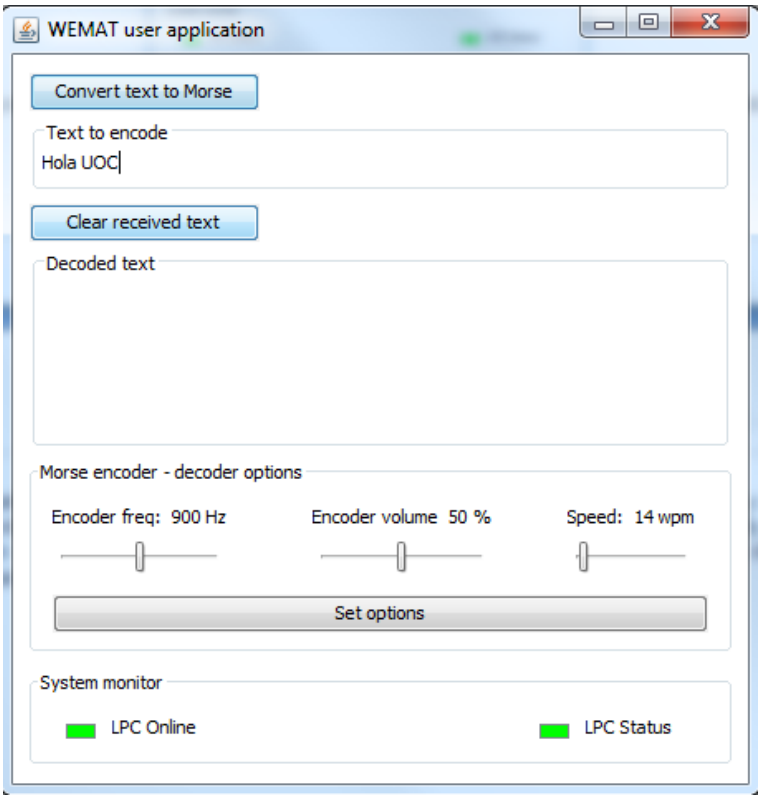


Figura : Detalle de envío de texto al sistema empotrado

Nota: La aplicación es *no case sensitive*, es decir, no se preocupe si el texto incluye mayúsculas y/o minúsculas.

- Para configurar los controles de *tono*, *volumen* y *velocidad wpm* simplemente deslice los controles al valor deseado y pulse el botón *Set options*.

- Por último, la recepción de texto en la caja *Decoded text* comienza tan pronto como el sistema empotrado capte y decodifique código Morse. Para borrar dicha caja de texto, pulse el botón *Clear received text*.

Nota: Si la aplicación Java se cierra y se vuelve a abrir, los controles puede que no reflejen el valor actual de los mismos en el sistema empotrado. Para sincronizar, simplemente ajuste los controles y pulse *Set options*. Ahora el sistema empotrado se encontrará configurado con los valores que muestra la aplicación.

10.4 Importación del Workspace y compilación en LPCXpresso

Para importar el *Workspace* del proyecto, vaya al menú de *LPCXpresso* y seleccione:

File --> Import

y en la pestaña General, seleccione *Archive* y pulse *Next*:

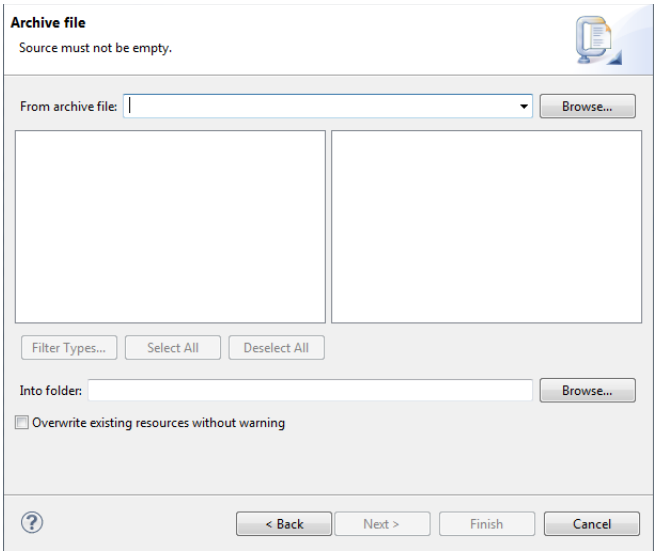


Figura : Detalle de importación de Workspace en LPCXpresso IDE

Tan sólo queda pulsar *Browse* en la caja *From archive file* y el IDE importará el proyecto pulsando el botón *Finish*.

Una vez en el proyecto, seleccione en el menú:

Project --> Build All

LPCXpresso mostrará una barra de progreso que indica el estado de la compilación.