Trabajo final de Grado

Grado de Ingeniería en Tecnologías Industriales

Diseño y construcción de un detector de metales basado en un microcontrolador

MEMORIA

Autor: Carlos Martínez Cantón

Director: Rosa Rodríguez

Convocatoria: Mayo 2016

Escuela Técnica Superior
de Ingeniería Industrial de Barcelona

Resumen

La detección de metales tiene una gran variedad de aplicaciones en la industria actual, en materia de seguridad y también en el ocio. Pero, en general, su coste elevado no está al alcance de todo el público. En este proyecto se ha diseñado y construido un detector de metales utilizando componentes electrónicos económicos y con buenas prestaciones. El detector se ha basado en un microcontrolador de la familia PIC del fabricante Microchip, el PIC16F690, y la detección del campo magnético se ha realizado mediante el sensor HMC5883L. Además, se ha creado una aplicación móvil para teléfonos con sistema operativo Android con la que establecer comunicación con el dispositivo. El usuario se conecta al detector de metales mediante tecnología bluetooth y se controla el encendido, el apagado y se monitoriza su estado.

El código de programación del microcontrolador se ha realizado en lenguaje de programación c mediante el editor MPLAB X IDE y el software para la aplicación móvil se ha implementado a través de la plataforma online de diseño de aplicaciones móviles App Inventor 2. Finalmente se ha puesto a prueba el funcionamiento del detector de metales con una serie de experimentos para medir su rango de detección. La fiabilidad del aparato se ha visto determinada por la sensibilidad del sensor utilizado.

Sumario

RESUMEN 1

SUMARIO 3

1. GLOSARIO 5

2. PREFACIO 8

2.2. Origen del proyecto 8

2.3. Motivación 8

2.4. Requerimientos previos 8

3. INTRODUCCIÓN 10

3.2. Objetivos generales 10

3.3. Objetivos específicos 10

3.4. Alcance del proyecto 10

4. EL DETECTOR DE METALES 12

4.1. Tipos de detector de metales 12

5. INTRODUCCIÓN AL DISEÑO DE APLICACIONES MÓVILES 15

5.1. Aplicación móvil para el control del detector de metales 21

6. MÓDULO BLUETOOTH HC-05 31

7. EL MICROCONTROLADOR PIC16F690 32

7.1. Módulo EUSART del PIC16F690 36

7.1.1. Modo asíncrono de la EUSART 37

7.1.2. Velocidad de transmisión de datos 40

7.1.3. Recepción de datos 42

7.1.4. Transmisión de datos 49

8. EL SENSOR DE CAMPO MAGNÉTICO HMC5883L 55

8.1. Protocolo de comunicación I^2C 58

9. FIABILIDAD DEL DETECTOR DE METALES 71

9.1. Rango de medición 71

10. EL REGULADOR DE TENSIÓN 77

11. PRESUPUESTO _____ **78**

12. IMPACTO AMBIENTAL _____ **80**

CONCLUSIONES _____ **81**

BIBLIOGRAFÍA _____ **82**

1. Glosario

a	Radio de una bobina eléctrica
EIA	Evaluación de Impacto Ambiental
AND	Puerta lógica tipo Y
ANSEL	Analog Select Register
ANSELH	Analog Select High Register
BAUDCTL	Baud Rate Control Register
BOREN	Brown-out Reset Enable bit
BRG	Baud Rate Generator Register
Bus	La función de un bus es permitir la conexión lógica entre distintos subsistemas de un sistema digital, enviando datos entre distintos circuitos
CREN	Continuous Receive Enable bit
DIA	Declaración de impacto ambiental
EUSART	Enhanced Universal Synchronous Asynchronous Receiver Transmitter
FIFO	First In First Out
FOSC	Frequency Oscillator bit
GIE	Global Interrupt Enable bit
I	Intensidad de corriente
INTCON	Interrupt Control Register
L	Longitud de una bobina eléctrica
LED	Light Emitting Diode
LSb	Least Significant bit

Magnetoresistencia	Propiedad que tienen ciertos materiales de variar su resistencia eléctrica cuando son sometidos a un campo magnético
MSb	Most Significant bit
N	Número de espiras de una bobina eléctrica
NRZ	Non return to zero
OERR	Overrun Error bit
PEIE	Peripheral Interrupt Enable bit
PIC	Peripheral Interface Controller
PIR1	Peripheral Interrupt Request Register
PORTC	Port C Register
Protoboard	Placa de prototipos de circuitos electrónicos
PWRTE	Power-up Timer Enable bit
QR	Quick Response Code
R	Resistencia óhmica
RCFI	Receive Flag Interruption bit
RCIE	EUSART Receive Interrupt Enable bit
RCREG	Eusart Receive Data Register
RCSTA	Receive Status and Control
RSR	Receive Shift Register
SCL	Serial Clock
SDA	Serial Data
SPBRG	Serial Port Baud Rate Generator Register
SPEN	Serial Port Enable bit

SYNC	Synchronous Enable bit
TRISC	Port C Tri-State Register
TRMT	Transmit Shift Register Status bit
TXEN	Transmit Enable bit
TXIE	EUSART Transmit Interrupt Enable bit
TXIF	Transmit Interrupt Flag
TXREG	EUSART Transmit Data Register
TXSTA	Transmit Status and Control
USB	Universal Serial Bus
V	Voltaje
V_{DD}	Tensión de alimentación positiva
V_{SS}	Tensión nula o tierra
μ_0	Permeabilidad magnética del vacío

2. Prefacio

2.2. Origen del proyecto

Cada vez aparecen más situaciones en las que se debe incluir un protocolo de seguridad que proteja al ciudadano, ya sea en transporte, en procesos industriales o en el ocio. Este hecho está bien asumido pero el coste que implica la tecnología necesaria es en ocasiones un gran problema para el usuario.

Es el caso de la detección de metales, que hoy en día tiene un gran abanico de aplicaciones en materia de seguridad. En la industria alimentaria se requiere su uso para la detección de elementos que pueden ser nocivos para el consumidor y en determinados medios de transporte se utilizan para la seguridad del propio pasajero y del resto. No hay más que ir a cualquier aeropuerto, donde para entrar a la sala de embarque se debe pasar antes por un control de seguridad en el que se detectan metales con una puerta para tal función o con un dispositivo detector manual. También está extendido su uso para la detección de minas explosivas, situación en la que la precisión y fiabilidad del aparato deben estar bien garantizadas, lo cual equivale a un coste mayor del dispositivo. Así pues, se ha querido investigar en este tema e intentar abaratar el diseño y construcción de uno de estos aparatos con unas prestaciones similares a las que puede ofrecer un detector de metales del mercado actual.

2.3. Motivación

El elevado coste, en general, que supone un aparato detector de metales y la gran variedad de componentes electrónicos existentes en el mercado con los que se podría implementar uno propio, manifiesta la necesidad de diseñar y construir uno más económico y con muy pocos elementos. Además, existe la motivación de poner en práctica todos los conocimientos adquiridos en el Grado, especialmente en el campo de la electrónica, a parte de adquirir muchos nuevos.

2.4. Requerimientos previos

Para diseñar y construir el detector de metales se emplearán los componentes electrónicos que han sido proporcionados por el departamento de electrónica de la escuela: el microcontrolador

PIC16F690, el sensor magnético HMC5883L y el módulo bluetooth HC-05. Todos ellos de coste muy económico y con suficientes prestaciones para cumplir con los objetivos del proyecto.

3. Introducción

Un detector de metales es un aparato capaz de detectar la presencia de elementos metálicos y realizar una señal audible o visual, hecho que tiene gran aplicación para la seguridad de eventos, edificios, transporte público, procesos industriales, entre otros muchos.

3.2. Objetivos generales

Se quiere diseñar y construir un detector de metales basado en un microcontrolador de la familia PIC del fabricante Microchip. La detección del campo magnético se realizará en base a un sensor de tipo MPU6050 y el control de este aparato detector se hará a través de un dispositivo móvil con sistema operativo Android y tecnología bluetooth.

3.3. Objetivos específicos

- Diseñar y construir un dispositivo detector de metales mediante el uso del microcontrolador PIC16F690, el sensor de campo magnético HMC5883L y el módulo bluetooth HC-05.
- Crear una aplicación móvil para sistemas Android con el fin de controlar y monitorizar el estado del aparato detector de metales.
- Fácil manejo de la aplicación móvil.
- Fiabilidad en la detección de metales.

3.4. Alcance del proyecto

El diseño y construcción de este detector de metales se divide en dos partes. Primero se creará la aplicación móvil mediante la plataforma online y gratuita App Inventor 2. Ésta se hará intuitiva para facilitar el control del detector. Seguidamente se programará el microcontrolador PIC16F690 para que procese las señales procedentes de esta aplicación y del sensor de campo magnético. El código de programación del microprocesador se implementará en lenguaje de programación c mediante el editor MPLAB X IDE y se compilará con el compilador XC8.

Las conexiones entre los distintos componentes electrónicos que conforman el detector de metales se pondrán a prueba en una placa de montaje de prototipos de circuitos electrónicos protoboard. Finalmente, una vez efectuado todas las pruebas necesarias y comprobado el correcto funcionamiento del aparato detector, se procederá con la conexión de los elementos sobre una placa de soldadura.

4. El detector de metales

Un detector de metales es un aparato capaz de detectar las variaciones del campo magnético terrestre producidas por la presencia de elementos metálicos. Este tipo de elementos, por sus propiedades intrínsecas, producen una intensidad de campo magnético que perturba el campo magnético de la Tierra.

El campo magnético terrestre **[1]** se origina en el núcleo del planeta, donde fluyen corrientes de hierro líquido que son conductoras de la electricidad. Este movimiento de cargas eléctricas es el causante de este campo y como todo campo magnético, tiene dos polos opuestos. Imaginando un imán colocado en el centro de la Tierra, las líneas de flujo del campo salen por el polo norte y entran por el polo sur del imán (figura 4.1). Este campo varía entre aproximadamente 25 y 65 µT (0.25 – 0.65 G), lo cual se trata de valores muy pequeños si se compara con cualquier elemento metálico o el típico imán de nevera, que ronda en los 100 G (gauss). Debido a este valor tan pequeño del campo magnético terrestre, se considerará que es constante, de modo que el aparato detector de metales que se implemente en el proyecto no se verá afectado en la lectura de las medidas tomadas por el sensor debido a estas pequeñas variaciones.

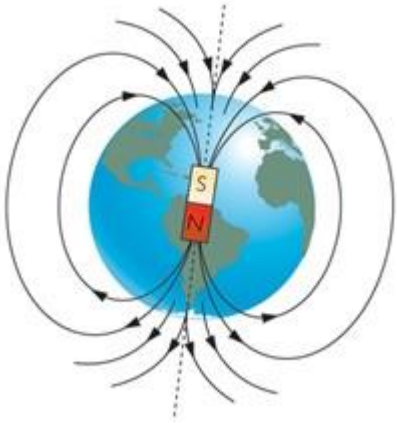


Figura 4.1 – *Líneas de flujo del campo magnético terrestre*

4.1. Tipos de detector de metales

Actualmente se utilizan diferentes tipos de detector de metales para diferentes aplicaciones, entre los cuales, los más extendidos son:

- El detector de metales tipo manual usado en el área de seguridad de muchos establecimientos, empresas, grandes eventos, etc. Éste es muy manejable e indica la presencia de metales mediante una señal sonora. Su precio pueda variar mucho según el fabricante y la precisión del aparato, pero se pueden encontrar a partir de unos 30 € con unas prestaciones mínimas.

Figura 4.2 – *Detector de metales manual*

- El detector de metales usado para encontrar cualquier objeto metálico enterrado como monedas, tesoros o minas explosivas. Su precio también varía bastante según la marca y tipo de metales que es capaz de detectar. Aun así, su coste es mucho más elevado que el detector manual, desde los 50 € hasta más de 1000 €.

Figura 4.3 – *Detector de metales para uso personal*

- El detector de metales de tipo pórtico usado en la entrada de algunos edificios, instalaciones o aeropuertos.

Figura 4.4 – *Detector de metales pórtico*

- Los detectores de metales utilizados en la industria alimentaria, química o farmacéutica, entre otras muchas. Éstos se usan para mejorar la calidad final del producto y/o mejorar la seguridad de los trabajadores.

Figura 4.5 – *Detector de metales usado en procesos industriales*

5. Introducción al diseño de aplicaciones móviles

Uno de los objetivos principales es controlar el detector de metales con un teléfono móvil con sistema operativo Android, el cual se conectará al dispositivo mediante el sistema bluetooth del teléfono. Por lo tanto, habrá que incorporar un módulo bluetooth al detector para establecer la conexión teléfono-detector y poder realizar el intercambio de datos necesario.

La intención es crear una aplicación móvil con la que el usuario pueda conectarse y desconectarse del dispositivo detector, activarlo o desactivarlo y que se monitorice su estado en la pantalla de dicha aplicación. Es decir, que se muestre mediante señales visuales y/o sonoras la detección de algún metal. Para tal fin, se dispone gratuitamente de la plataforma online App Inventor 2 **[2]**, creada por el MIT (Instituto tecnológico de Massachusetts) y desarrollada por Google Labs. Elaborada principalmente para un público no familiarizado con la programación informática, esta web permite diseñar aplicaciones para móviles con sistema operativo Android, de forma visual a través de un sistema de bloques y herramientas básicas. Las aplicaciones resultantes son sencillas pero cubren un gran número de necesidades básicas en un dispositivo móvil, suficiente para cumplir los objetivos de este proyecto.

Antes de empezar con el diseño de la aplicación para del detector de metales, es necesario conocer el entorno de esta plataforma y las herramientas con las que se trabajará.

El primer paso, es registrarse con una cuenta gmail para poder acceder a la web de App inventor 2. Una vez hecho, simplemente se tiene que entrar a la dirección url http://ai2.appinventor.mit.edu (figura 5.1) para comenzar a diseñar la aplicación.

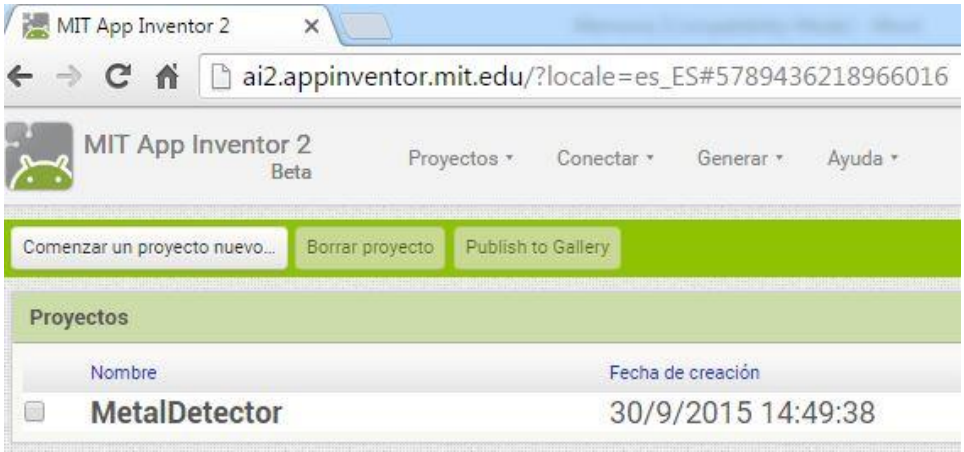


Figura 5.1 - *Página de inicio de la web App Inventor 2*

.

En este caso, el proyecto se ha nombrado MetalDetector. A continuación, si se hace clic sobre su nombre, aparece la ventana Diseñador (figura 5.2), desde donde se añaden los elementos junto con sus propiedades para crear la aplicación. Esta ventana también se utiliza para configurar la apariencia de la aplicación.

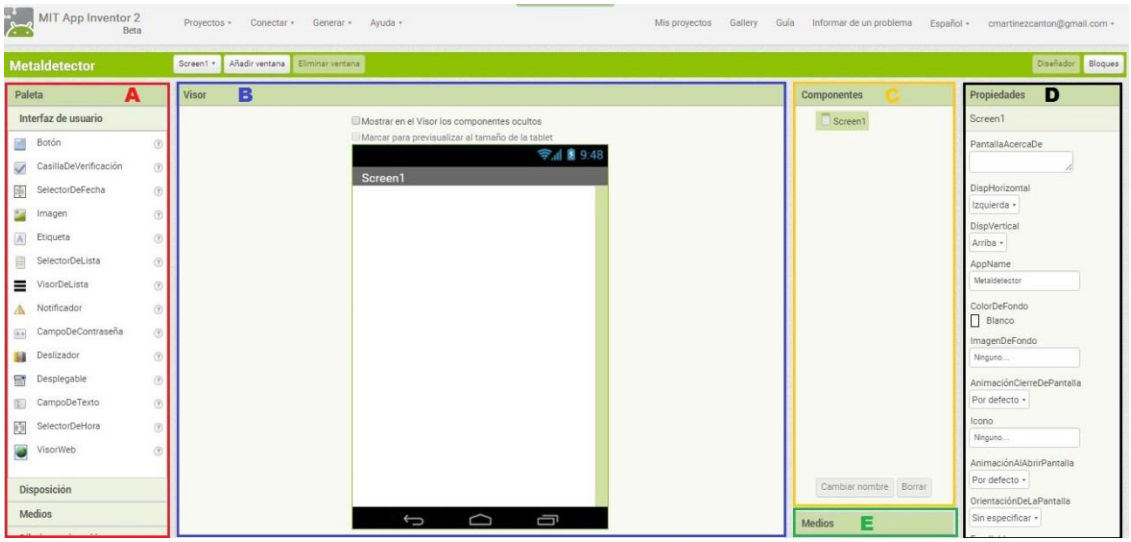


Figura 5.2 - *Ventana de diseño de App Inventor 2*

A la izquierda de la pantalla (ver recuadro A en rojo de la figura 5.2) se encuentra la paleta del proyecto, donde aparecen clasificados por familias los diferentes elementos que se pueden incluir en la aplicación y, con los que más adelante, se implementará el software mediante un sistema de bloques. La clasificación de estos componentes es la siguiente: interfaz de usuario, disposición, medios, dibujo y animación, sensores, social, almacenamiento y conectividad.

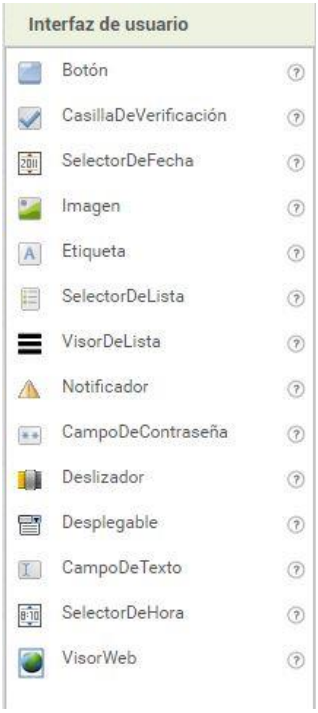
En el centro de la página (ver recuadro B en azul de la figura 5.2) está el visor, el cual simula la pantalla del dispositivo móvil y donde se ubican, a conveniencia del usuario, los elementos que se han seleccionado previamente. Sólo se tiene que arrastrar cada componente deseado de la paleta al visor para que éste se incluya en el proyecto.

Los elementos seleccionados aparecen en el apartado componentes (recuadro C en amarillo de la figura 5.2), desde donde se puede cambiar su nombre o eliminarlos. Debajo (recuadro E en verde), se situan los medios utilizados, ya sean archivos de audio, videos o imagenes que sube el propio usuario.

Por último, en la pestaña Propiedades (ver recuadro D en negro de la figura 5.2) se pueden configurar las propiedades de cada uno de los elementos escogidos: apariencia, duración de archivos de audio, etc.

Para crear la aplicación móvil que controla el detector de metales, únicamente se requieren elementos de los siguientes tipos:

- **Interfaz de usuario:** concretamente de este tipo se utilizan el botón, la imagen, la etiqueta y el selector de lista.



El botón detecta cuándo se hace clic sobre él. Esto es útil para ejecutar una determinada acción al pulsarlo, como por ejemplo, encender el aparato detector de metales.

El selector de lista es un botón que al presionarlo muestra una lista de textos entre los cuales el usuario puede escoger. Estos textos se pueden definir a través de la ventana de diseño, configurando una de las propiedades de este componente, o desde el editor de bloques, especificando en la propiedad Elementos el nombre de una lista.

La imagen es un componente para mostrar fotografías. Mediante el diseñador o el editor de bloques se pueden definir tanto la foto que hay que mostrar, como otras características que afectan a su apariencia.

Figura 5.3 - *Elementos de interfaz de usuario*

La etiqueta muestra un texto, definido en su propiedad Texto. Utilizando otras propiedades, que pueden manejarse en el diseñador o en el editor de bloques, es posible modificar la apariencia y la ubicación del texto.

- **Disposición:** son elementos para colocar el resto de componentes de forma ordenada en el visor.

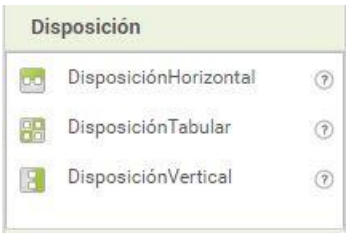


Figura 5.4 - *Elementos de disposición*

- **Medios:** de este tipo se utilizará exclusivamente un componente de sonido. Éste es un campo multimedia que reproduce sonidos y opcionalmente vibra el número de milisegundos especificado en el editor de bloques.

A diferencia del elemento Reproductor, que puede reproducir archivos de sonido más largos, como canciones, el componente Sonido es más adecuado para archivos de corta duración, como efectos de sonido. En el caso del detector de metales se utilizará un efecto de sonido que simule el pitido típico al detectar un metal por este tipo de aparatos.

Figura 5.5 - *Medios disponibles*

- **Sensores:** se hará uso del Reloj, que es un componente no visible en el visor, el cual permite utilizar un temporizador. Será útil para ejecutar una serie de acciones cíclicamente. En las propiedades del reloj (figura 5.6) se configura el intervalo del temporizador en milésimas de segundo.

Figura 5.6 - *Sensores disponibles en App Inventor 2*

- **Conectividad:** en el proyecto está presente el componente ClienteBluetooth, que permite establecer la conexión del dispositivo móvil con el detector de metales a través del sistema bluetooth de ambos aparatos.

Figura 5.7 - *Componentes de conexión de App Inventor 2*

Una vez seleccionado los componentes deseados, el siguiente paso a realizar es el código de programación a través del editor de bloques (figura 5.8). En el apartado Bloques, a la izquierda de la pantalla, aparecen el conjunto de comandos de programación con los que se puede implementar el código. Estos están clasificados por familias, desde comandos de control hasta procedimientos. Para incluir uno de ellos en el programa, se debe arrastrar hasta el recuadro Visor, donde se combinarán los distintos comandos escogidos como si de un puzzle se tratara, con el fin de que se ejecuten determinadas acciones al realizarse u ocurrir otras.

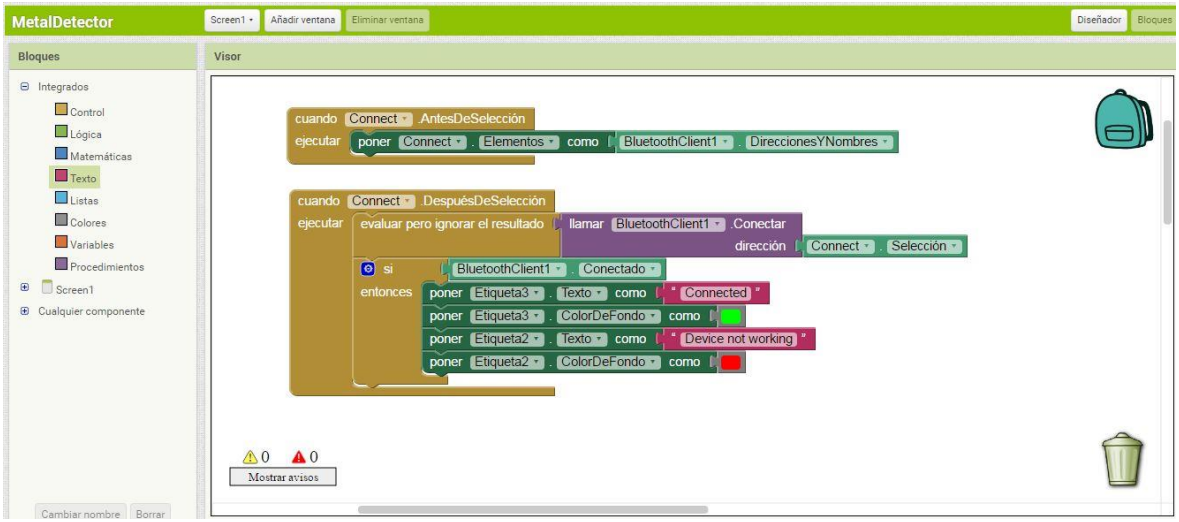


Figura 5.8 - *Editor de bloques de App Inventor 2*

En el apartado Bloques también aparecen los componentes que se han elegido desde el Diseñador de App Inventor 2. Al pulsar uno de estos elementos, aparece una ventana en el Visor con las diferentes instrucciones disponibles para ese componente (figura 5.9).

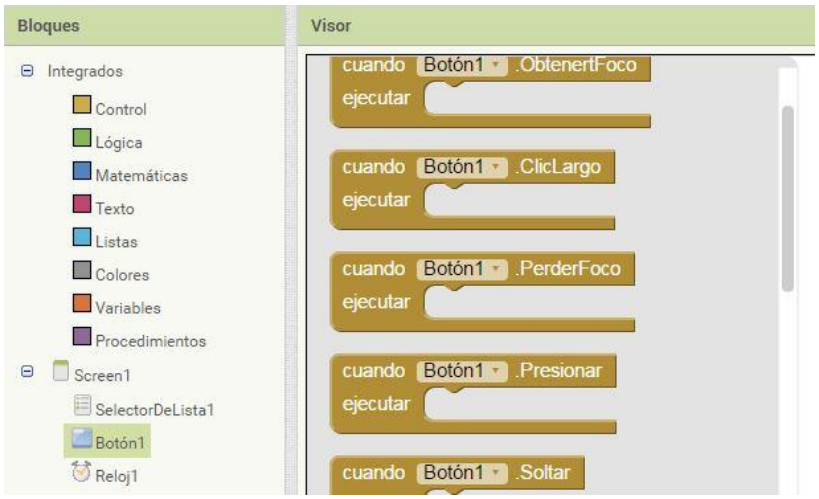


Figura 5.9 – *Instrucciones de código disponibles para el botón*

De los integrados (figura 5.10), básicamente se necesitan comandos de control, de lógica, de matemáticas y de texto para conformar el programa de la aplicación móvil del detector de metales.

Figura 5.10 – *Instrucciones de código para los integrados de control*

Una vez finalizado el diseño y programación de la aplicación móvil, se debe generar un código QR desde la pestaña Generar de la página de inicio del proyecto (figura 5.11).

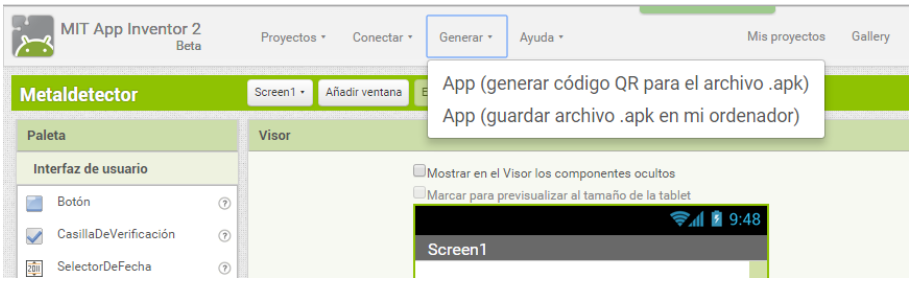


Figura 5.11 – Ventana desplegable del botón generar

Se deberá descargar gratuitamente la aplicación móvil MIT AI2 Companion (disponible en Play Store) con el fin de instalar la aplicación diseñada en el teléfono mediante el escaneo del código QR que se haya generado (figura 5.12).

Figura 5.12 – *Código QR generado mediante App Inventor 2*

5.1. Aplicación móvil para el control del detector de metales

Una vez conocido el entorno de trabajo de App Inventor 2, se ha llevado a cabo el diseño de la aplicación móvil para el control y monitorización del dispositivo detector de metales. Dicha aplicación se ha nombrado Metal Detector y su apariencia después de múltiples cambios y mejoras es la de la figura 5.13.

Figura 5.13 – *Aplicación móvil Metal Detector*

De la paleta de componentes disponibles para la creación de aplicaciones Android, los principales elementos seleccionados han sido:

- Un **selector de lista** nombrado “Connect to bluetooth device”, con el que el usuario puede conectarse vía bluetooth a otros dispostivos. Al hacer clic sobre él, se abre una ventana con la lista de dispositivos detectados. Para clarificar su funcionalidad, se ha colocado al lado del selector el logotipo de la tecnología bluetooth.

- Cuatro **botones**. El primero de ellos, nombrado “Disconnect bluetooth device”, se utiliza para deshabilitar la conexión bluetooth entre el teléfono móvil y el dispostivo previamente seleccionado con el selector de lista anterior. A la derecha de este botón, aparece una imagen del logotipo de bluetooth con una franja roja que simboliza la desconexión.

  Al pulsar el botón “Start”, se pone en marcha el detector de metales. Por el contrario, el botón “Stop” detiene su funcionamiento. Estos dos botones se han coloreado en verde y rojo respectivamente para que el usuario comprenda mejor su utilidad.

  Por último, presionando el botón “EXIT” se sale de la aplicación.

- Un efecto de **sonido** similar al que realiza un detector de metales del mercado actual.

- Elementos no visibles en la pantalla del teléfono como el **clienteBluetooth**, para poder realizar las conexiones con otros dispositivos mediante tecnología bluetooth. También

un **Reloj** cuyo temporizador se ha programado para que salte cada medio segundo. La función de estos elementos se entenderá mucho mejor cuando se analice el sistema de bloques que conforma el código de programación de la aplicación.

- Cuatro **etiquetas.** Una de ellas es la que incluye el título de la aplicación, Metal Detector. El resto, cuya ubicación se indica en la figura 5.13, sirven para monitorizar el estado del detector de metales: mostrar si se ha conectado o desconectado el teléfono móvil del detector, si éste se encuentra en funcionamiento o no y si se ha detectado algún elemento metálico.

La aplicación móvil, a medida que se ha ido diseñando, se ha testeado con un prototipo del detector de metales que se ha montado sobre una placa protoboard (figura 5.14). Este prototipo está formado por un microcontrolador (rodeado en rojo), un sensor magnético (rodeado en amarillo) y un módulo bluetooth (rodeado en verde). El conjunto se ha alimentado a 5V mediante un programador PICkit 2 conectado vía USB a un ordenador. De momento, no fijarse en el resto de componentes que aparecen en la placa, tales como resistencias óhmicas y cables. Estos se han utilizado previamente para realizar pruebas durante la programación del microcontrolador y se explicará con detalle su función en siguientes puntos de la memoria del proyecto. Ahora únicamente pensar que sobre la protoboard se ha realizado la conexión de los dispositivos que conforman el detector de metales y la aplicación móvil Metal Detector es la que se encarga de controlarlo.

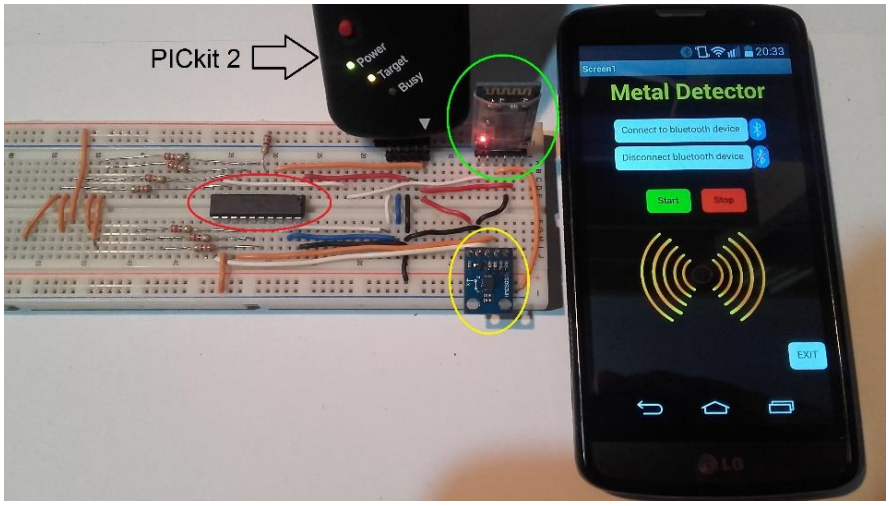


Figura 5.14 – *Detector de metales montado en la placa protoboard y la aplicación móvil*

A continuación se expone por partes el programa que implementa esta aplicación junto con imágenes de muestra donde se puede ver su funcionamiento junto al prototipo.

En la figura 5.15 se observa el conjunto de bloques correspondiente a las instrucciones a realizar antes y después de seleccionar el selector de lista “Connect to bluetooth device” (Connect en el sistema de bloques). Antes de seleccionarlo, es decir, al abrir la aplicación móvil y activar el sistema bluetooth del teléfono, se almacena en esta lista las distintas direcciones de los dispostivos bluetooth que hay al alcance del móvil. Una vez se haga clic sobre el selector, se abrirá una ventana con la lista mencionada y allí se podrá seleccionar el dispositivo con el que se quiere establecer la conexión bluetooth. En este caso aparecerá en la lista la dirección del módulo bluetooth HC-05 incorporado al detector de metales (figura 5.16).

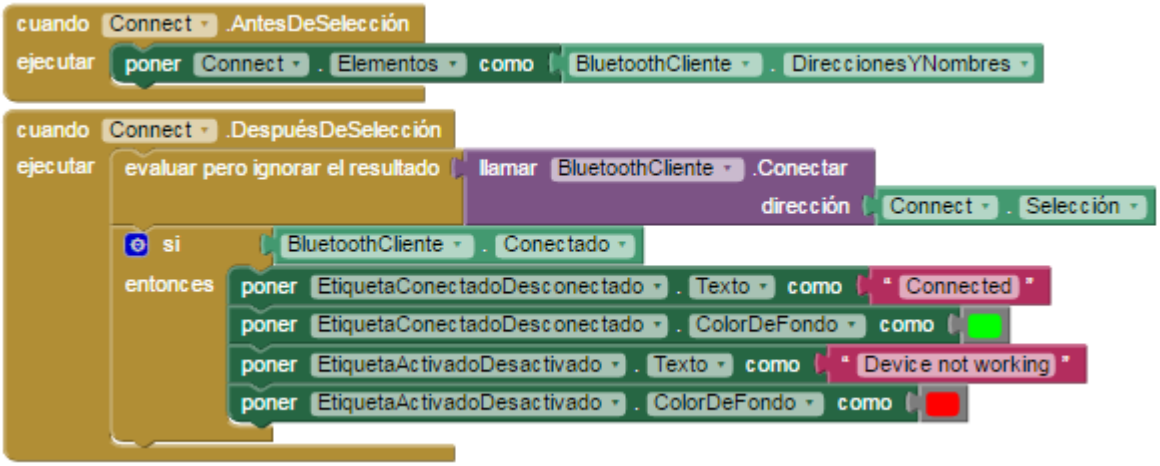


Figura 5.15 – *Código de programación del selector de lista “Connect to bluetooth device”*

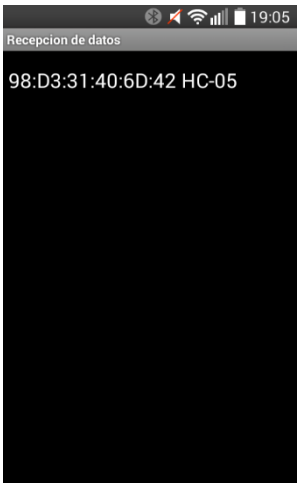


Figura 5.16 – *Dirección del módulo bluetooth incorporado al detector de metales.*

Para conectarse a él se debe seleccionar. Si la conexión tiene éxito, aparecerá en la pantalla el texto “Connected” con un fondo verde y el texto “Device not working” para indicar que el detector no está en funcionamiento.

En la figura 5.17 se muestra el funcionamiento de este código. En ella, ya se ha conectado el teléfono al detector de metales y, por tanto, aparecen los dos textos en verde y rojo indicando su estado.

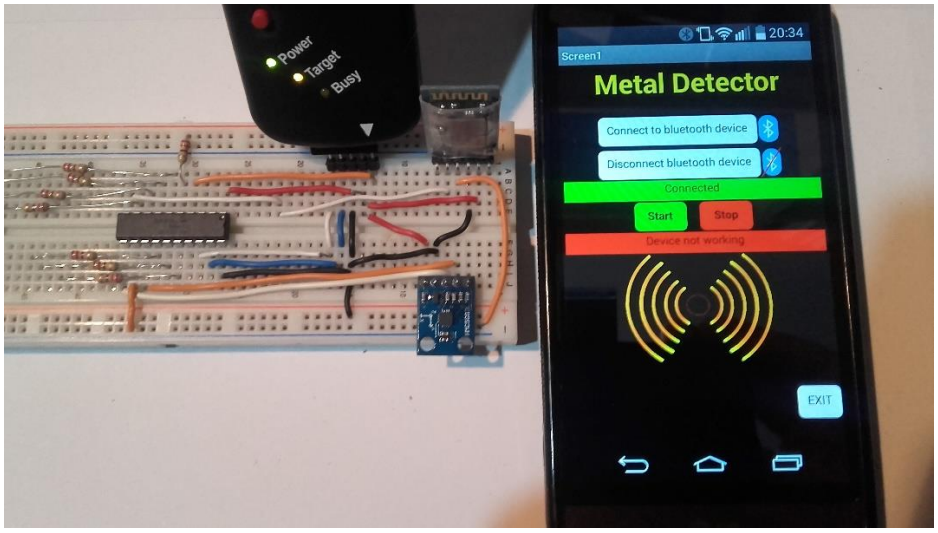


Figura 5.17 – *Conexión establecida entre el teléfono y el módulo bluetooth*

En el siguiente sistema de bloques (figura 5.18) se declaran las instrucciones a realizar con el botón “Disconnect bluetooth device” (Disconnect en el conjunto). Al presionarlo, se desconecta el teléfono móvil de aquel aparato al que se ha conectado previamente. Si se desconecta correctamente, aparece en la pantalla el texto “Disconnected” con fondo rojo y se limpian los textos del resto de etiquetas (la de estado de funcionamiento y la de detección de un metal). También se desactiva el efecto de sonido en el caso de que estuviera detectando un metal y de golpe se desconectara.

Figura 5.18 – *Código de programación para el botón "Disconnect bluetooth device"*

En la figura 5.19 se puede ver como la desconexión entre el teléfono y el detector de metales se indica mediante el texto "Disconnected" en fondo rojo y también evita que aparezca el texto "Metal detected" al acercar un elemento metálico como el destornillador.

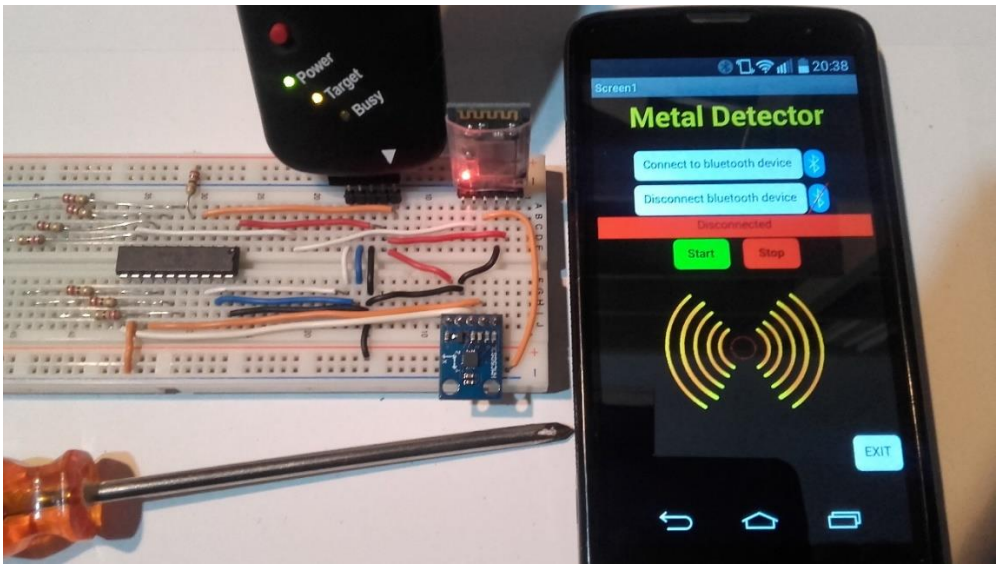


Figura 5.19 – *Desconexión teléfono-módulo bluetooth*

Al hacer clic en el botón verde "Start" (figura 5.20), si el detector está conectado al teléfono, éste último le envía el byte de valor decimal 1, el cual es procesado por el microcontrolador como una señal de encendido del sensor magnético para empezar la detección de metales. En la pantalla de la aplicación aparecerá el texto "Device working" con fondo verde para indicar este estado.

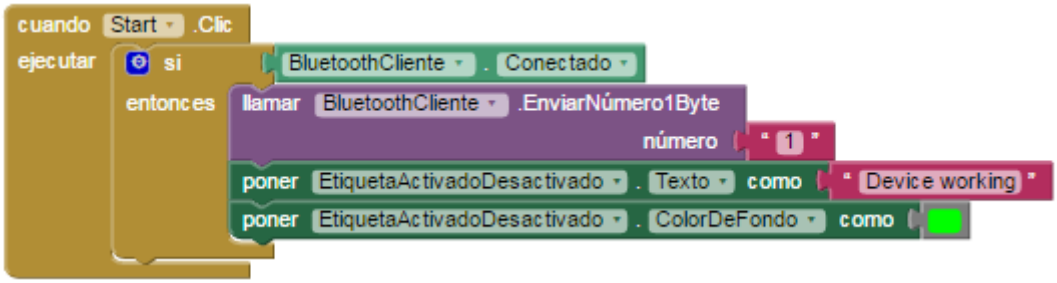


Figura 5.20 – *Código de programación para el botón "Start"*

En la figura 5.21, se observa como una vez conectado el teléfono al detector de metales (aparece texto "Connected" con fondo verde), si se pulsa el botón "Start" se activa el sensor del detector de metales y se comienza a realizar la detección de elementos metálicos ("Device working" con fondo verde).

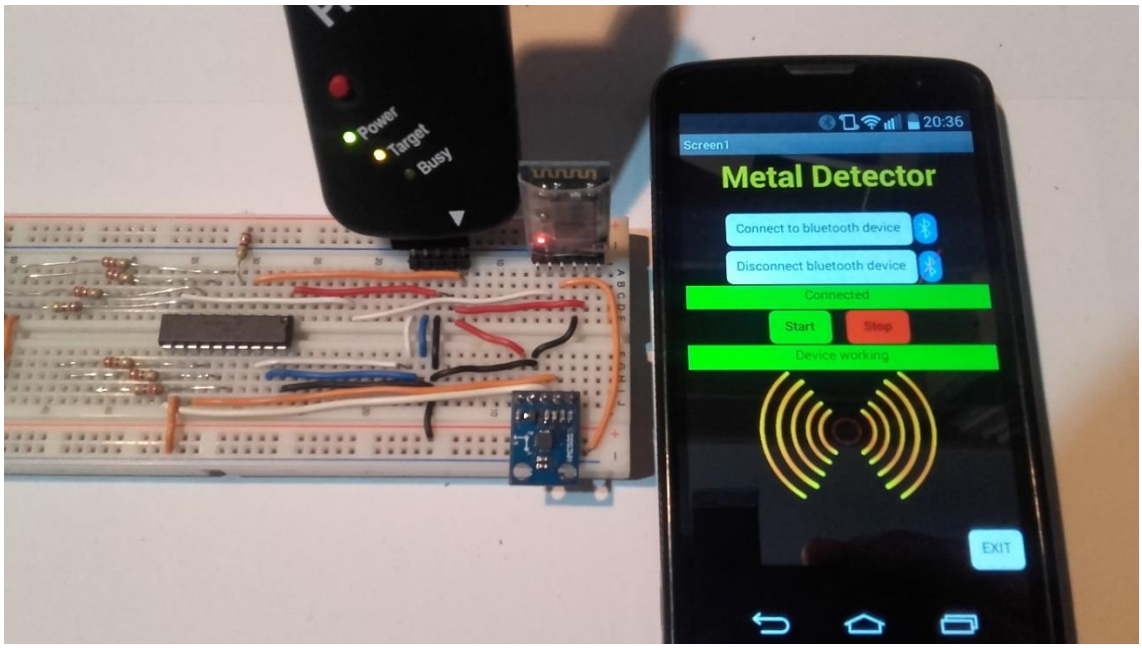


Figura 5.21 – *Sensor magnético activado*

Una vez el detector de metales está en funcionamiento, hace falta indicar la presencia de metales. En este caso, se muestra mediante el texto "Metal detected" con fondo amarillo más el efecto de sonido y una vibración del teléfono móvil (figura 5.22). Esta señal la debe enviar el detector de metales al móvil, el cual si recibe el byte con valor decimal 1 quiere decir que se ha detectado un metal y, por tanto, se activan los avisos anteriores. En cambio, si recibe el byte con valor decimal 0, no se ha detectado ningún elemento metálico y se desactivan todos ellos.

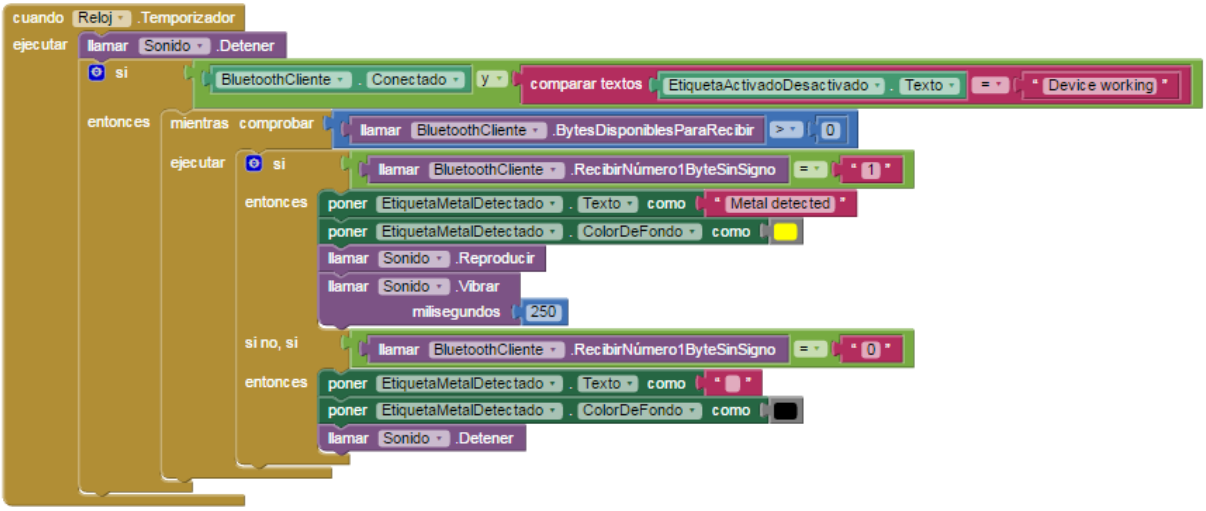


Figura 5.22 – *Código de programación referente a las señales que el móvil recibe*

Este proceso es cíclico debido al reloj que se ha incluido, cuyo temporizador se ha configurado en 500 milésimas de segundo. Por lo tanto, cada medio segundo, si la conexión bluetooth teléfono-detector está habilitada, el detector está en funcionamiento y el dispositivo móvil puede recibir bytes sin bloquearse, se activarán las señales correspondientes a un metal detectado o no.

Como se aprecia en la figura 5.23, el teléfono está conectado al detector de metales (“Connected” con fondo verde), éste está en funcionamiento (“Device working” en verde) y se ha detectado la presencia de un elemento metálico, el destornillador (“Metal detected” en amarillo), por lo cual, el móvil también realiza la señal acústica y vibra durante un instante.

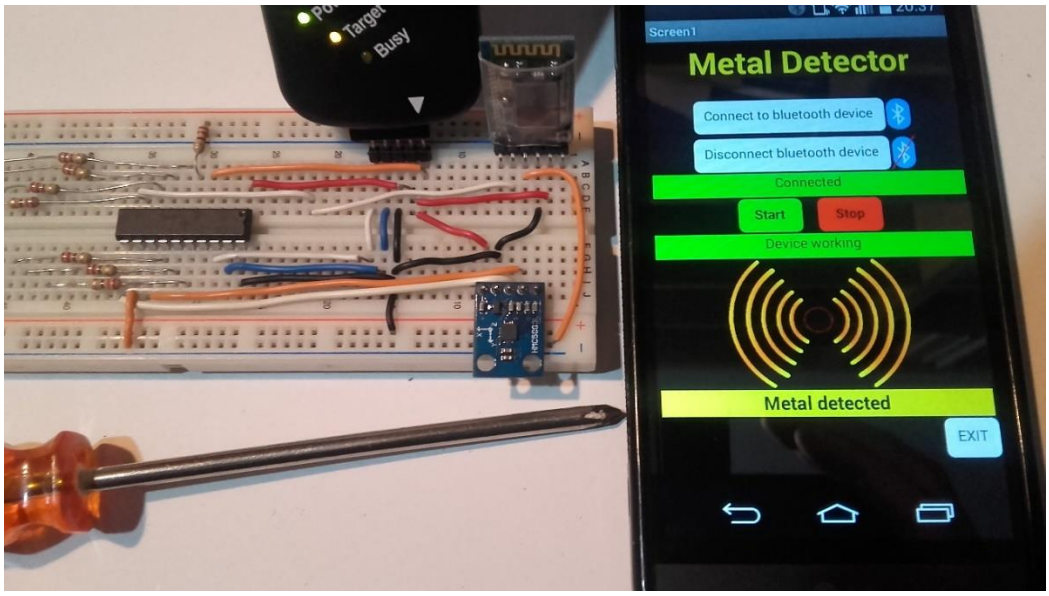


Figura 5.23 – *Detección de un elemento metálico*

Para detener el funcionamiento del detector de metales, se debe hacer clic en el botón rojo "Stop". Tal y como se indica en el sistema de bloques (figura 5.24), si la conexión teléfono-detector está establecida, al pulsar "Stop" el móvil le envía el byte de decimal 0 y el microcontrolador lo procesa como una señal de apagado del sensor magnético. Este estado se indica en el teléfono mediante el texto "Device not working" con fondo rojo. En el caso de que previamente se estuviera detectando un metal, se limpiaría el texto "Metal detected" y se detendría el efecto de sonido.

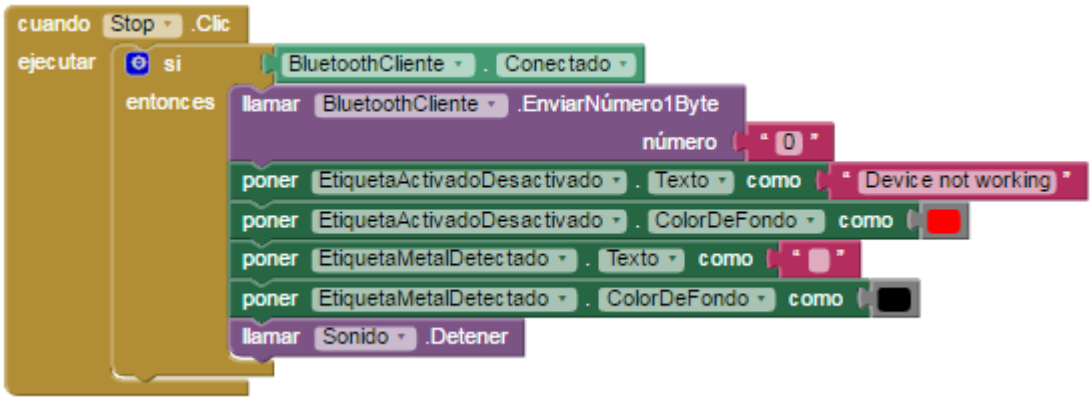


Figura 5.24 – *Código de programación del botón "Stop"*

Como se observa en la figura 5.25, se ha pulsado el botón "Stop" y el detector de metales ha dejado de funcionar (texto "Device not working" con fondo rojo"). Por lo tanto, aunque haya un destornillador cerca del sensor, éste no es detectado.

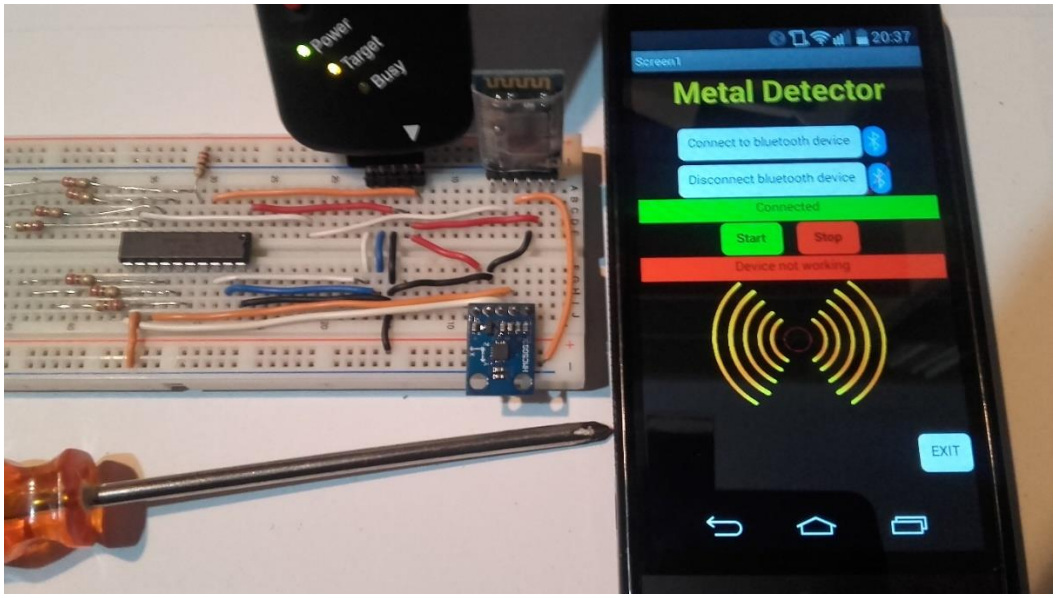


Figura 5.25 – *Detención del detector de metales*

Finalmente, para salir de la aplicación simplemente hay que pulsar el botón “EXIT” (figura 5.26).

Figura 5.26 – *Código de programación del botón “EXIT”*

6. Módulo bluetooth HC-05

Para el diseño y construcción del detector de metales se ha empleado el módulo bluetooth HC-05 (figura 6.1) **[3]**, que trabaja como elemento intermediador en la transferencia de datos entre el teléfono móvil y el microcontrolador. Para ello deben conectarse los pines RX y TX del módulo bluetooth a los pines TX y RX del microchip respectivamente. También se debe alimentar el HC-05 a la misma tensión que el microcontrolador con los pines +5V y GND (tierra).

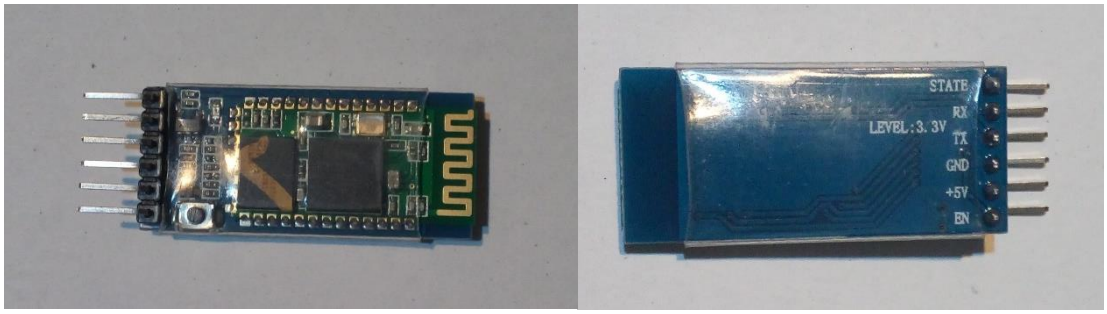


Figura 6.1 – *Módulo bluetooth HC-05*

Los pines RX son los pines receptores de cada dispositivo. Por ellos reciben los datos. En cambio, los pines TX son los pines transmisores de datos. Por lo tanto, el módulo bluetooth transmite datos a través de su pin TX hacia el pin RX del microcontrolador y el microcontrolador transmite datos por su pin TX que llegan al pin RX del bluetooth.

Mediante la aplicación móvil Metal Detector se establece conexión con el módulo bluetooth HC-05. Al pulsar los botones “Start” y “Stop” de la aplicación, se transmite el dato correspondiente vía tecnología bluetooth al dispositivo HC-05 y finalmente éste lo transmite por su pin TX hacia el pin RX del microcontrolador. En el caso que sea el microcontrolador quien transmite un dato, éste llegará primero al módulo bluetooth y a continuación se enviará al teléfono móvil.

7. El microcontrolador PIC16F690

Para desarrollar el detector de metales se ha empleado un microcontrolador de la familia PIC del fabricante Microchip. Un PIC (Controlador de Interfaz Periférico) consiste en un circuito integrado o microchip contenido en un encapsulado de plástico o cerámica. En resumen, consiste en un conjunto de circuitos electrónicos que mediante su programación se es capaz de automatizar procesos eléctricos y mecánicos. Un microcontrolador se compone de puertos, una unidad aritmética lógica, contadores de programa, temporizadores e interrupciones, una memoria de programa y una memoria de datos, bancos de datos, un conversor análogo digital, etc., los cuales son esenciales para su programación y automatización de procesos tales como el simple encendido de un LED, el encendido de un motor, el muestreo de una señal de datos, un sin fin de cosas que tienen aplicaciones hoy en día.

Se ha escogido el PIC16F690 ya que cumple con las prestaciones que requiere el detector de metales y es muy barato, además de contener variedad de funciones que facilitan su programación. En la figura 7.1 se observa el diagrama de pines de este microcontrolador. Del encapsulado salen veinte conductores metálicos que corresponden a los veinte pines que contiene, apropiados para conectar circuitos externos. Para ponerlo en funcionamiento se debe alimentar a una tensión de 5 V mediante los pines 1 y 20, siendo el pin V_{DD} la tensión alta y el V_{SS} la tensión nula o tierra.

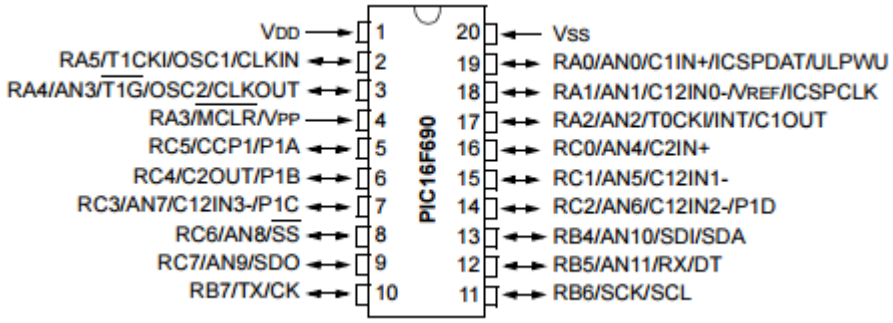


Figura 7.1 – *Diagrama de pines del PIC16F690*

El PIC16F690 está formado por módulos con una funcionalidad diferente y que trabajan con una serie de pines y registros. Los registros son espacios de memoria en los que se almacenan los datos a procesar por el microcontrolador. Este microchip es de tipo 8-bit, por lo tanto, estos registros contienen datos formados por ocho bits, que equivalen a un byte. El valor de un bit es binario: ‘1’ o ‘0’.

Por ejemplo, uno de los registros que se usará más adelante es el registro TRISC, el cual se utiliza para configurar como salidas o entradas los pines del puerto C, que son los ocho pines del microcontrolador nombrados RC0 a RC7 (pines número 5, 6, 7, 8, 9, 14, 15 y 16). Si uno de estos pines está configurado como entrada, se habilita la entrada de señales por el conductor metálico correspondiente. Esta señal procederá de algún dispositivo conectado al microcontrolador. Por otro lado, si un pin es configurado como salida, se habilita la salida de señales a través de él.

El byte que se almacena en el registro TRISC es el correspondiente al valor binario de los ocho bits TRISC7 a TRISC0, siendo el bit TRISC7 el más significativo y el bit TRISC0 el menos significativo. De forma que cuando uno de estos bits vale '0' se configura el pin correspondiente del puerto C como salida y cuando vale '1' se configura como entrada. Es decir, si TRISC5=1, el pin RC5 se configura como una entrada de datos al PIC.

R/W-1	R/W-1	R/W-1	R/W-1	R-1	R/W-1	R/W-1	R/W-1
TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
bit 7							bit 0

Tabla 7.1 - *Bits del registro TRISC*

Según el registro, se podrá escribir en él un dato (enviar ocho bits y que los almacene), leer un dato (recibir los ocho bits que contiene el registro) o ambas acciones. La programación de estos registros es clave para construir el código de programación del microcontrolador.

Para la construcción del detector de metales se conectarán al microcontrolador dos dispositivos: el módulo bluetooth HC-05 y un sensor de campo magnético. Entre estos tres elementos debe establecerse una comunicación para el envío y recibimiento de datos. El sensor capta variaciones de campo magnético, procesa estos datos y los envía al microcontrolador, que a su vez los procesa y los envía al módulo bluetooth para ser enviados finalmente vía esta tecnología al teléfono móvil. Y viceversa para conectar, desconectar, encender o apagar el detector desde la aplicación móvil. Esta comunicación entre distintos dispositivos implica que haya unas señales de reloj **[4]** que coordinen los sistemas.

Del PIC16F690 se hará uso del módulo EUSART (Enhanced universal synchronous asynchronous receiver transmitter) **[5]**. Con este módulo se configuran las opciones de transmisión y recepción de datos entre el microcontrolador y los dispositivos que se conecten a él (otros circuitos integrados).

Para realizar el código de programación se utilizará el editor MPLAB X IDE versión 3.10 **[6]**, destinado a productos de la marca Microchip. Este editor se basa en el lenguaje de

programación c para implementar el código. Para compilarlo, es decir, para pasar el conjunto de instrucciones que constituyen el programa de lenguaje de programación c a lenguaje máquina (a código binario, ceros y unos), se hará uso del compilador XC8 versión 1.35 **[7]**. Una vez compilado, se transferirá del ordenador a la memoria de programa del microcontrolador mediante el dispositivo programador PICkit2 (figura 7.2) **[8]**, que contiene seis pines de salida pero únicamente se requiere conectar los cinco primeros al microchip para programarlo. Los pines del PIC a los que se debe conectar el PICkit2 son aquellos nombrados de igual manera.

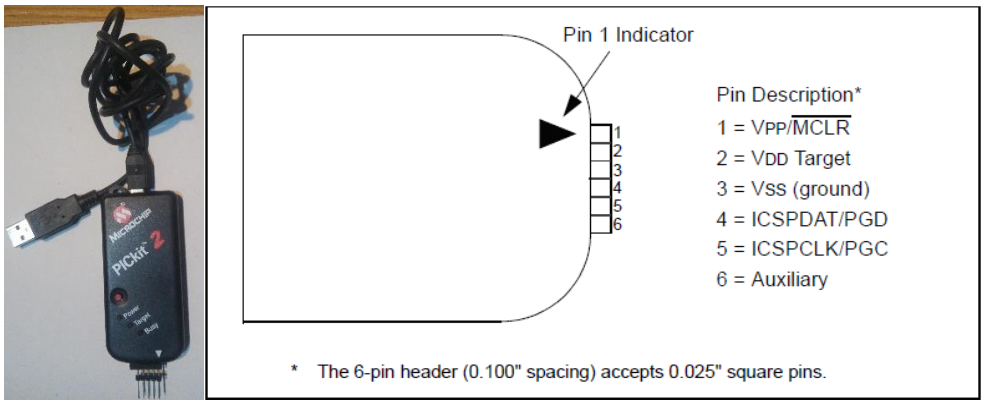


Figura 7.2 - *Dispositivo programador PICkit2 y su diagrama de pines*

Una vez conocidas las principales características del PIC16F690 y los medios que se usarán para programarlo, véase el diagrama de flujo de la figura 7.3, el cual representa las principales acciones que ejecutará según las señales que reciba de la aplicación móvil y según las lecturas que haga el sensor magnético.

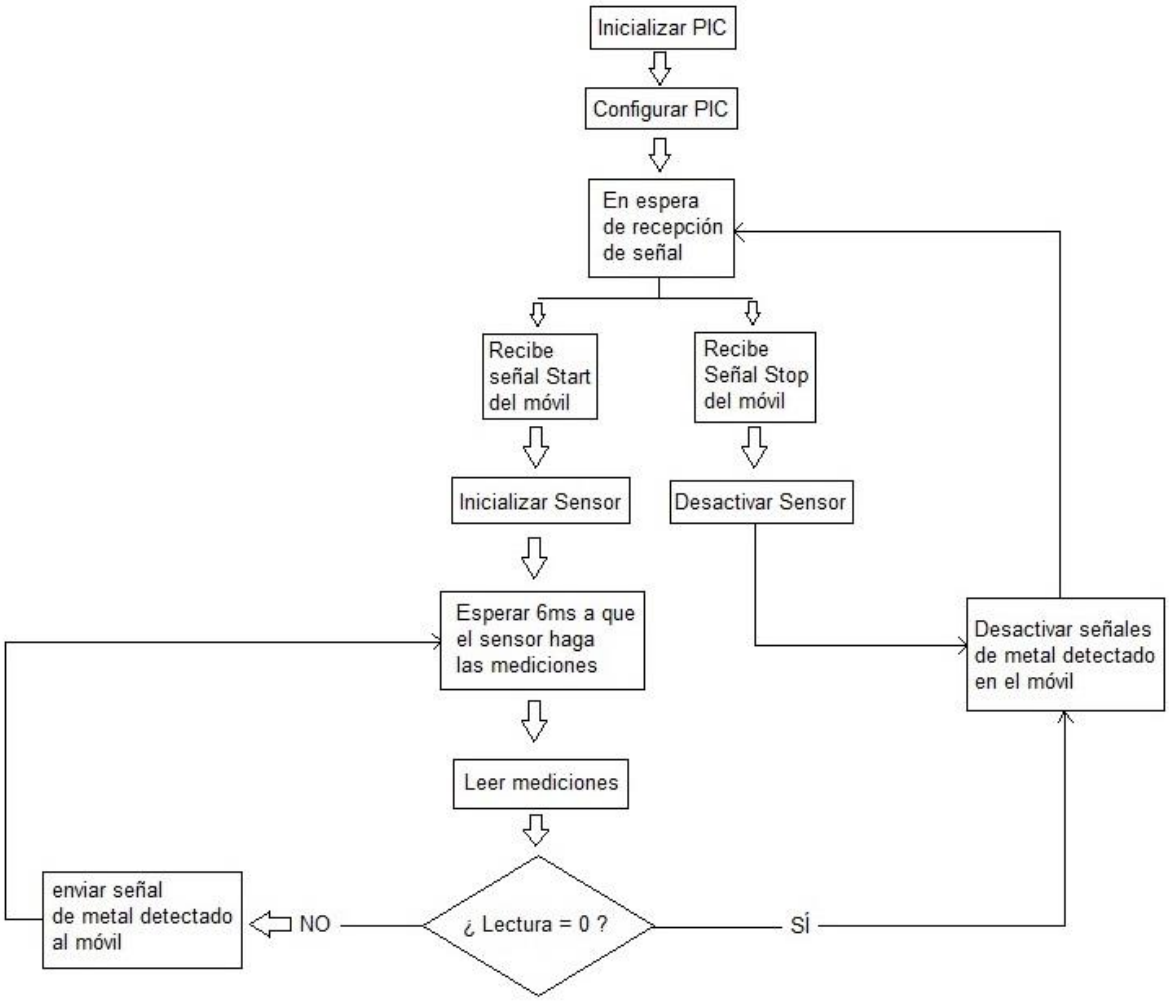


Figura 7.3 – *Diagrama de flujo del conjunto de acciones que realiza el PIC16F690*

Ahora, el primer paso al realizar el código con el editor MPLAB, es incluir las librerías “xc.h” y “pic16f690.h” (figura 7.4). Éstas hacen que MPLAB y el compilador XC8 carguen automáticamente los registros y parámetros del microcontrolador PIC16F690, de forma que al escribir en el programa el nombre de un pin, de un bit o de un registro del microchip, éste sea identificado automáticamente.

```
1  #include <xc.h>
2  #include <pic16f690.h>
3
4  //Inicio de la configuracion de bits del PIC:
5  #pragma config FOSC = INTRCIO
6  #pragma config WDTE = OFF
7  #pragma config PWRTE = OFF
8  #pragma config MCLRE = OFF
9  #pragma config CP = OFF
10 #pragma config CPD = OFF
11 #pragma config BOREN = OFF
12 #pragma config IESO = OFF
13 #pragma config FCMEN = OFF
14 //Fin de la configuracion de bits
15
16 #define _XTAL_FREQ 4000000
```

Figura 7.4 – *Inicio del código de programación*

A continuación se deben configurar los bits de configuración del microcontrolador, que permiten reconfigurar ciertos aspectos generales de su funcionamiento tales como un temporizador de encendido (bit PWRTE), reinicio en caso de apagón (bit BOREN), etc. Se han desactivado todos ellos (estado OFF) ya que no son relevantes para el buen funcionamiento del detector de metales, a excepción del bit FOSC. Éste es el bit de selección de oscilador de frecuencia y se ha configurado de tipo INTRCIO, lo cual indica que el oscilador es interno. El oscilador de frecuencia es el circuito que indica la velocidad de trabajo del microcontrolador y, por tanto, es un elemento indispensable de todo microprocesador o microcontrolador. Se trata de una señal de reloj que indica cada cuanto tiempo se debe ejecutar una instrucción del programa que se ha grabado en el microchip. Cada cuatro ciclos o periodos de esta señal se ejecuta una instrucción del código de programación. Así pues, si se configura la frecuencia en 4 MHz, cada instrucción se ejecutará en 1 µs (ec. 7.1).

$$\frac{tiempo}{instrucción} = 4 \cdot T = 4 \cdot \frac{1}{f} = 4 \cdot \frac{1}{4 \cdot 10^6\ s^{-1}} = 10^{-6} s = 1\mu s \qquad \text{(ec. 7.1)}$$

Al ser interno este oscilador de frecuencia, no se requiere el uso de circuitos electrónicos externos para generar esta señal de reloj, reduciendo así las conexiones y cableado del detector de metales.

El oscilador de frecuencia del PIC16F690 es de tipo “XT” (XTAL), cuya frecuencia máxima es de 4 MHz (la que se ha utilizado). En el código se debe introducir mediante la instrucción #define, seguido del tipo de oscilador (XTAL_FREQ) y el valor de frecuencia en hercios (ver figura 7.4).

Todos estos bits de configuración se guardan en una localización específica de la memoria de programa del PIC y se graban al mismo momento que el código de programación.

A continuación se exponen los diferentes módulos que contiene el microcontrolador y su configuración necesaria para la implementación del detector de metales. También se explica el código propuesto para tal efecto.

7.1. Módulo EUSART del PIC16F690

El módulo EUSART del PIC16F690 es aquel utilizado para la comunicación en serie con los periféricos que se conecten a él, ya sean otros microcontroladores o cualquier otro circuito integrado. En este caso, para construir un detector de metales y controlarlo con tecnología bluetooth, es necesario conectar al microchip un sensor de campo magnético y un módulo

bluetooth. De tal manera que configurando adecuadamente la EUSART se transmitirán y recibirán datos entre el PIC y estos dos dispositivos externos.

Se trata de habilitar la comunicación entre distintos circuitos electrónicos, por lo tanto, son necesarias unas señales de reloj para coordinar estos circuitos. Estas señales de reloj las contiene el módulo EUSART, además de los registros de almacenamiento necesarios para la transferencia de datos.

Los registros con los que opera este módulo son los siguientes:

- **TXSTA** (Transmit Status and Control): registro utilizado para la transmisión de datos.
- **RCSTA** (Receive Status and Control): registro utilizado para la recepción de datos.
- **BAUDCTL** (Baud Rate Control): registro dedicado al soporte de operaciones síncronas y asíncronas del módulo EUSART.

7.1.1. Modo asíncrono de la EUSART

La EUSART se puede configurar en modo asíncrono o síncrono. En el modo asíncrono, las señales de reloj necesarias para sincronizar el microcontrolador con los dispositivos externos conectados a él son internas. Es decir, cada aparato genera su propia señal de reloj. En cambio, en el modo síncrono es el microcontrolador quien debe generar la señal de reloj para el resto de dispositivos que se conecten a él, de modo que se requiere un cable más (una conexión más) por donde se envíe esta señal de reloj.

Para la construcción del detector de metales se ha conectado al PIC un sensor de campo magnético y un módulo bluetooth, ambos con señales de reloj internas. Por lo tanto, se ha configurado el módulo EUSART en modo asíncrono, que corresponde a la instrucción SYNC=0.

El módulo EUSART transmite y recibe datos utilizando el formato estándar non-return-to-zero (NRZ), que se implementa con dos valores, uno alto equivalente a un bit ‘1’ y uno bajo equivalente a un bit ‘0’. NRZ se refiere al hecho de que en la transmisión consecutiva de bits con el mismo valor se mantiene este valor sin pasar por un valor neutral entre la transmisión de un bit a otro. Por lo tanto, si se están transmitiendo, por ejemplo, tres bits consecutivos de valor

‘1’, la salida mantiene el valor alto entre el paso del primer bit al segundo y entre el paso del segundo bit al tercero.

Los pines que utiliza este módulo para transmitir y recibir información de dispositivos que estén conectados al microcontrolador son los pines transmisor TX y receptor RX (figura 7.5).

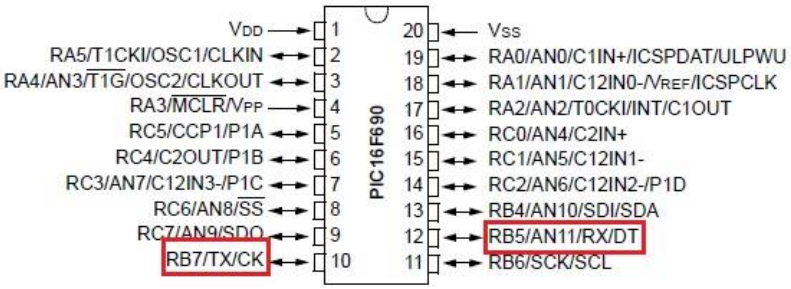


Figura 7.5 – *Pines transmisor y receptor del PIC16F690*

La EUSART transmite y recibe esta información en forma de carácteres. Un carácter consiste en un bit de arranque de valor ‘0’ (Start bit) seguido de ocho bits que corresponden al dato transferido y finalizado por un bit de parada de valor ‘1’ (Stop bit). Estos bits de arranque y parada se utilizan para sincronizar los dos dispositivos entre los que se realiza la transferencia del dato (entre el microcontrolador y el sensor por ejemplo), de forma que con el bit inicial se indica cuando se va a transferir un dato y con el bit final se indica cuando termina esa transferencia. Cada transmisión de un bit perdura 1/(Baud Rate) segundos, donde el Baud Rate es la velocidad de transmisión en bits por segundo, la cual se configura mediante el registro BAUDCTL.

Primero se transmite y recibe el bit menos significativo (LSb) de los ocho bits que conforman un dato. El transmisor y el receptor del módulo trabajan independientemente pero comparten el mismo formato de datos y frecuencia de transmisión de datos (Baud Rate).

Para aclararlo vease la figura 7.6, en la que se simula la transmisión de un carácter a través del módulo. Si se configura la frecuencia de transmisión a 9600 bits/segundo, el período resultante es el de la ecuación 7.2. Es decir, cada 104 microsegundos se transmite un bit.

$$T = \frac{1}{f} = \frac{1}{9600\ \frac{bits}{segundo}} \simeq 104\ \frac{\mu s}{bit} \qquad \text{(ec. 7.2)}$$

La EUSART se alimenta a la misma tensión que el PIC, 5V. Así pues, el valor lógico alto o bit ‘1’ corresponde a un voltaje de 5V y el valor lógico bajo o bit ‘0’ a 0V. Supóngase ahora el

estado inicial de la EUSART en 5V. Para marcar el inicio de transmisión de un dato (en la figura 7.6 se transmite el byte 00011111, de bit más significativo a menos significativo) se transmite primero el bit de arranque. Al cabo de 104 microsegundos se transmite el bit menos significativo del dato (LSb) y otros 104 microsegundos más tarde se transfiere el siguiente. Así sucesivamente hasta haber transmitido el bit más significativo (MSb). Finalmente, se envía el bit de parada para indicar el fin de la transmisión del dato.

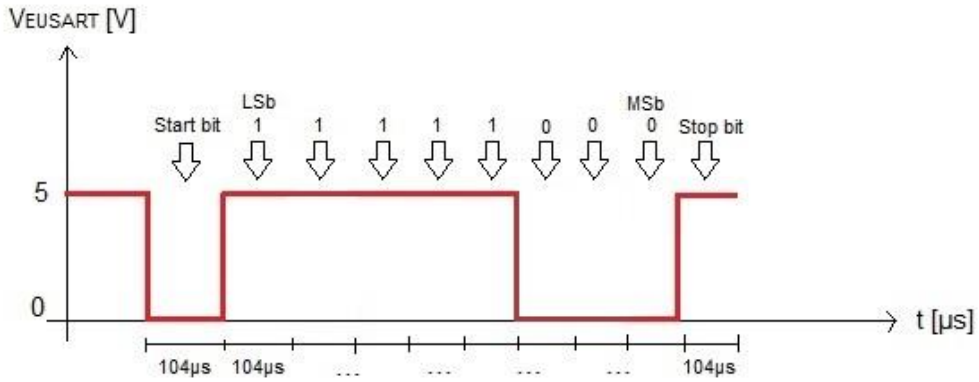


Figura 7.6 – *Formato de transmisión de datos del módulo EUSART*

Para comprobar que realmente se realiza así la transmisión de datos, se ha visualizado la señal del pin transmisor TX del microcontrolador a través de un osciloscopio (figura 7.7). Como se puede apreciar, cada recuadro del eje de abscisas del osciloscopio corresponde a un intervalo de 250 µs, por lo que en cada uno de ellos aparecen algo más de dos bits. El Start bit es el primer bit de valor bajo que se ve en la pantalla, seguido de cinco bits con valor alto y otros tres bits de valor bajo. Finalmente aparece el bit de parada, que siempre tiene valor '1'.

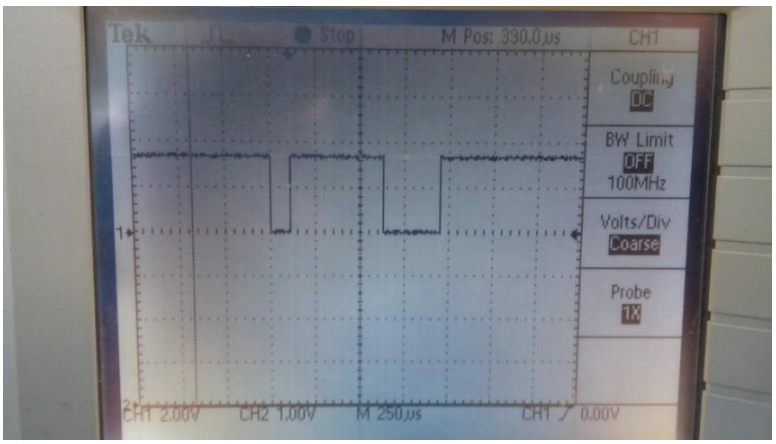


Figura 7.7 – *Visualización de la señal TX en el osciloscopio*

7.1.2. Velocidad de transmisión de datos

La EUSART también se encarga de generar la velocidad de transmisión con la que se transfieren los datos entre el microcontrolador y el resto de dispositivos conectados.

El generador de velocidad de transmisión (BRG) **[9]** es un temporizador de 8 bits o 16 bits utilizado para el correcto funcionamiento del módulo EUSART, tanto en modo asíncrono como síncrono. El BRG es el encargado de sincronizar la comunicación entre el microcontrolador y los dispostivos conectados a él.

Los registros asociados a este generador son los de la tabla 7.2 **[9]**, donde también se indican los ocho bits correspondientes a cada uno de ellos.

TABLE 12-4: REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
BAUDCTL	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	01-0 0-00	01-0 0-00
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0	0000 0000	0000 0000
SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8	0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010

Legend: x = unknown, – = unimplemented read as '0'. Shaded cells are not used for the Baud Rate Generator.

Tabla 7.2 – *Registros asociados al generador de velocidad de transmisión*

Los registros SPBRGH y SPBRG determinan el período de velocidad de este temporizador. En modo asíncrono se configura a través del bit BRGH del registro TXSTA y el bit BRG16 del registro BAUDCTL.

En la documentación del BRG están tabulados los ratios típicos y su error para varios modos asíncronos. Es conveniente escoger un ratio alto (imponiendo BRGH = 1) con el fin de reducir el error. En el caso del PIC16F690, cuyo oscilador de frecuencia es de 4 MHz, si se activa el generador de velocidad de transmisión en modo 8-bit (envía y recibe datos de ocho bits de longitud) y se deshabilita el bit BRG16 (BRG16 = 0) resultan los valores de ratio que se observan en la tabla 7.3 **[9]**.

TABLE 12-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)
300	—	—	—	—	—	—	—	—	—	300	0.16	207
1200	1202	0.16	207	1200	0.00	191	1202	0.16	103	1202	0.16	51
2400	2404	0.16	103	2400	0.00	95	2404	0.16	51	2404	0.16	25
9600	9615	0.16	25	9600	0.00	23	9615	0.16	12	—	—	—
10417	10417	0.00	23	10473	0.53	21	10417	0.00	11	10417	0.00	5
19.2k	19.23k	0.16	12	19.2k	0.00	11	—	—	—	—	—	—
57.6k	—	—	—	57.60k	0.00	3	—	—	—	—	—	—
115.2k	—	—	—	115.2k	0.00	1	—	—	—	—	—	—

Tabla 7.3 – *Ratios típicos de velocidad de transmisión en modo asíncrono*

El error es del 0.16 % para la mayoría de valores, así que se ha decidido escoger una velocidad de transmisión de 9600 bits/segundo. Tal y como se indica en la tabla anterior, es necesario escribir en el registro SPBRG el valor decimal 25, que corresponde al valor binario 00011001.

Así pues, el código de programación correspondiente a lo explicado en el apartado 7.1.2. se ha incluido en la función EUSART_Inicialización (figura 7.8), implementada mediante la instrucción void, que es utilizada para funciones que no retornan un valor. Esta función está definida en el fichero secundario “EUSART.h”, el cual está incluido en el fichero principal del programa “Main.c” a través de la instrucción #include “EUSART.h”.

EUSART.h | Main.c

Source | History

```
1
2   void EUSART_Inicializacion()
3   {
4     BAUDCTLbits.BRG16 = 0;    //Modo 8-bit activado
5     TXSTAbits.SYNC = 0;       //Modo asincrono activado
6     TXSTAbits.BRGH = 1;       //Velocidad de transmision alta
7     SPBRG = 25;               //Velocidad de 9600 bits/segundo
8   }
```

Figura 7.8 – *Definición de la función EUSART_Inicialización*

Para habilitar (darle valor lógico alto ‘1’) o deshabilitar (darle valor lógico bajo ‘0’) un bit de un registro se usa la instrucción de la ecuación 7.3, donde Y es el nombre del bit, X es el registro al que pertenece y Z es el valor lógico que se le quiere dar.

$$\mathrm{Xbits.Y} = \mathrm{Z} \qquad \text{(ec. 7.3)}$$

7.1.3. Recepción de datos

Una vez configurada la velocidad de transmisión de datos con la que trabaja el microcontrolador, hace falta configurar la recepción y transmisión de estos datos. En este apartado se explica detalladamente cómo el módulo EUSART recibe los datos de los dispositivos que se conectan al microprocesador y el código de programación que se ha implementado.

A través de la aplicación móvil Metal Detector mostrada en el apartado 5.1, el teléfono envía las señales de encendido o apagado y puesta en marcha o parada del detector de metales. Estas señales se transmiten desde el bluetooth del móvil al módulo bluetooth HC-05 conectado al microcontrolador a través de su pin receptor RX. Finalmente el HC-05 las transmite por su pin transmisor TX al pin receptor RX del PIC para que sean procesadas.

Los registros asociados al circuito receptor del módulo EUSART son los de la tabla 7.4 **[5]**.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
BAUDCTL	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	01-0 0-00	01-0 0-00
INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 000x	0000 000x
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
RCREG	EUSART Receive Data Register								0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0	0000 0000	0000 0000
SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8	0000 0000	0000 0000
TRISB	TRISB7	TRISB6	TRISB5	TRISB4					1111 ----	1111 ----
TXREG	EUSART Transmit Data Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010

Legend: x = unknown, – = unimplemented read as '0'. Shaded cells are not used for Asynchronous Transmission.

Tabla 7.4 – *Registros asociados al circuito receptor del módulo EUSART*

Y el diagrama de bloques correspondiente a la recepción de datos a través de este módulo es el de la figura 7.9.

Para habilitar el receptor de la EUSART en modo asíncrono se requiere la siguiente configuración de estos tres bits de control:

- **CREN = 1**. Así se activa el circuito receptor de la EUSART permitiendo la recepción continua de datos.
- **SYNC = 0**. Se configura la EUSART en modo asíncrono.
- **SPEN = 1**. Se configura el pin RX del microcontrolador como entrada de datos.

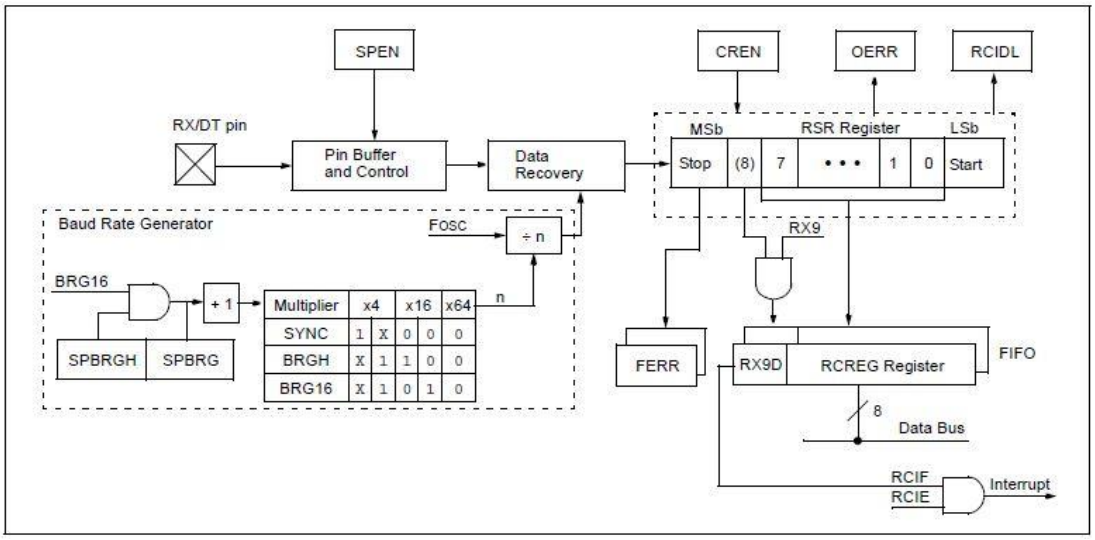


Figura 7.9 – *Diagrama de bloques de recepción de datos del módulo EUSART*

Como se puede ver en la figura 7.9, los datos se reciben a través del pin RX del microcontrolador y se transmiten al registro RSR (Receive Shift Register). Una vez se han transmitido todos los bits del carácter (Start bit + 8 bits + Stop bit), éstos se transfieren immediatamente a la memoria FIFO (First-In-First-Out), la cual permite almacenar hasta dos carácteres completos.Tanto la FIFO como el registro RSR no son accesibles directamente por software, de forma que el acceso a los datos recibidos se hace mediante la lectura del registro RCREG.

En cuanto el carácter se ha transferido a la FIFO, el bit RCIF del registro PIR1 se habilita. Para que un carácter salga de la FIFO es necesario leer el registro RCREG. Una vez se haya leido el carácter y no haya ninguno más pendiente por leer, el bit RCIF volverá a tomar un valor bajo ‘0’.

Como se observa en la misma figura, los bits RCIF y RCIE son las entradas binarias de una puerta lógica AND que tiene como salida una interrupción. Por lo tanto, para activar esta interrupción es necesario que ambas entradas tengan un valor lógico ‘1’. Generar una interrupción significa hacer una pausa en la ejecución del programa principal del microcontrolador para ejecutarse la función de interrupción implementada en este programa. Esta función hará que se lea el registro RCREG hasta que no haya ningun dato por leer en la FIFO, hecho que hará desactivar el bit RCIF y por tanto se finalizará la ejecución de la función de interrupción. Entonces el programa principal volverá a ejecutarse immediatamente por donde lo había dejado previamente. De esta manera, siempre que se reciban datos, se activará el bit RCIF y saltará la interrupción, de modo que los datos se leerán.

Para que suceda esta interrupción se requiere pues la activación del bit RCIE, además de los bits PEIE y GIE del registro INTCON.

En el caso de que la FIFO reciba por completo un tercer carácter antes de que haya sido leído el RCREG, se genera un error por sobresaturación que impide la recepción de más datos, pero sin impedir la lectura de los que ya están escritos en la FIFO. Cuando esto sucede se habilita el bit OERR del registro RCSTA. Para reanudar la recepción de datos se requiere eliminar el error mediante la limpieza del bit CREN del registro RCSTA (CREN=0).

El código de programación que se ha implementado para la recepción de datos con el módulo EUSART es el de las figuras 7.10 y 7.11. En la función EUSART_Inicialización del fichero "EUSART.h" (figura 7.10) se ha añadido la habilitación de los bits SPEN y CREN. También se ha creado una nueva función llamada EUSART_Lectura para leer el dato que se almacene en el registro RCREG. Esta función devuelve un valor de tipo char (8 bits), que corresponde con el byte que contiene el registro RCREG.

EUSART.h | Main.c

Source | History

```
1
2   void EUSART_Inicializacion()
3   {
4     BAUDCTLbits.BRG16 = 0;        //Modo 8-bit activado
5     TXSTAbits.SYNC = 0;           //Modo asincrono activado
6     TXSTAbits.BRGH = 1;           //Velocidad de transmision alta
7     SPBRG = 25;                   //Velocidad de 9600 bits/segundo
8     RCSTAbits.SPEN = 1;           //Se habilita el pin RX como entrada de datos
9     RCSTAbits.CREN = 1;           //Se activa la recepcion continua de datos
10  }
11
12  char EUSART_Lectura()      //Funcion para leer el dato almacenado en el RCREG
13  {
14      return RCREG;
15  }
```

Figura 7.10 - *Código de programación del fichero "EUSART.h"*

Por otro lado, en el fichero "Main.c" (figura 7.11) se incluye la función principal del programa llamada main, que es la que se ejecuta al alimentar el microcontrolador. En ella aparecen las isntrucciones ANSEL=0 y ANSELH=0, necesarias para configurar todos los pines del PIC como entradas o salidas digitales, puesto que por ellos se envian y reciben datos en forma de bytes. Incluso cuando se conecte el sensor de campo magnético, éste también enviará bytes al microcontrolador. El sensor capta variaciones de tensión debido a la variación del campo magnético por la presencia de elementos metálicos. Estas variaciones de voltaje se procesan y

se convierten en bytes, que posteriormente se envían al PIC. Más adelante se hablará en detalle del sensor y su configuración.

También se han habilitado los pines del puerto C como salidas mediante la instrucción TRISC=0 (así se escribe un '0' en todos los bits del puerto C) y se han activado los bits PEIE, GIE y RCIE para habilitar las interrupciones. Estas interrupciones corresponden a las instrucciones de la función Interrupción. Por un lado, cuando se active el bit RCIF (lo cual sucede si se ha recibido un dato, se ha almacenado en la FIFO y aun no se ha leído), entonces se leerá el dato con la función EUSART_Lectura y además se escribirá en el puerto C. Por otro lado, si hay un error de sobresaturación de la FIFO (activación del bit OERR), se deshabilita la recepción de datos (CREN=0) para limpiar el error y seguidamente se rehabilita la recepción continua de datos (CREN=1).

EUSART.h | Main.c

Source | History

```
#include <xc.h>
#include <pic16f690.h>
#include "EUSART.h"

//Inicio de la configuracion de bits del PIC
#pragma config FOSC = INTRCIO
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = OFF
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = OFF
#pragma config IESO = OFF
#pragma config FCMEN = OFF
//Fin de la configuracion de bits

#define _XTAL_FREQ 4000000

void main()
{
  ANSEL = 0;                //Configuracion de todos los pines del PIC como entradas o salidas digitales
  ANSELH = 0;
  TRISC = 0;                //Los pines del puerto PORTC se configuran como salidas de datos
  EUSART_Inicializacion();  //Se inicializa la EUSART con velocidad de transmision de 9600 bits/segundo
  INTCONbits.PEIE = 1;      //Se activa la interrupcion de perifericos
  INTCONbits.GIE = 1;       //Se activan las interrupciones globales
  PIE1bits.RCIE = 1;        //Se activan las interrupciones en la recepcion de datos
}

void interrupt Interrupcion()      //Funcion que atiende las interrupciones
{
    if(PIR1bits.RCIF)              //Si hay algun dato en la FIFO pendiente por leer (RCIF=1)
    {
        PORTC = EUSART_Lectura();  //Se lee el dato y se escribe en el puerto C
    }

    if(RCSTAbits.OERR)             //Si se activa el bit de error OERR
    {
        RCSTAbits.CREN = 0;       //Se limpia el bit CREN para limpiar el error
        RCSTAbits.CREN = 1;       //Se rehabilita la recepcion continua
    }
}
```

Figura 7.11 – *Código de programación del fichero "Main.c"*

El siguiente paso es diseñar una aplicación móvil de prueba para transmitir bytes del teléfono al PIC y visualizarlos a través de unos LEDs acoplados al microcontrolador. De esta manera se sabrá si el código anterior es correcto.

Puesto que al enviar un byte al microcontrolador, éste se escribe o almacena en los ocho bits del puerto C mediante la instrucción PORTC = EUSART_Lectura() (figura 7.11), se conectarán ocho LEDs a los ocho pines del puerto C, que han sido configurados como salida de datos. De modo que si uno de estos bits tiene valor '1' el LED se encenderá y si tiene valor '0' se mantendrá apagado. Para limitar la corriente que circula por los LEDs se debe conectar en serie una resistencia de valor apropiado. En este caso, se han colocado resistencias de 220 Ω. Si se alimentan a la tensión del microcontrolador (5V) la intensidad resultante que circula por ellos, según la ley de Ohm, es de 23 mA (ec. 7.4), apto para su correcto funcionamiento.

$$I = \frac{V}{R} = \frac{5V}{220\Omega} = 0.023\ A = 23\ mA \qquad \text{(ec. 7.4)}$$

Esta aplicación de prueba, de similar aspecto que la versión final vista en el apartado 5.1, incluye como componentes un selector de lista (Connect to bluetooth device), cuatro botones (Disconnect bluetooth device, Start, Stop y Close) y una etiqueta no visible en la figura 7.12.

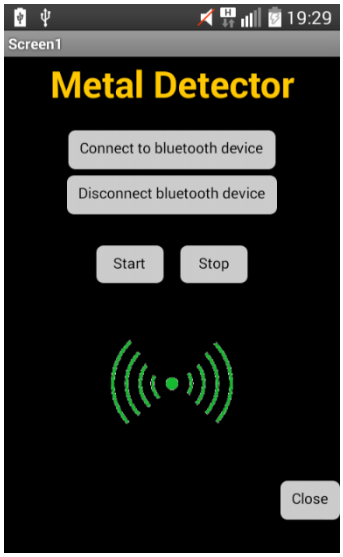


Figura 7.12 – *Aplicación móvil para transmitir datos del teléfono al microcontrolador.*

La función de "Connect to bluetooth device" y "Disconnect bluetooth device" es la misma que en la versión final del punto 5.1, port lo tanto, su código de programación en App Inventor 2 mediante el sistema de bloques también lo es. El selector de lista permite conectarse a los dispositivos bluetooth detectados por el teléfono y el botón sirve para desconectarse.

En cambio, la utilidad de los botones “Start” y “Stop” ahora es distinta. Como se observa en el código de programación de la figura 7.13, cuando se haga clic en el botón “Start” y el móvil esté conectado al módulo bluetooth incorporado al PIC, se enviará el byte correspondiente al valor decimal 13, que es el 00001101. Si se pulsa el botón “Stop”, se enviará el byte correspondiente al valor decimal 242, que es el 11110010.

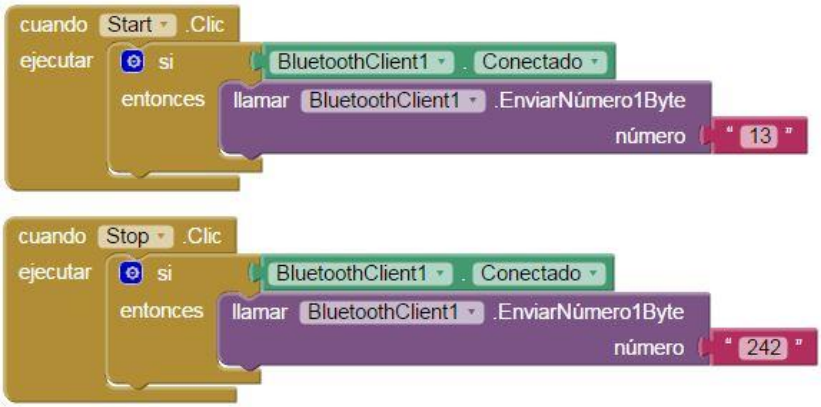


Figura 7.13 – *Código de programación de los botones Start y Stop*

Una vez enviado cualquiera de estos dos valores al módulo bluetooth HC-05, éste lo transmite a través de su pin transmisor TX al pin receptor RX del PIC. Finalmente, el PIC escribe el byte en su puerto C.

El circuito construido es el de la figura 7.14, donde el LED de más a la izquierda corresponde al bit más significativo del byte recibido y el de más a la derecha corrresponde al bit menos significativo.

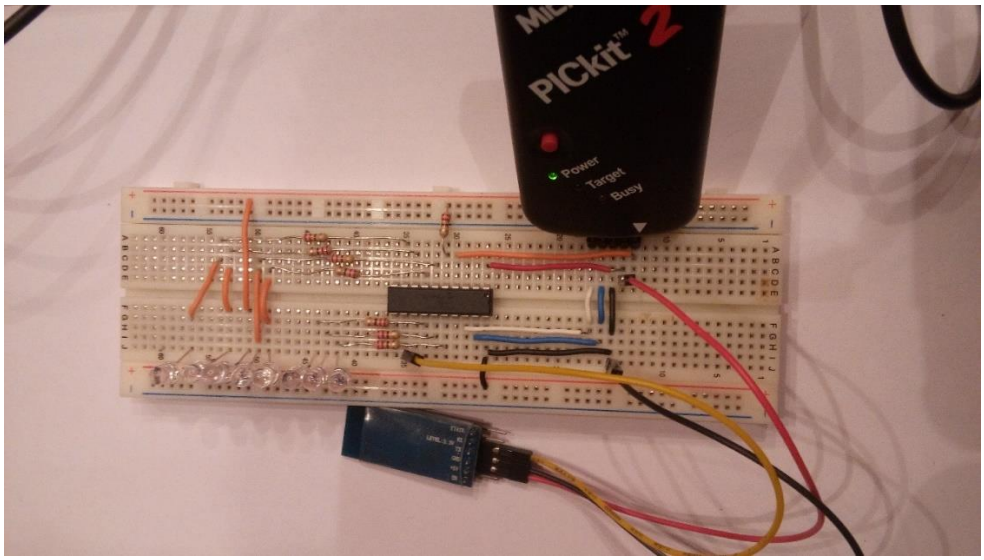


Figura 7.14 – *Prototipo para enviar un byte del teléfono al PIC y visualizarlo por los LEDs*

Como se puede apreciar, el pin transmisor TX del módulo bluetooth HC-05 se ha conectado al pin receptor RX del microcontrolador (cable amarillo), además de alimentar el periférico a la misma tensión que el PIC (cables rojo y negro).

Una vez conectado el teléfono móvil al HC-05, al pulsar el botón “Start” se envía el valor binario 00001101 (de bit más significativo a menos siginificativo). Los bits de valor ‘1’ harán que se encienda su correspondiente LED (figura 7.15).

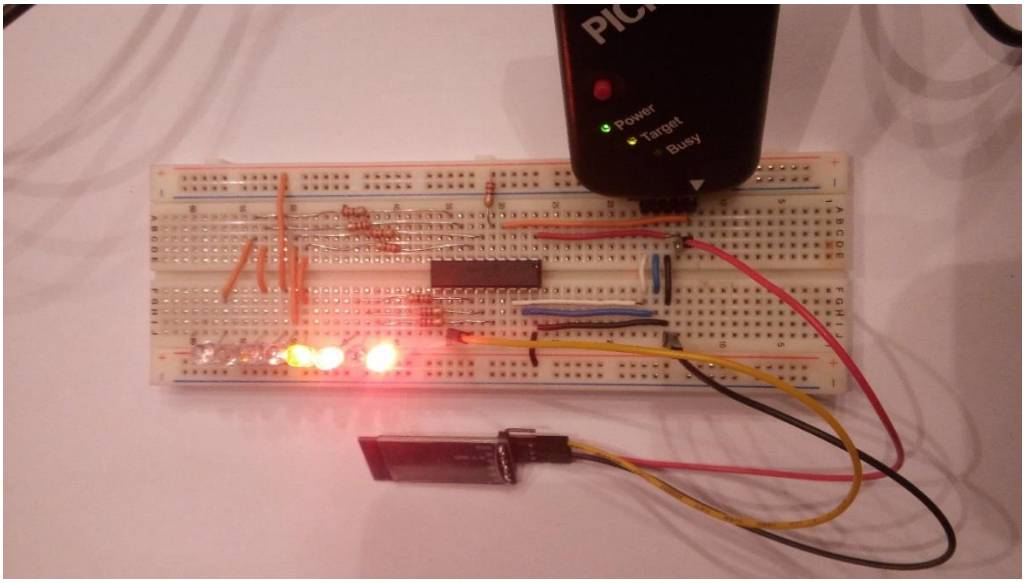


Figura 7.15 –*Visualización del byte con valor decimal 13 (00001101) por los LEDs*

Y al presionar el botón “Stop” se envía el valor binario 11110010 (figura 7.16).

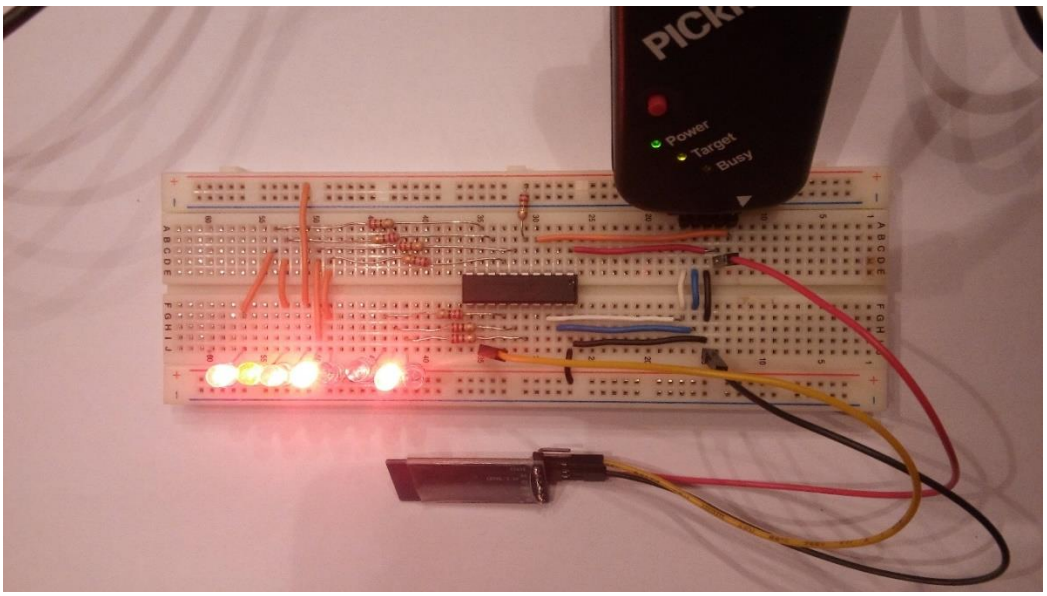


Figura 7.16 –*Visualización del byte con valor decimal 242 (*11110010*) por los LEDs*

7.1.4. Transmisión de datos

El último aspecto a tener en cuenta del módulo EUSART es cómo realiza la transmisión de datos. El microcontrolador recibirá datos del sensor de campo magnético y los procesará. Seguidamente enviará una señal a través del módulo bluetooth HC-05 al dispositivo móvil, que monitorizará el estado del detector. Esta señal se trata de un byte determinado.

El PIC envía los datos a través de su pin transmisor TX (figura 7.17).

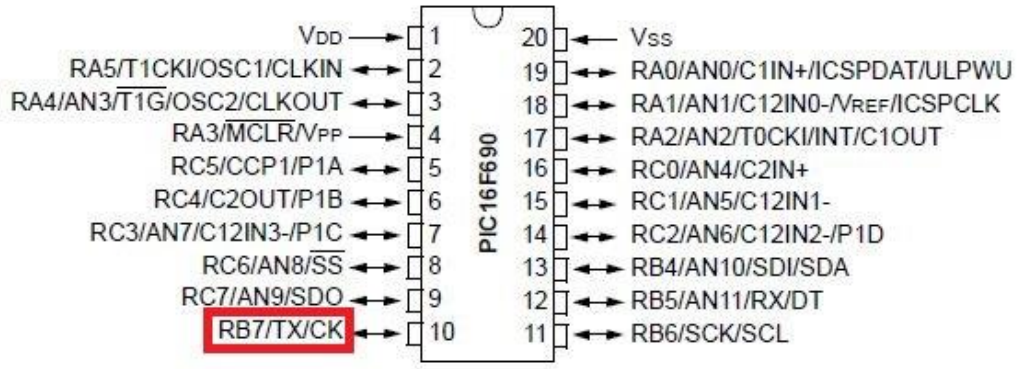


Figura 7.17 – *Pin TX del PIC16F690*

Este pin se conecta al pin receptor RX del módulo bluetooth (figura 7.18), ya que es el microchip quien transmite un dato que sale por su pin TX y llega al pin RX del bluetooth. Además es necesario alimentar el bluetooth a la misma tensión que el microcontrolador (figura 7.19).

Figura 7.18 – *Módulo bluetooth HC-05*

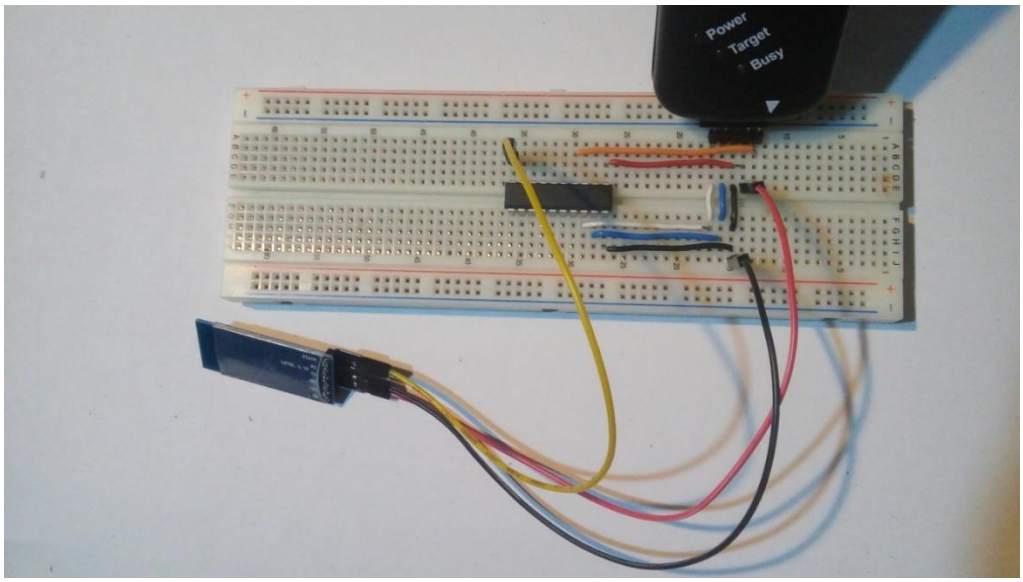


Figura 7.19 – *Conexión entre el microcontrolador y el módulo bluetooth para la transmisión de datos del módulo EUSART*

Los registros asociados a la transmisión de datos con el módulo EUSART son los indicados en la tabla 7.5 **[5]**, donde se indican también los ocho bits correrspondientes a cada registro.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
BAUDCTL	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	01-0 0-00	01-0 0-00
INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 000x	0000 000x
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
RCREG	EUSART Receive Data Register								0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0	0000 0000	0000 0000
SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8	0000 0000	0000 0000
TRISB	TRISB7	TRISB6	TRISB5	TRISB4					1111 ----	1111 ----
TXREG	EUSART Transmit Data Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010

Legend: x = unknown, – = unimplemented read as '0'. Shaded cells are not used for Asynchronous Transmission.

Tabla 7.5 – *Registros asociados a la transmisión de datos con el módulo EUSART*

El circuito transmisor se habilita para operaciones en modo asíncrono configruando los siguientes bits:

- **TXEN = 1**. Así se habilita el circuito transmisor de la EUSART.
- **SYNC = 0**. Se configura la EUSART en modo asíncrono.

- **SPEN = 1**. Se configura el pin TX del PIC como salida. Los datos salen del chip hacia el módulo bluetooth.

La transmisión de datos mediante el módulo EUSART se indica de forma esquemática en el diagrama de la figura 7.20 **[5]**.

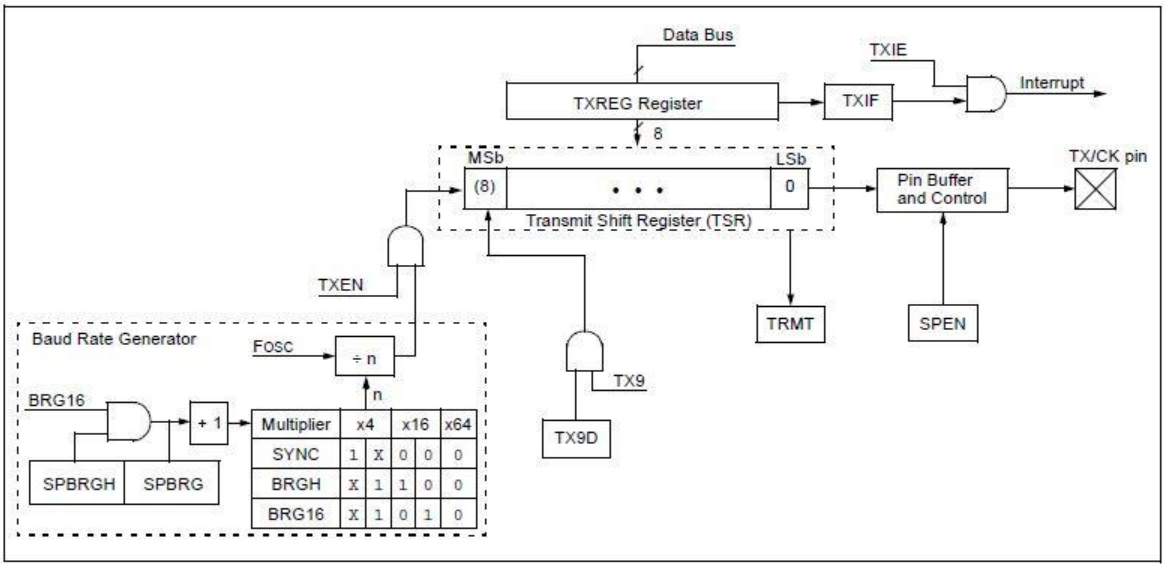


Figura 7.20 - *Diagrama de bloques de transmisión de datos del módulo EUSART*

El corazón del transmisor de la EUSART es el registro TSR (Transmit Shift Register), que no es directamente accesible por software. El TSR obtiene del registro TXREG el dato que debe transmitir por el pin TX.

La transmisión se inicia escribiendo un carácter (Start bit + 8 bits + Stop bit) en el registro TXREG. Si éste es el primer carácter o el anterior ya ha salido del TSR (es decir ya se ha transmitido por TX), se transfiere immediatamente al registro TSR. En caso de que el TSR esté ocupado transmitiendo otro dato previo, el TXREG almacena el caracter hasta que vuelva a estar libre, es decir hasta que haya transmitido el Stop bit de ese carácter anterior.

El bit TXIF del registro PIR1 es un bit de lectura, no se puede imponer su valor y se habilita (valor alto ‘1’) cuando no hay ningún caracter almacenado en TXREG esperando a ser transmitido al TSR. Como se observa en la figura 7.20, hay una puerta lógica AND con entradas los bits TXIF y TXIE y como salida una interrupción. Es posible activar esta interrupción habiendo habilitado previamente el bit TXIE del registro PIE1. En todo caso, para la transmisión de datos no se ha requerido el uso de esta interrupción.

Por otro lado está el bit de lectura TRMT, que indica el estado del registro TSR. Si TMRT = 1 entonces TSR está vacío y si TMRT = 0, se está transfieriendo un dato del TXREG al TSR.

TRMT permanece en ‘0’ hasta que todos los bits del carácter han sido transmitidos del TSR al pin TX.

Para poner en práctica la teoria, se ha implementado un código que transmite datos del PIC al módulo bluetooth y éste envia una serie de bytes al dispositivo móvil.

Primero vease el fichero “EUSART.h” (figura 7.21), donde se ha añadido a la función ya existente EUSART_Inicialización la instrucción TXSTAbits.TXEN = 1 para habilitar el circuito transmisor de la EUSART.

EUSART.h | Main.c

Source | History

```
1
2   void EUSART_Inicializacion()
3   {
4     BAUDCTLbits.BRG16 = 0;        //Modo 8-bit activado
5     TXSTAbits.SYNC = 0;           //Modo asincrono activado
6     TXSTAbits.BRGH = 1;           //Velocidad de transmision alta
7     SPBRG = 25;                   //Velocidad de 9600 bits/segundo
8     RCSTAbits.SPEN = 1;           //Se habilita el pin TX como salida y el pin RX como entrada
9     RCSTAbits.CREN = 1;           //Se activa la recepcion continua
10    TXSTAbits.TXEN = 1;           //Se activa el circuito transmisor
11  }
12
13  void EUSART_Escritura(char dato)  //Funcion que escribe un byte en el registro TXREG
14  {
15      if(PIR1bits.TXIF)             //Si no hay ningun dato almacenado en TXREG
16      {
17         TXREG = dato;             //Se escribe el byte "dato" en TXREG
18      }
19  }
```

Figura 7.21 – *Código de programación del fichero “EUSART.h”*

Seguidamente se ha definido la función EUSART_Escritura, que escribe un byte en el registro TXREG siempre que no haya ninguno almacenado en él, es decir, cuando TXIF=1.

En el fichero “Main.c” (figura 7.22) se ha añadido a la función principal main un bucle infinito mediante la instrucción while(1). El bucle consiste en escribir consecutivamente los bytes correspondiente a los valores decimales 10, 20, 30, 40, 50, 60 y 70 en intervalos de un segundo. Estos valores se mandan a través del pin TX del microcontrolador al módulo bluetooth y éste los transmite al teléfono móvil, por el cual se irán visualizando a medida que los reciba.

EUSART.h | Main.c

Source | History

```
#include <xc.h>
#include <pic16f690.h>
#include "EUSART.h"

//Inicio de la configuracion de bits del PIC
#pragma config FOSC = INTRCIO
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = OFF
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = OFF
#pragma config IESO = OFF
#pragma config FCMEN = OFF
//Fin de la configuracion de bits

#define _XTAL_FREQ 4000000

void main()
{
  ANSEL = 0;                //Configuracion de todos los pines del PIC como entradas o salidas digitales
  ANSELH = 0;
  TRISC = 0;                //Los pines del puerto PORTC se configuran como salidas de datos
  EUSART_Inicializacion();  //Se inicializa la EUSART con velocidad de transmision de 9600 bits/segundo
  INTCONbits.PEIE = 1;      //Se activa la interrupcion de perifericos
  INTCONbits.GIE = 1;       //Se activan las interrupciones globales
  PIE1bits.RCIE = 1;        //Se activan las interrupciones en la recepcion de datos

  while(1)
  {
      EUSART_Escritura(10);
      __delay_ms(1000);
      EUSART_Escritura(20);
      __delay_ms(1000);
      EUSART_Escritura(30);
      __delay_ms(1000);
      EUSART_Escritura(40);
      __delay_ms(1000);
      EUSART_Escritura(50);
      __delay_ms(1000);
      EUSART_Escritura(60);
      __delay_ms(1000);
      EUSART_Escritura(70);
      __delay_ms(1000);
  }
}
```

Figura 7.22 – *Código de programación del fichero "Main.c"*

Una vez conformado el código de programación para el microcontrolador, se ha realizado la aplicación móvil que muestra estos valores en la pantalla (figura 7.23). Ésta contiene el botón Conectar para conectarse a otros dispositivos bluetooth y dos etiquetas: una con contorno azul que muestra si se ha conectado y otra con contorno rojo para mostrar el dato que recibe.

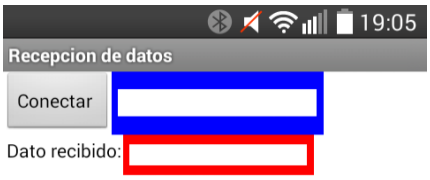


Figura 7.23 – *Aplicación móvil para recibir datos del PIC*

Puesto que el microcontrolador va enviando los datos a intervalos de un segundo de forma indefinida, la aplicación móvil debe incluir un reloj que haga repetir la acción de recibir un dato y mostrarlo por la etiqueta de contorno rojo (figura 7.24). El temporizador del reloj debe estar programado a menos de un segundo, con el fin de poder mostrar todos los datos que se envían desde el PIC. En este caso se ha configurado a medio segundo.

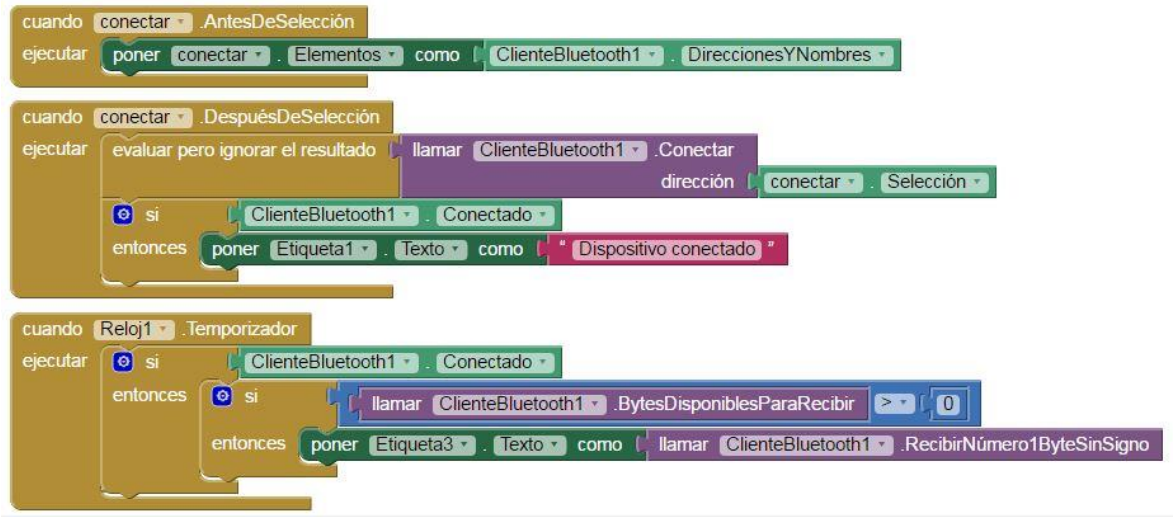


Figura 7.24 – *Código de programación de la aplicación móvil*

Una vez conectado el teléfono al módulo bluetooth, se irá mostrando cada segundo uno de los valores que se envían desde el microchip. En la figura 7.25 se muestra el valor 10 y un segundo más tarde aparece el valor 20.

Figura 7.25 – *Visualización del valor recibido 10 y un segundo más tarde el valor 20*

8. El sensor de campo magnético HMC5883L

El sensor de campo magnético escogido para diseñar y construir el detector de metales es el HMC5883L **[10]**. Éste detecta las variaciones de campo magnético de la Tierra y manda la señal correspondiente al microcontrolador. El sensor tiene cinco pines que permiten conectarlo al microchip (figura 8.1).

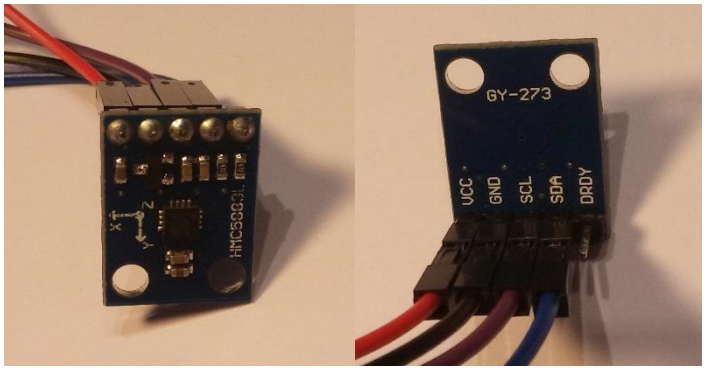


Figura 8.1 – *Sensor de campo magnético HMC5883L*

El HMC5883L utiliza magnetoresistores para detectar las variaciones de campo magnético, uno por cada eje (X, Y y Z). Esta tecnología ofrece ventajas que otros sensores magnéticos no tienen, ya que la anisotropía de sus sensores implica precisión y linealidad para cada eje. Su diseño permite medir tanto la dirección como la magnitud de los campos magnéticos de la Tierra. Este sensor de la compañía Honeywell se encuentra entre los más sensibles y fiables de la industria de sensores de campo magnético de bajo coste.

Con una tensión de alimentación, el sensor convierte los campos magnéticos incidentes en las tres direcciones en una diferencia de potencial de salida. Esta diferencia es debida al cambio de valor de resistencia de los magnetoresistores (efecto magnetoresistor). Este voltaje se convierte en una señal digital mediante un convertidor analógico-digital y se almacena en forma de byte en los registros de almacenamiento del sensor. Los cambios de voltaje son positivos cuando el campo magnético incrementa en la dirección de las flechas indicadas para cada eje (ver en la figura 8.1), por lo tanto, el valor decimal del byte también sera creciente. Puesto que el voltaje de salida es directamente proporcional a la componente del campo magnético a lo largo del eje, el dispositivo incluye otros sensores en direcciones ortogonales que dan más precisión en la medida del campo magnético en cualquier dirección.

El control de este sensor se realiza a través del bus I^2C **[11]**, que se explicará con detalle en el siguiente apartado. El HMC5883L se conecta a este bus como esclavo bajo el control de un maestro, que será el microcontrolador.

Los registros de memoria del sensor se indican en la tabla 8.1 **[10]** junto a su dirección en valor decimal y si son de lectura, escritura o ambas. Todos ellos son registros de ocho bits, es decir, almacenan un byte.

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

Tabla 8.1 – *Registros del sensor HMC5883L*

Al aplicarse un campo magnético, el sensor toma las medidas y las almacena en los seis registros de salida nombrados Data Output Register. Se graban dos bytes por eje, uno alto (Data Output X MSB Register para el eje X) y otro bajo (Data Output X LSB Register para el eje X). Estos seis registros son únicamente de lectura y el microcontrolador será quien los lea.

Hay dos registros de configuración del sensor: Configuration Register A y Configuration Register B. El primero se utiliza para configurar el número de muestras promediadas por medición, el ratio al cual se escriben los datos medidos en el conjunto de registros de salida y el modo de medición. Estos datos están tabulados junto con su correspondiente configuración de bits en la documentación del sensor magnético HMC5883L **[10]**. Para el detector de metales, se han escogido ocho muestras promediadas por medición, un ratio de salida de 15 Hz y el modo normal de medición. Así pues, el byte resultante a escribir en este registro es el 0x70 en hexadecimal.

Se ha escogido un número de muestras promediadas por medición de ocho ya que, como se ve en la figura 8.2 **[10]**, a mayor número de muestras, menor desviación estandar en las mediciones y ,por lo tanto, menor dispersión en los valores medidos.

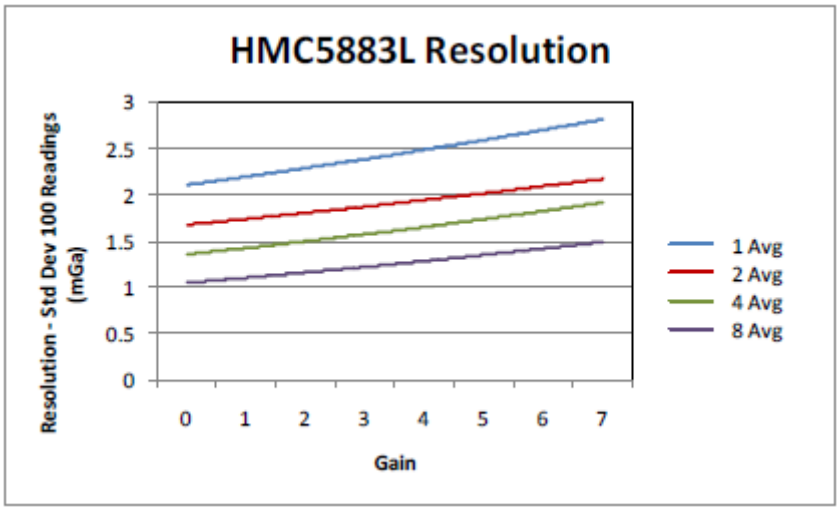


Figura 8.2 – *Resolución según las muestras promediadas por medición y la ganancia*

El registro de configuración B sirve para establecer la ganancia del dispositivo. Las ganancias nominales y su configuración de bits correspondiente se encuentran tabuladas en la documentación del sensor **[10]**.

Con el fin de evitar saturaciones en alguno de los registros de salida es conveniente escoger un valor de ganancia baja. Basándose en el gráfico Ganancia-Resolución de la figura 8.3 y habiendo escogido previamente un total de ocho muestras promediadas por medición, se ha decidido escoger la ganancia número 5, con un valor de 390 LSb/Gauss.

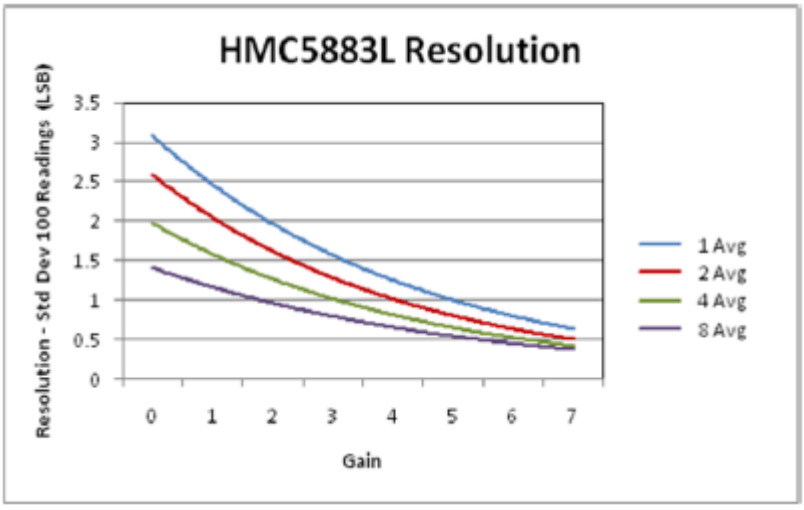


Figura 8.3– *Resolución según las muestras promediadas por medición y la ganancia*

En el registro Mode register se puede escribir o leer un dato. Este registro es usado para seleccionar el modo operacional del sensor. Según la configuración de los ocho bits que

contiene este registro **[10]**, se habilitará el modo de medición contínua o el modo único de medición o el modo de parada. Con el primero de ellos, el dispositivo toma medidas constantemente y las escribe en los registros de salida. Con el modo único de medición se realiza una sola medición y a continuación se pone en modo de parada. Las medidas tomadas permanecen en los registros de salida para que puedan ser leídas. Y en el modo de parada, como indica su nombre, el dispositivo deja de realizar mediciones. Para implementar el detector de metales se usará el modo de medición contínua.

El resto de registros son irrelevantes para implementar el código de programación del detector de metales.

Hay que tener en cuenta que después de haber inicializado el sensor, se le debe otorgar un tiempo de 6 ms para que tome las medidas. Una vez transcurrido este tiempo, el microcontrolador puede comenzar con la lectura de los registros de salida del sensor (los Data Output Register), donde se almacenan estas medidas.

8.1. Protocolo de comunicación I^2C

Como se ha comentado anteriormente, el control del sensor HMC5883L se realiza mediante el bus I^2C. El bus I^2C (Inter-Integrated-Circuit). es un bus de datos en serie que permite la comunicación entre distintas partes de un circuito, como por ejemplo en este caso, entre el microcontrolador y el sensor HMC5883L.

El I^2C está diseñado como un bus maestro-esclavo, donde uno de los dispositivos hace de maestro y el resto de esclavos. En el caso del detector de metales, el microcontrolador es el maestro y el sensor es el único esclavo. Este bus sólo necesita el uso de dos señales para sincronizar los distintos dispositivos (además de la señal de tierra o masa). Estas dos señales son la SDA (serial data o señal de datos) y la SCL (serial clock o señal reloj).

Mediante la SDA se transmiten los datos entre los dispositivos y con la señal de reloj se sincronizan.

Para entender mejor la comunicación entre estos dos dispositivos electrónicos observese la figura 8.4. Ambos se alimentan a la misma tensión y la transferencia de datos se realiza a través de la señal SDA. Para sincronizar los dos circuitos se utiliza la señal de reloj SCL.

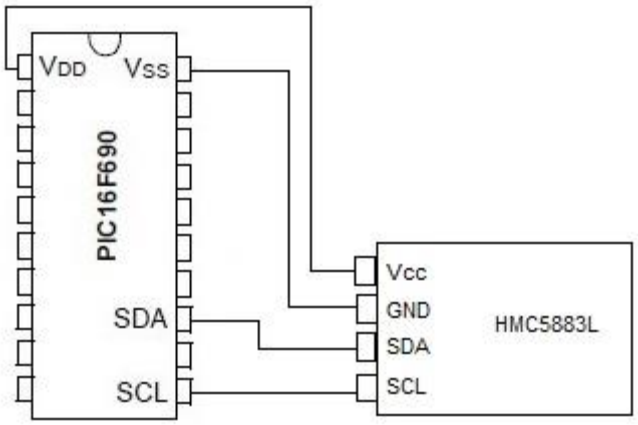


Figura 784 – *Conexión entre el microcontrolador y el sensor*

Los pines SCL y SDA del microcontrolador corresponden a los pines número 11 y 13 respectivamente, que equivalen a los pines RB6 y RB4. En la figura 8.5 se ha esquematizado la circuitería interna del PIC realtiva a estos pines.

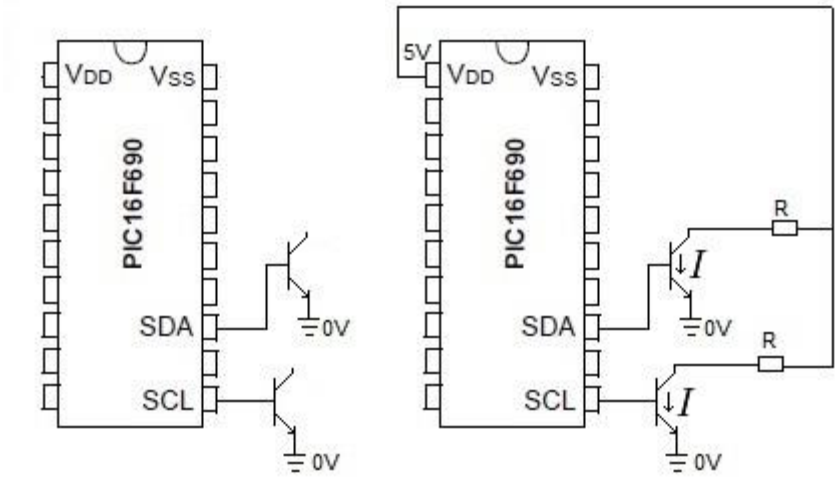


Figura 8.5 – *Pines SDA y SCL del microcontrolador y su conexión*

Los pines SDA y SCL del PIC son de tipo colector abierto **[12]**, es decir, formados por transistores abiertos por arriba. Las señales SDA y SCL son señales binarias, o toman un valor lógico ‘1’ o un valor lógico ‘0’, que concuerda con que los pines tengan una tensión alta (próxima a los 5V) o una tensión baja (0V) respectivamente. Para que los pines tomen este valor alto es necesario conectar una resistencia entre el pin y la alimentación de 5V, de forma que cuando no circule corriente por el transistor (I=0), la tensión del pin sea casi de 5V (no exactamente 5V debido a la caída de tensión en la resistencia) o lo que es lo mismo, en alta impedancia. Por contra, si circula corriente, el pin se encontrará a una tensión nula. Estas resistencias ya están incluidas en el circuito integrado del sensor.

Para que los pines tomen un valor lógico ‘1’ será necesario configurarlos como entradas. Para ello habrá que configurar los bits correspondientes al registro al que pertenecen, que es el registro TRISB. El pin SDA se configura mediante el bit TRISB4 de ese registro y el SCL con el bit TRISB6. La instrucción a aplicar en el código es TRISB4=1 y TRISB6=1.

En cambio, para que tomen un valor ‘0’ habrá que configurarlos como salidas y darles un valor bajo: TRISB4=0 y a continuación RB4. Lo mismo para el RB6.

Él código de programación referente a estos pines, se ha escrito en el fichero secundario llamado “i2c.h”, donde se incluyen todas las funciones que definen el protocolo de comunicación I^2C. Antes, pero, en el fichero principal “Main.c” (figura 8.6) se han definido las variables SDA_PIN, SCL_PIN, SDA_DIR y SCL_DIR para hacer referencia a los pines y bits de registro de las señales SDA y SCL.

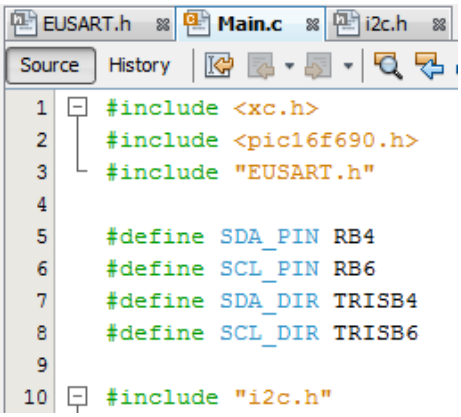


Figura 8.6 – *Fichero “Main.c”*

Como se puede ver en el fichero “i2c.h” (figura 8.7), las funciones i2c_SDA_alto, i2c_SCL_alto, i2c_SDA_bajo e i2c_SCL_bajo se utilizan para que las señales SDA y SCL tengan un valor lógico ‘1’ o un valor lógico ‘0’.

El maestro (el PIC) es el que crea la señal de reloj SCL para sincronizarse con el esclavo (sensor). Mientras no haya comunicación entre los dos dispositivos, las señales SDA y SCL se mantienen en valor alto.

EUSART.h | Main.c | i2c.h

Source | History

```
1   // Funciones que definen el protocolo de comunicacion del bus I2C:
2
3   void i2c_SDA_alto()
4   {
5      SDA_DIR = 1;       //Se configura el pin RB4 o SDA como entrada digital.
6   } //Los pines SDA y SCL se encuentran en ALTA IMPEDANCIA (1) (debido a las dos resistencias situadas entre
7    //estos pines y Vdd, ya incluidas en el magnetometro) cuado no se establece comunicacion a traves del bus i2c
8
9   void i2c_SCL_alto()
10  {
11     SCL_DIR = 1;       //Pin RB6 o SCL como entrada digital. Pin SCL en alta impedancia(1)
12  }
13
14  void i2c_SDA_bajo()
15  {
16     SDA_DIR = 0;      //Pin SDA como salida
17     SDA_PIN = 0;      //Pin SDA en estado logico 0
18  }
19
20  void i2c_SCL_bajo()
21  {
22      SCL_DIR = 0;     //Pin SCL como salida
23      SCL_PIN = 0;     //Pin SCL en estado logico 0
24  }
```

Figura 8.7 – *Fichero "i2c.h" donde se define el protocolo de comunicación I^2C*

El maestro inicia una transmisión enviando un bit de arranque a través de la señal SDA, que consiste en que SDA cambie a valor lógico '0' mientras la señal SCL valga '1' (figura 8.9).

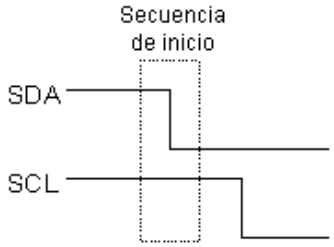


Figura 8.9 – *Secuencia de inicio*

Esta secuencia de inicio se ha implementado en MPLAB mediante la función i2c_inicio (figura 8.10).

```
26  void i2c_inicio()   //Funcion para iniciar la transferencia de datos
27  {
28     i2c_SDA_alto();
29     i2c_SCL_alto();
30     i2c_SDA_bajo();  //Condicion de inicio de transmision: SDA cambia a 0 cuando SCL = 1
31     i2c_SCL_bajo();
32  }
```

Figura 8.10 – *Función i2c_inicio del fichero "i2c.h"*

A continuación, el maestro debe enviar siete bits que corresponden a la dirección del esclavo al cual se dirige, es decir, aquel esclavo con el que quiere establecer comunicación (en este caso sólo hay uno, el sensor), seguido de un octavo bit que indica el tipo de operación que quiere realizar. Si este octavo bit vale '0' se trata de una operación de escritura en la que el maestro escribe un dato en los registros del sensor. Si el bit es '1' consiste en una operación de lectura en la que el maestro lee algún dato de los registros del sensor, o lo que es equivalente, el esclavo escribe un dato en los registros del maestro. La dirección del sensor ya viene predefinida en la hoja de documentación del mismo **[10]**. Para operaciones de lectura, la dirección del sensor es 0x3D (en hexadecimal) y para operaciones de escritura es 0x3C. Estas dos direcciones ya incluyen el octavo bit indicador del tipo de operación. 0x3D en binario es 00111101, donde el octavo bit o bit menos significativo vale '1' (operación de lectura). 0x3C en binario es 00111100, con un bit '0' como bit menos significativo (operación de escritura).

Tanto estas dos direcciones como las direcciones del resto de registros del sensor HMC5883L (tabla 8.1) se han definido en variables en otro fichero secundario llamado "HMC5883L.h" (figura 8.11) con el fin de que el código de programación sea más comprensible. En este fichero se incluirán las funciones pertenecientes a las operaciones de escritura, lectura, conexión y desconexión del sensor. Más adelante se hablará de ellas.

EUSART.h | Main.c | i2c.h | HMC5883L.h

Source | History

```
1  //Se definen las variables que contienen las direcciones del HMC5883 para operaciones
2  //de lectura y escritura:
3  #define HMC5883L_LECTURA_DIR     0x3D
4  #define HMC5883L_ESCRITURA_DIR   0x3C
5
6  //Se definen las variables que contienen las direcciones de los registros del HMC5883L:
7  #define Config_Reg_A   0x00
8  #define Config_Reg_B   0x01
9  #define Mode_Reg       0x02
10 #define X_MSB_Reg      0x03
11 #define X_LSB_Reg      0x04
12 #define Z_MSB_Reg      0x05
13 #define Z_LSB_Reg      0x06
14 #define Y_MSB_Reg      0x07
15 #define Y_LSB_Reg      0x08
```

Figura 8.11 – *Definición de variables en el fichero "HMC5883L.h"*

Para que el maestro pueda enviar un byte con la dirección de un registro del sensor se ha implementado en el fichero "i2c.h" la función i2c_enviar_byte (figura 8.12), la cual envía el byte que se le indique, bit a bit, a través de la señal SDA: si el bit es un '1' se pondrá la señal SDA en alto y si el bit es un '0' se pondrá la señal SDA en bajo. Para operaciones de lectura, en las que el microcontrolador debe recibir un byte que envía el sensor, se ha definido la función i2c_recibir_byte.

```
void i2c_enviar_byte(char byte) //Funcion para enviar un byte, bit a bit, a traves de la señal SDA.
{
   char n = 0;
   while(n<8)               //Se envia bit a bit, de más significativo a menos significativo
   {
      if((byte&0x80) == 128)    //0x80 es 10000000. Por lo tanto, si el bit mas significativo de byte es 1.
      {                         //128 (decimal) en binario es 10000000
         i2c_SDA_alto();        //Se transfiere el bit 1 por la señal SDA
      }
      else
      {
         i2c_SDA_bajo();        //Se transfiere el bit 0 por la señal SDA
      }
      i2c_SCL_alto();
      i2c_SCL_bajo();   //Aqui termina el pulso del clock. Es en el pulso donde se coge el valor del bit del SDA
      byte = byte << 1;  //Se desplaza el byte un bit hacia la izquierda para poder enviar el siguiente bit
      n++;
   }
   i2c_SDA_alto();  //Se libera la señal SDA para que el esclavo pueda enviar el bit de confirmacion (ACK)
}
char i2c_recibir_byte()  //Funcion para leer el byte almacenado en un registro del HMC5883L
{
   char i_byte = 0, n = 0;
   i2c_SDA_alto();   //Pin SDA habilitado como entrada. El esclavo envia automaticamente los bits,-
   while(n<8)        //-que entran al PIC por SDA
   {
      i2c_SCL_alto();  //Empieza la lectura del bit

      if (SDA_PIN)     //Si SDA = 1
      {
         i_byte = (i_byte << 1) | 0x01;  //Se añade un 1 en el byte "i_byte"
      }
      else            //Si SDA = 0
      {
         i_byte = i_byte << 1;          //Se añade un 0 en el byte "i_byte"
      }

      i2c_SCL_bajo();  //Finaliza la lectura del bit
      n++;
   }
   return(i_byte);
}
```

Figura 8.12 – *Función i2c_enviar_byte del fichero "i2c.h"*

Cabe destacar que todos los cambios de valor en la señal SDA se producen cuando la señal SCL tiene valor bajo. De manera que el valor que tiene un bit resulta el que tiene SDA durante el pulso de la señal SCL (cuando SCL vale '1'). Que la SDA cambie cuando la SCL está alta ocurre para dos únicas condiciones: la del bit de arranque (Start bit) y el bit de parada (Stop bit). Por cada bit transmitido en la señal SDA sucede un pulso de reloj.

Una vez se ha mandado la dirección del esclavo por la SDA, el esclavo responde con un bit de confirmación (ACK) registrado en el noveno pulso de reloj. Es en este momento cuando el esclavo enviará un '0' a través de la señal SDA para indicar que la conexión maestro-esclavo se ha establecido. En el caso de que la conexión sea errónea, la línea SDA se mantiene en alto (NACK). Estas dos acciones se han implementado en el fichero "i2c.h" con las funciones i2c_ack y i2c_nack respectivamente (figura 8.13).

```
void i2c_ack()  //Funcion para que el maestro haga el 9º pulso de la señal SCL y asi el esclavo envie el ACK
{               //El esclavo envia automaticamente el ACK, es decir, pone a 0 la señal SDA
    i2c_SCL_alto();
    i2c_SCL_bajo();
}

void i2c_nack()
{
    i2c_SDA_alto(); //Se pone SDA en alto por si es el master quien finaliza la transmision (cuando recibe datos)
    i2c_SCL_alto();
    i2c_SCL_bajo();
}
```

Figura 8.13 – *Funciones i2c_ack y i2c_nack del fichero "i2c.h"*

Una vez se hayan conectado, se manda a través de la SDA el dato a escribir en el esclavo o el esclavo manda por la misma línea el dato que quiere leer el maestro. Si se quiere volver a enviar o leer un nuevo dato, el maestro debe enviar otro nuevo bit de arranque.Y de nuevo se manda el bit de confirmación ACK para indicar la correcta transferencia del nuevo dato. En el caso de que sea el maestro quien lea un dato del esclavo, es el esclavo quien manda este bit de confirmación; cuando el maestro lee un dato del sensor, es el maestro quien envia este bit.

Para finalizar la transmisión, el maestro envía un bit de parada (Stop bit) con el que la SDA pasa de valor '0' a '1' mientras la señal SCL valga '1' (figura 8.14). La función correspondiente a esta secuencia es la i2c_fin (figura 8.15).

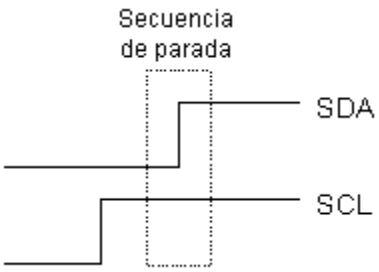


Figura 8.14 – *Secuencia de parada*

```
void i2c_fin()    //Funcion para finalizar la transferencia de datos
{
    i2c_SDA_bajo();
    i2c_SCL_alto();
    i2c_SDA_alto();     //Condicion de fin de transmision: SDA cambia a 1 cuando SCL = 1
}
```

Figura 8.15 – *Función i2c_fin del fichero "i2c.h"*

En resumen, el sensor de campo magnético detecta las variaciones de campo magnético debido a la presencia de elementos metálicos que perturban este campo, convierte la señal analógica en digital (byte) y la almacena en sus seis registros de salida (Data Output Register),

dos bytes por cada dirección. Es entonces cuando el microcontrolador debe establecer la comunicación con el sensor vía el bus I^2C y mandar las señales oportunas para leer sus registros.

En la figura 8.16 se muestran las funciones que se han incluido en el fichero secundario "HMC5883L.h". La función HMC5883L_escribir permite escribir un byte en el registro deseado, de modo que con la función HMC5883L_Inicializar se configurará el sensor escribiendo los bytes de configuración necesarios en el registro de configuración A, en el registro de configuración B y el registro de modo. Se ha configurado el sensor con una velocidad de escritura de los registros de salida de 15 Hz, una ganancia de 390 LSb/Gauss y el modo de medición continua.

```
void HMC5883L_escribir(char reg_direccion, char valor)  //Para escribir el byte "valor" en el
{                                                       //registro de direccion "reg_direccion"
   i2c_inicio();
   i2c_enviar_byte(HMC5883L_ESCRITURA_DIR);
   i2c_ack();
   i2c_enviar_byte(reg_direccion);
   i2c_ack();
   i2c_enviar_byte(valor);
   i2c_nack();
   i2c_fin();
}

char HMC5883L_leer(char reg)              //Para leer un byte del registro con direccion "reg"
{
   char val = 0;
   i2c_inicio();
   i2c_enviar_byte(HMC5883L_ESCRITURA_DIR);
   i2c_ack();
   i2c_enviar_byte(reg);
   i2c_ack();
   i2c_inicio();
   i2c_enviar_byte(HMC5883L_LECTURA_DIR);
   i2c_ack();
   val = i2c_recibir_byte();
   i2c_nack();
   i2c_fin();
   return(val);
}
void HMC5883L_inicializar()
{
   HMC5883L_escribir(Config_Reg_A, 0x70);  //Data output rate = 15Hz
   HMC5883L_escribir(Config_Reg_B, 0xA0);  //ganancia de 390 LSB/Gauss
   HMC5883L_escribir(Mode_Reg, 0x00);      //Modo de medicion continua
}

void HMC5883L_desconectar()
{
   HMC5883L_escribir(Mode_Reg, 0x02);
}
```

Figura 8.16 – *Funciones definidas en el fichero "HMC5883L.h"*

Para entender mejor la función HMC5883L_escribir, vease los pasos que se realizan con la instrucción HMC5883L_escribir(Config_Reg_A, 0x70) que está contenida dentro de la función HMC5883L_Inicializar (figura 8.17). Con esta instrucción se escribe el byte 0x70 en el registro de configuración A del sensor para tener un ratio de salida de datos de 15 Hz. Para ello, el microcontrolador inicia la transmisión enviando el bit de arranque. Seguidamente envia la dirección de escritura del sensor (0x3C en hexadecimal, que equivale en binario a 00111100). A continuación, en el noveno pulso del reloj, el sensor responde con el bit de confirmación ACK y posteriormente el PIC envía la dirección del registro al que se dirige (0x00). De nuevo el esclavo responde con el ACK y se envía el dato 0x70 (01110000 en binario). Una vez el sensor lo ha recibido, éste envía el bit de confirmación negativo NACK para indicar al maestro que ya no puede recibir más datos, ya que la posición del registro de configuración A ha sido ocupada por el byte 0x70. Finalmente el microcontrolador finaliza la transmisión con el bit de parada.

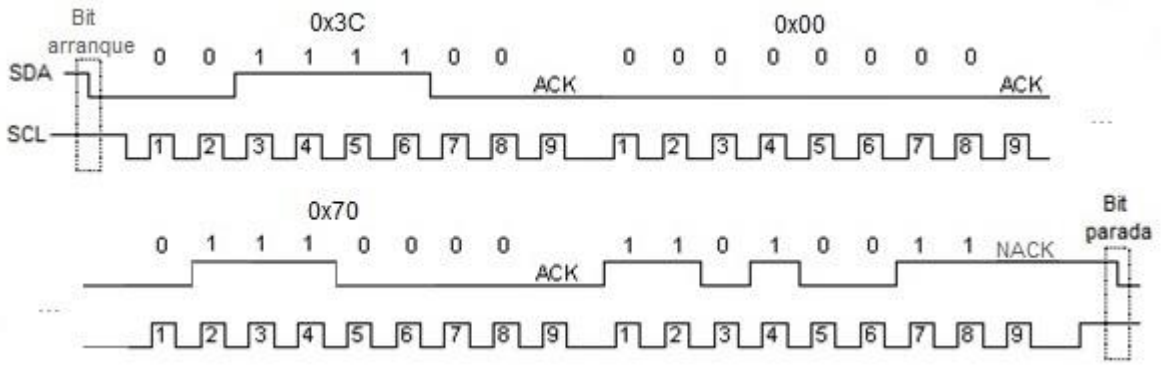


Figura 8.17 – *Señales SDA y SCL correspondientes a la instrucción HMC5883L_escribir(Config_Reg_A, 0x70)*

Siguiendo con la explicación del código de programación del fichero “HMC5883L.h”, la función HMC5883L_desconectar cambia el modo de medición continua a modo Sleep o parada, con lo que el sensor deja de tomar medidas. Finalmente la función HMC5883L_leer permite leer el byte que contiene el registro que se especifique. Con esta función se hará que el microcontrolador lea los registros de salida del sensor, los cuales contienen la información medida.

Así pues, la versión final del fichero principal “Main.c” para programar el microcontrolador e implementar el detector de metales es la de las figuras 8.18 y 8.19 . En ellos se han declarado seis variables para los datos que se van a medir con el sensor.

EUSART.h | Main.c | i2c.h | HMC5883L.h

Source | History

```
#include <xc.h>
#include <pic16f690.h>
#include "EUSART.h"

#define SDA_PIN RB4
#define SCL_PIN RB6
#define SDA_DIR TRISB4
#define SCL_DIR TRISB6

#include "i2c.h"
#include "HMC5883L.h"

char LECTURA_X_L; // Se declaran 6 variables para los datos que se van a medir
char LECTURA_X_M;
char LECTURA_Y_L;
char LECTURA_Y_M;
char LECTURA_Z_L;
char LECTURA_Z_M;

//Inicio de la configuracion de bits del PIC
#pragma config FOSC = INTRCIO
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = OFF
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = OFF
#pragma config IESO = OFF
#pragma config FCMEN = OFF
//Fin de la configuracion de bits

```

Figura 8.18 – Fichero “Main.c”

También se ha añadido la variable de ocho bits (char) interruptor (figura 8.19), que se utiliza para guardar los datos que recibe el microcontrolador a través del módulo bluetooth incorporado. Estos datos serán las señales que se envían desde la aplicación móvil Metal Detector enseñada en el punto 5.1, concretamente las señales generadas al pulsar los botones “Start” y “Stop”, que corresponden con eviar el byte con valor decimal 1 y el byte con valor decimal 0 respectivamente.

```
#define _XTAL_FREQ 4000000

char interruptor;

void interrupt Interrupcion()      //Funcion que atiende las interrupciones
{
    if(PIR1bits.RCIF)          //Cuando hay algun dato en la FIFO pendiente por leer (RCIF=1)
    {
        interruptor = EUSART_Lectura();  //Se lee el dato y por tanto RCIF=0
        if (interruptor)                 //Si se pulsa el boton Start de la aplicacion movil
            {
                HMC5883L_inicializar();
                __delay_ms(6);           // Se espera 6 ms para tener listas las medidas tomadas por el sensor
                while(!PIR1bits.RCIF)    //mientas RCIF=0
                {
                    LECTURA_X_M = HMC5883L_leer(X_MSB_Reg);   //Se leen las 6 medidass
                    LECTURA_X_L = HMC5883L_leer(X_LSB_Reg);
                    LECTURA_Y_M = HMC5883L_leer(Y_MSB_Reg);
                    LECTURA_Y_L = HMC5883L_leer(Y_LSB_Reg);
                    LECTURA_Z_M = HMC5883L_leer(Z_MSB_Reg);
                    LECTURA_Z_L = HMC5883L_leer(Z_LSB_Reg);
                    __delay_ms(67);

                    PORTC=LECTURA_Y_M;  //Se muestran los datos del eje Y por el puerto C
                    if (PORTC != 0)
                    {
                        EUSART_Escritura(1);   //se activa efecto de sonido y vibracion en el movil
                    }
                    else
                    {
                        EUSART_Escritura(0);  //Se desactiva el sonido y vibracion
                    }
                 }
            }

        else if(!interruptor)   //Si se pulsa el boton Stop de la aplicaicon movil
            {
                HMC5883L_desconectar(); //Se desconecta el sensor
                EUSART_Escritura(0);    //Se desactiva el sonido y vibracion
            }
    }

    if(RCSTAbits.OERR)         //Si se activa el bit de error OERR
    {
        RCSTAbits.CREN = 0;    //Se limpia el bit CREN
        RCSTAbits.CREN = 1;    //Se rehabilita la recepcion continua de datos
    }
}

void main()
{
  ANSEL = 0;                //Configuracion de todos los pines del PIC como entradas o salidas digitales
  ANSELH = 0;
  TRISC = 0;                //Los pines del puerto PORTC se configuran como salidas
  EUSART_Inicializacion();  //Se inicializa la EUSART con velocidad de transmision de 9600 bits/segundo
  INTCONbits.PEIE = 1;      //Se activa la interrupcion de perifericos
  INTCONbits.GIE = 1;       //Se activan las interrupciones globales
  PIE1bits.RCIE = 1;        //Se activan las interrupciones en la recepcion de datos
}
```

Figura 8.19 – Continuación del fichero “Main.c” de la figura 8.18

La función principal main del programa se mantiene igual que en la última modificación. Esta función es la que se ejecuta al alimentar el microcontrolador. Habilitará los pines del puerto C como salidas digitales, inicializará el módulo EUSART con una velocidad de transmisión de datos de 9600 bits/segundo y se activarán los bits de interrupción. Al habilitar estos bits, cuando

se active el bit RCIF (hecho indicador de que el PIC ha recibido un dato) o el bit de error OERR (indicador de que hay sobresaturación en la recepción de datos), la función main dejará de ejecutarse para saltar a la función de interrupción.

Por lo tanto, si se pulsa el botón “Start” de la aplicación móvil, se envía el byte decimal 1 al microcontrolador y se activa el bit RCIF. Consecuentemente se ejecuta la función de interrupción y debido a que la variable interruptor toma el valor 1, se pone en marcha el sensor. Seis milésimas de segundo después (tiempo necesario para que el sensor tome las medidas y las almacene en sus registros de salida), mientras no llegue otra señal al microcontrolador, es decir, mientras no se pulse ni el botón “Start” y “Stop” de la aplicación, el PIC lee las medidas tomadas por el sensor y las guarda en las variables de lectura declaradas en el inicio del fichero. Finalmente se guarda la lectura del eje Y en el puerto C y si esta resulta ser diferente a 0, indicador de que se ha detectado un elemento metálico, se envía el byte decimal 1 al teléfono, lo cual activará el efecto de sonido, la vibración y la etiqueta indicadora “Metal Detected”. Por lo contrario, si la lectura Y es 0, indicador de que no ha detectado un metal, se envía el byte decimal 0 al móvil, que hará detener el efecto de sonido, la vibración y eliminará la etiqueta “Metal Detected”.

Toda esta secuencia se repite hasta que se pulse el botón “Stop”. En este momento el teléfono envía el byte decimal 0 al microcontrolador, se activará el bit RCIF y la variable interruptor tomará el valor decimal 0. Por lo tanto, se ejecutará el else if de la función de interrupción, con el que se desactiva el sensor. El programa volverá entonces a ejecutar la función principal main hasta el momento en que el PIC reciba de nuevo una señal del móvil. Según su valor (byte decimal 1 o 0) se activará el sensor o se desactivará.

La lectura del valor medido en el eje Y se ha guardado en el puerto C mediante la instrucción PORTC=LECTURA_Y_M (figura 8.19). De esta manera, acoplando ocho LEDs en los pines del puerto C se puede observar, siempre que se active el sensor, esa medición. Cuando no se detecta ningún metal la lectura vale 0, ya que no incide ningún campo magnético y, por tanto, no hay ninguna diferencia de potencial detectada por el sensor. En este momento, pues, los LEDs deben estar apagados (figura 8.20).

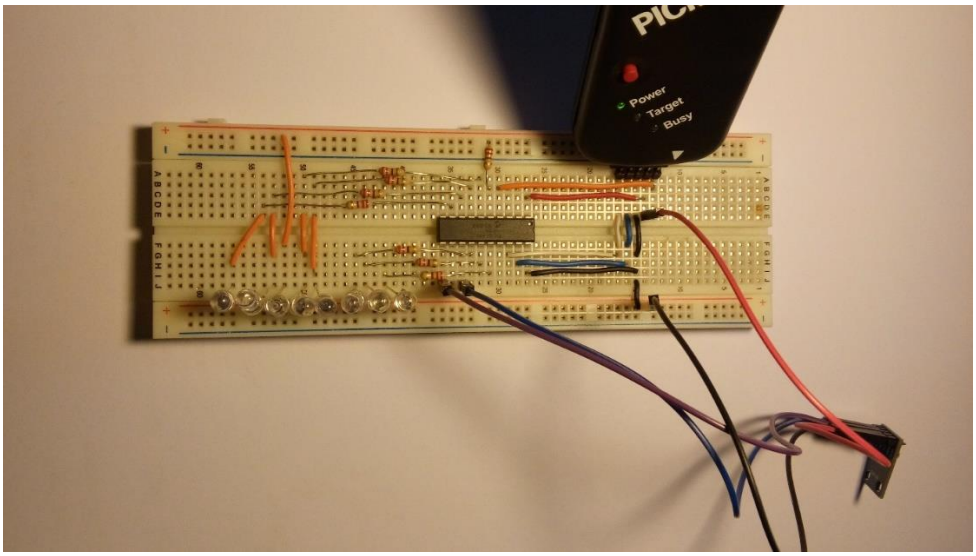


Figura 8.20 – *Lectura nula al no detectar ningún metal*

En cambio, al acercar un elemento metálico como un destornillador o un imán, la lectura no es cero y por tanto, los LEDs se encienden según el byte medido (figura 8.21).

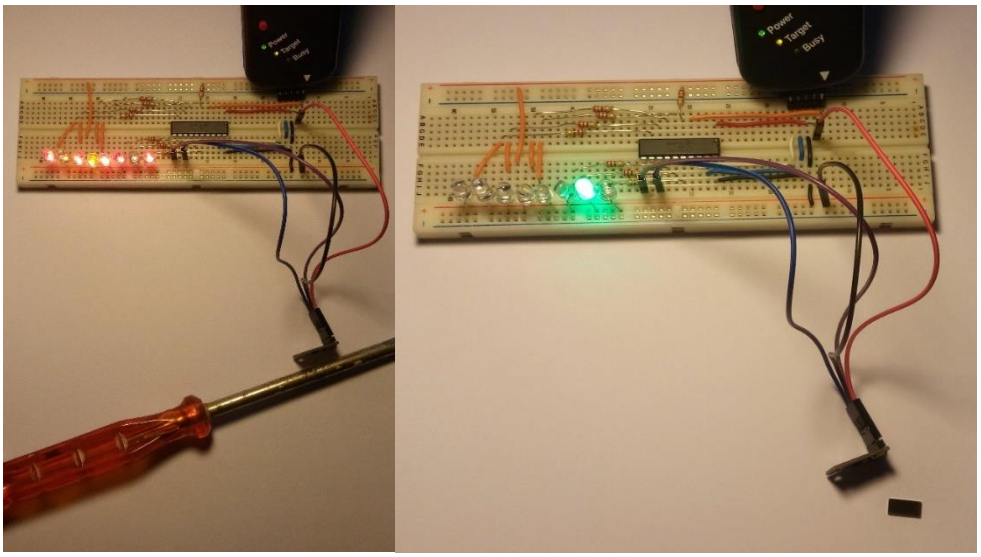


Figura 8.21 – *Lectura no nula al detectar elementos metálicos o imanes*

9. Fiabilidad del detector de metales

Como se ha visto en el apartado 8, únicamente se indica la presencia de un metal en función de la medición realizada en la dirección Y (PORTC=LECTURA_Y_M). Esto es debido a que después de muchas pruebas de verificación, se ha observado que la lectura útil del sensor es la correspondiente al movimiento del objeto en esa dirección. Con las lecturas del resto de ejes, había momentos en los que no se acercaba ningún elemento metálico y todos los LEDs acoplados a las salidas del puerto C del microcontrolador estaban encendidos. Por esta razón, finalmente se ha decidido hacer sólo la lectura del eje Y.

El inconveniente de utilizar sólo la medición respecto un eje es que no se aprovecha todo el potencial que ofrece el sensor. Aun así, se es capaz de detectar metales. Sólo hay que tener en cuenta que la detección de perturbaciones del campo magnético se realizan en la dirección Y.

9.1. Rango de medición

La intensidad con la que se detecta el metal depende de la intensidad de campo magnético del elemento metálico y de su distancia al sensor. Cuanto más intenso sea el campo magnético que incide en el detector (cuanto más cerca esté el metal), más variará la resistencia de los magnetoresistores del sensor y, por tanto, más diferencia de potencial habrá en sus salidas. Consecuentemente, el valor digital o byte obtenido del convertidor analógico-digital será más elevado. Este fenomeno se aprecia en las figuras 9.1, 9.2 y 9.3.

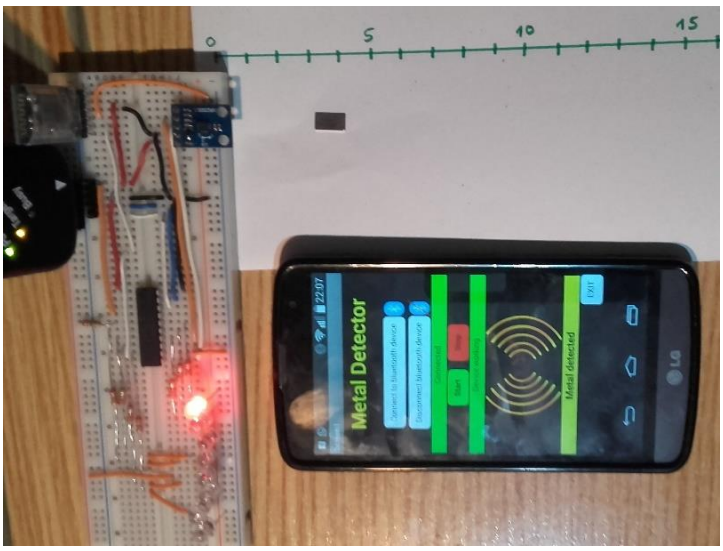


Figura 9.1 – *Detección del valor decimal 1 (en binario 00000001) con imán a 3 cm del sensor*

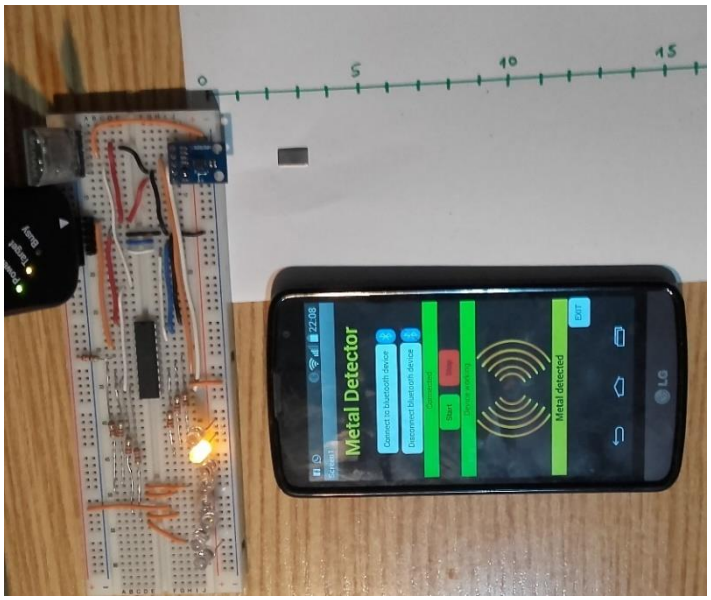


Figura 9.2 – *Detección del valor decimal 2 (en binario 00000010) con imán a 2 cm del sensor*

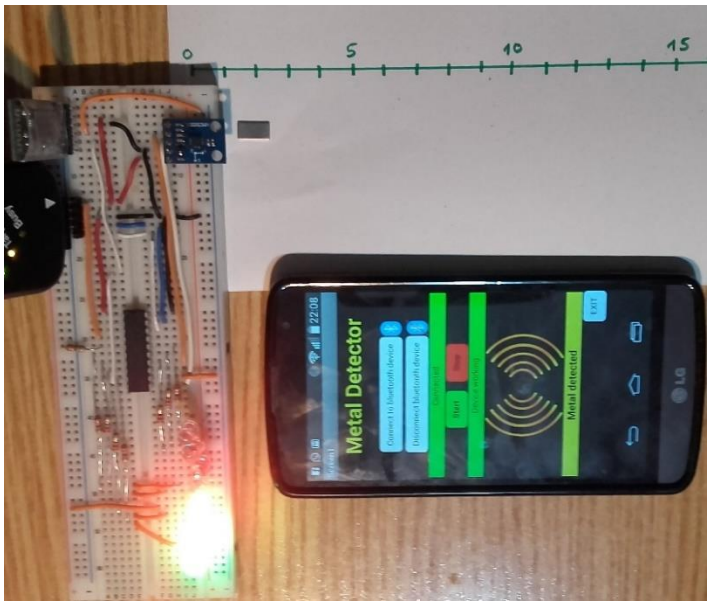


Figura 9.3 – *Detección del valor decimal 248 (en binario 11111000) con imán a 1 cm del sensor*

Como se observa en ellas, se ha dibujado unas marcas en centímetros representativas de la distancia a la que se encuentra el objeto del sensor, en este caso un pequeño imán de neodimio, cuya intensidad de campo es bastante intensa. Primero se ha colocado encarando uno de sus polos al sensor, de modo que al ir acercándose cada vez más, la lectura tomada por el sensor, que se indica mediante los LEDs, va aumentando de valor (el byte aumenta de valor decimal). En la figura 9.1 el imán se encuentra a 3 cm del sensor y el byte saliente es un 1 decimal (en binario 00000001). A los 2 cm de cercanía, el byte pasa a ser un 2 decimal

(00000010) y sucesivamente hasta que llega un punto en el que se encienden todos los LEDs (255 en decimal es 11111111).

En cambio, si se cambia el imán de orientación y se encara el polo contrario al sensor, el valor decimal de las lecturas decrece (figuras 9.4, 9.5 y 9.6). Esto es debido a que en esta orientación, el sentido de la fuerza magnética resultante que ejerce el imán sobre el sensor es de sentido opuesto a la situación anterior.

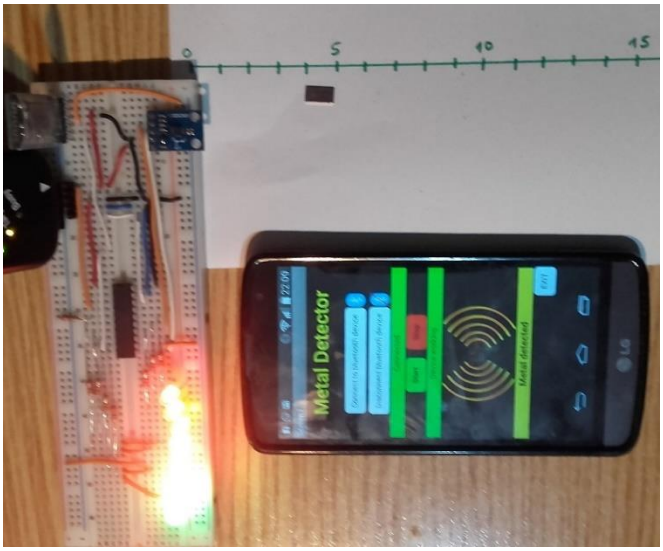


Figura 9.4 – *Detección del valor decimal 255 (en binario 11111111) con imán a 4 cm del sensor*

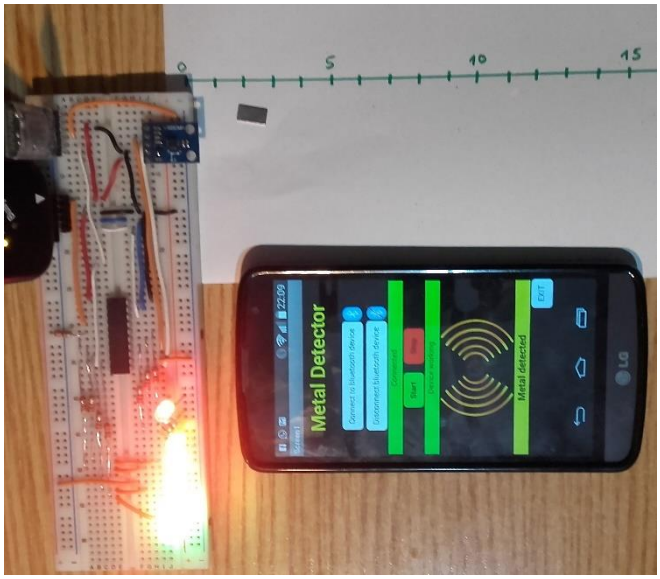


Figura 9.5 – *Detección del valor decimal 253 (en binario 11111101) con imán a 2 cm del sensor*

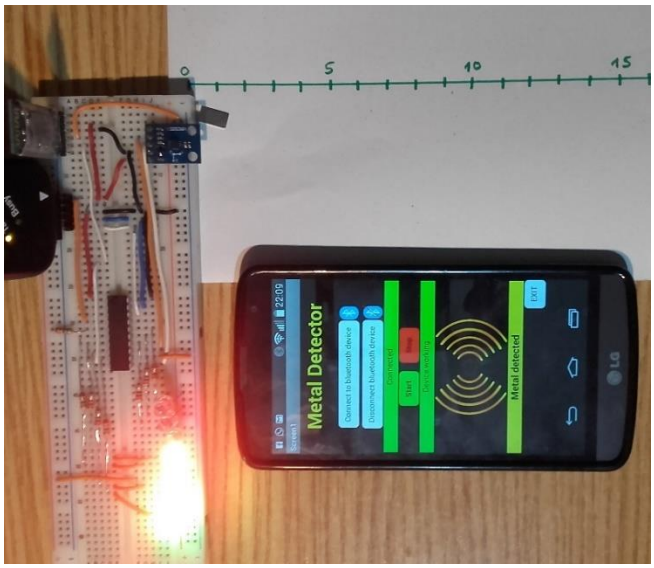


Figura 9.6 – *Detección del valor decimal 252 (en binario 11111100) con imán a 0,5 cm del sensor*

Ahora bien, para saber el rango aproximado de distancias en el que trabaja el detector de metales se ha realizado el siguiente experimento de la figura 9.7. Se ha hecho circular una corriente por una bobina de N espiras, de tal modo que se ha generado un campo magnético cuyo valor es el de la ecuación 9.1 **[13]**, donde μ es la permeabilidad magnética del vacío ($4\pi \cdot 10^{-7}$ NA^{-2}), I la intensidad que circula por la bobina an amperios, L la longitud de la bobina en metros y α y β los ángulos especificados en la figura 9.8. Al ir acercando la bobina al sensor, según el campo generado se ha detectado a una distancia diferente.

$$B = \frac{\mu I N}{2L} \cdot (\cos\alpha - \cos\beta) \qquad \text{(ec. 9.1)}$$

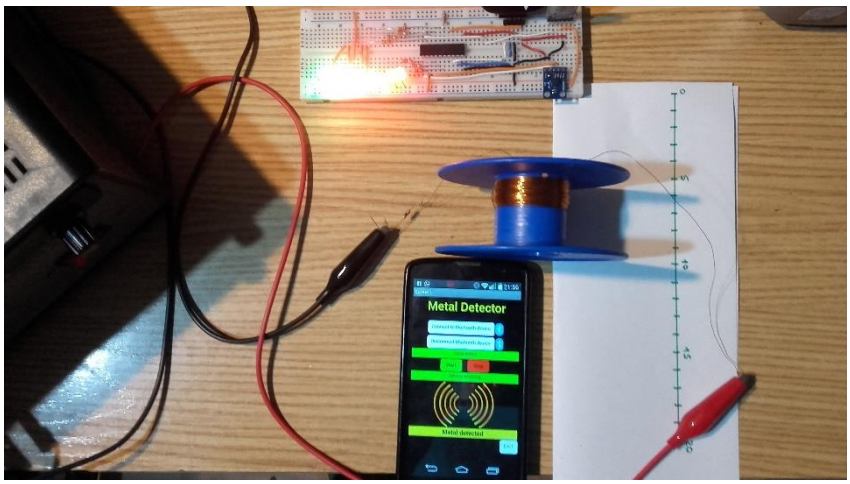


Figura 9.7 – *Experimento realizado para conocer el rango de detección del dispositivo*

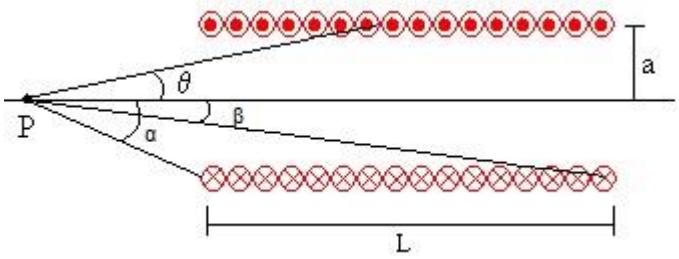


Figura 9.8 – *Campo producido por un solenoide en un punto de su eje*

Se ha construido un bobina de 150 espiras, de longitud L igual a 1,2 cm y radio a de 1,75 cm. A continuación se ha conectado mediante una resistencia de 10 Ω a una fuente de alimentación y se ha ido variando la tensión para conseguir diferentes corrientes según la ley de Ampere (ec. 9.2).

$$I = \frac{V}{R} = \frac{V}{10\Omega} \qquad \text{(ec. 9.2)}$$

Con el valor de esta corriente se ha podido calcular el valor de campo magnético que genera la bobina. A partir de aquí, para unos valores de campo conocidos, se ha ido variando la distancia de la bobina para ver a qué distancia máxima era detectada. Al aumentar la tensión aumenta la intensidad de corriente y el campo magnético generado por la bobina, de modo que cuanta más corriente a más distancia será detectado este campo.

En la tabla 9.1 se muestran las medidas tomadas. La distancia se ha considerado del sensor al inicio de las espiras de la bobina.

Intensidad [A]	Campo magnético [T]	Campo magnético [Gauss]	**Distancia al sensor[m]**
0,3	-0,00611	-6,11	0,03
0,4	-0,00814	-8,14	0,04
0,5	-0,00102	-10,2	0,05
0,75	-0,00153	-15,3	0,08
1,2	-0,00244	-24,4	0,1

Tabla 9.1 – *Distancia máxima de detección según el campo magnético generado por la bobina*

La distancia máxima respecto al sensor ha sido de 10 cm con un campo magnético generado por la bobina de -24,4 Gauss (figura 9.9).

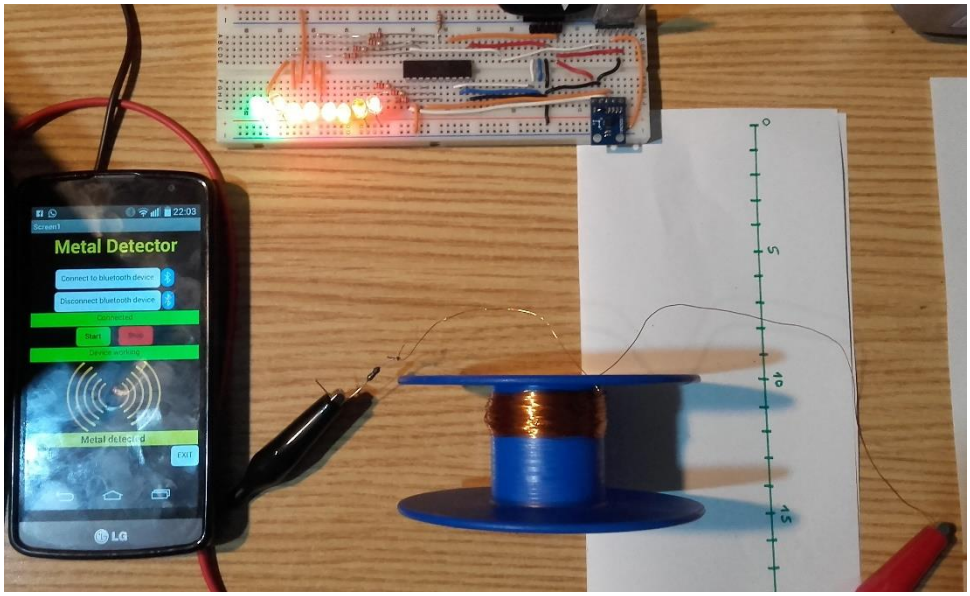


Figura 9.9 – *Campo magnético de -24,4 Gauss captado a una distancia de 10 cm*

Aun así, las mediciones han estado más claras en el rango de 0 a 8 cm de distancia respecto el sensor. En este intervalo el detector de metales ha captado el campo magnético de la bobina en todas las situaciones de la tabla 9.1 (como la figura 9.10). No se han probado valores de campo magnético más elevados puesto que 24,4 Gauss ya es un valor muy elevado. Así que el rango final de fiablidad del detector de metales se declara en una distancia de 0 a 8 cm máximo.

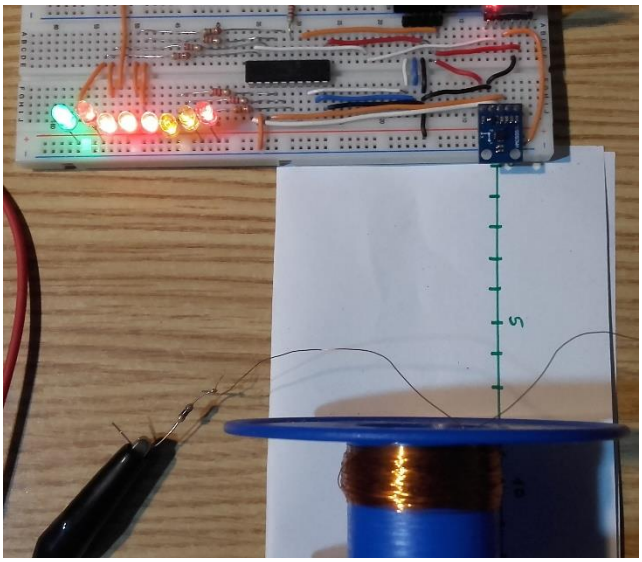


Figura 9.10 – *Campo magnético de -15,3 Gauss captado a una distancia de 8 cm*

10. El regulador de tensión

Una vez realizado el código de programación necesario para el microcontrolador y hecho las pruebas de verificación y fiabilidad del aparato detector de metales a través de la placa protoboard, se ha procedido con la soldadura de los distintos elementos que componen el dispositivo detector sobre una placa de soldadura.

Para ello se debe tener en cuenta que el conjunto debe incluir un regulador de tensión, debido a que ahora el conjunto no estará alimentado a los 5V de tensión que ofrece el programador PICkit2, sinó que se alimentará a través de una pila de 9V. El regulador se encargará de proveer al conjunto de esos 5V necesarios.

El componente utilizado para regular la tensión es el L7805CV (figura 10.1) **[14]**, con un total de tres pines. Uno de ellos utilizado como entrada de los 9V de la pila, otro como toma de tierra y el último para la salida regulada de los 5V.

Figura 10.1 – *Regulador de tensión L7805CV*

11. Presupuesto

Para la confección del presupuesto de la tabla 11.1, se han tenido en cuenta todos los elementos que han permitido su diseño y construcción, tanto componentes electrónicos como materiales y programas informáticos. Además de los recursos humanos. Todos los precios incluyen el IVA.

Se ha especificado el precio por hora de un ingeniero senior en 15 €/hora y las horas invertidas en el proyecto en función del número de créditos (12 créditos a 25 horas por crédito). Para el profesor titulado el precio por hora es de 50 €.

Material	Precio [€/unidad]	Unidades	Total[€]
Microcontrolador PIC16F690	2,68	1	2,68
Sensor magnético HMC5883L	4,07	1	4,07
Módulo bluetooth HC-05	4,90	1	4,90
Regulador de tensión L7805CV	0,31	1	0,31
Resistencia 220 Ω	1,19	1	1,19
LED bajo consumo	0,25	8	2
Pila 9V	2,33	1	2,33
Cables de conexión	3	1	3
Zócalo	0,16	1	0,16
Protoboard	7,18	1	7,18
Alambre de estaño	2,45	1	2,45
Hilo de cobre	2,45	1	2,45
Soldador eléctrico	7,48	1	7,48
Smartphone	31,19	1	31,19
Ordenador portátil	245,26	1	245,26

Editor MPLAB X IDE	0	1	0
Compilador XC8	0	1	0
Diseñador de aplicaciones móviles App Inventor 2	0	1	0
Programador PICkit2	24,99	1	24,99
Ingeniero Senior	15	300	4500
Profesor titulado	50	10	500
TOTAL			**5341,64**

Tabla 11.1 – *Presupuesto del detector de metales*

A la hora de hacer el presupuesto no se ha tenido en cuenta el precio del osciloscopio utilizado durante las pruebas de verificación en el laboratorio, debido a que es un aparato de muy alto coste y su uso es prescindible para el desarrolo del detector de metales.

12. Impacto ambiental

Según las directivas de la Unión Europea **[15]**, se exige una evaluación de impacto ambiental (EIA) de todos aquellos proyectos que interfieren en el medio ambiente de manera positiva o negativa. El documento que recoje esta estudio o evaluación es la declaración de impacto ambiental (DIA). Este documento identifica y evalua las etapas del proyecto que pueden producir un impacto ambiental. En el caso del detector de metales, se encuentran las etapas de diseño, construcción y uso del producto.

En la fase de diseño se han utilizado componentes que ya están fabricados como el microontrolador, el sensor, el módulo bluetooth, etc. Estos elementos deben cumplir previamente estas directivas marcadas.

Por otro lado, la construcción del prototipo se ha hecho manualmente, así que el impacto ambiental ha sido nulo. En cambio, para realizar el montaje final sobre una placa de soldadura se han tenido que soldar los distintos elementos a ella. En este proceso se ha quemado estaño, con la dispersión de gases correspondiente. Aún así, el periodo de tiempo ha sido corto generando muy poco impacto sobre la atmósfera.

Finalmente, durante la etapa de uso del detector de metales, éste no genera nigún impacto en el ambiente. Pero al terminar la vida útil del producto se debe deshacer de manera adecuada de los componentes electrónicos. Para ello existen empresas como Recyclia **[16]**, que se encargan de gestionar los residuos eléctricos y electrónicos y las pilas usadas. Estas empresas recogen este tipo de elementos en puntos limpios y comercios y los llevan a la planta de tratamiento. En ella se realizará el proceso adecuado y se reciclan o reaprovechan algunos materiales, de forma que se ahorra dinero y no se contamina.

Conclusiones

Se ha llevado a cabo el diseño y construcción de un detector de metales basado en tres componentes electrónicos: el microcontrolador PIC16F690, el sensor magnético HMC5883L y un módulo bluetooth HC-05 para poder controlar este dispositivo mediante un teléfono móvil con sistema operativo Android. Para ello se ha creado la aplicación móvil Metal Detector, de uso intuitivo y que permite conectarse vía tecnología bluetooth al aparato detector con el fin de contolar su encendido y apagado. También se ha conseguido monitorizar continuamente el estado del detector de metales a través de señales visuales y sonoras generadas por el teléfono.

El código de programación del microcontrolador se ha realizado en lenguaje de programación c con el editor MPLAB X IDE. En él se han creado distintos ficheros para definir diferentes aspectos generales del microchip, como la transmisión y recepción de datos con el sensor magnético y el bluetooth HC-05 a través del módulo EUSART, el protocolo de comunicación tipo I^2C con estos dispositivos y el procesamiento de las medidas tomadas por el sensor para mandar la señal correspondiente al teléfono móvil.

Tras varias pruebas de verificación del código con el prototipo montado sobre la placa protoboard, se ha observado que la lectura útil del sensor es la correspondiente al movimiento del objeto en la dirección Y, por esta razón sólo se ha tenido en cuenta una de las direcciones de medida. Con la medición resultante en esta dirección es suficiente para que el detector haga su función, detectar metales. Para conocer el rango aproximado de medición fiable del montaje final, se ha realizado un experimento con una bobina de N espiras, con la que se han generado distintos valores de campo magnético conocidos. Los resultados han manifestado un rango de medición apto de 0 a 8 cm de distancia máxima entre el sensor magnético y el elemento metálico detectado.

Así pues, el detector es capaz de detectar metales aunque no se haya aprovechado todo el potencial que ofrece su sensor magnético. Para obtener unas medidas más precisas y fiables se debería mejorar el código de programación o calibrar correctamente tal componente ante la presencia de interferencias o ruidos electromagnéticos en el entorno.

Bibliografía

[1] GUASCH I VALLCORBA, Miquel; BORREGO RONCAL, Marina; JORDAN ARIAS, Jordi. Electrotècnia. Mc Graw Hill, Madrid, 2009.

[2] MASSACHUSETTS INSTITUTE OF TECHNOLOGY. *MIT App Inventor 2* [en línea]. <http://ai2.appinventor.mit.edu> [Consulta: 28 de noviembre de 2015]

[3] ITEAD STUDIO. *HC-05: Bluetooth to Serial Port Module* [en línea]. <http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf, 18/06/2010> [Consulta: 2 de marzo de 2016]

[4] SISTEMAS DIGITALES. *Señal de reloj* [en línea]. <https://sistemasdigitales2.files.wordpress.com/2011/11/semana3.pdf> [Consulta: 10 de enero de 2016]

[5] MICROCHIP. *PIC16F690 Datasheet*: *Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART),* pág.148-163 [en línea]. <www.microchip.com/wwwproducts/en/PIC16F690> [Consulta: 12 de octubre de 2015]

[6] MICROCHIP. *Editor MPLAB X IDE versión 3.10* [en línea]. <http://www.microchip.com/mplab/mplab-x-ide> [Consulta: 15 de octubre de 2015]

[7] MICROCHIP. *Compilador XC8 versión 1.35* [en línea]. <http://www.microchip.com> [Consulta: 15 de octubre de 2015]

[8] MICROCHIP. *PICkit2 Programmer/Debugger User's Guide* [en línea]. <http://ww1.microchip.com/downloads/en/DeviceDoc/51553E.pdf> [Consulta: 15 de octubre de 2015]

[9] MICROCHIP. *PIC16F690 Datasheet: EUSART Baud Rate Generator (BRG*), pág. 160-163 [en línea]. <http://www.microchip.com/wwwproducts/en/PIC16F690> [Consulta: 17 de enero de 2016]

[10] HONEYWELL. *Three-Axis Digital Compass IC HMC5883L* [en línea]. <http://www.farnell.com/datasheets/1683374.pdf> [Consulta: 5 de marzo de 2016]

[11] J. CARLETTI, Eduardo. *Comunicación - Bus I2C: Descripción y funcionamiento* [en línea]. <http://robots-argentina.com.ar/Comunicacion_busI2C.htm> [Consulta: 10 de marzo de 2016]

[12] MICRO COMPUTER CONTROL CORPORATION. *What is an Open-Collector/Open-Drain Circuit* [en línea]. < http://www.mcc-us.com/open-collectorfaq.htm> [Consulta: 12 de marzo de 2016]

[13] GUASCH I VALLCORBA, Miquel; BORREGO RONCAL, Marina; JORDAN ARIAS, Jordi. Electrotècnia - Unitat 3: Magnetisme i electromagnetisme. Mc Graw Hill, Madrid, 2009.

[14] FARNELL. *Regulador de tensión L7805CV Datasheet* [en línea]. <http://www.farnell.com/datasheets/1805459.pdf> [Consulta: 25 de abril de 2016]

[15] Unión Europea. Directiva 2011/92/UE relativa a la evaluación de las repercusiones de determinados proyectos públicos y privados sobre el medio ambiente. Diario oficial de la Unión Europea, 28 de enero de 2012.

[16] Portal oficial de Recyclia [en línea]. <http://www.recyclia.es/> [Consulta: 1 de mayo de 2016]