Universidad de Granada

ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Luis Reyes González Industrialización de aparato medidor de la sedimentación en fluidos magnéticos y magnetorreológicos

16/17

Proyecto fin de carrera

ugr Universidad de Granada

Industrialización de aparato medidor de la sedimentación de geles magnetorreológicos

Luis Reyes González

Ingeniería de telecomunicación

Tutor: Andrés María Roldán Aranda

Curso 2016/2017

ESTUDIOS DE INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

"Industrialización de aparato medidor de la sedimentación en fluidos magnéticos y magnetorreológicos"

CURSO: 2016/2017

Luis Reyes González

ESTUDIOS DE INGENIERÍA DE TELECOMUNICACIÓN

"Industrialización de aparato medidor de la sedimentación en fluidos magnéticos y magnetorreológicos"

REALIZADO POR:

Luis Reyes González

DIRIGIDO POR:

Andrés María Roldán Aranda

DEPARTAMENTO:

Electrónica y Tecnología de los Computadores

El tribunal constituido para la evaluación del proyecto PFC titulado:

"Industrialización de aparato medidor de la sedimentación en fluidos magnéticos y magnetorreológicos"

Realizado por el alumno: Luis Reyes González

Y dirigido por el tutor: Andrés María Roldán Aranda

Ha resuelto asignarle la calificación de:

☐ SOBRESALIENTE (9 - 10 puntos)

☐ NOTABLE (7 – 8.9 puntos)

☐ APROBADO (5 – 6.9 puntos)

☐ SUSPENSO

Con la nota[1]: _____ puntos.

El Presidente:

El Secretario:

El Vocal:

Granada, a ____ de _____ de 2017

[1] Solamente con un decimal.

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Proyecto Fin de Carrera de D. Luis Reyes González,

Informa:

que el presente trabajo, titulado:

"Industrialización de aparato medidor de la sedimentación en fluidos magnéticos y magnetorreológicos"

ha sido realizado y redactado por el mencionado alumno bajo nuestra dirección, y con esta fecha autorizo a su presentación.

Granada, a 22 de Febrero de 2017

Fdo.

Los abajo firmantes autorizan a que la presente copia de Proyecto Fin de Carrera se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 22 de Febrero de 2017

Fdo.

Industrialización de aparato medidor de la sedimentación en fluidos magnéticos y magnetorreológicos

Luis Reyes González

PALABRAS CLAVE:

PIC, PCB, USB, *rotary encoder*, Python, Altium, magnetorreológico, SMD, Solidworks, CCS.

RESUMEN:

Este proyecto tendrá como objetivo la industrialización de un prototipo que mide la sedimentación en fluidos magnéticos y magnetorreológicos. Se realizará un análisis de la documentación del prototipo, de manera que estudiando sus características se puedan añadir mejoras funcionales. El sistema incorporará *hardware*, *software* y parte mecánica, realizado con herramientas específicas como Altium, Solidworks, el compilador CCS y Python.

KEYWORDS:

PIC, PCB, USB, rotary encoder, Python, Altium, magnetorheological, SMD, Solidworks, CCS.

ABSTRACT:

This project will aim the industrialization of a prototype that measures sedimentation in magnetic and magnetorheological fluids. An analysis of the documentation of the prototype will be carried out, so that by studying its characteristics, functional improvements can be added. The system will incorporate hardware, software and mechanical part, made with specific tools like Altium, Solidworks, the compiler CCS and Python.

Dedicado a

Lourdes, por ser mi brújula cuando me pierdo.

Agradecimientos:

En primer lugar, quisiera expresar todo mi agradecimiento a mi familia. Gracias a mis padres por el apoyo constante y la confianza depositada, por haber estado siempre a mi lado y por todo el esfuerzo que han dedicado a formarme. No quisiera olvidarme del resto de mi familia, gracias a todos ellos por su afecto.

Gracias a Lourdes por estar siempre dispuesta a recordarme los beneficios del trabajo. Mención también a mis amigos y compañeros, los de toda la vida y los que se van descubriendo día a día.

Mostrar también mis agradecimientos al Departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, y muy especialmente a mi tutor, Andrés María Roldán Aranda. Gracias por tu constante dedicación, atención y supervisión durante todo este tiempo. Por tu trato cercano y amable, y por todo el conocimiento transmitido.

A todos, de corazón, muchas gracias.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE LISTADOS

LIST OF TABLES

CAPÍTULO

1

INTRODUCCIÓN.

1.1 Contexto

Llegado el momento de cursar la asignatura Proyecto Fin de Carrera correspondiente a la titulación de Ingeniería de Telecomunicación, se pretende realizar un trabajo que esté dentro de todos los ámbitos presentados en la Normativa de Realización de Proyectos Fin de Carrera de la Universidad de Granada, y en el que se desarrolle una solución real a un problema que puede encontrarse en el ejercicio de la profesión.

Esta solución consistirá en el análisis, diseño y optimización de un prototipo (figura 1.1) para llevar a cabo la industrialización del mismo. El prototipo en cuestión es un aparato que mide la sedimentación de los fluidos magnetorreológicos cuya patente pertenece al departamento de Física de la Facultad de Ciencias de la Universidad de Granada.

1

Figura 1.1 – *Prototipo.*

La medición se realiza en base a la variación de la frecuencia de resonancia de un circuito LC, cuyas bobinas se pretende que se puedan conmutar por software, siendo el núcleo de la bobina el fluido magnetorreológico. De esta manera se establece una relación entre la frecuencia de oscilación y la concentración en la muestra.

1.1.1 Principio de operación

Los fluidos magnetorreológicos son aquellos cuyas propiedades reológicas se ven alteradas por la presencia de un campo magnético (figura 1.2), es decir, al aplicar un campo magnético su viscosidad se puede regular desde un estado de viscosidad baja hasta uno similar al lodo en viscosidad [1]. Están formados por partículas magnetizables en suspensión en un líquido portador como puede ser aceite mineral, agua, alcohol etc, o incluso un portador sólido con elasticidad suficiente para permitir la orientación de los dipolos ante el campo magnético externo. Sus aplicaciones se dan en diversos campos, como por ejemplo en la automoción se usan en frenos y amortiguadores, también se usan en construcción civil como amortiguadores en puentes, edificios con dispositivos antiseísmos, en medicina se usan en el sistema de control en las prótesis de rodillas, amortiguación en electrodomésticos, adaptando la rigidez del amortiguador a las distintas fases del ciclo de lavado y carga de ropa...

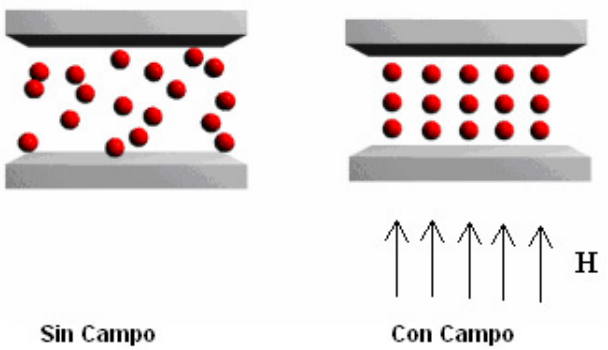


Figura 1.2 – *Respuesta de fluidos magnéticos a un campo magnético exterior.*

Pero las partículas en suspensión estan sometidas a la fuerza de la gravedad, por lo que interesa saber el tiempo que tardan en sedimentar a la hora de elegir un fluido para una aplicación. Antes de este prototipo, los aparatos que medían la sedimentación de los fuidos magnetorreológicos eran ópticos, medían la sedimentación mediante un láser y un receptor colocado al otro lado de la muestra. El método propuesto por el departamento de ciencias consistía, como ya hemos mencionado, en obtener la velocidad de sedimentación mediante el cambio de valor de una bobina cuyo núcleo es el fluido a medir. Esta variación en el valor se debe al cambio de la permeabilidad magnética, μ , inducido por la presencia de partículas magnetizables. Teniendo en cuenta que el valor de L (ec. 1.1.1 donde L μH, A en cm^2 y l en cm) es proporcional al valor de μ, y que este depende de la concentración de partículas en el fluido queda deducida la relación.

$$L = \mu \cdot \frac{N^2 A}{l} \tag{1.1.1}$$

La variación del valor de la bobina se obtendrá a partir de la variación de la frecuencia de resonancia de un resonador LC

1.2 Objetivo

El objetivo principal e inmediato del sistema propuesto es la medida de la sedimentación de los geles magnetorreológicos. Para conseguir el objetivo se han seguido tres líneas de diseño: mecánica, *hardware* y *software*.

En la parte mecánica se engloba el diseño exterior del aparato y su sistema de movimiento.

El *hardware* se basa en un oscilador LC, y en como la frecuencia varía al introducir en el núcleo de la bobina un gel magnetorreológico.

En el campo del *software* englobamos toda la programación para la comunicación entre la GUI y el *hardware*.También incluye la GUI, que sirve para la visualización de los datos y la interactuación con el PIC.

1.3 Contenido y estructura capitular

1

Introducido ya el tema principal de este proyecto, seguiremos las siguientes etapas para llegar a la solución requerida (figura 1.3). Para lo cual, primero analizaremos los requisitos impuestos por el proyecto, para continuar por el análisis de las soluciones adoptadas con el fin de cumplir tales requisistos, después se realizará el proceso de diseño e implementación del sistema. Una vez desarrollado el mismo se procederá a la evaluación y validación del sistema.

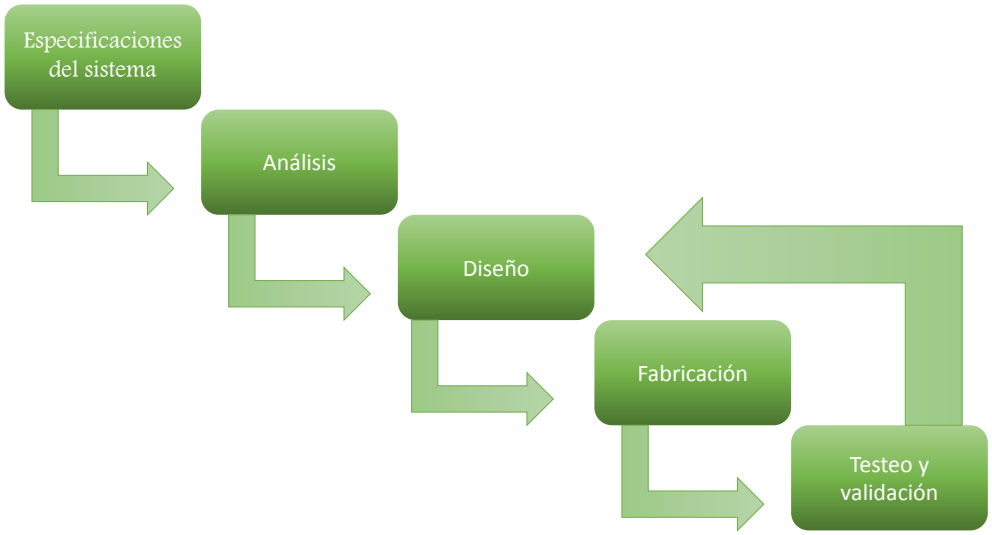


Figura 1.3 – *Fases de trabajo.*

Basándonos en dicho diagrama, se numerarán los diferentes capítulos de esta memoria:

- *Capítulo 2:* **Especificaciones del sistema:** en este capítulo realizaremos un minucioso estudio de los requisistos del sistema a fabricar. Diferenciaremos entre requisitos mecánicos, *hardware* y *software*.
- *Capítulo 3:* **Análisis del sistema:** en este punto se plantean las posbiles soluciones que podemos adoptar para cada uno de los requerimientos y limitaciones observados en las especificaciones del sistema.
- *Capítulo 4:* **Diseño del sistema:** Se expondrán los procedimientos de diseño mecánico, *hardware* y *software*. En esta sección analizaremos las soluciones a los requisitos propuestos en el capítulo 2.
- *Capítulo 5:* **Fabricación** Una vez revisado el diseño configurado en la fase anterior, procederemos a fabricar las placas de nuestro proyecto y las distintas piezas mecánicas

e implementaremos el *software*.

- *Capítulo 6:* **Test y validación** Validaremos el sistema al completo mediante la puesta en funcionamiento de nuestra PCB y analizaremos los resultados obtenidos.
- *Capítulo 7:* **Conclusión y líneas futuras** Breve conclusión y mención a las líneas futuras asociadas al proyecto.

1

CAPÍTULO

2

ESPECIFICACIONES DEL SISTEMA

En este capítulo se expondrán los requisitos y especificaciones del producto a fabricar, desde los iniciales hasta los que han ido surgiendo a medida que se avanzaba en el diseño. Vamos a clasificar los diferentes requisitos en: mecánicos, *hardware* y *software*, dentro de estos últimos haremos una mención especial a los requisitos de la GUI.

2.1 Requisitos mecánicos:

A nivel mecánico se nos pide:

- Realizar un sistema en el que la bobina fuera móvil para escanear la muestra.

- Inmovilizar el tubo para el escaneo.

- Motor de suficientes revoluciones para realizar el escaneo completo en un tiempo inferior a 30 segundos.

- Diseñar modelo $3D$ atractivo visualmente para su industrialización.

2.2 Requisitos *hardware*

En lo relativo al *hardware* tenemos las siguientes exigencias:

- Conmutación de bobinas controladas por PIC.
- Motor que opere a una intensidad menor a 500 mA y su driver correspondiente.
- LCD para la visualización de variables.
- *Buzzer* para emisión de señales acústicas.
- Integración de un sistema ICSP.
- *Rotary encoder* para controlar el aparato sin necesidad de la GUI.
- Final de carrera para prevenir movimientos imposibles del motor.
- Comunicación con el ordenador a través de un puerto USB.
- Necesidad de un microcontrolador con un módulo CCP para la captura de la frecuencia de oscilación del circuito resonador LC.

2.3 Requisitos *software*

La parte *software* tiene las siguientes exigencias:

- Comunicación con el PIC para conmutar bobinas, subir y bajar motor e iniciar análisis.
- Lectura de la posición del motor y de la frecuencia en cada instante de tiempo deseado.
- Cálculo de la concentración de la muestra a partir de la frecuencia leída.
- Guardado de los datos leídos en una base de datos.

2.3.1 Requisitos GUI

En lo referente a la GUI, debe ser capaz de:

- Visualización gráfica de los datos.
- Escoger rango de tiempo en la visualización.
- Comunicación con el software para ejecutar las distintas funciones de este.
- Recuperar sesión guardada.
- Descargar los datos para su posible posprocesado con otro *software*.

2.4 Fases del proyecto

En esta sección expondremos las fases del proyecto (figura 2.1), describiendo su contenido de manera generalizada.

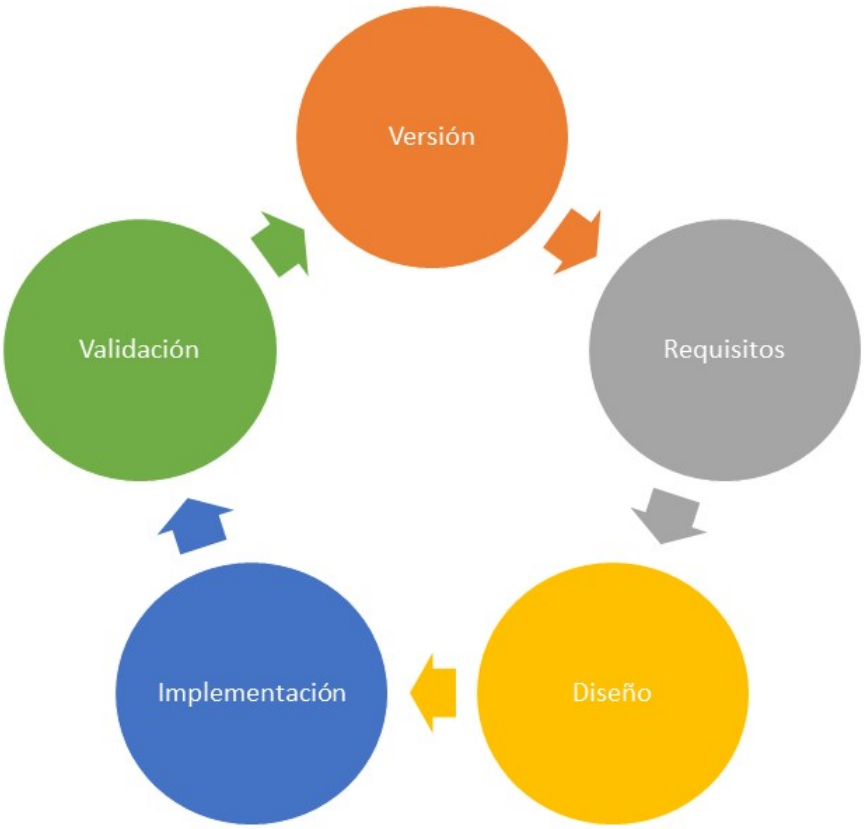


Figura 2.1 – *Principales fases de un proyecto.*

Las distintas fases son:

- **FASE I- Especificaciones del sistema:** estudio detallado de los requerimientos del sistema.
- **FASE II- Análisis y Diseño:** una vez se han determinado todos los requerimientos y espeficicaciones en la fase anterior concebimos la composición del sistema. Esto incluye considerar todos los elementos *hardware*, *software* y mecánicos.Determinaremos la utilidad dentro del sistema de los distintos dispositivos eléctricos o mecánicos utilizados. Diseñaremos y simularemos mediante un software específico.
- **FASE III- Desarrollo e Implementación:** fabricamos todo lo diseñado en la fase

II. Una vez hecho se desarrolla el algorítmo con el que programaremos nuestro PIC y el *software* asociado.

- **FASE IV- Testeo y Validación:** después de concluir el proceso de fabricación e implementación, solo queda validar el funcionamiento global del conjunto, resolviendo defectos y anomalías *hardware* y depurando posibles fallos *software* y aportando algunas mejoras puntuales. Una vez hecho esto pasamos a la obtención de datos y resultados.

- **FASE V- Versiones:** una vez el sistema esta terminado y es estable se trabaja en futuras versiones que incluyan alguna mejora adicional.

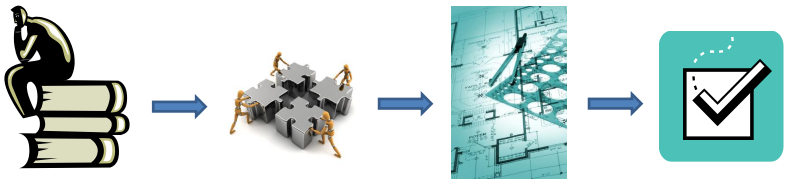


Figura 2.2 – *Analogía de fases.*

CAPÍTULO

3

ANÁLISIS DEL SISTEMA.

En este capítulo estudiaremos las soluciones que se van a adoptar para cumplir con los requisitos y especificaciones que se plantean en el capítulo 2.Para ello estudiaremos y analizaremos los recursos necesarios para el movimiento del sistema, el control del mismo y PIC encargado de controlar el sistema *hardware*.

Adicionalmente vamos a realizar una planificación del proyecto desgranando las distintas fases y fijando un tiempo concreto para su desarrollo.

3.1 Análisis del sistema

3.1.1 Sistema micro-controlador

El primer paso para el diseño de un prototipo es el análisis del PIC que se va a usar para el control del sistema y los periféricos. Debido a que la finalidad del proyecto es crear un producto industrializable hemos decidido usar un PIC, en concreto el 18f4550 (figura 3.1). Esta decisión viene impuesta por la finalidad del producto, puesto que necesitamos plena libertad a la hora de diseñar la PCB en forma, tamaño y en componentes, lo que nos hizo descartar los sistemas de placas de desarrollo, como podría ser Arduino.

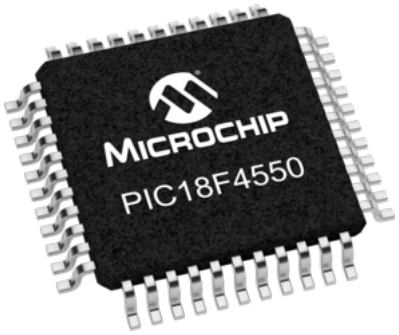


Figura 3.1 – *PIC 18F4550.*

La elección se realizó por pasos:

- Primero decidimos el tamaño de registro, elegiendo 8 bits, puesto que la cantidad de datos a tratar no va a ser lo suficiente elevado como para hacer uso de todos los registros, por tanto, en caso de ser necesario almacenar una variable mayor a 255, se podría hacer usando varios registros.

- Una vez seleccionado el tamaño de registro pasamos a la búsqueda de una familia que soportase la comunicación USB, lo que nos hizo decantarnos por la familia 18f, ya que son los más ecónomicos con esta característica.

- Finalmente hablamos con el departamento para ver el stock de microcontroladores de esa familia, para cruzarlo con las características restantes que debía de tener nuestro PIC y elegir uno. La elección final fue el 18f4550 (figura 3.1).

Este PIC dispone de comunicación USB, 1 K-byte de RAM para este, tres interrupciones externas, cuatro modulos de Timer, dos módulos CCP y puede ser programado mediante ICSP, por lo que según lo expuesto en el capítulo 2, cumple todos los requisitos pedidos.

3.1.2 Análisis del oscilador

El oscilador del que partimos es el del prototipo que mencionamos en el capítulo 1. En la figura 3.2 podemos observar el oscilador encuadrado [7].

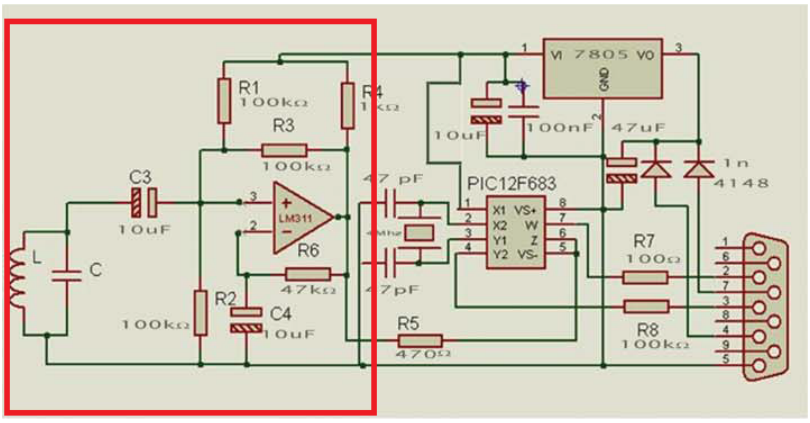


Figura 3.2 – *Electrónica prototipo.*

3

Una vez tenemos de donde partimos, aislamos el oscilador(figura 3.3) para su análisis a mano.

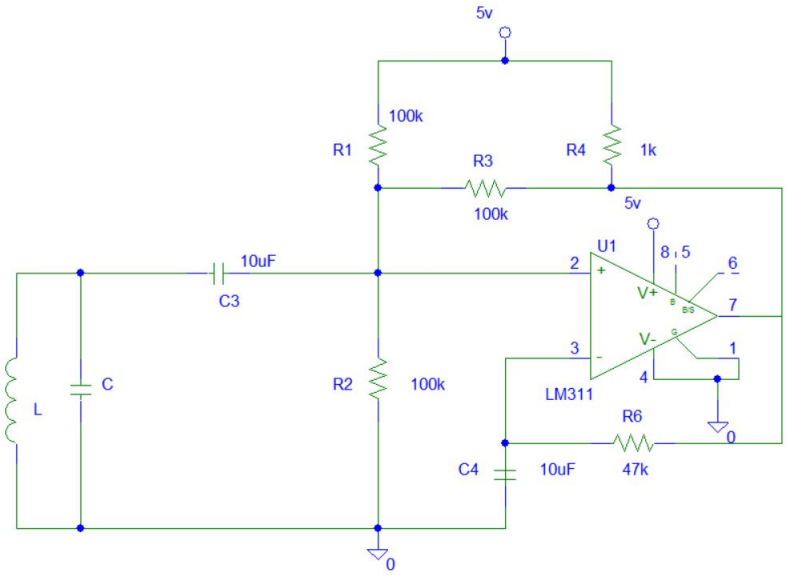


Figura 3.3 – *Oscilador a analizar.*

El análisis se hará en dos partes diferenciadas, primero el conjunto LC que nos fijan la frecuencia de oscilación de la señal en V_+ mediante el condensador de desacoplo C3. Después el circuito restante que aislado actúa como un oscilador debido a la carga y descarga del condesador C4. Demostremos que la frecuencia de oscilación en V_+ es la que viene impuesta por LC con C3 en desacoplo, sabemos que esta frecuencia es la frecuencia de oscilación del circuito LC, que es aquella a la que la impedancia del circuito es real, es decir, la reactancia inductiva es igual a la reactancia capacitiva. Teniendo en cuenta que en nuestro circuito L y C se encuentran en paralelo trabajaremos en términos de admitancia(3.1.1):

$$
\begin{aligned}
Y(j\omega) = j\omega C + 1/(j\omega L) = 0 \\
-\omega^2 CL/(j\omega L) + 1/(j\omega L) = 0 \\
\omega^2 CL = 1 \\
\omega = \frac{1}{\sqrt{LC}}\ rad/s \\
\omega = 2\pi f \\
f = \frac{1}{2\pi\sqrt{LC}}\ Hz
\end{aligned}
\tag{3.1.1}
$$

Podemos ver como la oscilación final del circuito se corresponde con la oscilación del resonador LC (figura 3.4), apoyando la teoría con una simulación en PSPIECE:

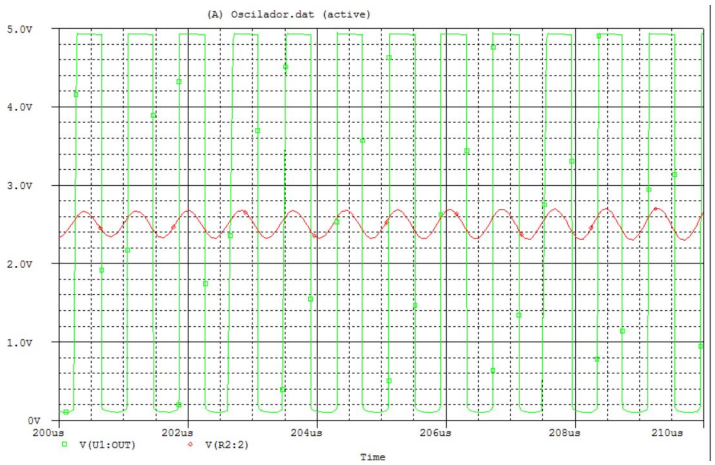


Figura 3.4 – *Comparativa señal en oscilador LC y a la salida del comparador.*

Analizaremos ahora el resto del circuito(figura 3.5) para su mejor comprensión:

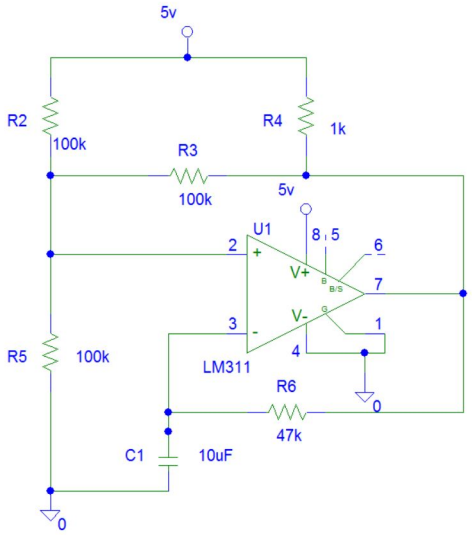
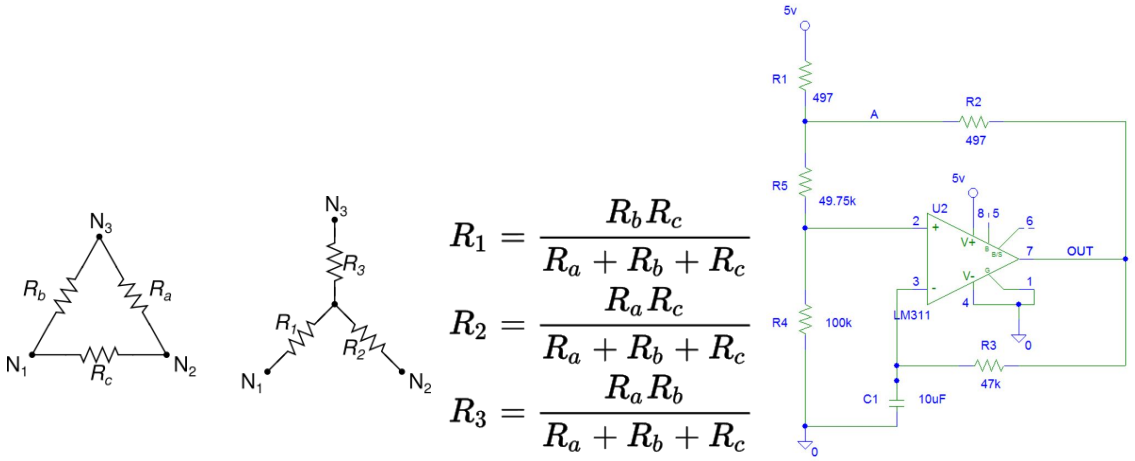


Figura 3.5 – *Oscilador LM311.*

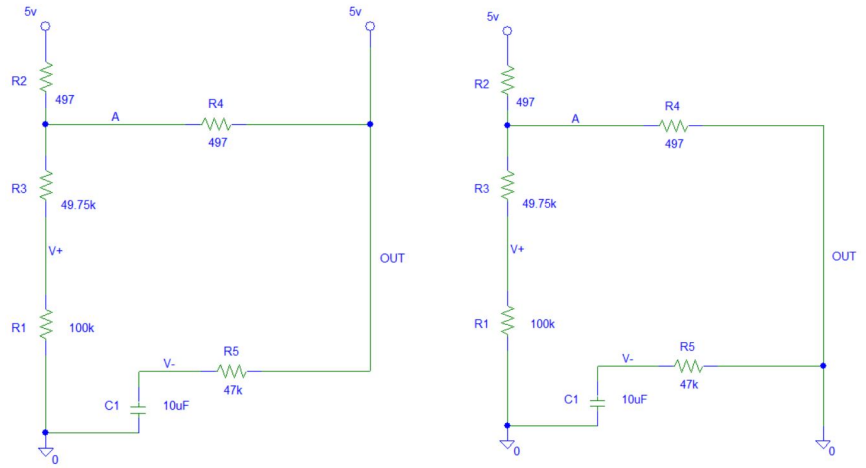
Al iniciar el análisis podemos apreciar como las resistencias R2, R3 y R4 se encuentran formando un triángulo por lo que aplicando el teorema de Kennelly (figura 3.6) obtenemos el circuito equivalente (figura 3.6):



(a) *Teorema de Kennelly* **(b)** *Circuito equivalente*

Figura 3.6 – *Teorema de Kennelly y el circuito equivalente resultante*

Dependiendo de las entradas V^+ y V^- del comparador LM311 la salida, el nodo OUT, estará a 5V, cuando $V^+ > V^-$, o 0V en caso contrario, lo que cortocircuitaría la salida virtualmente a la alimentación o a masa (figura 3.7):



(a) *Salida en alta* **(b)** *Salida en baja*

Figura 3.7 – *Circuitos equivalentes según salida LM311*

Por lo tanto el problema queda reducido a la carga y descarga de un condensador, C1. Mientras el condensador se está cargando $V^+ > V^-$y por tanto la salida está en alta, una vez que el condensador ha almacenado suficiente energía $V^- > V^+$ la salida cambia a baja por lo que C1 comienza a descargarse hasta que $V^+ > V^-$. Calculemos cual es el punto de

corte V^+ para el cual conmuta el comparador.

Si nos encontramos en la situación en la que $V^+ > V^-$(figura 3.7) apreciamos como las resistencias R1 y R2 se encuentran en paralelo, por tanto el circuito a analizar sería un divisor de tensión con $R = R1||R2$ en serie con R3 (figura 3.8), y ya que $R1 = R2$ entonces $R = R1/2$ por tanto $V+$ quedaría como (3.1.2):

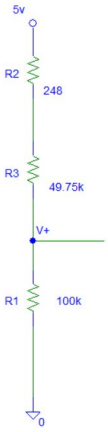


Figura 3.8 – *Divisor de tensión resultante.*

$$V^+ = Vcc\left(\frac{R4}{R + R4 + R3}\right) = 3.3\ V \tag{3.1.2}$$

En el caso en el que $V^- > V^+$ tenemos dos divisores de tensión seguidos (figura 3.9), por lo tanto obtendríamos V^+ (3.1.3) encadenando los cálculos de ambos:

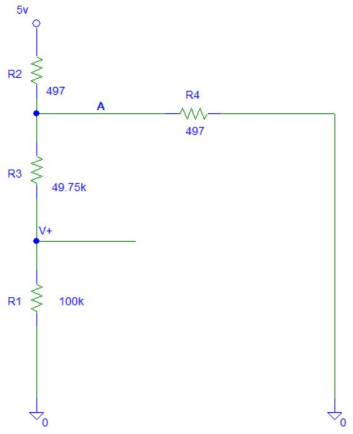


Figura 3.9 – *Divisores de tensión resultantes.*

$$\begin{aligned} V^A &= Vcc\left(\frac{R2}{R1+R2}\right) = 2.5\ V \\ V^+ &= V^A\left(\frac{R4}{R3+R4}\right) = 1.67\ V \end{aligned} \tag{3.1.3}$$

Analicemos ahora la carga y descarga de C1.Supongamos que el comparador acaba de conmutar a alta y por tanto el condensador está completamente descargado, como $v_c(0) = 0$ ya que la tensión ha de mantenerse continua comienza la carga del condensador (3.1.4):

$$\begin{aligned} V &= iR + v_c \\ i &= C\frac{dv_c}{dt} \\ RC\frac{v_c}{dt} + v_c &= V \\ v_c(t) = V(v_c(0) - V)&\left(1 - e^{\left(\frac{-t}{RC}\right)}\right) \\ v_c(t) &= V\left(1 - e^{\left(\frac{-t}{RC}\right)}\right)\ V \end{aligned} \tag{3.1.4}$$

En el momento que $V^- > V^+$ empieza a descargarse el condensador (3.1.5) y como la tensión ha de mantenerse continua $v_c(0) = V$:

$$\begin{aligned} 0 &= iR + v_c \\ i &= C\frac{dv_c}{dt} \\ RC\frac{v_c}{dt} + v_c &= 0 \\ v_c(t) &= Ve^{\left(\frac{-t}{RC}\right)}\ V \end{aligned} \tag{3.1.5}$$

Podemos ver como los cálculos hechos a mano se corresponden con lo simulado con PSPIECE (figura 3.10):

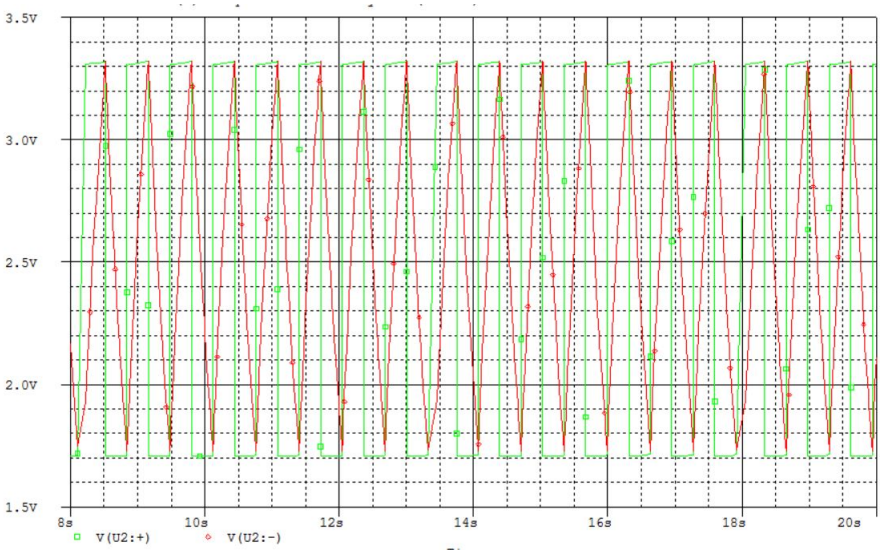


Figura 3.10 – *Resultados simulación.*

3

3.1.3 *Rotary encoder*

Los *rotary encoders* (figura 3.11) son dispositivos electromecánicos que se usan para convertir la posición angular de un eje a un código digital. Existen principalmente dos tipos:

- Absolutos: producen un código digital único para cada ángulo distinto del eje.
- Relativos o incrementales: miden el cambio de ángulo a partir de un ángulo dado.

Figura 3.11 – *Rotary con pulsador.*

Puesto que a nosotros nos interesa usar el rotary encoder para controlar el aparato para realizar medidas esporádicas sin estar necesidad de que este esté conectado a un ordenador vamos a usar uno relativo. Ya que existe la opción y vamos a necesitar un pulsador elegimos el modelo de ALPS $EC12D$ que incluye un rotary encoder con pulsador. Este modelo tiene 5 patillas, 2 para el pulsador y 3 para la lectura de pulsos.

Para detectar el giro conectamos una patilla común a masa y las otras dos a la la alimentación con una resistencia de *pull up*, esto hace que al girar el encoder conecte la patilla A y la B a masa con un pequeño desfase [5]. Observando la patilla que baja antes detectamos el sentido del movimiento (figura 3.12).

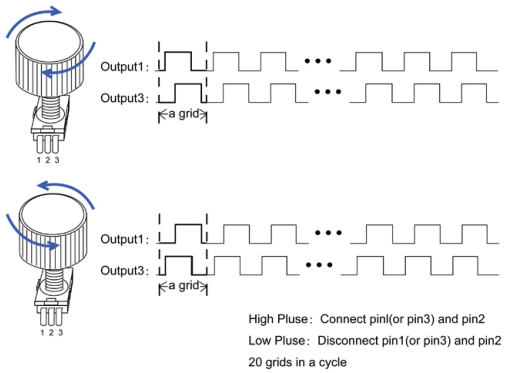


Figura 3.12 – *Pulsos generados al mover el rotary.*

3.1.4 Comunicación USB

A la hora de determinar el tipo de comunicación USB debemos fijarnos en los tipos permitidos por el PIC elegido. En este caso permite el estándar 2.0 y sus dos tipos de configuraciones: Low Speed (1.5 Mb/s) y Full Speed (12 Mb/s). Debido a que el volumen de datos con el que vamos a trabajar es pequeño podríamos trabajar con configuración Low Speed, pero, a la hora de programar sería más tedioso por lo que recurriremos a una opción incluida en el *datasheet* [8] llamada Streaming Parallel Port (SSP), mediante la cual el PC reconocerá el USB como un puerto serie. Debido a que es un estándar más antiguo se dispone de mucha más información para trabajar con SSP, además, Python tiene una librería dedicada exclusivamente al puerto serie por lo que la comunicación a nivel software se ve simplificada.

3.1.5 Conmutación de bobina

Dentro de la multitud de posbiles soluciones para usar un dispositivo como interruptor controlado por un PIC nos quedamos con dos opciones:

- Interruptor CMOS: Esta opción se consideró inicialmente como válida pero se descartó puesto que teníamos una señal del orden de ± 50 mV necesitaríamos un CMOS con alimentación dual para no perder toda la señal por debajo de 0 V, por lo que por consideraciones de coste descartamos esta opción.

- MOSFET como interruptor: puesto que el MOSFET permite que fluya la corriente a través de él no tenemos el problema anterior de un voltaje demasiado pequeño, por lo que esta ha sido la solución final.

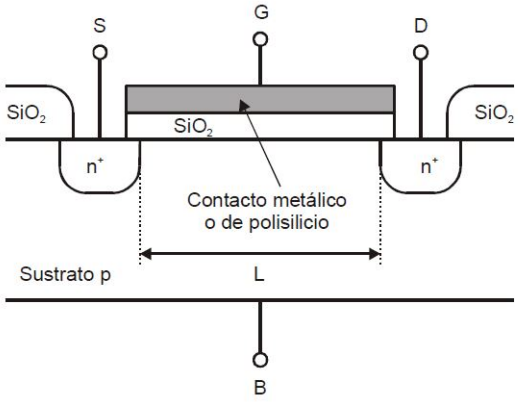


Figura 3.13 – *Estructura interna MOSFET.*

El MOSFET dispone de cuatro terminales (figura 3.13) llamados puerta (G, gate), drenador (D, drain), fuente (S, source) y sustratro (B, bulk). Para que exista corriente entre la fuente y el drenador es necesario establecer un camino por el cual fluya la misma, lo cual sólo es posible si se crea una lámina de inversión de carga de electrones que una las dos regiones N^+ (figura 3.14).

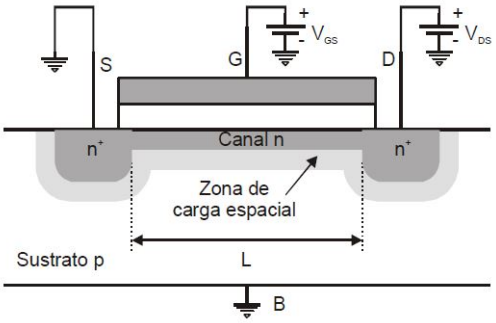


Figura 3.14 – *MOSFET con canal creado entre D y S.*

Existen cuatro tipos de MOSFET según el tipo de canal y si es de enriquecimiento o de deplexión. En nuestro caso nos hemos decantado por un MOSFET de canal N de enriquecimiento en el que el sustrato semiconductor es tipo P y no existe canal a menos que apliquemos la tensión necesaria a la puerta para que esto ocurra.

3.2 Planificación del proyecto

3.2.1 Fases constitutivas del proyecto

El trabajo del proyecto se dividirá en distintas fases que se irán sucediendo de manera secuencial hasta tener el primer prototipo, a partir de ese momento se podrá trabajar de manera concurrente. De esta manera pretendemos simular un organigrama de tareas que se asemeje al ritmo de trabajo habitual en la creación de proyectos a nivel empresarial.

Las fases constitutivas que formarán el proyecto (figura 3.15) son:

- Análisis
- Diseño *hardware*
- Diseño *software*
- Diseño mecánico
- Implementación
- Test del producto
- Validación

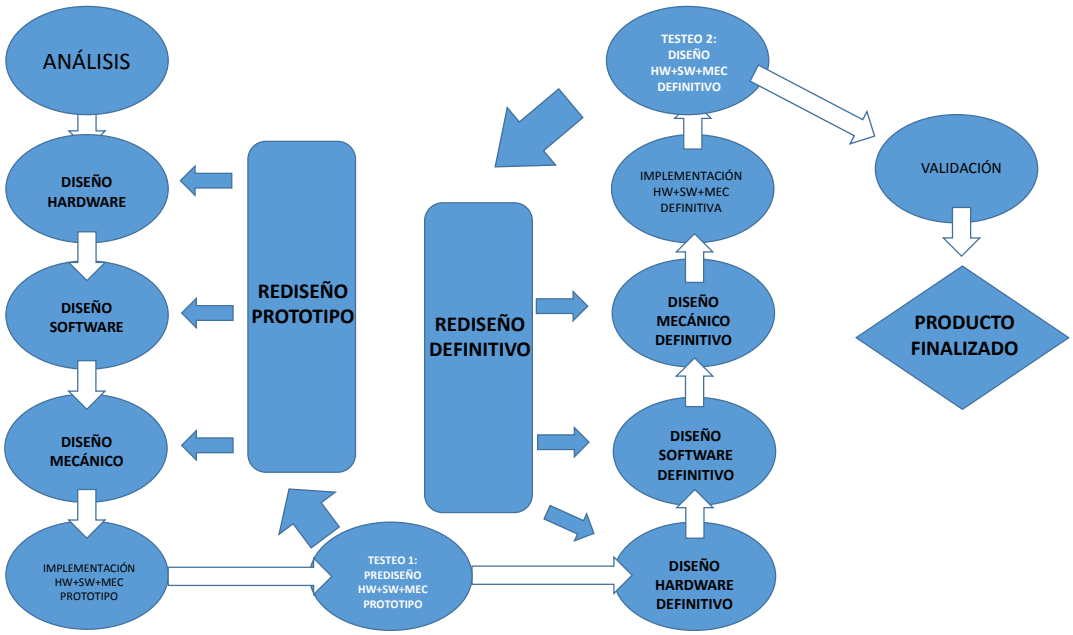


Figura 3.15 – *Fases del proyecto.*

Una vez hecho el análisis documentado en este capítulo iniciamos el diseño *hardware* del producto. Analizados y definidos los bloques funcionales del proyecto pasamos a buscar soluciones específicas a nivel eléctrico, es decir, buscamos los componentes que cumplen nuestros requerimientos.

El primer prototipo con el que trabajamos fue facilitado por el Departamento de Electrónica, consistente en un $PIC18F4550$ con una conexión USB y preparado para su programación mediante ICSP en una placa de baquelita en la que los pines A y B del PIC erán accesibles para conexión o medición con osciloscopio. A este primer prototipo fuimos añadiendo distintos módulos para ir probando las distintas partes de nuestro sistema.

Una vez teníamos el primer prototipo pasamos al diseño de *software*, en el que en englobamos la programación del PIC, la programación en Python para la conexión del PIC con el PC y la programación en HTML para la GUI. Aquí al igual que en el diseño *hardware* obtuvimos un primer prototipo funcional y lo implementamos para ir añadiéndole pequeñas mejoras funcionales.

Una vez el hardware y el software están listos pasamos a la parte mecánica. En este punto la forma de trabajar fue siempre la misma, diseñar en Solidworks, imprimir en $3D$ y probar, para retocar y mejorar las partes del diseño que veamos necesarias en la siguiente iteración.

Una vez listo el primer prototipo con los distintos módulos funcionando podemos pasar al diseño definitivo, en el que vistas las carencias de nuestro primer prototipo las solucionamos y creamos un prototipo definitivo. Volvemos a testear todo en nuestro nuevo prototipo y en caso de que todo este correcto pasamos al diseño final del producto.

La siguiente fase sería la validación, en la que se comprueba si todo es correcto damos el producto por finalizado, en caso contrario habría que volver atrás, introducir los cambios necesarios y repetir este paso.

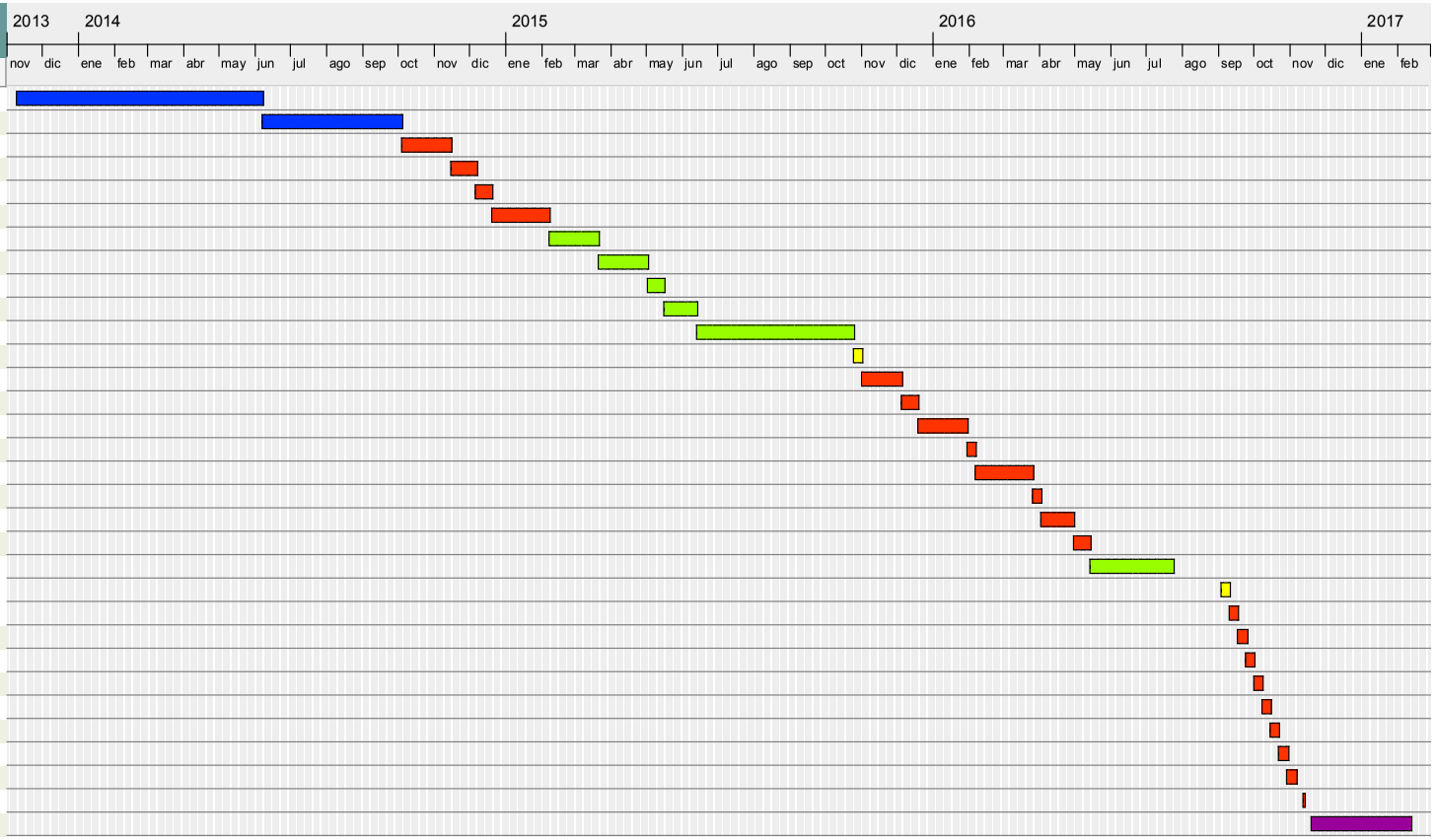
3.2.2 Planificación temporal

En la siguiente página incluimos un diagrama de Gantt con los plazos propuestos para el desarrollo del proyecto. En azul están las tareas dedicadas a documentación, en rojo las dedicadas a *hardware*, en verde las dedicadas a *software*, en amarillo las dedicadas a la parte mecánica y en morado las que engloban ambas partes.

Planificación proyecto

Diagrama de Gantt

Nombre	Fecha de inicio	Fecha de fin
Familiarización herramientas	9/11/13	7/06/14
Documentación proyecto	7/06/14	4/10/14
Diseño esquema placa oscilador	4/10/14	15/11/14
Diseño PCB placa oscilador	15/11/14	7/12/14
Fabricación placa oscilador	6/12/14	20/12/14
Testeo placa oscilador	20/12/14	7/02/15
Programación puerto serie	7/02/15	21/03/15
Programación rutina lectura frecuencia	21/03/15	2/05/15
Programación movimiento motor	2/05/15	16/05/15
Programación señal PWM.	16/05/15	13/06/15
Creación GUI	13/06/15	25/10/15
Diseño modelo 3D	25/10/15	1/11/15
Diseño esquema placa base	1/11/15	5/12/15
Diseño esquema placa motor	5/12/15	19/12/15
Diseño PCB placa base	19/12/15	30/01/16
Diseño PCB placa motor	30/01/16	6/02/16
Fabricación placa base	6/02/16	26/03/16
Fabricación placa motor	26/03/16	2/04/16
Testeo placa base	2/04/16	30/04/16
Testeo placa motor	30/04/16	14/05/16
Unificación del software	14/05/16	24/07/16
Resiseño final modelo 3D	3/09/16	10/09/16
Rediseño esquema final placa oscilador	10/09/16	17/09/16
Rediseño esquema final placa base	17/09/16	25/09/16
Rediseño esquema final placa motor	24/09/16	1/10/16
Rediseño PCB final oscilador	1/10/16	8/10/16
Rediseño PCB final placa base	8/10/16	15/10/16
Rediseño PCB final placa motor	15/10/16	22/10/16
Fabricación placa final oscilador	22/10/16	30/10/16
Fabricacion placa base final	29/10/16	6/11/16
Fabricacion placa final motor	12/11/16	13/11/16
Ensamblaje y testeo final	19/11/16	12/02/17



3

CAPÍTULO

4

DISEÑO DEL SISTEMA.

Una vez analizados los requisitos y decididas las soluciones que vamos a adoptar en la realización del producto (figura 4.1), en el presente capítulo abordaremos las cuestiones concernientes al diseño *hardware*, mecánico, y *software*.

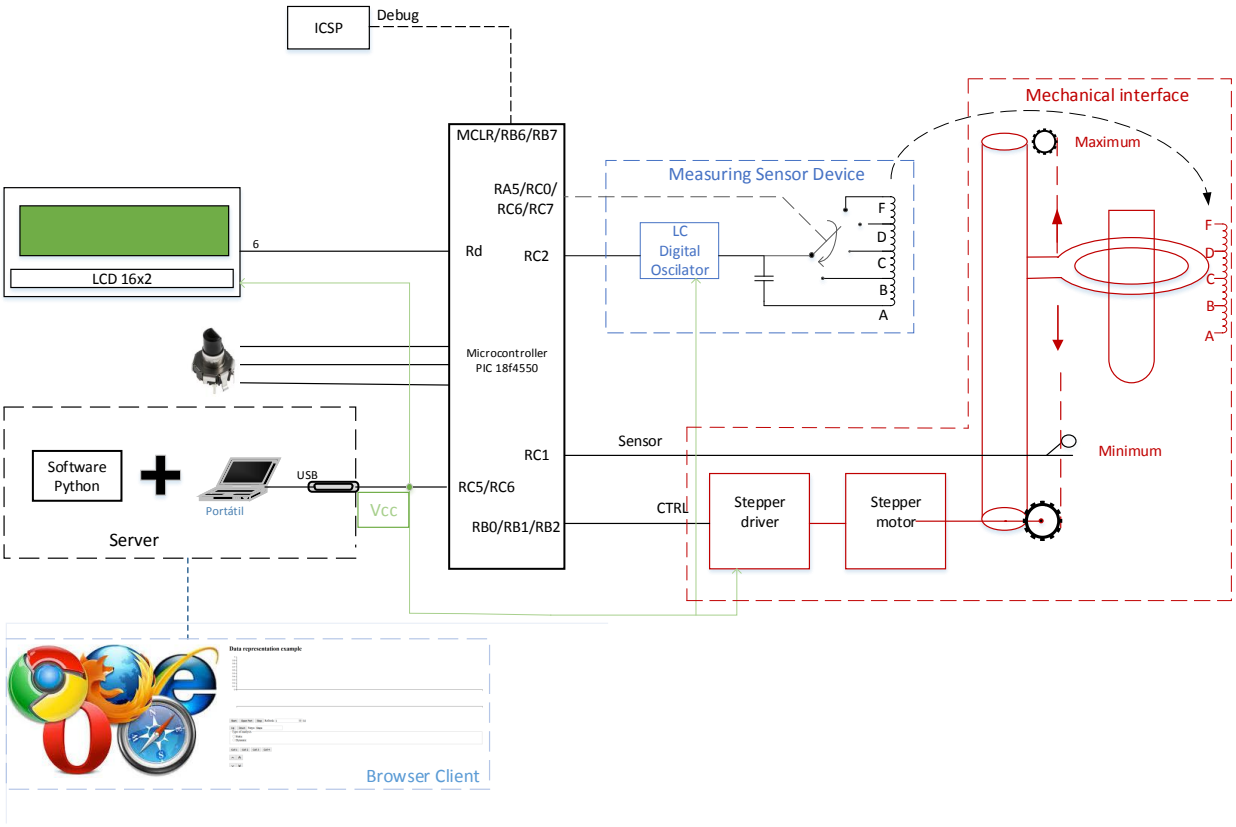


Figura 4.1 – *Diagrama del MagnetoScanner*

4.1 Diseño *Hardware*

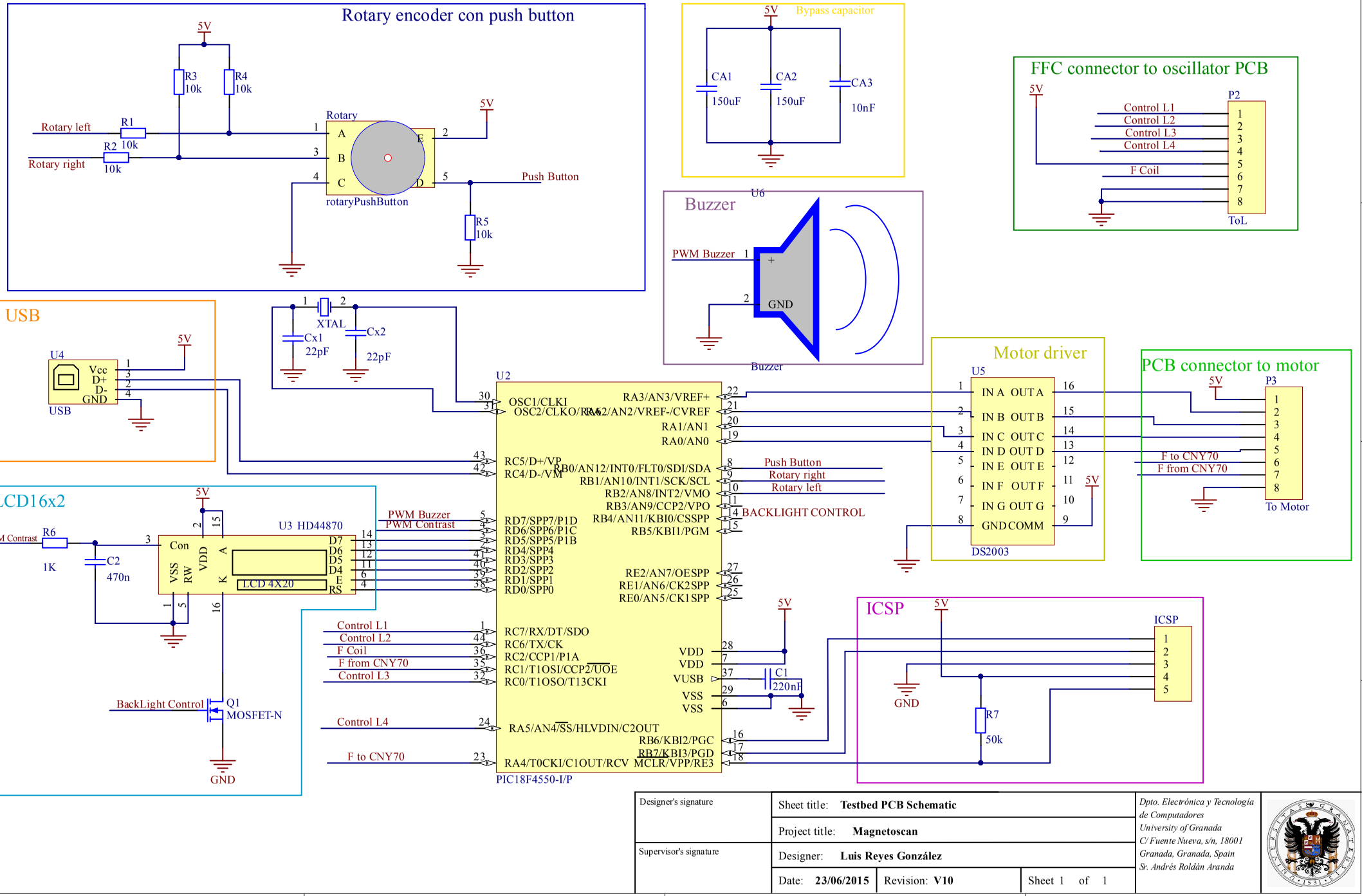
Para explicar el funcionamiento de todo el sistema hemos decidido dividir el mismo en tres esquemáticos distintos, uno por cada placa que construiremos. Antes de nada enumeremos las placas para su posterior análisis:

- Placa base.
- Placa motor.
- Placa oscilador LC.

En cada sección veremos el esquemático del circuito y el diseño $2D$ y $3D$ de la placa, todos ellos creados con la herramienta *software* Altium. Finalmente, incluiremos un apartado dedicado al cálculo del valor de la bobina puesto que tendremos que hilar manualmente esta.

4.1.1 Esquemáticos placa base

En la siguiente página incluimos el esquemático con el *pinout* del PIC $18f4550$ además de los módulos necesarios para el control y la comunicación del sistema. En este esquemático incluimos el *pinout* del PIC $18f4550$ además de los módulos necesarios para el control y la comunicación del sistema. En la siguiente página mostraremos el esquemático completo con el fin de tener una primera vista general del mismo, para luego explicar cada parte por separado:


Rotary encoder con push button
5V
R3 10k
R4 10k
Rotary left
R1
R2 10k
Rotary right
10k
Rotary
A
B
C
D
E
1
2
3
4
5
rotaryPushButton
Push Button
R5 10k
Bypass capacitor
CA1 150uF
CA2 150uF
CA3 10nF
FFC connector to oscillator PCB
P2
Control L1
Control L2
Control L3
Control L4
F Coil
ToL
Buzzer
U6
PWM Buzzer
GND
USB
U4
Vcc
D+
D-
GND
XTAL
Cx1 22pF
Cx2 22pF
U2
OSC1/CLKI
OSC2/CLKO/RA6
RA3/AN3/VREF+
RA2/AN2/VREF-/CVREF
RA1/AN1
RA0/AN0
RC5/D+/VP
RC4/D-/VM
RB0/AN12/INT0/FLT0/SDI/SDA
RB1/AN10/INT1/SCK/SCL
RB2/AN8/INT2/VMO
RB3/AN9/CCP2/VPO
RB4/AN11/KBI0/CSSPP
RB5/KBI1/PGM
RE2/AN7/OESPP
RE1/AN6/CK2SPP
RE0/AN5/CK1SPP
RC7/RX/DT/SDO
RC6/TX/CK
RC2/CCP1/P1A
RC1/T1OSI/CCP2/UOE
RC0/T1OSO/T13CKI
VDD
VUSB
VSS
RA5/AN4/SS/HLVDIN/C2OUT
RB6/KBI2/PGC
RB7/KBI3/PGD
RA4/T0CKI/C1OUT/RCV
MCLR/VPP/RE3
PIC18F4550-I/P
Push Button
Rotary right
Rotary left
BACKLIGHT CONTROL
Motor driver
U5
IN A OUT A
IN B OUT B
IN C OUT C
IN D OUT D
IN E OUT E
IN F OUT F
IN G OUT G
GND COMM
DS2003
PCB connector to motor
P3
F to CNY70
F from CNY70
To Motor
LCD16x2
U3 HD44870
PWM Contrast
R6 1K
C2 470n
Con
VSS
VDD
RW
A
K
LCD 4X20
D7
D6
D5
D4
E
RS
PWM Buzzer
PWM Contrast
Control L1
Control L2
F Coil
F from CNY70
Control L3
Control L4
F to CNY70
BackLight Control
Q1
MOSFET-N
GND
C1 220nF
ICSP
R7 50k
Designer's signature
Supervisor's signature
Sheet title: Testbed PCB Schematic
Project title: Magnetoscan
Designer: Luis Reyes González
Date: 23/06/2015
Revision: V10
Sheet 1 of 1
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andrés Roldán Aranda

4.1.1.1 PIC18F4550

En la sección 3.1.1 ya se explicó el motivo de la elección de este PIC haciendo referencia a todas sus características, por lo que no es necesario repetirlas en este punto. El *pinout* del $18F4550$ podemos verlo en la figura 4.2 junto con las conexiones necesarias para el correcto funcionamiento del mismo.

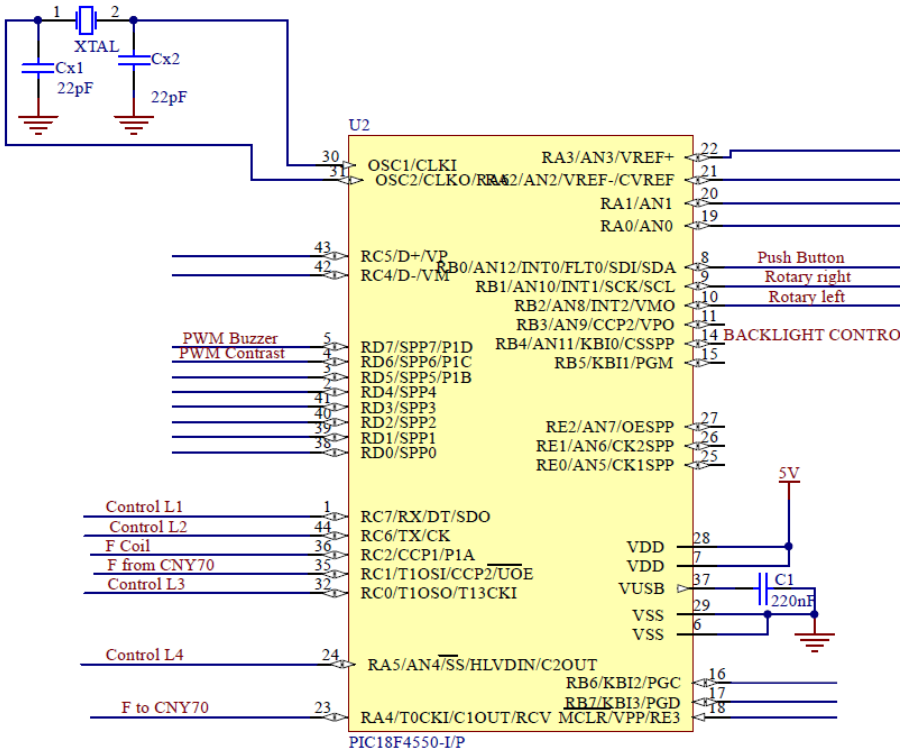


Figura 4.2 – *Pinout y conexiones del 18F4550.*

A pesar de tener un oscilador incorporado, el microcontrolador no puede funcionar sin componentes externos que estabilizan su funcionamiento y determinan su frecuencia (velocidad de operación del microcontrolador). Dependiendo de los elementos utilizados así como de las frecuencias el oscilador puede funcionar en cuatro modos diferentes:

- Crystal de bajo consumo(LP).
- Crystal/Resonator(XT).
- Crystal/resonator de alta velocidad (HS).
- Resistencia/condensador (RC).

En nuestro caso hemos usado un reloj de cuarzo de alta velocidad a 16 Mhz. Esta decisión se tomó cruzando los datos de los valores que el cristal puede tomar para el correcto funcionamiento de la comunicación USB, presentes en el datasheet [8], con los valores presentes en el departamento de electrónica. Los valores de los condensadores también fueron obtenidos a partir del datasheet del componente (figura 4.3).

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
XT	4 MHz	27 pF	27 pF
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	20 MHz	15 pF	15 pF

Capacitor values are for design guidance only.

These capacitors were tested with the crystals listed below for basic start-up and operation. **These values are not optimized.**

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following this table for additional information.

Crystals Used:
4 MHz
8 MHz
20 MHz

Figura 4.3 – *Valores de los condensadores según frecuencia cristal*

El condensador $C1$ entre VUSB y masa es debido al modo de alimentación USB elegido(figura 4.4), en el que todo el sistema está alimentado por los 5V del USB, lo que nos limita la intensidad a 500 mA [6].

17.6.1 BUS POWER ONLY

In Bus Power Only mode, all power for the application is drawn from the USB (Figure 17-10). This is effectively the simplest power method for the device.

FIGURE 17-10: BUS POWER ONLY

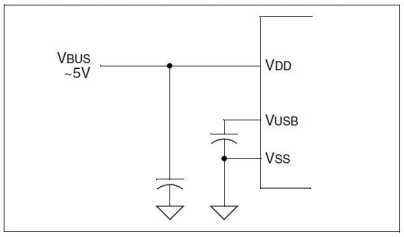


Figura 4.4 – *Método de alimentación sólo por bus.*

4.1.1.2 LCD 2x16

Para la visualización de la información básica se ha escogido una pantalla LCD de 2 líneas y 16 carácteres en cada línea, comunmente conocida como LCD $2x16$. Se decidió por este tamaño debido a que su pequeño tamaño, los datos a mostrar serán los menús y la información relativa al análisis como puede ser la posición y la concentración por lo que con 32 carácteres será suficinente. La conexión se ha realizado según lo mostrado en la figura 4.5:

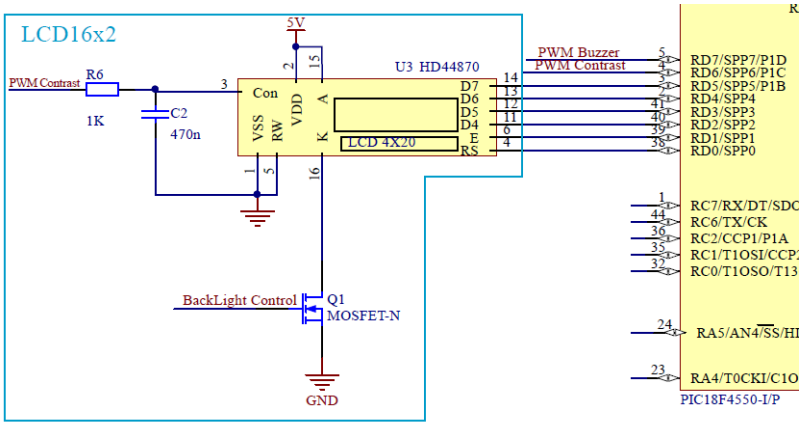


Figura 4.5 – *Conexión LCD.*

La comunicación entre el PIC y la LCD se hace mediante los pines $RD4$ a $RD7$. Como no vamos a leer nada de la LCD el pin RW se conectará directamente al valor 0 que indica escritura, por lo que lo conectamos a masa. $RD0$ controlará la selección de registro y finalmente $RD1$ controlará el inicio de la escritura mediante un valor en alta, desactivando todas las funciones si se coloca en baja.

El *backlight* es encendido o apagado mediante un MOSFET-N controlado desde el PIC.

El control del contraste se hace un filtro paso baja que a la entrada tiene una señal PWM controlada por el PIC, variando el tiempo en alta de la señal obtenemos los diferentes voltajes a la salida del filtro que nos dan los diferentes contrastes. La principal ventaja que tiene este método es que no hay que tocar la electrónica para cambiar el contraste, como se hacía antes al controlarlo mediante un potenciómetro.

4.1.1.3 USB

La alimentación y la conexión con el PC se hacen mediante USB. La conexión USB tiene cuatro pines: alimentación, masa, $D+$ y $D-$. La conexión con el PIC(figura 4.6) se hace interconectando los pines $D+$ y $D-$ con $RC5$ y $RC4$ respectivamente.

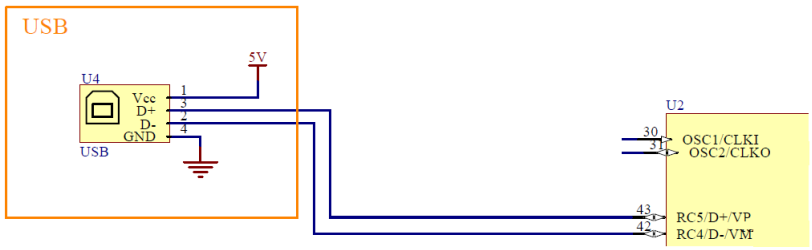


Figura 4.6 – *Conexión USB.*

4.1.1.4 Condensadores de desacoplo

Con el fin de evitar el rizado en la alimentación incluimos una etapa de desacoplo (figura 4.7) en la que tenemos condensadores electrolíticos de alta capacidad (150 μF)para desacoplar a bajas frecuencias y de baja capacidad (10 nF) para hacerlo a las altas.

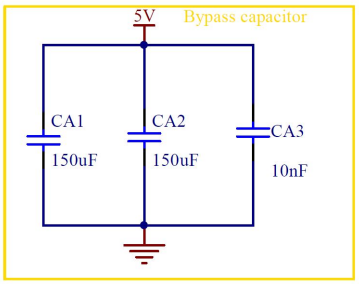
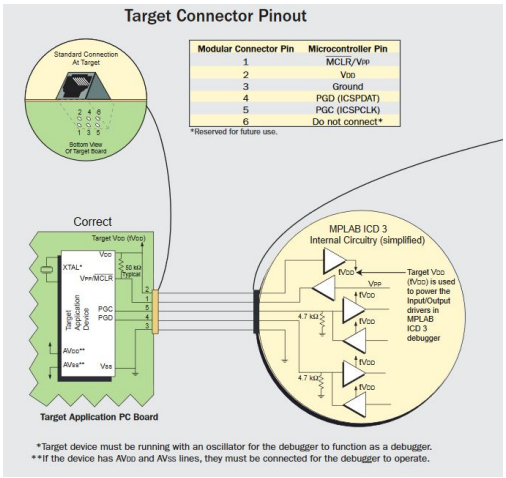
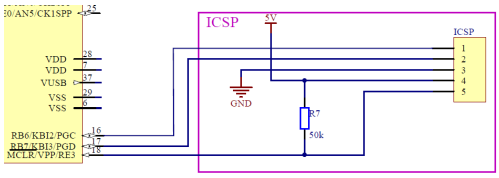


Figura 4.7 – *Etapa de desacoplo.*

4.1.1.5 ICSP(In-Circuit Serial Programming)

La programación serie en circuito se hará con un programador y debugeador de alta velocidad facilitado por el departamento de electrónica, en concreto el MPLAB ICD 3 de Microchip. En la siguiente imagen (figura 4.8) podemos ver en la parte de arriba la propia documentación del dispositivo en la que se indica como conectarlo y la conexión realizada en nuestra placa:





(a) *Documentación MPLAB ICD 3* **(b)** *Esquema montado en nuestra placa.*

Figura 4.8 – *Conexiones ICSP.*

4.1.1.6 Motor *driver*

El manejo del motor se realiza mediante un *driver* de Darlington, el cual se caracteriza por tener una gran ganancia en corriente debido a la incorporación de dos transistores bipolares en tándem. En concreto hemos usado el $DS2003$ de Texas Instruments puesto que facilitaban

muestras de este modelo. La conexión la podemos ver en la figura 4.9:

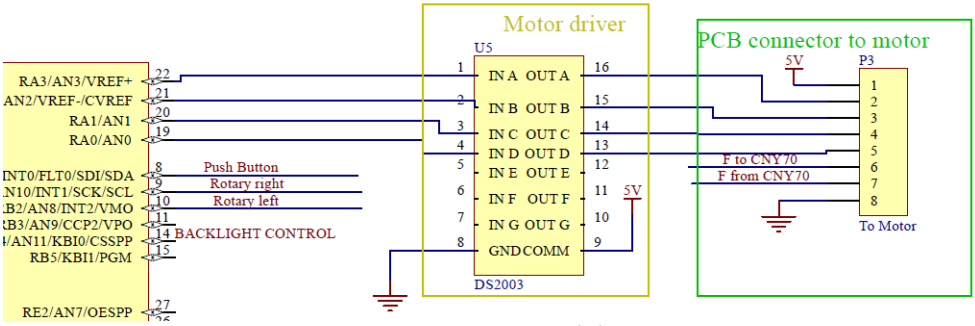
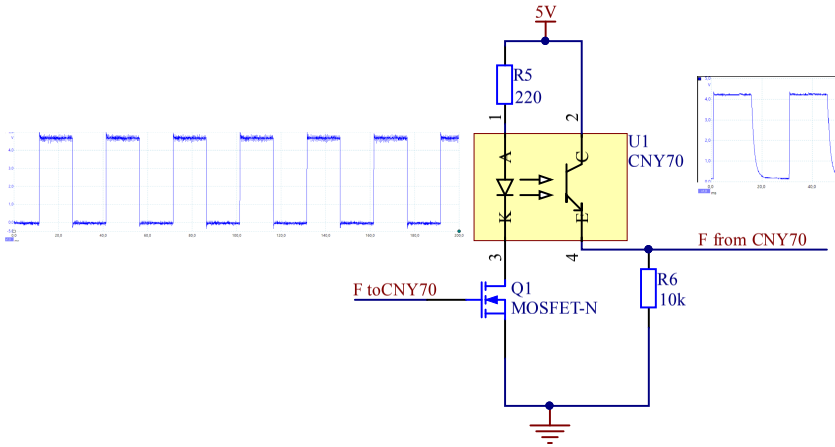


Figura 4.9 – *Driver del motor.*

4.1.2 Esquemáticos placa motor

En esta sección incluiremos la placa en la que va el motor y el final de carrera óptico. En la siguiente página incluimos el esquemático completo:

Optical EndStop

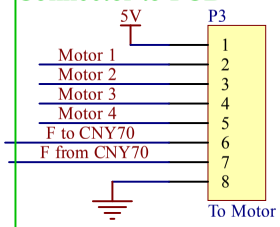


BASIC CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

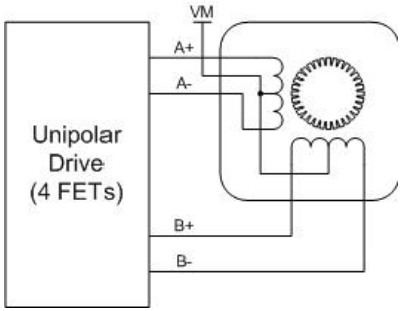
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
COUPLER						
Collector current	V_{CE} = 5 V, I_F = 20 mA, d = 0.3 mm (figure 1)	I_C (2)	0.3	1.0		mA
Cross talk current	V_{CE} = 5 V, I_F = 20 mA, (figure 2)	I_{CX} (3)			600	nA
Collector emitter saturation voltage	I_F = 20 mA, I_C = 0.1 mA, d = 0.3 mm (figure 1)	V_{CEsat} (2)			0.3	V
INPUT (EMITTER)						
Forward voltage	I_F = 50 mA	V_F		1.25	1.6	V
Radiant intensity	I_F = 50 mA, t_p = 20 ms	I_e			7.5	mW/sr
Peak wavelength	I_F = 100 mA	λ_P	940			nm
Virtual source diameter	Method: 63 % encircled energy	d		1.2		mm
OUTPUT (DETECTOR)						
Collector emitter voltage	I_C = 1 mA	V_{CEO}	32			V
Emitter collector voltage	I_E = 100 µA	V_{ECO}	5			V
Collector dark current	V_{CE} = 20 V, I_F = 0 A, E = 0 lx	I_{CEO}			200	nA

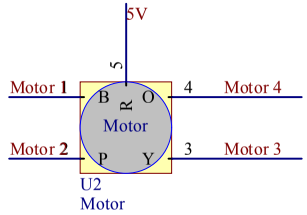
Notes
(1) Measured with the "Kodak neutral test card", white side with 90 % diffuse reflectance
(2) Measured without reflecting medium

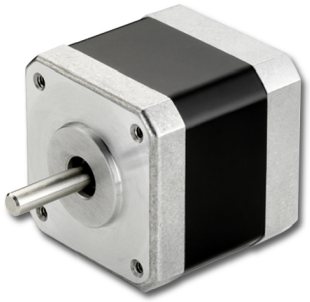
Connector to PCB



Stepper Motor







Designer's signature	Sheet title: **Testbed PCB Schematic**		*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **Magnetoscan**		
Supervisor's signature	Designer: **Luis Reyes González**		
	Date: **23/06/2015**	Revision: **V10**	Sheet * of *

El final de carrera seleccionado ha sido un modelo óptico debido a que estos son menos propensos a averías al no tener partes móviles, en conreto el dispositivo que usaremos es el $CNY70$. Contiene un emisor de radiación de luz infrarroja, es decir, un fotodiodo, y un recetor, fototransistor. En nuestro caso utilizamos el sensor como una entrada analógica puesto que únicamente nos interesa si está activo o no, así cuando está en baja indica que no existe nada que haga contacto con el fotodiodo y, por tanto, refleje la luz infrarroja por lo que el final de carrera no está activado. En el momento en el que existe una superficie para reflejar la luz infrarroja lo suficientemente cerca la señal pasa a estar en alta, lo que activa el final de carrera.

La configuración montada con el MOSFET tipo N y la señal periódica de frecuencia conocida para activar el PIC se usa para evitar que el final de carrera sea erróneamente activado por la luz solar, al tener esta un componente infrarrojo. Mediante este MOSFET cortocircuitamos la salida del LED infrarrojo a masa cuando la señal que le llega desde el PIC está en alta y dejamos en abierto cuando está en baja lo que implica que el LED se irá encendiendo y apagando a la frecuencia de la señal. Por tanto, en el fotorreceptor tendremos una señal con la misma frecuencia de la señal de control, que podremos leer en el PIC para determinar si el final de carrera está activo por una superficie cercana.

La resistencia colocada en el LED sirve para limitar la corriente que lo atraviesa, el valor se ha calculado a partir del *datasheet* [9] en el que nos da una corriente máxima de 50 mA por lo tanto puesto que alimentamos a 5 V la resistencia deberá ser mayor a 100 , el valor escogido es 220 Ω.

La resistencia colocada en el fototransistor se ha elegido para maximizar el voltaje que va al PIC, teniendo en cuenta que la corriente de colector es de 50 mA al iluminar el dispositivo según del *datasheet* [9] con una resistencia de 10 $k\Omega$ obtendríamos un valor de 5 V en el PIC.

La conexión del motor se realiza conectando la toma intermedia de las bobinas a alimentación y el inicio y final de cada bobina a cada uno de los pines de salida del *driver* del motor (figura 4.10).

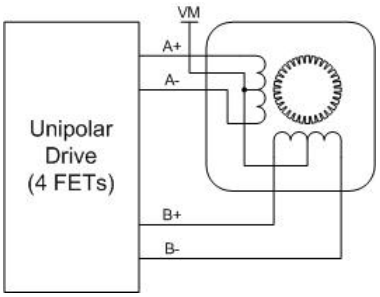
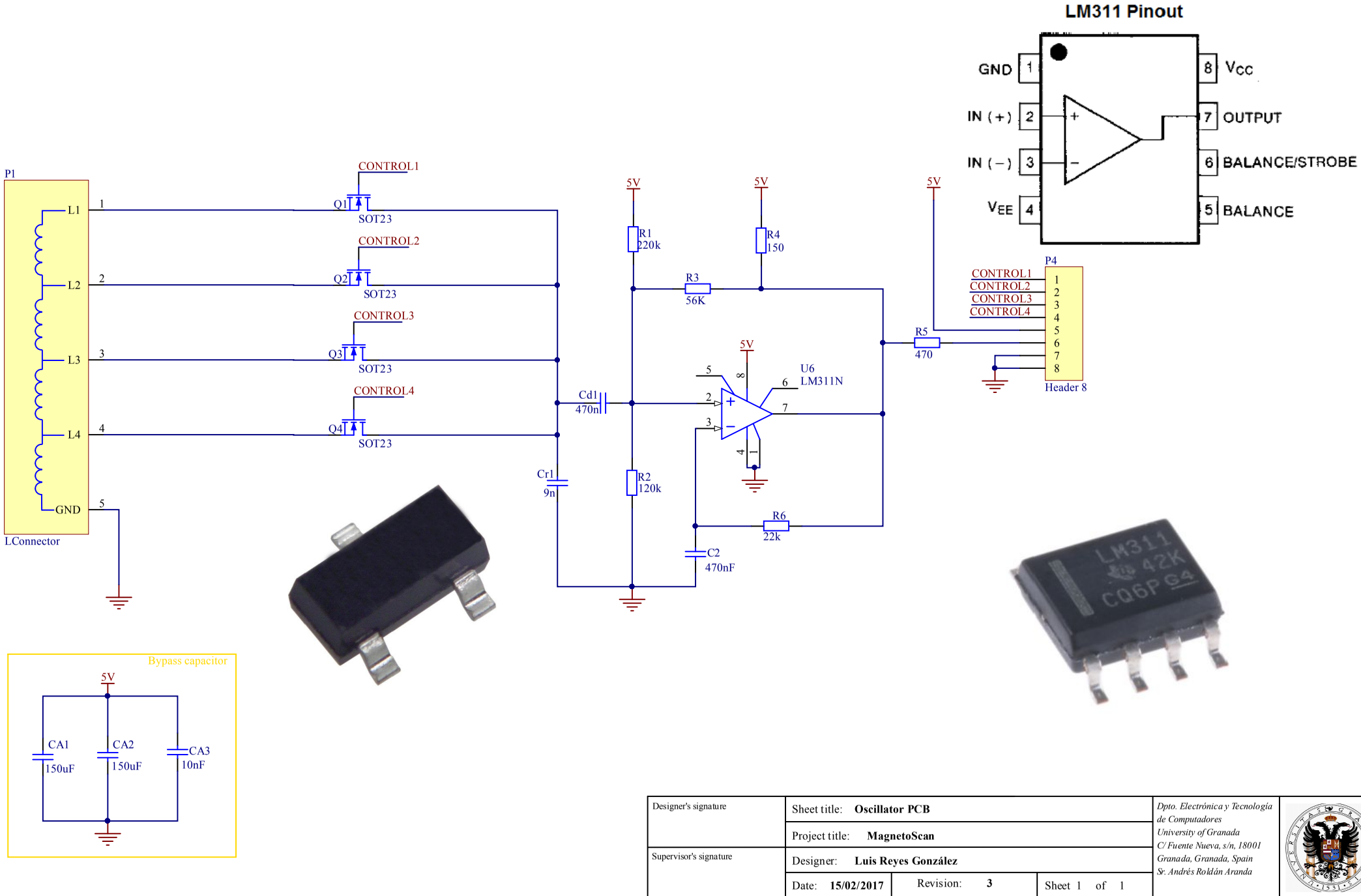


Figura 4.10 – *Conexión motor.*

4.1.3 Esquemáticos placa oscilador LC

El oscilador completo analizado en el capítulo 3 tiene su PCB propia. En un principio se barajó la idea de conmutar las bobinas desde la placa base y montar en ella el oscilador,

pero esto traería graves problemas de ruido al ser la señal proveniente de la bobina de valores muy bajos y la longitud del cable bastante alta (en torno a 20 cm) por lo que se decidió que la señal que debia mandarse a la placa base era la salida del $LM311$ con una amplitud de 5 V para evitar estos problemas de ruido. En la siguiente página incluimos un esquema completo de la placa:



Designer's signature	Sheet title: **Oscillator PCB**		*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **MagnetoScan**		
Supervisor's signature	Designer: **Luis Reyes González**		
	Date: **15/02/2017**	Revision: **3**	Sheet 1 of 1

Puesto que ya se analizó completamente el oscilador en el capítulo 3 no es necesario volver a hacerlo. El único cambio incluido ha sido añadir cuatro MOSFETs de canal N para conmutar la bobina *multi-tap*. Esta bobina consiste en una única con cuatro tomas distintas, lo que permite con un sólo enrollado obtener cuatro valores distintos para la inductancia. Estos MOSFETs son activados por el PIC para conducir o no conducir y, en cada momento, únicamente debe de estar uno activo para evitar interferencias en los valores de la inductancia.

Con el fin de minimizar el ruido la señal de salida de nuestro oscilador ha sido colocada en el conector entre la alimentación y masa, para que así, si indujese algún ruido en las señales contiguas, con los condensadores de desacoplo se eliminaría.

4.1.4 Diseño PCBs

En este apartado incluimos tanto el diseño 2D como el diseño 3D de los esquemáticos mostrados en la sección anterior. Antes de nada emplazaremos en esta sección los diferentes requisitos impuestos por la manera de fabricar las distintas PCBs, es decir, impuestos por la máquina de control numérico ProtoMat *S*62:

- *Track width*: el ancho de la pista debe ser mayor o igual a 15 mils. El modo de operación del plotter nos obliga a fijar ese valor para finalmente tener un ancho de pista mínimo de 10 mils. Esto es así porque la máquina no tiene en cuenta el cobre desalojado por la fresadora al crear la pista, por lo que en el diseño tenemos que tenerlo en cuenta nosotros.

- *Clearance*: es la separación mínima entre pistas, pads y vías. La fijaremos en 10 mils para evitar que puedan aparecer circuitos con los desechos de la creación de la placa, es decir, el cobre removido por la fresadora que puede quedarse en los surcos de las vías y hacer un cortocircuito.

- *Hole size* El tamaño de los agujeros (*drills*) de la placa deberá escogerse teniendo en cuenta las brocas existentes en la máquina a la hora de fabricar. En nuestro caso los distintos tamaños disponibles son: 0.7, 0.8, 0.9, 1, 1.1, 1.3 y 1.5 mm.

Una vez abordados los requisitos mecánicos vamos a describir el modo de operación a la hora de diseñar cada una de las placas:

1. Revisión exhaustiva de los encapsulados de todos los componentes para elegir el adecuado, en todos los casos posibles el encapsulado SMD.

2. Creación del componente. Los componentes propios usados han tenido que ser creados de cero en la mayoría de los casos, para ello:

   (a) Creamos el símbolo del esquemático del componente.

(b) Creamos el *footprint* del componente ya sea tomando las medidas del *datasheet* o con un calibre del propio componente. Una vez creado comprobamos que es correcto imprimiéndolo en un papel y posando el componente encima.

(c) Incluir el modelo 3D del componente.

3. Desde el esquemático actualizamos el archivo de la PCB para incluir todos los componentes.

4. Creamos las diferentes reglas que nos son impuestas por la manera de fabricación.

5. Seleccionamos el tamaño de la placa y colocamos los componentes con el máximo espaciado posible para facilitar su soldadura.

6. Interconectamos los componentes mediante pistas, siempre lo más cortas posibles y lo más anchas para reducir al mínimo la inductancia y la resistencia parásitas, y usamos la opción *teardrop* para aumentar la conexión con los componentes y así mejorar la conductividad eléctrica.

7. Ejecutamos el comando *Design rule check* para comprobar que nuestro diseño es correcto, en caso contrario realizamos los cambios pertinentes y volvemos a ejecutar el comando.

Una vez explicado el modo de diseño y los requisitos impuestos vamos a describir las tres PCBs diseñadas.

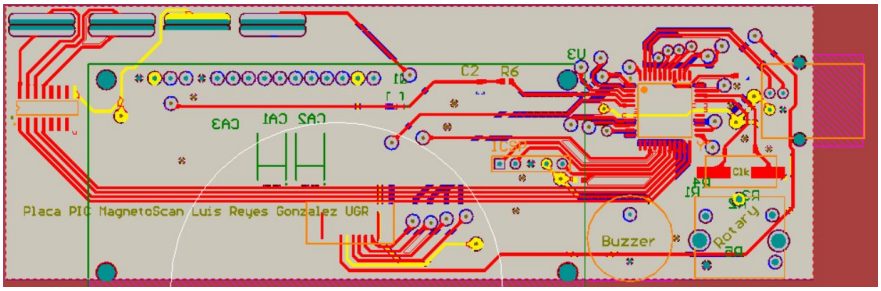
4.1.4.1 Placa base

El diseño de esta placa viene impuesto por el tamaño de la LCD, el emplazamiento del rotary, el del USB y de la apertura para la conexión con la placa motor.

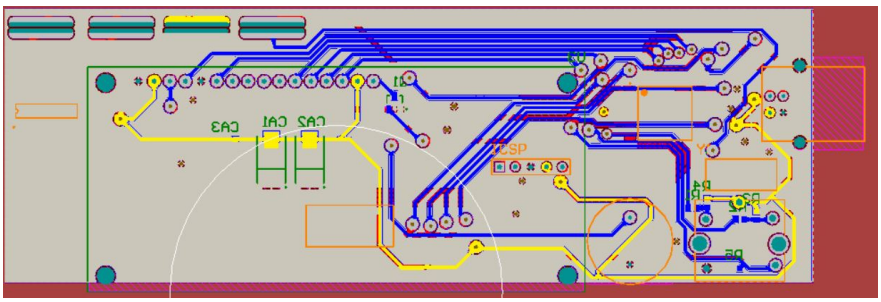
En el diseño de esta placa se han usado todos los los componentes SMD exceptuando el *rotary encoder*, el USB y el *buzzer* puesto que estos componentes no tenían la opción de usar ese encapsulado. Los conectores del ICSP y de la LCD son *true hole* también por necesidad impuesta por los componentes. En el caso de la conexion con la placa del oscilador usamos un conector FFC.

El ancho de las pistas se ha establecido en 20 mils. En los diseños hemos identificado la masa con el color gris y los 5 V con el color amarillo a fin de facilitar la comprensión visual de los diseños.

En la figura 4.11 podemos ver el diseño $2D$ de la placa:



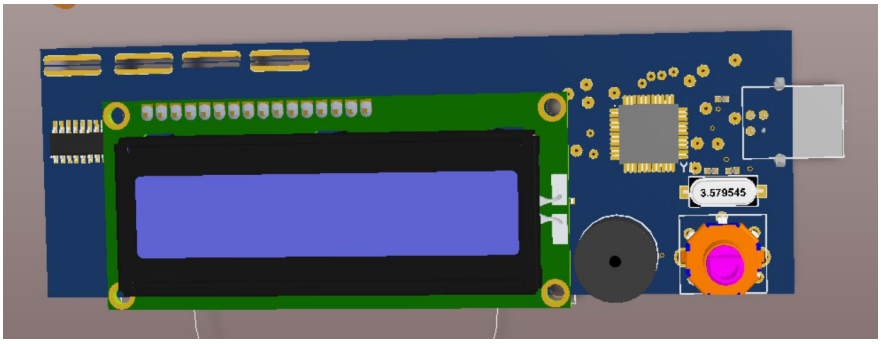
(a) *Vista top placa base*



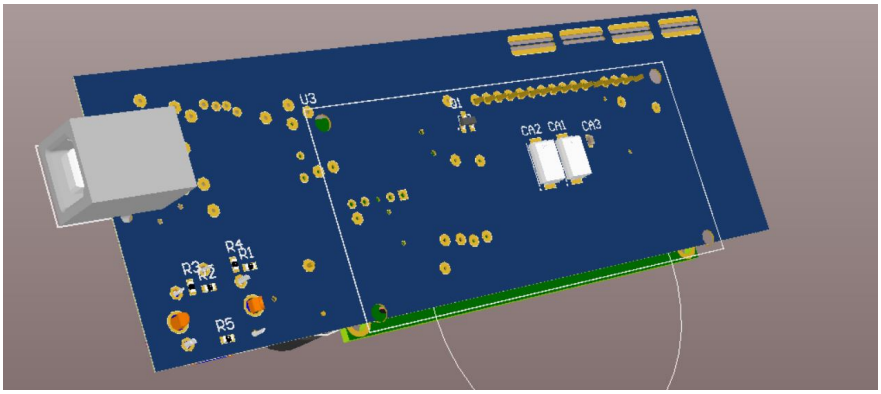
(b) *Vista bottom placa base*

Figura 4.11 – *Vistas diseño 2D placa base*

A continuación mostramos en la figura 4.12 el diseño 3*D* de la placa:



(a) *Vista top placa base*



(b) *Vista bottom placa base*

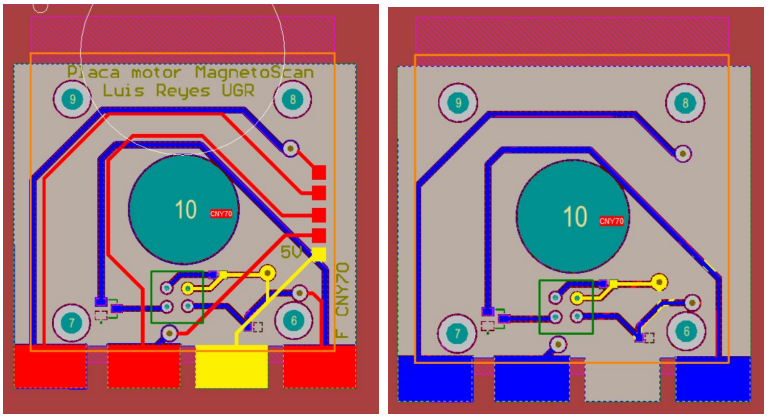
Figura 4.12 – *Vistas diseño 3D placa base*

4.1.4.2 Placa motor

El diseño de esta placa viene impuesto por el tamaño del motor, la abertura de la placa base en la que tiene que encajar los salientes de esta y la colocación del $CNY80$. Para darle rigidez al diseño esta placa se ha diseñado creado unos salientes que encajan perfectamente en el conector de la placa base, así la conexión se realiza soldando las dos placas a 90^O.

En el diseño se han usado resistencias SMD de tamaño 0605, el MOSFET es de encapsulado $SOT23$ y finalmente el $CNY70$ es *true hole* puesto que no existe encapsulado SMD para este componente.

A continuación mostramos en la figura 4.13 el diseño $2D$ de la placa del motor, como en el apartado anterior, el ancho de las pistas se ha establecido a 20 mils y el color amarillo identifica a la *net* 5 V y el gris a la de masa.



(a) *Vista top placa motor* **(b)** *Vista bottom placa motor*

Figura 4.13 – *Vistas diseño 2D placa motor*

En la figura 4.14 podemos ver el diseño $3D$ de la placa:

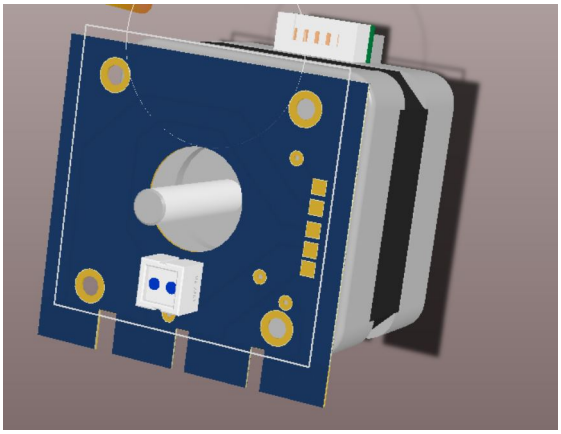


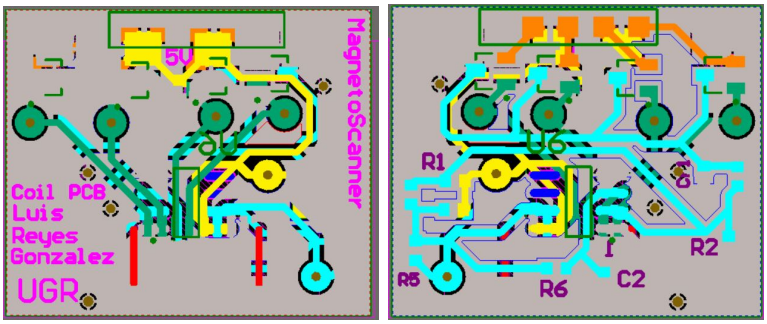
Figura 4.14 – *Visualización modelo 3D placa motor*

4.1.4.3 Placa oscilador

Finalmente diseñamos la placa en la que va nuestro oscilador. Puesto que esta PCB va incluida en una parte móvil deberá ser lo más pequeña posible. Debido a esta imposición todos los componentes usados en esta placa tienen encapsulado SMD. Se ha usado un conector de cable plano (FFC).

Una vez más la anchura de pistas es de 20 mils, y el gris corresponde al plano de masa mientras que el amarillo corresponde a la *net* 5 V. Además se le ha dado colores a las distintas *nets* para facilitar la lectura, el color naranja corresponde a la bobina *multi-tap*, al control de los MOSFETs le corresponde el verde oscuro y el color aguamarina a todos los componentes del oscilador .

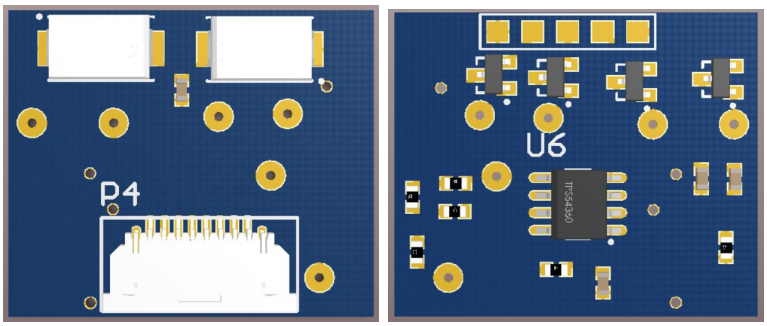
A continuación podemos ver la figura 4.15 donde se muestran las vistas *top* y *bottom* de la placa:



(a) *Vista top* **(b)** *Vista bottom*

Figura 4.15 – *Vistas diseño 2D placa oscilador*

Finalmente mostramos las vistas del diseño 3*D* (figura 4.16) de nuestra placa, para hacernos una idea de como quedará al montarla en el carro ya que esta placa no estará oculta a la vista.



(a) *Vista top* **(b)** *Vista bottom*

Figura 4.16 – *Vistas diseño 3D placa oscilador*

4.1.5 Diseño bobinas

Puesto que necesitamos unas bobinas muy especificas tendremos que fabricarlas a mano. Para ello enrollaremos una sola capa alrededor de un cilindro plástico, por lo que la inductancia responde a la ecuación 4.1.1 [10]:

$$L = \frac{0.0395a^2N^2}{l} \ \mu henrys \tag{4.1.1}$$

Donde a y l están dados en cm. Puesto que la longitud de nuestra bobina será de 8 mm y el radio de 9 mm, la relación entre ambos es de 1.125.Para bobinas cortas, aquellas cuya relación a/l es mayor que 1, es necesario aplicar el factor de Nagaoka [10](ecuación 4.1.2).

$$L = K\frac{0.0395a^2N^2}{l} \ \mu henrys \tag{4.1.2}$$

El valor que toma la constante de Nagaoka lo obtenemos según el diagrama de la figura 4.17.

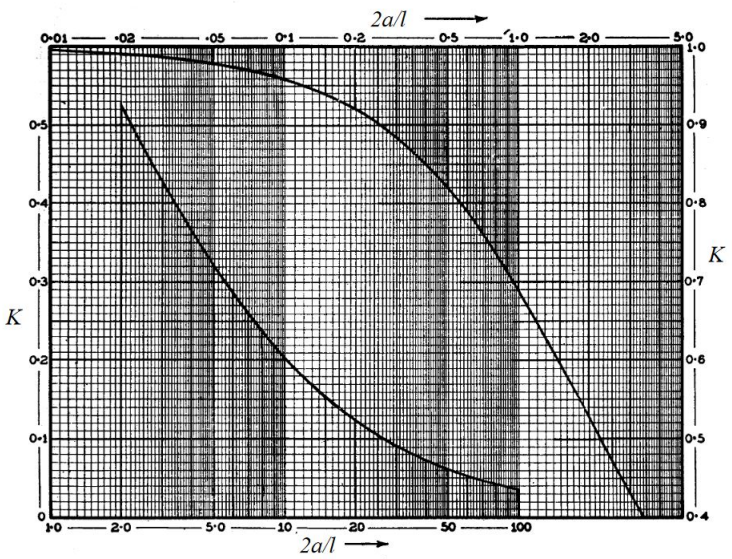


Figura 4.17 – *Valor constante Nagaoka*

En nuestro caso el valor sería de 0.51, por tanto, sustituyendo valores, la inductancia dependerá del número de vueltas según la ecuación 4.1.3:

$$L = 0.02039N^2 \ \mu henrys \tag{4.1.3}$$

4.2 Diseño *Software*

En este punto debemos hacer una distinción entre las tres partes que engloba nuestro *Software* (figura (4.18):

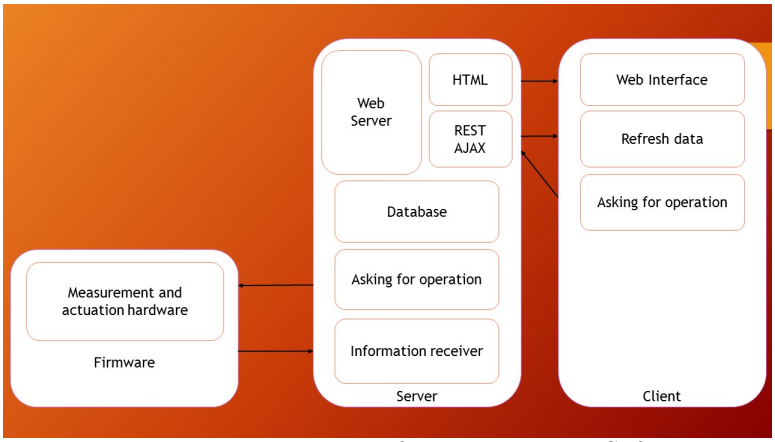


Figura 4.18 – *Esquema funcionamiento Software*

4

- *Firmware*: engloba toda la programación relativa al microcontrolador.

- Servidor: conexión del microcontrolador con el PC, incluye todo el *Software* necesario para que el ordenador se comunique con el microcontrolador y a su vez con la GUI.

- Cliente: permite la visualización de datos y el control del equipo a través de una GUI.

4.2.1 *Firmware*

Aquí explicaremos el funcionamiento y la programación del PIC. El primer paso para trabajar es configurar el oscilador así que una vez más recurrimos al *datasheet* [8] del componente para elegir la configuración adecuada (figura 4.19). Necesitamos una frecuencia de 4 Mhz para obtener los 96 Mhz a partir de los cuales se obtienen los 48 con los que funciona el USB, por tanto dividimos nuestra frecuencia incial entre 4 mediante la opción *PLL*4. Una vez tenemos esa frecuencia tenemos un *posescaler* que es capaz de dividir la frecuencia si queremos trabajar con los periféricos a menor frecuencia, en nuestro caso lo hemos configurado entre 4 para volver a tener los 16 Mhz que teníamos a la entrada.

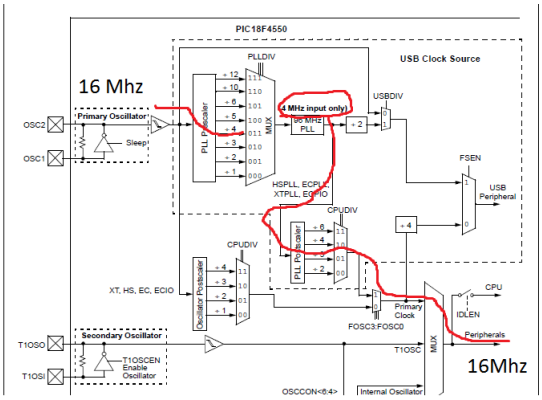


Figura 4.19 – *Esquema del datasheet para la configuración de la frecuencia.*

Una vez hecha la configuración inicial del reloj pasamos a determinar las distintas funciones que tiene que desempeñar el PIC:

- Comunicación USB.
- Lectura de frecuencia del oscilador.
- Lectura frecuencia $CNY70$.
- Señal para control $CNY70$.
- Señal para control del *buzzer*.
- Señal PWM para control contraste.
- Control motor.
- Escritura LCD.
- Rutina *rotary* y pulsador.

4.2.1.1 Comunicación USB.

Como ya se explicó en el capítulo 3 vamos a trabajar con un puerto serie virtual creado en el mismo USB, para la realización del cual nos apoyamos la librería prediseñada de PIC C compiler usb_cdc. Las funciones necesarias para trabajar son las siguientes:

- $Usb\_cdc\_init()$ Inicia el servicio de comunicación CDC (*Communication Device Class*).
- $usb\_init()$ Inicia la comunicación USB.
- $usb\_task()$ Se utilizad para establecer la comunicación.

- *usb_enumerated*() Devuelve *true* o *false* dependiendo de si esta correctamente conectado o no. Hay que incluirlo en un bucle infinito que compruebe cada vez que se accede a él que seguimos conectados.

- *usb_cdc_getc*() Lee un carácter del *buffer*.Existen otros métodos, basados en este, proporcionados también por microchip para leer datos según el tipo, en concreto usamos $get\_string\_usb(char * s, intmax)$

- *usb_cdc_putc*() Envía un carácter. .

Por tanto la comunicación es necesaria establecerla en el método *main* llamando a los métodos $Usb\_cdc\_init()$ y $usb\_init()$. Una vez inicializada llamamos al método $usb\_task()$ dentro de un bucle infinito, acto seguido se comprueba si está correctamente conectado con *usb_enumerated*(). Finalmente si existe conexión decidimos si queremos escribir en el *buffer* USB mediante *usb_cdc_putc*() o si, lo que queremos es leer decidimos entre un carácter o un string completo y usamos el método *get* acorde.

4

4.2.1.2 Lectura de frecuencia.

Tanto la lectura de frecuencia del oscilador como la del $CNY70$ se harán con los módulos CCP (*Capture Compare PWM*). Estos módulos se configuraran ambos con el *timer1*, podemos hacerlo así puesto que las interrupciones no estarán activas a la vez, ya que al mover el motor no leemos frecuencia proveniente del oscilador y viceversa. Sin embargo, ambas configuraciones serán distintas, ya que mientras que la frecuencia del $CNY70$ tiene un valor bajo, la frecuencia leída del oscilador serán valores altos, por lo que el módulo $CCP1$ correspondiente a la lectura del oscilador se configurará para que salte la interrucpión cada 16 pulsos y el $CCP2$ en cada flanco de subida. Esto se hace así para evitar que el PIC esté continuamente dentro de esta interrupción. La manera de calcular la frecuencia leída será la siguiente (figura 4.20):

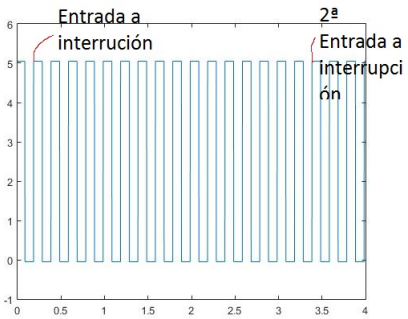


Figura 4.20 – *Esquema funcionamiento inturrupción CCP*

1. Después de 16 pulsos el PIC entra en la interrupción $CCP1$.

2. Eváluamos el *flag Startpulse* para determinar si es el inicio del pulso o es el final.

3. Si es el inicio ponemos a 0 el *timer*1 y cambiamos el *flag* de inicio de pulso.

4. En caso de que no sea inicio de pulso leemos el valor del registro $CCP1$ y cambiamos los valores de los *flags* de inicio de pulso y de pulso leído. Desactivamos la interrupción hasta que el programa requiera una nueva lectura de frecuencia.

5. Llamamos al método *readFrequency* (figura 4.21) en el que calculamos la frecuencia, para ello previamente hemos calculado los microsegundos que emplea el microcontrolador por *tick*, emplea 4 ciclos por acción por lo que dividimos $4/f_{PIC}$, y los multiplicamos por el número de pulsos, es decir, el valor del registro $CCP1$ al momento del fin del pulso. Puesto que la interrupción entra cada 16 pulsos dividimos el valor obtenido entre 16 para obtener la frecuencia de la señal. Puesto que al entrar en la interrupción tenemos que hacer evaluaciones de variables antes de leer o escribir los registros que nos interesan perdemos ciertos ciclos del PIC, lo que nos da un valor erróneo de la lectura de ciclos, por tanto corregimos ese valor manualmente en este método.

```
#int_CCP1
void rutina_CCP1(){
if(Startpulse==yes)
   {

   set_timer1(0);//If it's pulse start we set timer1 0 and change flag startpulse
   startpulse=No;

   }
else
   {
   endCCPR=CCP_1; //If it's end pulse we read the CCP_1 value and change readpulse and startpulse flags.
   Readpulse=yes;
   startpulse=yes;
   disable_interrupts(int_ccp1);
   }
}
```

(a) *Rutina interrupción CCP*

```
readFrequency(){

   if(readpulse==Yes)
   {
   pulses=endCCPR+50;//Pic losses 44 cycles on the interruption
   st=(uSxtick*pulses)/16;//Calculate period in uS
   freq=1000/st;//Calculate frequency in Khz
   printf(lcd_putc,"\f Periodo = %6.1f us \n Frequ = %g Khz \n Pulsos %Lu ", st,freq, pulses );//print at lcd
   printf(usb_cdc_putc,"%6.1f , %g , %Lu\n, %u", st,freq, pulses, cuenta );//print at usb period freq and pulses

   readpulse=No;//Change flag readpulse.


   }
}
```

(b) *Método para obtener la frecuencia*

Figura 4.21 – *Métodos necesarios para la lectura de frecuencia*

4.2.1.3 PWM contraste

La señal PWM que se encarga del control del contraste en la LCD está controlada por el *timer*0, este *timer* tiene un contador de tamaño seleccionable por *software* que puede tomar los valores de 8 o de 16 bits, sus registros son leibles y escribibles, tiene un *preescaler* dedicado

de 8 bits, permite seleccionar la fuente de reloj(interna o externa), y permite la interrupción por desbordamiento. Esta última opción es la que nosotros usaremos, para configurar el tiempo de desbordamiento deseado recurriremos al *datasheet* [8] (figura 4.22):

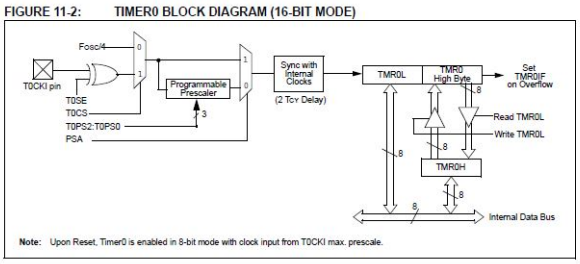


Figura 4.22 – *Diagrama de bloques del timer*0

A la vista del *datasheet* determinamos que el *timer0* se desborda siguiendo la ecuación 4.2.1, por lo que despejamos el valor del *timer0* para obtener una interrupción en el tiempo deseado.

$$T = \frac{4}{f_{osc}} * (2^{16} - ValueTimer0) * Prescaler$$
$$ValueTimer0 = 2^{16} - \frac{Tf_{osc}}{4 * Prescaler} \quad (4.2.1)$$

Si queremos desbordar el timer cada 125 microsegundos debemos de seleccionar el reloj interno y al prescaler asignarle un valor 1, es decir, no hacer prescaler. Si en este punto aplicamos la ecuación 4.2.1 obtenemos un valor del *timer0* de 65411, es decir, $FF83$. Una vez tenemos la interrupción creada pasamos a determinar el tiempo en alta de la señal, esto se hace mediante un contador auxiliar, temporizador, el cual se va incrementando en cada interrupción, cuando este valor es igual al valor que queremos que tenga el contraste la señal pasa a baja hasta que temporizador se reinicia, lo que nos permite variar el tiempo en alta de la señal (figura 4.23)

```
#int_TIMER0
void rutina_timer0(){
set_timer0(0xFF83);
   if(temporizador==0){
   output_high(PIN_D6);
   }
   if(temporizador==contraste)
   {
   output_low(PIN_D6);
   }
temporizador++;
   if(temporizador==15)
   {
   temporizador=0;
   }
}
```

Figura 4.23 – *Código interrupción timer*0

4.2.1.4 Señal de control $CNY70$

Configuramos el *timer*2 para el control de esta señal. El *timer*2 es un registro de 8 bits, por lo que sus valores van desde 0 hasta 255, consta de un *prescaler* y de un *postscaler*. Como siempre recurrimos al *datasheet* [8] para configurar el PIC (figura 4.24) :

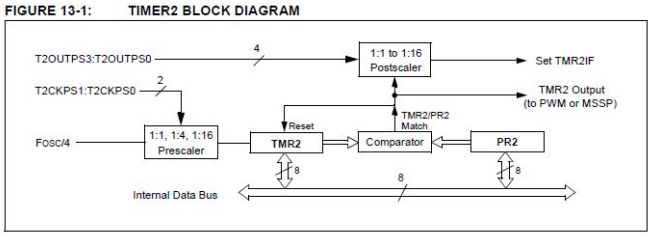


Figura 4.24 – *Diagrama de bloques del timer*2

A diferencia de los otros temporizadores, el temporizador Timer 2 no incrementa su cuenta hasta llegar a 0xFF y después al desborde sino que incrementa su cuenta desde 0x00 con cada ciclo de instrucción hasta que el valor del registro TMR2 coincide con el del registro PR2 y después, en el siguiente ciclo reinicia la cuenta desde 0x00.El *prescaler* tiene la misma función que en los otros dos timers y sirve como divisor de frecuencia antes de cada incremento. El *postscaler* funciona como un divisor de frecuencia después de cada coincidencia entre los registros TMR2 y PR2. El tiempo de desbordamiento viene dado por la ecuación 4.2.2:

$$\begin{aligned} T &= [Preescaler * (PR2 + 1) * Postscaler] * \frac{4}{f_{osc}} \\ PR2 &= \frac{\frac{T}{4/f_{osc}}}{Preescaler * Postscaler} - 1 \end{aligned} \tag{4.2.2}$$

4.2.1.5 Control motor

Un *stepper* motor consiste en un eje giratorio magnético, llamado rotor, y electroimanes en la parte estacionaria que rodea al motor. En la figura 4.25 podemos ver la rotación completa de un motor:

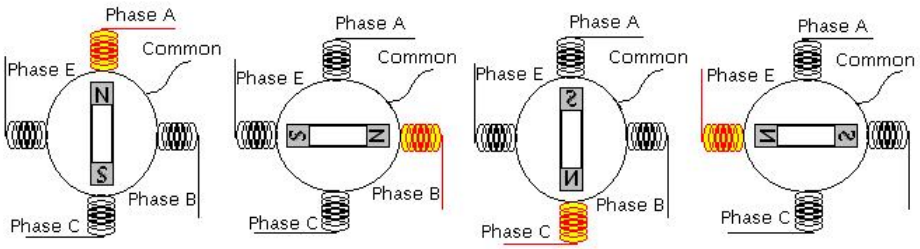


Figura 4.25 – *Movmiento stepper motor.*

Según el electroimán por el que circula corriente se alinea el rotor, por tanto el movimiento

del motor consiste en ir activando los electroimanes que rodean al motor por orden.Debido a esto existen tres tipos de movimiento según los electroimanes activados:

1. Medio paso: el rotor da medio paso en cada cambio de activación de electroimanes, esto se consigue activando el electroimán actual y el siguiente antes de apagar el anterior.

2. Paso simple: los electroimanes se van activando uno a uno por orden por lo que el motor da un paso completo a cada cambio.

3. Paso doble: los electroimanes se van activando dos a dos de manera sucesiva. Este modo será el que usemos por ser el más rápido a la hora de obtener revoluciones a cambio de sacrificar un poco de precisión debido a dar los pasos dobles.

El código usado para mover el motor lo podemos ver en la figura 4.26:

```
//Paso Doble
byte const bajar[4]={0b1001,
                     0b0011,
                     0b0110,
                     0b1100};

void goDown(){

      for(i=0;i<50;i++){
      if(finCarrera==No)
      {
            for(j=0;j<4;j++){
               output_a(bajar[j]);
               delay_ms(2);

            }
          }
        }
}
```

Figura 4.26 – *Código para mover el motor*

4.2.1.6 Escritura LCD

Para la escritura de la LCD hemos aprovechado los *drivers* facilitados por *PIC C Compiler* en *FlexLCD* donde definimos las conexiones de nuestra LCD con el PIC. Como pasaba con la conexión USB, estos *drivers* nos proporcionan dos métodos para el control de la LCD:

- *lcd_init*(): método que inicializa la LCD. Hay que llamarlo al iniciar el programa.
- *lcd_putc*(): método para escribir en la LCD.

4.2.1.7 Rutinas rotary encoder con pulsador

Aunque nos refiramos al mismo dispositivo son dos rutinas distintas, una para el pulsador y otra para la lectura del *rotary*. Tal y como explicamos en el capítulo 3 vamos a usar un *rotary* diferencial, es decir, aquel que determina el cambio de ángulo. Tal y como mostraba

la figura 3.11, necesitamos saber el puslo que se adelanta, para lo cual necesitamos conocer el estado del pulso en el instante anterior y en el actual. Esto se hace mediante una llamada a la función que se encarga de leer el *rotary* en la interrupción del *timer0*, al poner a 0 la variable temporizador usada para el control de contraste, lo que nos da una llamada cada 2 ms. Esta llamada únicamente se realiza cuando el PIC está dentro del menú, durante el análisis está deshabilitada. Cada vez que entra al método lee el estado de los pines conectados al *rotary*, los comprueba con los valores leídos anteriormente y modifica el valor de la variable cuenta, usada para el control de los distintos menús. Una vez hecha la comparación almacena el valor leído como el valor de estado anterior (figura 4.27).

```
void read_Rotary(){
Right= input(PIN_B1);
Left= input(PIN_B2);
if(!Right && RightBefore)
   {
   rotary=Yes;
   if(Left)
      {
      cuenta++;
      }
      else
      {
       cuenta--;
      }
     }
  RightBefore=Right;
  LeftBefore=Left;
  if(cuenta==5)
  {
  cuenta=0;
  }
 }
```

Figura 4.27 – *Código para leer el rotary*

En el caso del pulsador nos encontramos ante una interrupción externa por flanco de subida, ya que, según la configuración montada y explicada en el anterior apartado, al pulsar el botón se pone la patilla conectada al PIC en alta. Dentro de esta se realizan las acciones pertinentes según el estado actual del PIC, pasar del control por *software* al control manual, apagar el equipo o movernos por los menús.

4.2.2 *Server*

Como vimos en la figura 4.18 el *server* es el que se encarga de conectar la GUI con el aparato. En nuestro caso lo hemos configurado usando el lenguaje Python. Las funciones que implementa son las siguientes:

- Conexión USB.
- Conexión con cliente.
- Leer datos del PIC y guardarlos en una base de datos.
- Escribir PIC.
- Guardar datos en CSV.

4.2.2.1 Conexión USB

Aunque lo denominamos conexión USB recordemos que estamos simulando un puerto serie virtual, por lo que en lo que a python respecta estamos trabajando con un puerto serie, lo que nos permite trabajar con la librería Pyserial [4]. Por lo tanto para conectarnos al puerto llamamos al método *OpenPort()*, el cual lista todos los puertos COM del ordenador, detecta aquél en el que esta conectado el MagnetoScan y crea el objeto serial, facilitado por la librería, que crea la conexión con el PIC (figura 4.28). En caso de no poder realizar al conexión salta una excepción y nos devuelve error.

```
@app.route("/OpenPort")
def OpenPort():

        #------------------------------Connect Port-------------------
    ports = list(serial.tools.list_ports.comports())

    for p in ports:

        if "MagnetoScan" in p[1]:
            Port, Name, VID = p
            print(Port)
    global Conectado;
    Conectado = 0
    try:
        global PicData;
        PicData = serial.Serial(Port,9600)  # Creating our serial object named PicSerial
        Conectado = 1  # Tenemos comunicacion serie
        print("Conecta")
        return Response("ok")


    except SerialException:
        raise
        print('Imposible acceder al puerto')
        Conectado = 0
        return Response("fail")
```

Figura 4.28 – *Código para crear la conexión serie*

4.2.2.2 Conexión con cliente

La conexión con el cliente se realiza mediante *flask*[3]. *Flask* es un framework minimalista que permite crear aplicaciones web rapidamente y con un mínimo de línas de código. En el desarrollo siguiente veremos que todos los métodos usados van precedidos por una línea que indicala ruta de la aplicación (figura 4.29), lo que permite que con una llamada de *jQuery* el servidor ejecute el método que le pide el cliente. En el método main es necesario poner la dirección y el puerto en el que queremos ejecutar nuestro cliente (figura 4.29).

```
@app.route("/OpenPort")
```

(a) *Indicación ruta del método*

```
if __name__ == "__main__":
    # Run the server for any client IP and in the port 8000
    app.run(host='0.0.0.0',port=8000,debug=True, use_reloader=False)
    while True:
        pass
```

(b) *Método para lanzar el cliente en un puerto determinado*

Figura 4.29 – *Métodos necesarios para usar flask*

4.2.2.3 Leer datos del PIC y guardarlos en una base de datos

Para ello utilizamos el método *ReadSerie()* (figura 4.30). Los pasos a seguir son los siguietnes:

1. Esperamos mientras no existan datos en el *buffer* de entrada.
2. Una vez existe un dato leemos toda la línea que hay en el *buffer*.
3. Dividimos la línea en un array ya que sabemos como están delimitados los valores.
4. Insertamos cada valor en la tabla.

```
#---------------------------------Read data from serial port
def ReadSerie():

    while (PicData.inWaiting() == 0):
        pass
    PicCadena = PicData.readline()  # Read the line of text from the serial port
    dataArray = PicCadena.decode("utf-8").split(',')  # Split it into an array called dataArray
    Siz = len(dataArray)
    if Siz == 3:
        a = float(dataArray[0])  # Convert first element to floating number and save it into Concentration
        b = float(dataArray[1])  # Convert second element to floating number and save it into Frequency
        c = int(dataArray[2])  # Conver third element to integer number and save it into pulses

    # Get the date and time (with 3 decimal milliseconds)
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d,%H:%M:%S.%f")[:-3]

    # Return an dictionary with the table database schema
    yield str(a) + "," + str(b) + "," + str(c) + "," + str(timestamp)
    #yield {'Concentration':a,'Frequency':b,'Pulses':c,'timestamp':timestamp}
    #table.insert(a,b,c,timestamp)
    table.insert({'Concentration':a,'Frequency':b,'Position':c,'timestamp':timestamp})
```

Figura 4.30 – *Código para leer datos del puerto serie*

4.2.2.4 Escribir PIC

Para cualquier petición que se le hace al microcontrolador es necesario escribir en él, esto se hace mediante la función de la librería *Pyserial write*, esta función escribe en el *buffer*

USB los argumentos que se le dan. Para llamarla hay que hacerlo con un objeto de la clase *serial* creado previamente (figura 4.31).

```
#--------------------Request data-------------------------------
def requestOperation(requeriment):
    PicData.write(requeriment.encode())
```

Figura 4.31 – *Código para escribir en el microcontrolador.*

4.2.2.5 Guardar en formato CSV

Para guardar los datos en formato CSV es necesario acceder a la base de datos y escribir todos los datos en un *string* separados por comas (figura 4.32).

```
# Create a link to get all data in CSV format
@app.route("/csvData")
def csvData():
    # Init the data vectors
    timestamp = ['timestamp']
    Concentration = ['Concentration']
    Frequency = ['Frequency']
    Position = ['Position']

    csvString = "timestamp,Concentration,Frequency,Position\n"

    # Query the database with a SQL sentence: get all entries from data table
    # and order them by the timestap
    result = db.query('''SELECT * FROM data ORDER BY strftime('%Y-%m-%d %H%M:%f',timestamp) DESC ;''')
    for row in result:
        # Add the info to the csvString
        csvString += str(row['timestamp'][11:]) + "," + str(row['Concentration']) + "," + str(row['Frequency']) + "," + str(row

    # Return the result with CSV format
    return Response(csvString,mimetype='text/csv')
```

Figura 4.32 – *Código para obtener todos los datos en formato CSV.*

4.2.3 Cliente

El cliente consistirá en una página web HTML con las siguientes funcionalidades:

- Comunicación con servidor.
- Visualización de datos de manera gráfica.
- Visualización de datos de manera numérica.

4.2.3.1 Comunicación con el servidor

Como anticipamos en la sección anterior la comunicación se realiza mediante *jQuery*, que es una biblioteca multiplataforma de *JavaScript* para simplificar la manera de actuar con HTML. Pongamos el ejemplo de presionar el botón para activar la bobina 4 (figura 4.33).

```
$("#coil4").on("click",function (){

     if(analize==false){
    $.get("Coil4", function(data)
    {

          })}
    else
        {alert ('Please stop analisis before change coil')}
})
```

Figura 4.33 – *Ejemplo código jQuery.*

La forma de operar será la siguiente en todas las llamadas:

1. Asignamos la función al evento *onclick* del botón asociado a la bobina 4.

2. Realizamos las tareas pertinentes, en este caso comprobar si el análisis ya esta iniciado o no.

3. Nos comunicamos con el servidor mediante un *get*, en este caso le pedimos la bobina 4.

4. En caso de que debieramos hacer algo con la respuesta del servidor lo haríamos aquí, en este caso como únicamente nos responde una vez que esta recibido no es necesario tratar la respuesta.

4.2.3.2 Visualización de datos de manera gráfica.

La visualización de los datos de manera gráfica se hace usando $C3$ [2] una librería de gráficas reusable basada en $D3$. La elección de esta librería se ha debido a que la estética es sobria, que es lo que al final se busca en la representación gráfica de un aparato científico y muy personalizable, lo que nos permite mover los ejes, añadir trazas, borrar trazas..todo esto de manera *online*, sin tener que reiniciar el cliente.

La forma de generar el gráfico será (figura 4.34):

1. Crear una nueva variable $C3$.

2. La unimos a un apartado de la hoja de estilos

3. Generamos los ejes con el formato que deseamos y añadimos las columnas que va a tener nuestro gráfico.

4. Mostramos el gráfico.

```
//Generating a new chart
var chart = c3.generate({
    //Putting it in our div
    bindto: '#dataChart',
    transition: {
        duration: 0
    },
    //Axis, data and formats
    data: {
        x  : 'timestamp',
        type: 'spline',
        xFormat: '%H:%M:%S.%L', // 'xFormat' can be used as custom format of 'x'
        columns: [
            ['timestamp'],
            ['Position'],
            ['Concentration'],
            ['Frequency']

        ]
    },
    axis: {
        x: {
            type: 'timeseries',
            tick: {
                format: '%H:%M:%S.%L'
            }
        }
    },
  subchart: {
        show: true
}

});
```

Figura 4.34 – *Creación gráfico C3*

Ya tenemos el gráfico definido, es el momento de añadir los datos (figura 4.35), para lo cual pedimos los datos al servidor, dividimos el*string* obtenido en los cuatro datos que necesitamos y lo añadimos al gráfico con el comando *flow*.

```
$.get("onedata", function(data){


}).done(function(data){
        var A = data.split(",")
        var B = [
        [
          "timestamp",
          A[4]
        ],
        [
          "Concentration",
          parseFloat(A[0])
        ],
        [
          "Frequency",
          parseFloat(A[1])
        ],
        [
          "Position",
          parseFloat(A[2])
        ]
      ]
    //console.log(x);
    //Seleccionamos el gráfico que queremos

    chart.flow( {columns:B , length: 0});

})
```

Figura 4.35 – *Código para añadir datos al gráfico.*

4.2.3.3 Visualización de datos de manera numérica.

Puesto que ya teníamos creado el método que devolvía todos los datos en CSV le pasamos ese método a una zona de la hoja de estilo asociada y aparecen todos los datos leídos hasta el momento de forma numérica. Incluimos también un botón que permite descargar el archivo

para facilitar su tratamiento.

4.3 Diseño mecánico

Para satisfacer los requerimientos del diseño mecánico con el mínimo coste posible nos hemos decantado por el diseño para impresión $3D$. Esto conlleva una serie de problemas:

- Las tolerancias dependen de la boquilla de salida del plástico o extrusor, por lo que necesitaremos retocar los diseños para tener esto en cuenta.
- No se pueden crear partes voladas a 90^o.
- Al ser plástico extruido las piezas tienen que tener consistencia para que no se partan al mínimo movimiento.

Sin embargo el gran ahorro en costes y la posbilidad de disponer de una impresora $3D$ en el departamento de electrónica hicieron que nos decantásemos por esta opción.

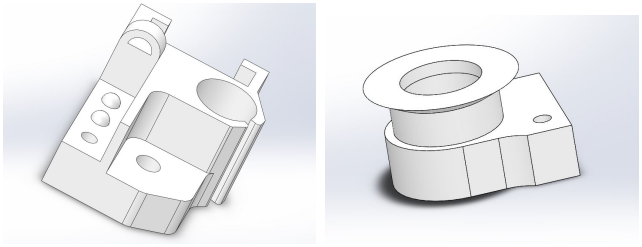
El diseño lo dividimos en varias partes que luego uniremos mediante tornillos:

- Creación de un soporte para la bobina y la placa.
- Movimiento del soporte de la bobina.
- Soporte tubo.
- Caja para la placa base.
- Solidez y estabilidad del producto final.

Veamos parte por parte el diseño mecánico.

4.3.1 Creación de un soporte para la bobina y la placa.

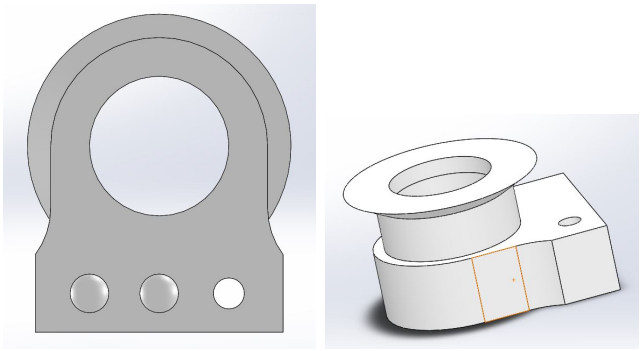
Necesitábamos un soporte móvil capaz de recorrer un tubo de muestra y albergar una bobina y una pequeña PCB. Inicialmente se intentó un diseño en una única pieza pero, puesto que esta pieza no era imprimible mediante el sistema de extrusión, se pasó a un diseño en dos piezas (figura 4.36):



(a) *Pieza inferior* **(b)** *Pieza superior*

Figura 4.36 – *Piezas del carro de la bobina*

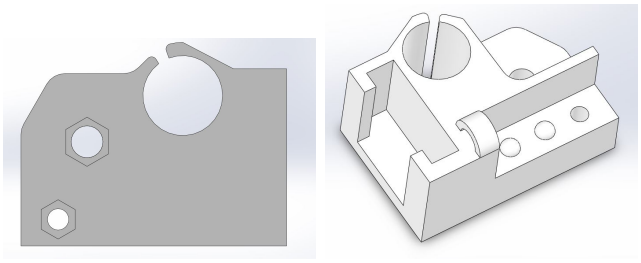
Analicemos el diseño del soporte de la bobina (figura 4.37). Tiene un espacio dedicado a bobinar con un tope por abajo que es el propio enganche con la otra pieza y en la parte superior un saliente en ángulo de 30^O para que sea imprimible. En la parte inferior tiene 3 aberturas, una de ellas pasante para la fijación mediante tornillo a la otra pieza. Las otras dos aberturas son cónicas para ayudar al ensamblamiento.



(a) *Vista inferior* **(b)** *Vista oblicúa*

Figura 4.37 – *Vistas soporte bobina.*

En cuanto al carro (figura 4.38) dispone de una abertura para el enganche con el soporte de la bobina, en el que existen unas aberturas con forma de cono y un agujero pasante coincidentes con los de la otra pieza para su perfecto ensamblaje. En la parte inmediatamente superior a esta abertura colocada entre este soporte y el de la placa, tiene un arco para recoger los hilos de cobre de la bobina. En el lateral está el sporte de la PCB. En paralelo a la conexión con el soporte de la bobina dispone de dos aberturas, la más pequeña para introducir una varilla roscada y la más grande para introducir un rodamiento. En la parte inferior dispone de dos huecos para encajar dos tuercas en la abertura para el tornillo y para la varilla roscada.

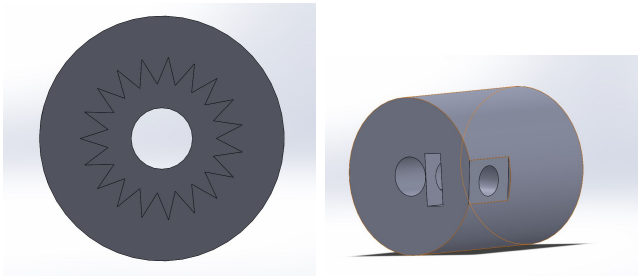


(a) *Vista inferior* **(b)** *Vista oblicua*

Figura 4.38 – *Vistas carro bobina.*

4.3.2 Movimiento del soporte de la bobina.

El movimiento de la bobina se ha ideado inspirándonos en el diseño de una impresora $3D$. Estas usan una varilla roscada con una tuerca y una varilla lisa con un rodamiento, el mecanismo se basa en unir la varilla roscada a un motor, al girar hacia un lado la tuerca se aprieta, por tanto avanza en la varilla roscada, lo que sube el carro. Al moverse el motor en sentido contrario la tuerca se afloja, por lo tanto avanza en el sentido contrario por lo que baja el soporte. Para este sistema de movimiento hemos usado una varilla roscada del 5 con la tuerca correspondiente, una varilla lisa de 8 mm con un rodamiento $LM8UU$. También necesitamos un tope en la parte superior para sostener las varillas y un acoplador(figura 4.39) para el motor y la varilla roscada.



(a) *Vista inferior* **(b)** *Vista oblicua*

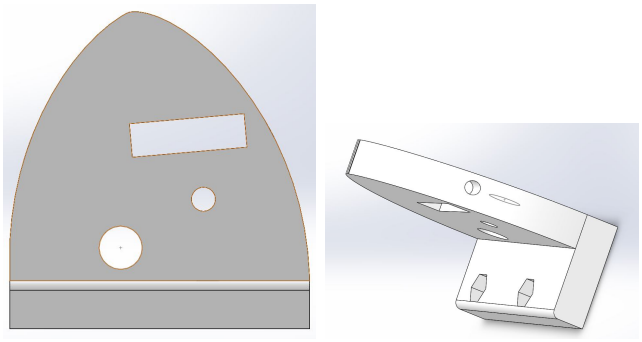
Figura 4.39 – *Vistas acoplador motor*

El acoplador consiste en un enganche por presión al engranaje del motor y una abertura en la parte superior, con un hueco en el que encaja una tuerca y un agujero pasante hasta la varilla. Para encajar la varilla apretamos el tornillo a la tuerca hasta que la varilla este fija al acoplador.

4.3.3 Soporte tubo.

Necesitamos un soporte para el tubo de ensayo que preferiblemente se adapte a distintos diámetros de tubo, ya que existen al menos dos con los que sabemos que se trabaja. Para solucionar este problema elegimos un soporte en dos piezas anclado al soporte superior de

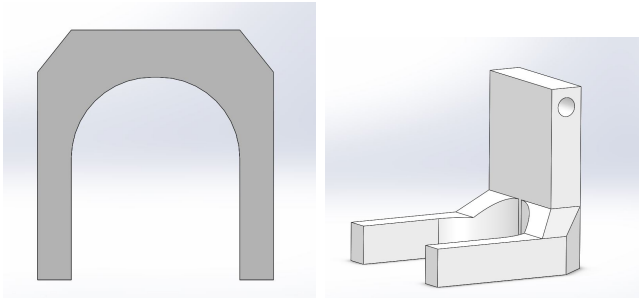
las varillas. El soporte superior (figura 4.40) incluye un anclaje al esqueleto del aparato mediante tornillos y unas aberturas para la varilla roscada, la varilla lisa y el soporte del tubo formando 90^o con los anclajes anteriores. El hueco dedicado al soporte del tubo está cruzado por un agujero pasante para sujetar el soporte con un pasador.



(a) *Vista inferior* **(b)** *Vista oblicua*

Figura 4.40 – *Vistas soporte superior*

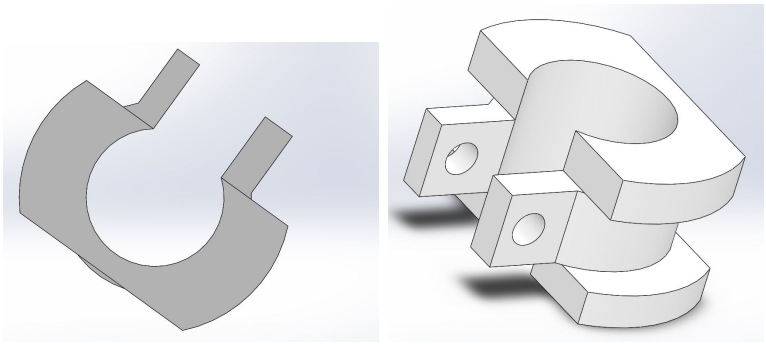
El soporte del tubo de ensayo se divide en dos piezas. La primera de ellas (figura 4.41) sirve para encajar la otra con el soporte superior encajando en la abertura antes mencionada con un agujero coincidente para anclarla con un pasador. Podemos observar como se han reforzado las partes más frágiles para evitar su ruptura.



(a) *Vista inferior* **(b)** *Vista oblicua*

Figura 4.41 – *Vistas soporte tubo parte de anclaje al sistema*

La otra parte del soporte sirve para agarrar en tubo de ensayo y encajarlo (figura 4.42).

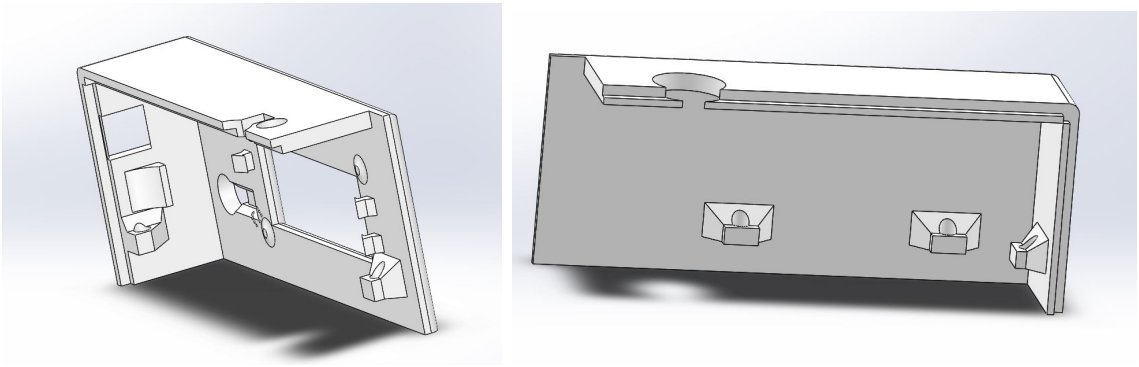


(a) *Vista inferior* **(b)** *Vista oblicua*

Figura 4.42 – *Vistas soporte tubo parte de sujección al tubo*

4.3.4 Caja para la placa base.

En este punto se diseño una caja para la placa base y el motor de manera que la placa base quedase perfectamente fijada, evitando cualquier movimiento durante el uso del aparato. Se tuvieron en cuenta para el diseño las aberturas necesarias para la conexión USB, el *rotary*, la LCD y el motor. Una vez más fue imposible hacer el diseño de una pieza por temas de impresión, por lo que dividimos el diseño en una parte frontal y una trasera. La frontal (figura 4.43) es donde van todas las conexiones y la mayor parte de la placa, mientras que la trasera (figura 4.43) sujeta la placa del motor. Podemos ver como se han colocado agarres para la LCD, un tope para la placa en la zona del USB y como los salientes para los tornillos están reforzados para evitar su ruptura.



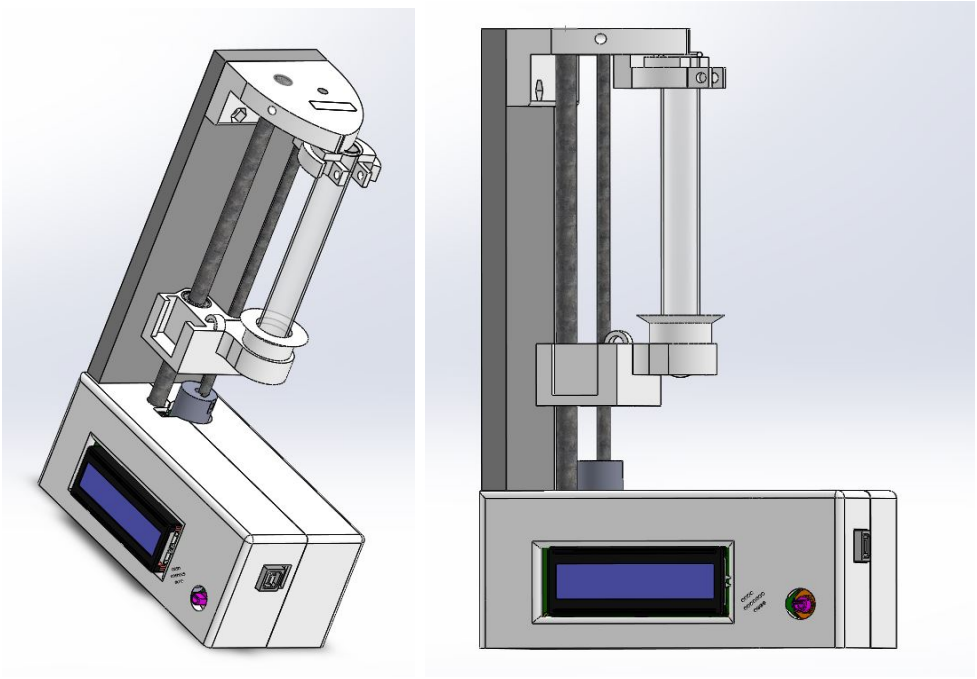
(a) *Vista inferior* **(b)** *Vista oblicua*

Figura 4.43 – *Vistas caja para la placa base*

4.3.5 Solidez y estabilidad del producto final.

Para terminar con el diseño $3D$ necesitamos un esqueleto en el que se apoyen todas las piezas que de consistencia al montaje. En nuestro caso hemos elegido metatrilato por es un material de facil acceso y tratamiento, además de visualmente atractivo. El resultado final

del diseño $3D$ lo podemos ver en la figura 4.44:



(a) *Vista oblicua* **(b)** *Vista frontal*

Figura 4.44 – *Resultado final diseño 3D*

CAPÍTULO

5

FABRICACIÓN

Una vez diseñado el sistema a sus tres niveles, hardware, software y mecánico, procedemos a explicar como se ha llevado a cabo el proceso de fabricación y montaje del mismo.

El desarrolo de la fabricación lo abordaremos tal y como hicimos con el diseño, es decir, diferenciando entre *hardware* y mecánico.

5.1 *Hardware*

El método de fabricación fue mediante una máquina de control numérico propiedad del departamento de electrónica, la PROTOMAT $S62$(figura 5.1), para la fabricación de las PCBs. Una vez creada la placa la soldadura se realizó con un soldador de mano y la ayuda de un microscopio (figura 5.1).

(a) *Protomat S62*

(b) *Microscopio y estación de soldadura*

Figura 5.1 – *Herramientas usadas para la fabricación hardware*

El proceso seguido fue el mismo en las tres placas:

1. Preparación de los archivos *GERBER y NCdrills*: estos archivos se exportan desde Altium al programa *CircuitCAM* donde se motan las distintas capas, se pueden incluir zonas de *rubout* si se desea para facilitar la soldadura a mano y evitar contactos no deseados, y finalmente, se aislan todas las capas. Una vez creado el archivo correctamente exportamos un archivo de extensión LMD.

2. El archivo .LMD es el que utiliza el *software BoardMaster*, que es el que se comunica con la máquina. Mediante este *software* calibramos la máquina para que las capas *top y bottom* esten bien alineadas.

5.1.1 Fabricación placa oscilador

Puesto que de la placa base teniamos un prototipo en baquelita, esta fue la primera placa que imprimimos.

En la figura 5.2 podemos ver una captura del *software* CircuitCAM en el que ya están todas las capas aisladas, listo para exportar al archivo LMD:

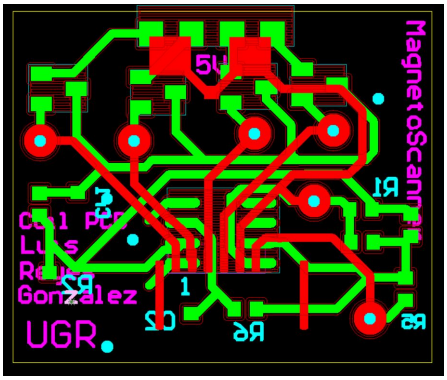
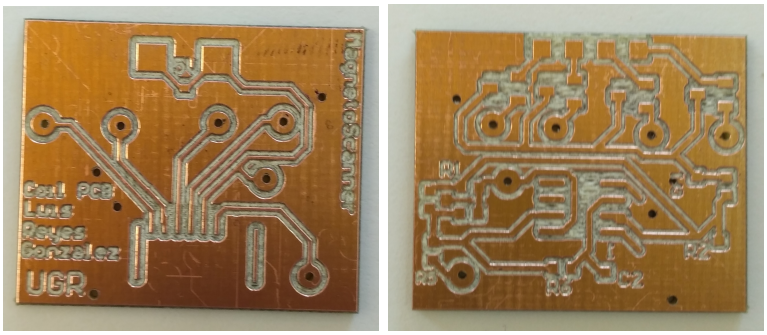


Figura 5.2 – *Captura circuitCAM placa oscilador*

Una vez realizada la placa queda como podemos ver en la figura 5.3:

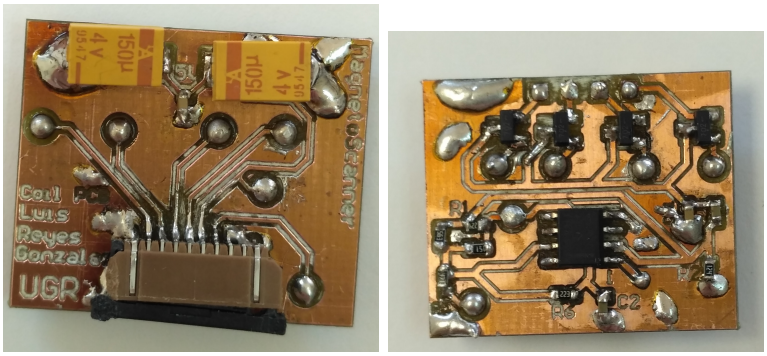


(a) *Top placa oscilador* **(b)** *Bottom placa oscilador*

Figura 5.3 – *Vistas de la placa oscilador impresa*

5

Procedemos a soldar la placa y vemos el resultado en la figura 5.4:



(a) *Top placa oscilador soldada* **(b)** *Bottom placa oscilador soldada*

Figura 5.4 – *Vistas de la placa oscilador soldada*

Si recordamos el diseño $3D$ de la placa obtenido en altium (figura 4.16) vemos que era una aproximación muy fiel a como quedaría una vez soldada la placa.

5.1.2 Fabricación placa base y placa motor

Una vez impresa la placa del oscilador pasamos a fabricar las otras dos placas que luego uniremos formando una sola.

En la figura 5.5 podemos ver una captura del *software* CircuitCAM en el que ya están todas las capas de la placa base aisladas aisladas, listo para exportar al archivo LMD:

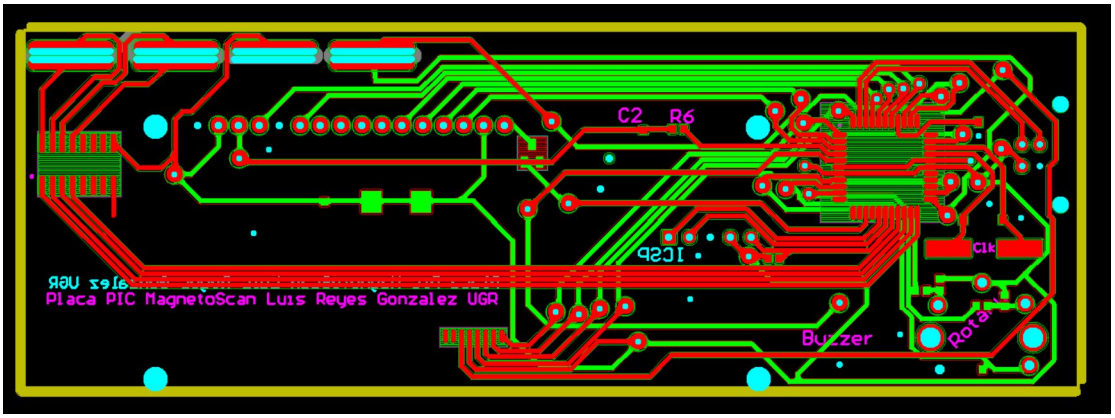


Figura 5.5 – *Captura circuitCAM placa base*

A continuación mostramos en la figura 5.6 una captura del *software* CircuitCAM en la que está la placa motor lista para su impresión.

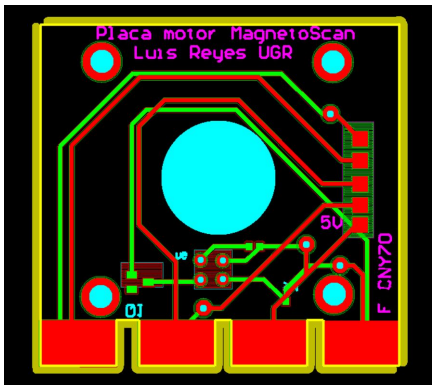
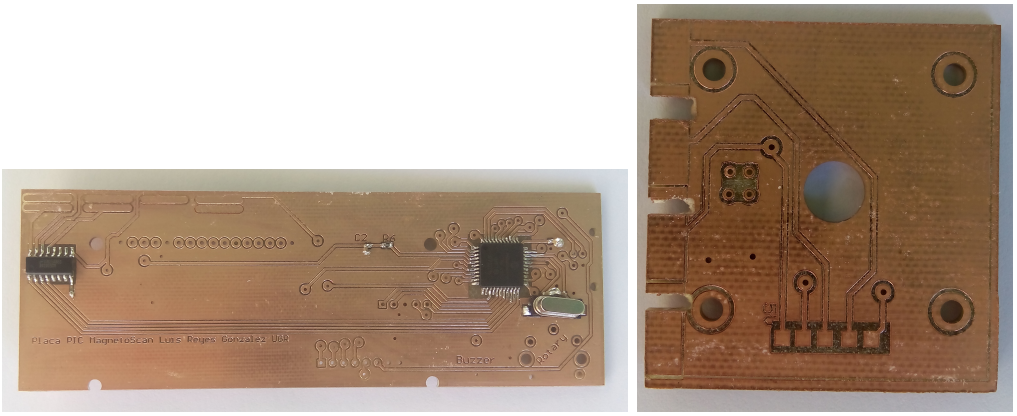


Figura 5.6 – *Captura circuitCAM placa motor*

Una vez realizadas las placas quedan como podemos ver en la figura 5.7:



(a) *Placa base* **(b)** *Placa motor*

Figura 5.7 – *Placas base y motor impresas*

Procedemos a soldar ambas placas y vemos el resultado en la figura 5.8:

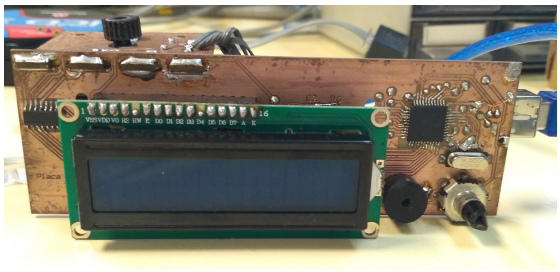


Figura 5.8 – *Ensamblaje de placas*

Podemos apreciar como la placa de motor sirve para darle rigidez al diseño y para sostener el motor, esto ha sido posible gracias al diseño de las aberturas y soldado de ambas placas.

5.2 Mecánica

En el capítulo 3 mencionamos que la herramienta de diseño $3D$ usada ha sido solidworks, pero, para la impresión $3D$ hemos usado un *software* de código libre llamado *Cura*. Se ha tomado esta decisión porque este *software* permite una ajuste sencillo de todos los parámetros importantes de la impresora, como puede ser el porcentaje de llenado, tamaño de la cama etc. La impresora usada para la impresión ha sido la *Prusa Mendel I3* (figura 5.9).

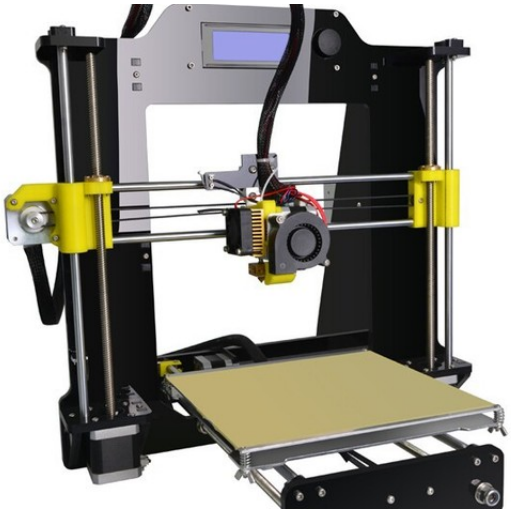
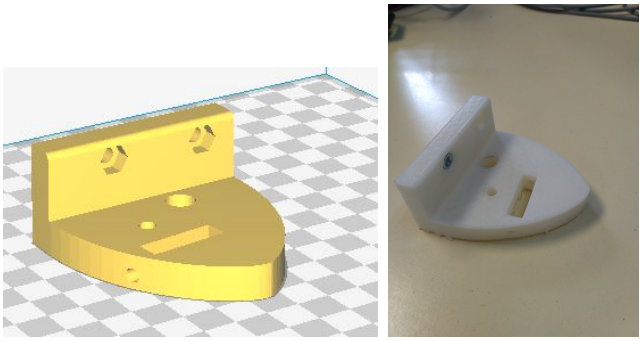


Figura 5.9 – *Impresora 3D*

El proceso seguido para todas las piezas ha sido siempre el mismo:

1. Exportar la pieza de altium a un formato .STL para trabajar en cura.
2. Colocar la pieza de manera que sea imprimible, es decir, no tenga partes voladas.
3. Seleccionar los parámetros adecuados a la impresión, por ejemplo, en piezas grandes la "cama" (espacio que se imprime de más de plástico para agarrar la pieza a la base de la impresora) debe tener mayor tamaño.
4. Imprimir y comprobar con calibre la variación de las medidas diseñadas a las reales.
5. Ajustar diseño y volver a repetir estos pasos.

En la figura 5.10 vemos un ejemplo de como hemos colocado el soporte superior en cura y el resultado una vez impresa:



(a) *Colocación en Cura* **(b)** *Pieza impresa*

Figura 5.10 – *Colocación en Cura y vista impresa*

CAPÍTULO

6

TEST Y VALIDACIÓN

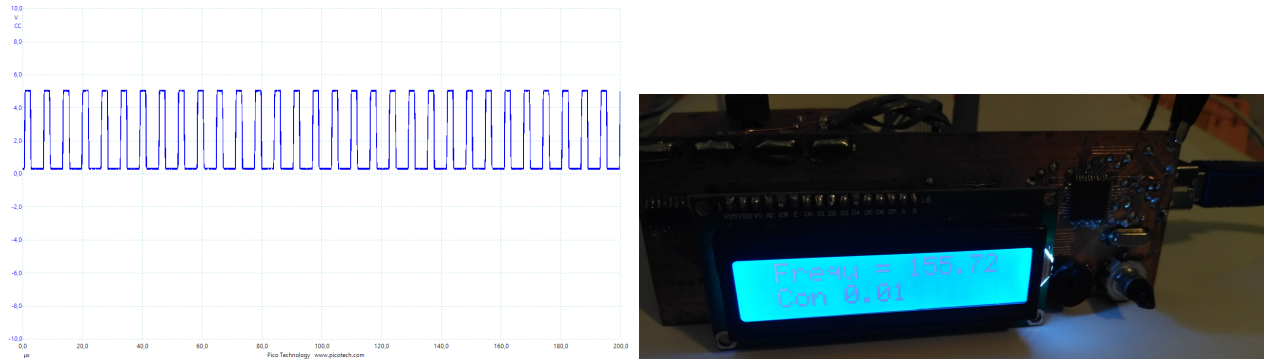
Una vez fabricadas y montadas nuestras PCBs pasamos a la fase de *test* de cada una de las funcionalidades del sistema. La manera de hacer esto es ir probando una a una las distintas funcionalidades comprobando con el osciloscopio que la señal es correcta para corroborar el buen funcionamiento del sistema, o, en caso contrario, arreglarlo.

Las funcionalidades que se probarán serán:

- Lectura frecuencia placa oscilador.
- Comprobación funcionamiento del $CNY70$.
- Comprobación funcionamiento USB.
- Comprobación funcionamiento LCD.
- Lectura rotary encoder.
- Movimiento motor.

6.1 Lectura frecuencia placa oscilador

Puesto que es la funcionalidad más importante la probaremos la primera. Una vez montado el sistema comprobamos probamos la rutina creada en el capítulo 4 y vemos el resultado en la figura 6.1.



(a) *Señal en el PIN CCP1 del PIC* **(b)** *Frecuencia leída por el PIC*

Figura 6.1 – *Señal osciloscopio y frecuencia leida PIC*

En el osciloscopio observamos la frecuencia de la señal, 155.7 Khz, mientras que en el PIC leemos la frecuencia como 155.72 Khz.

6.2 Comprobación funcionamiento del CNY70

La comprobación del correcto funcionamiento del $CNY70$ la haremos en varias partes. La primera aproximación la hicimos bajando la frecuencia de la señal de control y mediante una cámara de fotos observamos que si que se encendía y apagaba la luz infrarroja, después comprobamos con el osciloscopio en la figura 6.2 que efectivamente se enciende y apaga con la señal proveniente del PIC al MOSFET.

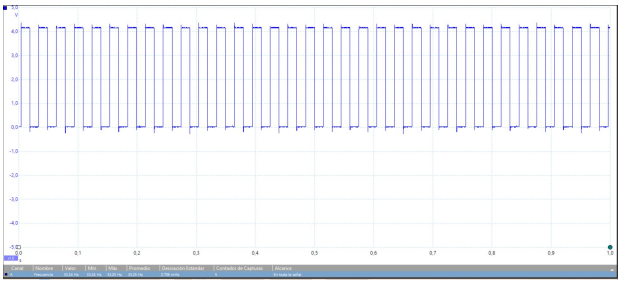


Figura 6.2 – *Señal en la masa del LED infrarrojo.*

Una vez comprobado activamos el final de carrera óptico y comprobamos que la señal que obtenemos tiene la misma que la señal de salida del PIC al MOSFET (figura 6.3). Finalmente comprobamos que el código funciona correctamente y detecta la interrupción.

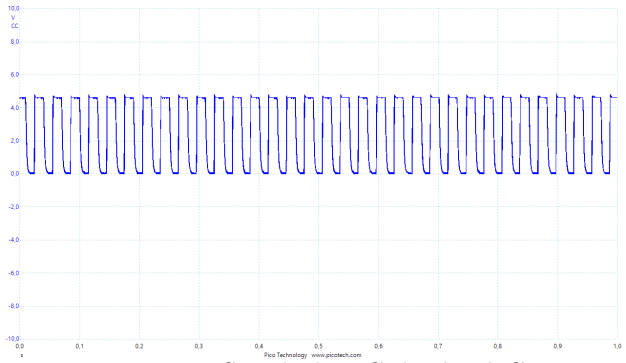


Figura 6.3 – *Señal al PIC desde el CNY70*

Efectivamente la frecuencia dela señal es la misma en ambos casos y, aplicando la misma rutina que para el caso del oscilador obtenemos la frecuencia de oscilación, en caso de coincidir con la frecuencia de entrada activa el *flag* de final de carrera, lo que impide el movimiento hacia abajo del motor.

6.3 Comprobación funcionamiento USB

El primer paso es conseguir que el ordenador detecte el USB como puerto serie. En este punto hubo un problema ya *windows 8* no permite la instalación de controladores sin firmar, y los proporcionados por microchip no están firmados. Esto se solucionó entrando en las opciones avanzadas del sistema y permitiendo la instalación de esos controladores.

Una vez detectado comprobamos que la lectura y escritura por parte del PIC es correcta.

6.4 Comprobación funcionamiento LCD

Comprobamos que fuera posible mostrar un mensaje y borrarlo.

6.5 Lectura rotary encoder

Primero comprobamos que la interrupción del botón funciona (figura 6.4).

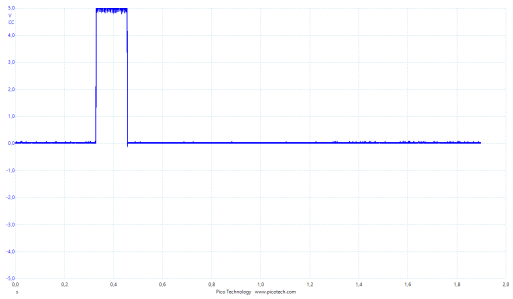
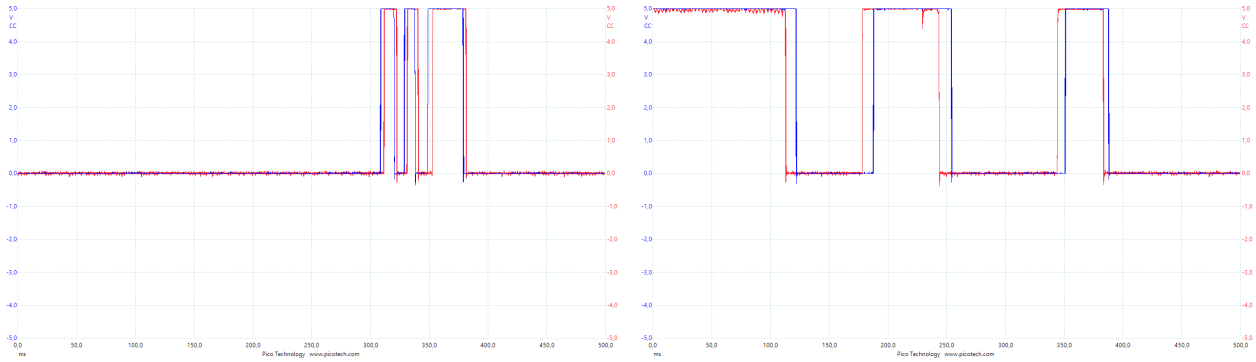


Figura 6.4 – *Señal originada al pulsar el botón.*

Al configurar la interrupción externa como cambio *low to high* entra en la interrupción en el flanco de subida, esto se ha configurado así para determinar también una pulsación larga (permanecer en estado alta durante 500 ms), cosa que habría sido imposible si detectasemos el flanco de bajada.

El siguiente paso es comprobar el movimiento del rotary (figura 6.5)



(a) *Señal al mover rotary hacia la izquierda* **(b)** *Señal al mover rotary hacia la derecha*

Figura 6.5 – *Comprobación funcionamiento rotary encoder*

Comprobamos como efectivamente según la dirección en la que movemos el rotary el cambio a alta se produce en órdenes distintos. Como ya explicamos en el capítulo 4 esto sirve para detectar la dirección del movimiento simplemente comprobando el estado de ambos pulsos y su estado anterior.

6.6 Movimiento motor

Comprobamos la señal en los pines del motor y si se mueve (figura 6.6).

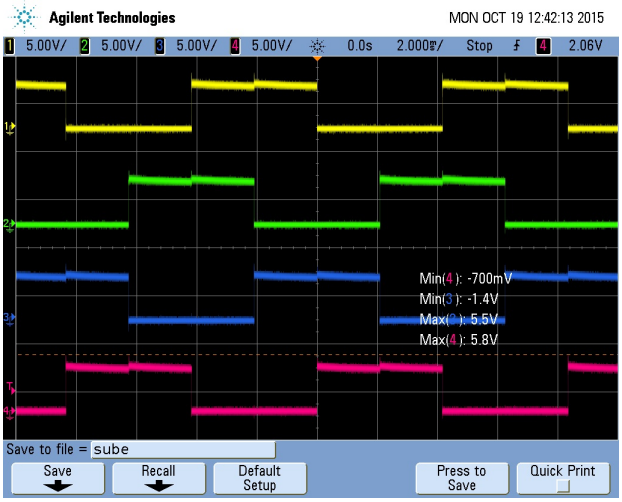


Figura 6.6 – *Señales en el motor*

Efectivamente las señales son correctas y el motor se mueve. Podemos fijarnos en como

se aprecia la configuración de paso doble en las señales, puesto que siempre hay dos pines en alta y van sucediéndose en orden.

6

CAPÍTULO

7

CONCLUSIÓN Y LÍNEAS FUTURAS.

El útimo capítulo ofrece una valoración personal del proyecto acerca de los resultados obtenidos. Adicionalmente haremos algunas consideraciones sobre futuras mejoras que se pueden realizar para continuar este trabajo.

Una vez terminado el proyecto es el momento de volver la vista atrás para ver todo el recorrido avanzado durante este proyecto. Como todo proyecto este empezó con un trabajo de documentación y análisis, puesto que partíamos de un prototipo al que había que había que realizar ciertas mejoras era necesario entenderlo completamente. Después comenzamos el diseño, una tarea difícil cuando uno no se ha enfrentado nunca a ello, ya que los conocimientos estaban ahí pero había que ordenarlos.Aunque poco a poco, con trabajo y esfuerzo se consiguió llegar a un primer prototipo con más errores que aciertos.

Superado ya el primer acercamiento cosas que al comenzar requerian mucho trabajo y tiempo hoy son casi como sumar, así ha sido por ejemplo todo lo concerniente al *Altium* y *Solidworks*. Herramientas con las que tuve el primer contacto a lo largo del proyecto y con las cuales me considero muy familiarizado a día de hoy.

Para terminar destacar el sentimiento de satisfacción tanto por el trabajo realizado, como por los conocimientos adquiridos durante él, conocimientos que da la experiencia, como por ejemplo los pasos a realizar para descubrir un fallo en un sistema extenso, como elegir componentes a la hora de realizar un diseño y un largo etcétera.

Una vez terminado el proyecto pasamos a enumerar las diferentes mejoras que podrían realizarse con el fin de complementar la funcionalidad del sistema diseñado:

1. Posibilidad de cargar sesiones anteriores.
2. Añadir una tarjeta de memoria SD para el trabajo de forma autónoma del prototipo.
3. Visualización en 3 dimensiones de la concentración, la posición y el tiempo.
4. Creación de un archivo instalable que incluya python, las librerias necesarias de python y los drivers del producto.
5. Añadir la opción de elegir el líquido portador en el programa.

7

BIBLIOGRAFÍA

[1] http://www.actimat.es/web/magnetoreologicos.asp.

[2] Documentacion c3js. http://c3js.org/reference.html.

[3] Flask. http://flask.pocoo.org/.

[4] Py serial's documentation. https://pythonhosted.org/pyserial/.

[5] Rotary encoder tutorial. http://www.hobbytronics.co.uk/rotary-encoder-tutorial.

[6] Axelson, J. *USB complete*, 3 ed. 2005.

[7] G. R. Iglesias, M. T. López-López, A. V. D., and Durán, J. D. G. Description and performance of a fully automatic device for the study of the sedimentation of magnetic suspensions.

[8] Microchip. Datasheet 18f4550. http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf.

[9] Vishay. Cny70. www.vishay.com/docs/83751/cny70.pdf.

[10] y José Mª Zamanillo Sáinz de la Maza, C. P. V. Diseño de bobinas. http://personales.unican.es/perezvr/pdf/Bobinas1.PDF.

ANEXO: ESTIMACIÓN DE COSTES

En el presente anexo haremos una estimación coherente de los costes que habría llevado realizar el proyecto en una empresa externa. Para ello clasificaremos los gastos en cuatro tipos: *hardware*, *software*, mecánicos y humanos, con un gasto total entre todos de 38664.38 €. El gasto en la creación de un segundo prototipo sería de 68.298 €, que sería el gasto correspodiente a los componentes y el plástico, más el coste humano asociado a la soldadura de las placas y el ensamblaje del producto.

Los diferentes recursos utilizados fueron detallados en el capítulo 4. Aunque los medios facilitados por el departamento de electrónica no han supuesto un coste real los tendremos en cuenta en este anexo.

Costes *software*

Las necesidades en *software* incluyeron las licencias de Solidworks, Altium y CircuitCam y la compra de un dispositivo MPLAB ICD 3, con un coste de 196.09 €, para programar y *debuggear* el microcontrolador.

Costes mecánicos

Los costes mecánicos son el plástico extruido, que tiene un coste de 20 € el rollo.

Costes *hardware*

- Una placa de cobre $AE16$ de $100x160x1.6$ mm: 4.16 €.
- Inventario de componentes (tabla 1): 44.138 €

Componente	Cantidad	Coste unitario	Coste total
LCD 16x2	1	8.16	8.16
Rotary encoder	1	0.87	0.87
Mosfet 2N7002	6	0.066	0.396
USB hembra tipo B	1	0.822	0.822
Cristal 16Mhz	1	0.51	0.51
Buzzer	1	2.21	2.21
Condensadores 0603	8	0.024	0.144
Condensador tantalio	4	1.854	7.416
Resistencias 0603	15	0.02	0.30
LM311	1	0.214	0.214
PIC18F4550	1	4.38	4.38
CNY70	1	1.016	1.016
Conector FFC	2	0.496	0.992
DS2003	1	0.449	0
Cable FFC	1	3.768	3.768
Stepper Motor	1	13.01	13.01

Table 1 – *Componentes usados en proyecto*

Costes humanos

La totalidad del proyecto ha sido desarrollado por una única persona. El coste del trabajo llevado a cabo para un perfil de ingeniero junior, considerando un coste de 20 € por hora y un tiempo de trabajo de 1920 horas aproximadamente, el cual se desglosa de la manera siguiente:

- Diseño *hardware*: 300 horas.
- Diseño *software*: 250 horas.
- Diseño mecánico: 100 horas.
- Implementación y testeo *hardware*: 450 horas.
- Implementación y testeo *software*: 400 horas.
- Implementación y testeo mecánico: 70 horas.
- Ensamblaje producto final y corrección de errores: 350 horas.

Lo que conllevaría un coste humano de 38400 €.

ABREVIATURAS Y SIGLAS

ACRÓNIMOS

C	Condensador	2, 3, 8, 13, 14
CCP	CAPTURE/COMPARE/PWM	8, 12
GUI	Graphical User Interface	xviii, 4, 7, 8, 24, 51, 58
ICSP	In-Circuit Serial Programing	8, 12, 24, 43
L	Bobina	2, 3, 8, 13, 14, 28
LCD	Liquid Crystal Display	xx, 8, 33, 34, 55, 57, 70, 79, 81

MOSFET	Metal oxide semiconductor eld e¤ect transistor	22, 23, 39, 42, 45, 48, 80, 81
PC	Personal Computer	22, 24, 34, 51
PCB	Printed Circuit Boards	5, 11, 42, 43, 48, 66, 67, 73, 79
PIC	Peripheral Interface Controller	12
PWM	Pulse-Width Modulation	34, 52, 55
RAM	Random Access Memory	12
SMD	Surface Mount Devices	xxix, 42, 43, 45, 48
USB	Universal Serial Bus	xviii, xx, 8, 12, 22, 24, 32, 34, 43, 51–53, 57–59, 61, 70, 79, 81

GLOSARIO

LISTADO DE SÍMBOLOS

Ω [Ohms]	Unidad de resistencia eléctrica.	xxix
π [pi]	ratio of circumference of circle to its diameter	xxix