Trabajo Fin de Grado

Diseño de un telémetro de bajo coste.

Autor

Cresko Yulvir Puyana Morón

Tutor

Alberto Francisco Hamilton

Escuela Superior de Ingeniería Industrial

2015/2016

Índice

1. Agradecimientos

Veo necesario dedicar un pequeño capítulo a hacer una reflexión acerca del camino que me ha llevado hasta aquí. Con la presentación de este Trabajo de Fin de Grado termino una etapa de mi vida; finalizar el Grado en Ingeniería Electrónica Industrial y Automática. Para mí esto no es nada trivial. Quizás, debido tanto a la situación en mi país de origen como a la situación de mis familiares más cercanos, soy consciente de la suerte que tengo al estar donde estoy.

Soy el primero de mi familia en conseguir una titulación universitaria. Cierto es que conseguirlo no ha sido sencillo. No obstante, para mí el verdadero reto han sido las distintas situaciones de mi vida con las que he tenido que lidiar.

Es por esto que he de agradecer con total sinceridad a tres familiares sin los cuales no lo hubiera logrado.

En primer lugar a mi padre, pues estoy plenamente convencido que que sin su influencia no habría conseguido ni mis metas académicas ni la personalidad que me permitirá obtener muchas más cosas a lo largo de mi vida.

En segundo lugar a mi madre. Todas las madres merecen un agradecimiento por los méritos que consiguen sus hijos, pero hay madres que a pesar de sus propias limitaciones hacen todo lo que esta en su mano por ayudar a sus hijos a lograr sus objetivos, no solo económicamente sino de otras formas mucho más importantes.

Finalmente, creo justo mencionar a mi tía Pilar, pues a pesar de estar muy lejos y de solo compartir conmigo una parte de la infancia, siempre ha hecho un esfuerzo por ayudar a mi familia. Va aun mas allá el interés que siempre ha mostrado por que yo pudiera completar mis estudios.

Por otro lado, durante la ejecución de este proyecto me encontré diversos obstáculos. Estos son los propios de ejercer la profesión de ingeniero: conocimientos que no tenemos y debemos adquirir para poder resolver un problema, cosas que aun conociendo su funcionamiento a la hora de la verdad no somos capaces de hacer funcionar etc.

Por ello, debo agradecer a varias personas la ayuda que me han prestado. En primer lugar a mi tutor Alberto Hamilton. Ha sido un profesor muy implicado e interesado en el desarrollo del proyecto. No solo ha cumplido como tutor si no que además siempre que he necesitado contactarle, su respuesta ha sido rápida y su paciencia infinita.

Otra persona que a pesar de no tener ninguna obligación de hacerlo, se mostró dispuesta a ayudarme en varias ocasiones, es la profesora Beatriz Rodríguez Mendoza quien me impartió la asignatura de Diseño y Tecnología de Circuitos Impresos.

Por último y para concluir, debo nombrar al técnico de laboratorio Manuel Fernández Vera, ya que siempre que necesitaba herramientas, materiales e incluso consejo, amablemente lo hacia. Este hecho aún fue más marcado durante el intento de fabricación de la PCB, en el que durante los repetidos intentos, tuvo mucha paciencia permitiéndome acceder a la insoladora y al resto de materiales, e incluso haciendo algún intento conmigo.

2. Abstract

The aim of this project is the obtainment of a low- cost device that takes distance measures and that can use them in order to get a system able to recognize the setting in which the device is placed. To do this, it has been created the telemeter LIDAR-Lite; it is a low- cost device which can take measures of its environment, using a higher accuracy than other devices and which result to be of a similar price but in which functioning is based on ultrasounds. Nonetheless, the LIDAR - Lite, due to its low price, has some limitations which a priori do not allow the device to carry out the different tasks that other commercials telemeter can. This is the reason why the necessity of making this project arose.

Some of the improvements that have been implemented throughout this project can be encompassed in four blocks:

- The necessity of taking distance measures in different angles: we have decided to couple the telemeter to the stepper motor axis and to control the rotation of it in order to take measures in different directions.
  The necessity of knowing the direction in which the telemeter carries out the measures: we have opted for the attachment of a sensor to the system as it can give us the proper information. The easier and cheaper solution that has been taken is the opto-interrupter.

- Programming: the brain in charge of the implementations of the necessary instructions for the correct functioning of the system is a Rasperry Pi. This apparatus has an operating system based on GNU/ Linux; it withstands naturally the programming by using the Python language, with which we have designed the algorithms needed for the control of the engine rotation, the interpretation of the information obtained from the opto-interrupter, the communication through I2C with the LIDAR-Lite and the processed of data.

- Design of the electronic circuit: we have made an electronic circuit able to provide the required tensions and streams for all the set. Furthermore, in order to fulfill the aim of having a system as compact as possible, we have designed a suitable circuit in the software Orcad Capture and Orcad Layout so as to obtain what is needed to fabricate a PCB.

- 3D structure design: through the union of all the components already mentioned, we have achieved a device which has the basic features of a commercial telemeter of a higher price. For the assembly and protection of all these components, it is highly important and necessary the design of the required fragments.

3. Introducción

Este proyecto de final de grado ha sido propuesto por el profesor Alberto Francisco Hamilton Castro del departamento de Ingeniería Informática y Sistemas y ha sido realizado por Cresko Yulvir Puyana Morón, alumno del Grado en Ingeniería Electrónica Industrial y Automática.

La finalidad de este proyecto es obtener un dispositivo de bajo coste que permita medir distancia y utilizarlas para obtener un sistema que sea capaz de reconocer el entorno. Para ello, se ha utilizado un dispositivo electrónico que realice dichas medidas; un telémetro LIDAR-Lite. Se trata de un dispositivo de bajo coste con una precisión mucho mayor que otros dispositivos de precio similar pero cuyo funcionamiento se basa en ultrasonidos. No obstante, debido a su bajo precio, presenta limitaciones que a priori no le permiten realizar las tareas que desempeñan otros telémetros comerciales, es por esto que surge la necesidad de este proyecto.

Así pues, a modo de resumen podemos decir que el objetivo de este proyecto es mejorar las características del LIDAR-Lite para obtener un producto que pueda reemplazar la tarea realizada por los telémetros comerciales de mayor precio.

Las mejoras que se han planteado a lo largo del desarrollo de este trabajo las podemos englobar en cuatro grandes bloques:

- Necesidad de tomar medidas en diferentes ángulos: este telémetro, al contrario que la mayoría de telémetros comerciales, no permite tomar medidas en más de una dirección, pues no cuenta con un mecanismo que permita que la lente que emite y recibe la luz gire en torno a un eje, para así, variar la dirección en la que se realiza la medida. Esto no tiene gran aplicación en robótica móvil ya que necesitamos conocer los obstáculos que nos rodean en todas las direcciones. Para solucionar esto, existe más de una alternativa. En este caso se ha optado por acoplar dicho telémetro al eje de un motor paso a paso y controlar el giro de éste, para así, poder tomar medidas en diferentes direcciones.

- Necesidad de conocer la dirección en la que el telémetro realiza la medida: los motores paso a paso, a pesar de que son convenientes para un posicionamiento muy preciso, no proveen del giro que han realizado con respecto al cero de un sistema de coordenadas. Es por esto que hemos optado por acoplar un sensor al sistema que nos pueda dar dicha información. La solución más sencilla y económica que se ha encontrado es un opto-interruptor.

- Programación necesaria: el cerebro encargado de ejecutar las instrucciones para el correcto funcionamiento del sistema es una Raspberry Pi. Este dispositivo corre un sistema operativo basado en GNU/Linux; soporta nativamente la programación usando el lenguaje Python, con el que se han diseñado los algoritmos necesarios para el control del giro del motor, la interpretación de la información obtenida del opto-interruptor, la comunicación mediante I2C con el LIDAR-Lite y el procesado de los datos obtenidos.

- Diseño del circuito electrónico: hemos diseñado un circuito electrónico que sea capaz de proporcionar las tensiones y corrientes necesarias para todo el conjunto. Partiendo de un alimentador de entrada de 12 voltios y 2 amperios, se ha diseñado un circuito apto para obtener los 5 voltios necesarios para alimentar las Raspberry y el telémetro. Por otro lado, para lograr el objetivo de tener un sistema los más reducido posible, se realizó el diseño del circuito en el software Orcad Capture y Orcad Layout para obtener lo necesario para fabricar una PCB lo más compacta posible.

• Diseño 3D de la estructura: como hemos visto, este proyecto se basa en utilizar componentes de bajo coste, los cuales por separado, no parecen que puedan cumplir con los requisitos de un sistema de reconocimiento del entorno. Sin embargo, mediante la unión de todos ellos se ha logrado un dispositivo que cumpla con las características básicas de un telémetro comercial mucho más costoso. Para el ensamblaje de todos estos componentes es fundamental crear una estructura. Se ha optado por una solución modular basada en tres piezas ya que debido a las limitaciones de las impresoras 3D, no era posible crearlo solo en una. No obstante, consideramos una ventaja que sea modular, pues en caso de rotura de una de las partes, no hay que imprimir de nuevo todo el diseño. Además permite un acceso más sencillo a los diferentes componentes.

Para afrontar la explicación de los pasos seguidos en la resolución de cada uno de los objetivos, hemos decidido dividir este documento en capítulos que engloben las acciones realizadas en cada una de las áreas. Estos capítulos tendrán en su inicio la descripción de los fundamentos teóricos necesarios para el entendimiento de las tareas desarrolladas.

Así pues, en el capítulo 4 hemos englobado todas las tareas llevadas a cabo en relación a la electrónica. En él se describen el funcionamiento de los motores paso a paso, el diseño de una placa de circuitos impresos, el funcionamiento básico de los convertidores DC/DC etc.

A continuación, el capítulo 5, está dedicado al diseño y fabricación de las piezas necesarias para el ensamblaje de todo el sistema. Para ello se ha hecho un breve comentario acerca del software de diseño y de la impresora utilizada. Para terminar, se realizara un recorrido por los diferentes prototipos desarrollados y explicando su evolución hasta el diseño final.

Seguidamente, dedicamos el capítulo 6 a la configuración y programación necesaria en este proyecto. Es una parte fundamental del trabajo, pues en él se describen los algoritmos diseñados y utilizados para llevar a cabo tanto el control de la posición del motor como la toma de medidas con el telémetro. Por último, se describe como se ha afrontado la creación de un mapa basándose en las medidas tomadas.

A continuación, he incluido un presupuesto de lo que costaría ejecutar este proyecto tanto en materiales, como en horas de trabajo. Terminaremos esta redacción con una valoración personal del trabajo realizado a modo de conclusión.

4. Electrónica

En este capítulo se explicará tanto el funcionamiento básico de los distintos elementos utilizados en este proyecto como el papel que desempeñan dentro del sistema. Para esto, empezaremos con una explicación teórica del funcionamiento de cada elemento por separado para ir obteniendo de manera progresiva una visión global del ensamblaje de todos ellos.

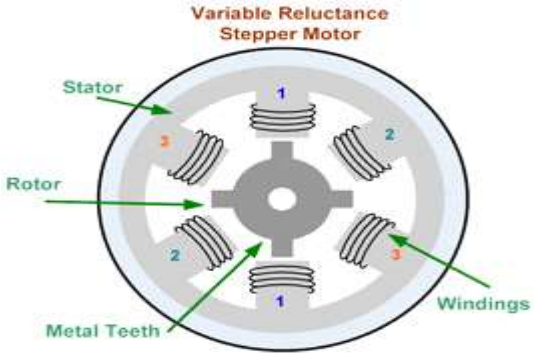
4.1. Motores paso a paso

Los motores paso a paso son ideales para la construcción de mecanismos en los que se requiera movimientos muy precisos. Sus principales aplicaciones se pueden encontrar en robótica de manipulación y posicionamiento, control de discos duros, unidades de CD, impresoras etc.

A pesar de que la velocidad de giro es más lenta que la de un motor de corriente continua, podemos controlar cuanto giran sin necesidad de un encoder, ya que sí podemos controlar con bastante precisión los pasos que giran. [1]

Los motores paso a paso se clasifican en tres tipos:

Motor de reluctancia variable: el rotor es de material magnético, pero no es un imán permanente. Éste tiene forma dentada y tendera a alinearse cuando se energice el estátor, de forma tal que minimice la reluctancia rotor-estátor.



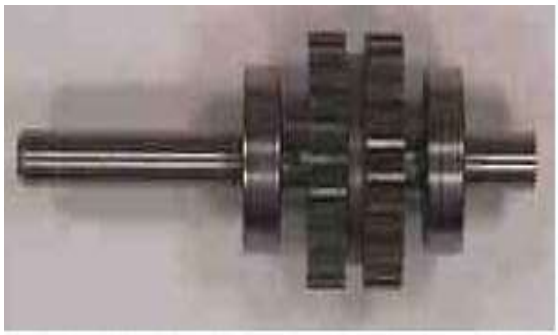
El par se produce como resultado de la atracción entre las bobinas y el rotor férrico. El rotor forma un circuito magnético con el polo del estátor. La reluctancia en un circuito magnético es el equivalente a la resistencia en un circuito eléctrico. Cuando el rotor está alineado con el estátor, el hueco entre ambos es muy pequeño y en este momento la reluctancia es mínima.

Por otro lado, la inductancia de este circuito también variará cuando el rotor gire. Si el rotor esta fuera de la alineación, la inductancia es muy baja y por tanto la corriente aumentará rápidamente. Si por el contrario, rotor y estátor están alineados, la inductancia será muy grande, siendo esto uno de los principales inconvenientes para controlar este tipo de motores.

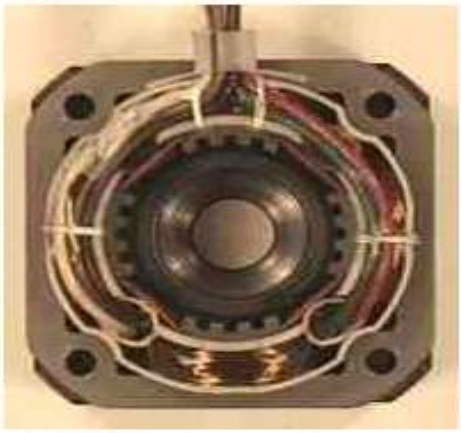
Estos motores no necesitan de un controlador externo para girar, pues todas las bobinas tienen un terminal común que se suele conectar al borne positivo y son alimentadas siguiendo una secuencia consecutiva.

Pueden diseñarse para conseguir pasos más pequeños que se consiguen mediante motores paso a paso de imán permanente, sin embargo, suelen tener un par menor.

Motor de magnetización permanente: son los más usados en robótica. Están formados por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas en su estátor.



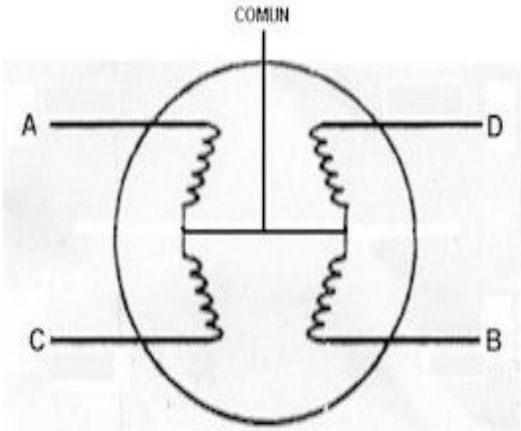
Rotor



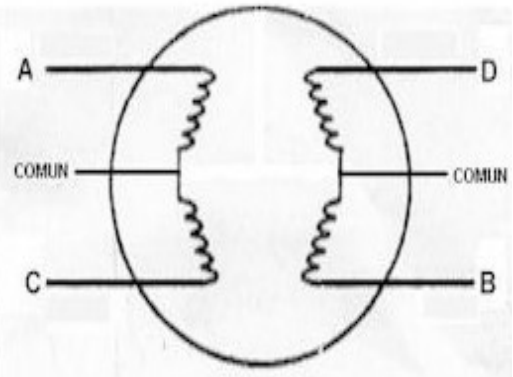
Estator de 4 bobinas

Existen dos tipos de motores paso a paso de imán permanente:

- Unipolares: suelen tener 5 o 6 cables de salida dependiendo de su conexionado interno.



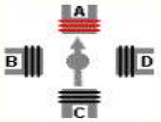
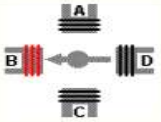
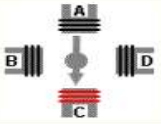
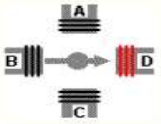
Motor P-P con 5 cables de salida



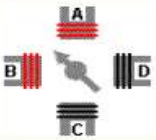
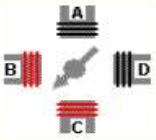
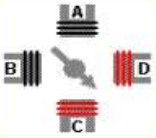
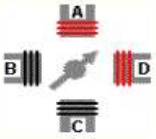
Motor P-P con 6 cables de salida

Este tipo se caracteriza por ser más simple de controlar que los bipolares ya que pueden ser controlados simplemente con transistores que permitan o no el paso de corriente a cada bobinado. Basándonos en esto, podemos conseguir de manera muy sencilla que el motor gire un paso o medio paso. Para explicar cómo se consigue esto, usaremos de ejemplo un motor unipolar de cuatro bobinas. El giro de un paso lo podemos conseguir energizando dos bobinas simultáneamente o energizando solo una bobina, aunque es preferible energizar dos bobinas a la vez ya que se consigue un par de giro y de retención mayor.

Energización de una única bobina para girar un paso

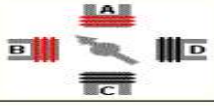
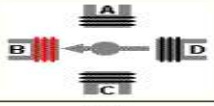
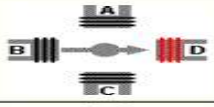
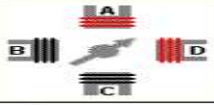
PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	ON	OFF	OFF	OFF	
2	OFF	ON	OFF	OFF	
3	OFF	OFF	ON	OFF	
4	OFF	OFF	OFF	ON	

Energización simultanea de dos bobinas para girar un paso

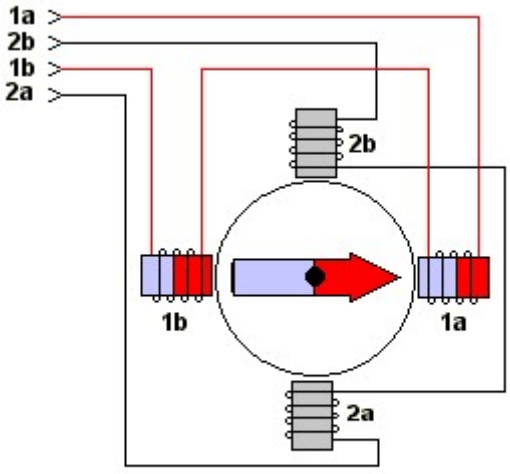
PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	ON	ON	OFF	OFF	
2	OFF	ON	ON	OFF	
3	OFF	OFF	ON	ON	
4	ON	OFF	OFF	ON	

Si se combinan las secuencias anteriores, se obtiene un paso de la mitad de longitud que el paso estándar. Para ello, primero se activan dos bobinas y luego solo una. Esto da lugar a una mayor precisión. No obstante, da lugar a una velocidad de giro más lenta, ya que para dar una vuelta completa hay que dar el doble de pasos.

Energización necesaria para dar medio paso

PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	ON	OFF	OFF	OFF	
2	ON	ON	OFF	OFF	
3	OFF	ON	OFF	OFF	
4	OFF	ON	ON	OFF	
5	OFF	OFF	ON	OFF	
6	OFF	OFF	ON	ON	
7	OFF	OFF	OFF	ON	
8	ON	OFF	OFF	ON	

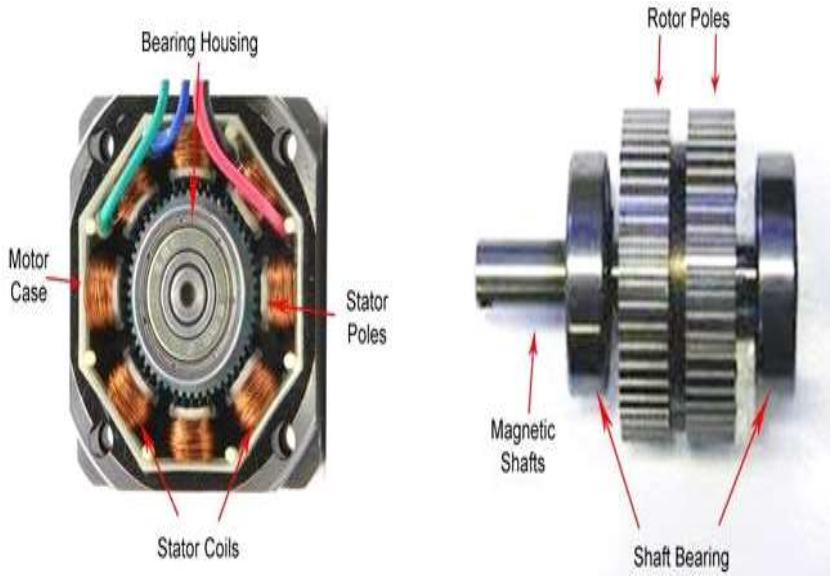
Bipolares: tienen generalmente cuatro cables de salida aunque los puede haber con 8 cables en aplicaciones en las que se necesite una gran cantidad de corriente eléctrica. Son más difíciles de controlar debido a que requieren del cambio de la dirección de flujo de la corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento, por lo que necesitaríamos un puente en H que nos permita ir cambiando a polaridad de los bobinados o recurrir a un controlador para esta tarea.



Conceptual Model of Bipolar Stepper Motor

Motor paso a paso híbrido: los motores híbridos paso a paso operan combinando los principios de los motores de imán permanente y los de reluctancia variable intentando obtener las características que destacan en cada uno de ellos. De esta forma resultan motores con unos ángulos de paso pequeños y un alto par de giro en un tamaño compacto.

Las características y formas del estátor son muy similares a los de un motor de imán permanente. La diferencia más importante se encuentra la estructura del rotor. Éste está formado por un disco cilíndrico imantado en posición longitudinal al eje. Las líneas magnéticas que genera el imán son guiadas por dos cilindros acoplados a los extremos de cada uno de sus polos (norte y sur), construidos generalmente por láminas de material ferro magnético y dentados que forman los polos del rotor. El motor híbrido produce un par por fuerza de reluctancia, igual que el motor de reluctancia variable. La diferencia entre ambos es el tipo de excitación utilizado. En el motor de reluctancia variable, la excitación es producida únicamente por medio del bobinado, mientras que en el motor híbrido, la excitación es conjunta entre el bobinado y el imán. Es por esto que los motores híbridos entran también en la misma clasificación de unipolares y bipolares.



4.2. Controladores motor paso a paso

Dependiendo del tipo de motor paso a paso utilizado se puede optar por más de una alternativa para controlar el energizado del bobinado del estátor, y por tanto el giro del motor. En la actualidad se pueden conseguir una gran variedad de controladores para estos motores; la mayoría para motores bipolares, ya que son los que requieren de una mayor complejidad para su control.

Su funcionamiento interno suele estar basado en puentes en H, pues esto permite el cambio de polaridad que necesitan estos motores. Una vez conocido mediante el apartado anterior el funcionamiento de los motores paso a paso, se puede entender con bastante facilidad la tarea que realizan estos controladores. Su tarea no es compleja y se podría hacer prescindiendo de ellos, pero debido al reducido precio y tamaño que han alcanzado resulta la opción más viable.

Por último, su uso facilita mucho la integración de los mismos con dispositivos programables como Raspberry pi, Arduino etc.

4.3.Telémetro LIDAR-Lite

Este producto es un telémetro láser fabricado por la compañía Pulsedlight3d, la cual actualmente no está muy claro a quién pertenece y si seguirá mejorando su principal producto, el LIDAR-Lite.

LIDAR-Lite es un producto dirigido a entornos donde se requiera un sensor de medidas de distancia óptico con un gran rendimiento, manteniendo un tamaño y peso muy reducidos. Sus principales aplicaciones están en el campo del mapeo 3D en vehículos autónomos tanto terrestres como aéreos.

Esta compañía diseñó este producto utilizando un único chip para procesar toda la información. Esto, acompañado del mínimo hardware necesario, dio lugar a un nuevo tipo de sensor que mejoró el rendimiento de los sensores actuales a un precio mucho más bajo.

Su principio de funcionamiento se basa en emitir un pulso de luz láser. La palabra "láser" viene del inglés "Light Amplification Stimulated by Emission of Radiation". La tecnología para medidas de distancia utilizada por la empresa PulsedLight3D esta basada en una medición muy precisa del tiempo transcurrido entre la emisión y la recepción de una señal. Esta técnica posee una alta fiabilidad, estableciendo la precisión de la medida por debajo de un centímetro. Para ello se basa en el promedio y digitalización de las dos señales.

Una señal de referencia es producida por el transmisor antes de enviar la señal que realizará la medida de distancia tras ser reflejada por el objetivo. El tiempo de retardo entre las dos señales guardadas se estima a través de un procesamiento de señales conocido como correlación. Su algoritmo de correlación calcula el tiempo de retardo. Esto se puede traducir en la distancia del objetivo basándose en la velocidad de la luz.

Este sensor debe ser alimentado entre 4.75 y 5.5 voltios y su consumo máximo está por debajo de los 100 miliamperios. La distancia máxima a la que es capaz de detectar un objeto, depende de la reflectividad de dicho objeto a la luz infrarroja. Objetos cuya reflectividad sea del 30% son detectados a una distancia máxima de 30 metros, mientras que objetos con reflectividad al infrarrojo del 90% serian detectados como máximo a 60 metros. El fabricante nos garantiza en esta toma de medidas un error máximo de +/- 2.5 centímetros y un tiempo de adquisición de dicha muestra siempre por debajo a 0.02 segundos.

Además, este dispositivo permite una comunicación serial basado en I2C en la cual se comporta como un esclavo y recibe órdenes en hexadecimal de un dispositivo maestro.

Cuenta con 6 cables de salida:

- Pin 1: es el positivo al que debemos conectar los 5 voltios de alimentación.
- Pin 2: es un pin activo a alta, si se usase, da la posibilidad de que el sensor trabaje con una alimentación de 3.3 voltios.
- Pin 3: en este caso es activo a baja y nos proporciona si el estado del sensor es ocupado o no.
- Pin 4: es utilizado para la comunicación mediante el bus I2C, concretamente se usa para el SCL (system clock), es la línea utilizada para los pulsos de reloj que sincronizan la comunicación.
- Pin 5: es utilizado para la comunicación mediante el bus I2C, concretamente se usa para el SDA (system data), es la línea por la que se envían los datos entre los dispositivos.
- Pin 6: es el cable utilizado para proporcionar la tierra (GND) al dispositivo.

4.4. Opto-interruptor

Es un sensor cuyo funcionamiento es muy sencillo. Consta de un emisor de luz infrarroja y un receptor de la misma, uno en frente del otro. de forma normal este haz de luz es detectado por el receptor. Si ocurriese que el receptor no fuera capaz de detectar el haz de luz, solo puede significar que entre emisor y receptor hay un objeto opaco que no permite la recepción. Sus aplicaciones son muy amplias y se puede encontrar en dispositivos tales como sensores de velocidad, contador de pulsos, finales de carrera etc.

4.5. Raspberry Pi

Es un ordenador de placa única (SBC: single board computer) de bajo coste. Se define un SBC a un ordenador construido en una sola placa de circuitos en el que incorpora todo lo necesario para su funcionamiento: microprocesador, memoria, entradas y salidas etc.

Su desarrollo comienza a principios de 2006 en Reino Unido con el objetivo de mejorar la enseñanza de la informática en las escuelas. Sin embargo, no es hasta febrero de 2012 cuando se realizaron las primeras ventas de un modelo completamente acabado y funcional.

En la actualidad encontramos un gran abanico de modelos Raspberry. Todos ellos siguen la característica común de mantener en un reducido espacio un ordenador plenamente funcional. El diseño de la Raspberry Pi B incluye un chipset Broadcom con una CPU a 700 Mhz, un procesador gráfico (GPU), un módulo de 512 Mb de memoria RAM, un conector RJ45 que nos proporciona acceso a la red a una velocidad de hasta 100Mbps, 2 buses USB 2.0, una salida de audio por Jack de 3.5 mm, salida digital de vídeo y audio por HDMI, pines de entrada y salida de propósito general y lector de tarjetas SD.

En su último modelo, Raspberry Pi 3 contamos con conectividad bluetooh y Wi-Fi y un nuevo procesador ARM cortex-A53 que funciona a 1,2 Ghz de 64 bits y la memoria RAM se aumenta hasta 1GB. Todo esto manteniendo el precio inicial de sus anteriores modelos, ya que su objetivo no es crear competencia entre sus distintos dispositivos, sino facilitar el aprendizaje de la informática de una forma económica.

La fundación Raspberry ha desarrollado un sistema operativo libre llamado Raspbian. Este sistema operativo se basa en Debian y esta optimizado para el hardware del dispositivo. Aunque existen muchas otras distribuciones para este mini-ordenador, Raspbian sigue siendo las mas completa, estable y que mejor rendimiento da en el dispositivo.

Sin embargo, debido a la popularidad de este ordenador, podemos encontrar un gran número de sistemas operativos capaces de funcionar en esta máquina, como Ubuntu Mate, OSMC, una versión optimizada de Windows 10, RaspAnd que permite ejecutar Android etc.

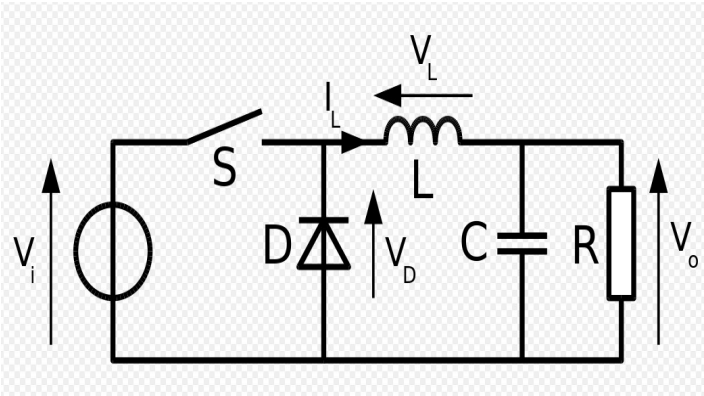
Debido a toda esta diversidad y a su filosofía de software libre, hay una gran comunidad en constante investigación que nos ofrecen usos infinitamente variados para este ordenador.

4.6. Convertidor DC/DC

Es un dispositivo que transforma corriente continua de una tensión a otra. La mayoría de los que se consiguen comercialmente se basan en transistores que funcionan en conmutación para reducir la potencia disipada por los mismos. Las frecuencias de conmutación suelen ser bastante elevadas para permitir reducir la capacidad de los condensadores y por tanto, reducir tamaño, peso y precio del producto final.

Hay varios tipos de convertidores DC/DC. Los podemos clasificar en tres grupos: reductores de tensión, elevadores de tensión y reductores-elevadores.

Se ha utilizado un circuito integrado LM2575 en el prototipo de la placa, siendo este un convertidor DC/DC reductor Buck. Éste es un convertidor de potencia que obtiene a su salida un voltaje continuo menor que el voltaje en su entrada. Es una fuente conmutada con dos semiconductores (un diodo y un transistor), un inductor y un condensador a su salida. En la siguiente imagen podemos ver el esquema básico.



El funcionamiento se basa en controlar la conexión del inductor mediante los dos semiconductores, alternando la conexión del inductor bien a la fuente de alimentación o bien a la carga se consigue reducir la tensión de salida.

El uso de convertidores DC/DC simplifica la alimentación del sistema, permitiendo generar las tensiones en el punto donde se necesitan, reduciendo así la cantidad de lineas de potencia necesarias. Por contra, generan ruido en la alimentación regulada que puede propagarse al resto del sistema si no es filtrado adecuadamente.

4.7. Paquete OrCAD

En la actualidad hay gran variedad de software para ordenador que nos permiten realizar el diseño de circuitos electrónicos, su simulación y la obtención de los ficheros necesarios para su producción, ya sea mediante fabricación en seco o mediante procedimientos químicos. Una de las alternativas que valoramos en este proyecto fue Fritzing. Este programa es de código abierto, por lo que mantiene la filosofía con la que hemos intentado desarrollar este proyecto. Sin embargo, debido a mis conocimientos previos en otra de las alternativas mas populares, optamos por realizar el diseño utilizando OrCAD.

OrcCAD es un paquete privado que tiene sus orígenes sobre los años 80. Su finalidad es permitir el diseño de circuitos electrónicos mediante ordenador. Para ello cuenta con varios programas integrados. OrCAD Capture nos permitirá realizar el conexionado de los distintos componentes de nuestro circuito. Con las conexiones realizadas podemos simular su funcionamiento gracias a PSPICE, otro de los programas que incluye OrCAD. Finalmente, si las conexiones realizadas son satisfactorias y la comprobación de su funcionamiento mediante simulación correcta, se puede utilizar Layout. Este programa recibirá la información necesaria acerca de los componentes y las conexiones para permitirnos realizar el diseño de una PCB y obtener lo necesario para su fabricación.

4.8. Trabajo realizado

Una vez conocidos todos los elementos que conformaran nuestro 'producto' final, nos disponemos a describir el trabajo realizado en orden cronológico.

4.8.1. Determinación tipo de motor

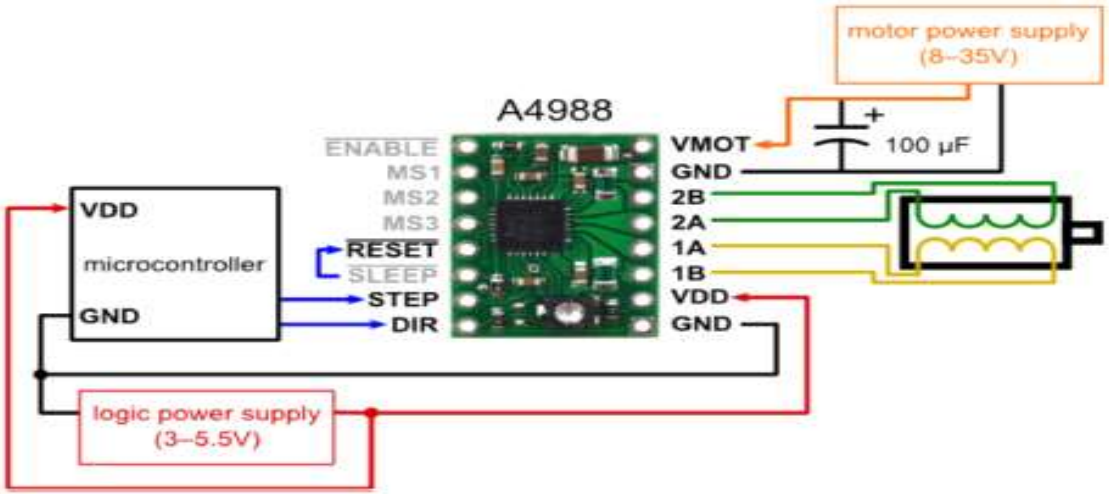
La universidad proporcionó un motor paso a paso modelo VEXTA PK244-02A-C58. El datasheet no proporciona demasiada información, por lo que tuvimos que basarnos en sus características. Se trata muy probablemente de un motor paso a paso de tipo híbrido en configuración unipolar.

Para concluirlo, nos basamos en el ángulo de giro de 1.8º por paso, el cual es bastante pequeño, (características de los motores híbridos). Por otro lado, abrimos el motor, quitando sus cuatro tornillos inferiores, y pudimos ver el aspecto del rotor, que era como el de la siguiente imagen:

Como ya se ha explicado anteriormente, este tipo de rotor corresponde a los motores híbridos. Por último, lo caracterizamos como unipolar debido a sus 6 cables de salida, 4 para el bobinado y 2 comunes para cada bobina.

4.8.2. Control elegido para el motor

Optamos por la sencillez; decidiendo controlar este motor como si fuese un motor bipolar. Para ello, solo se deben despreciar los dos cables comunes y conectar los otros cuatro cables al controlador utilizado. En este caso, dispusimos de un controlador comercial de nombre Pololu A4988. Se trata de un controlador para motores paso a paso bipolares. El siguiente esquema resulta muy útil para conocer las entradas y salidas del controlador, así como el cableado necesario para su control.

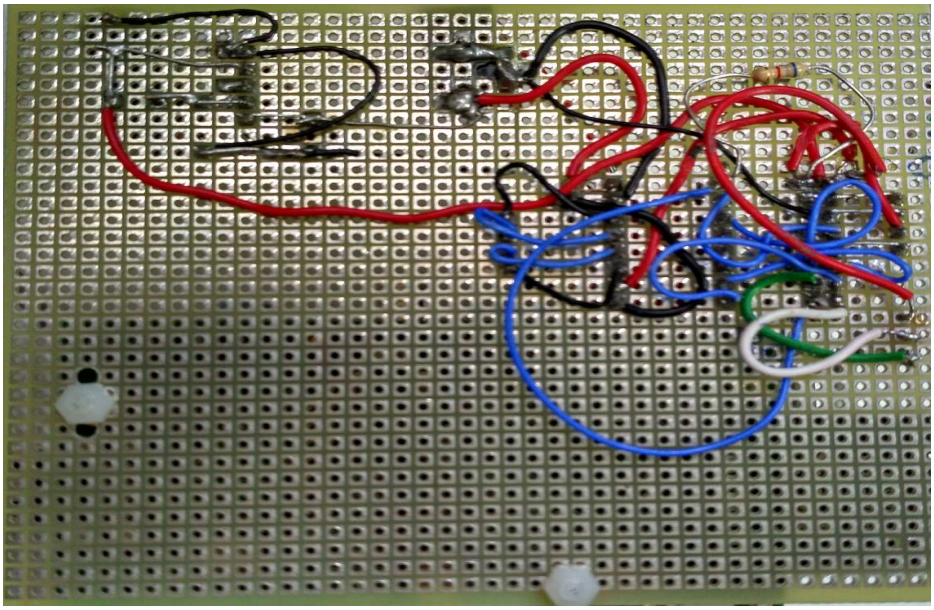


Como vemos en la imagen previa, este controlador debe ser alimentado con dos tensiones distintas. En este caso utilizamos los 12 voltios del conector de entrada y 3.3 voltios proporcionados mediante los GPIO de las Raspberry.

Además, la Raspberry y el circuito integrado que engloba el opto-interruptor deben ser alimentada con 5 voltios. De aquí surge la necesidad de utilizar el convertidor DC/DC para reducir la tensión.

Se creó una placa prototipo para establecer las conexiones eléctricas de los elementos, así como para crear el circuito necesario para el correcto funcionamiento del LM2575.

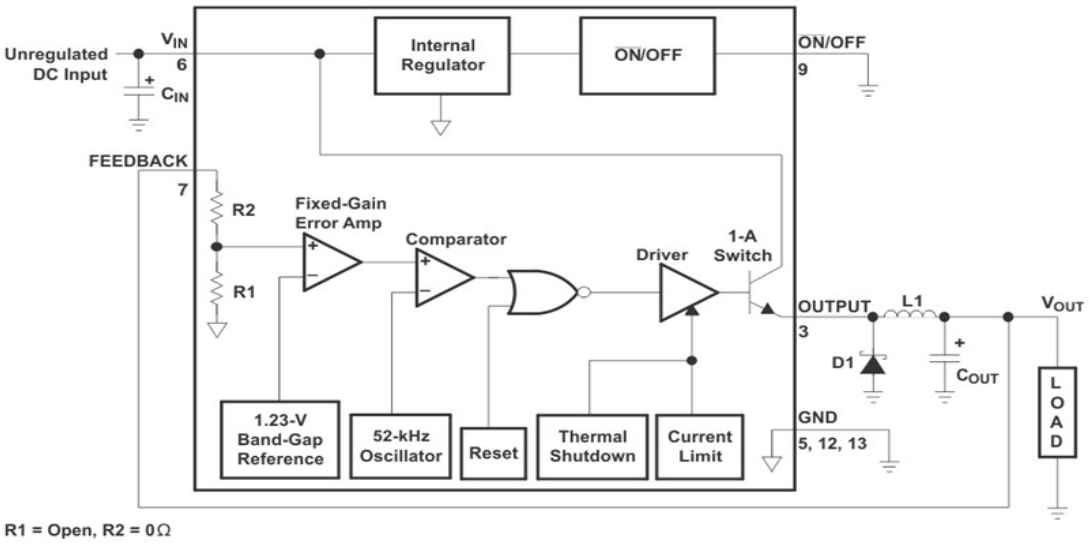
Las conexiones eléctricas, son un poco rudimentarias tal y como muestra la siguiente imagen, aunque su objetivo era comprobar que lo diseñado funcionaba correctamente.



Hasta este momento el trabajo se basaba en la conexión entre si de Raspberry, motor y su controlador, el opto-interrumputor y el telémetro.

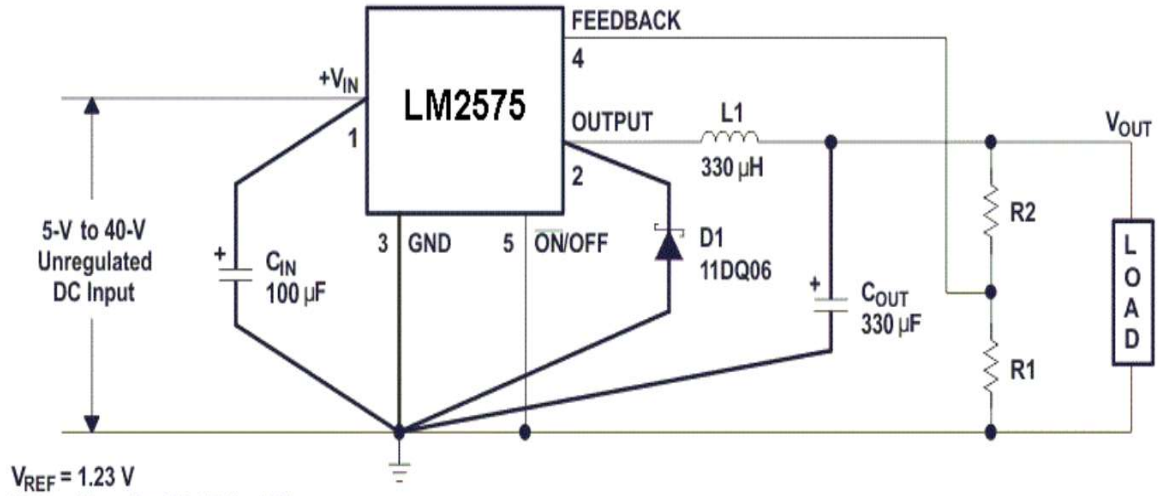
4.8.3. Convertidor DC/DC

El siguiente paso fue diseñar el circuito necesario para el LM2575. Para ello, se muestra en el datasheet como debe ser el circuito:

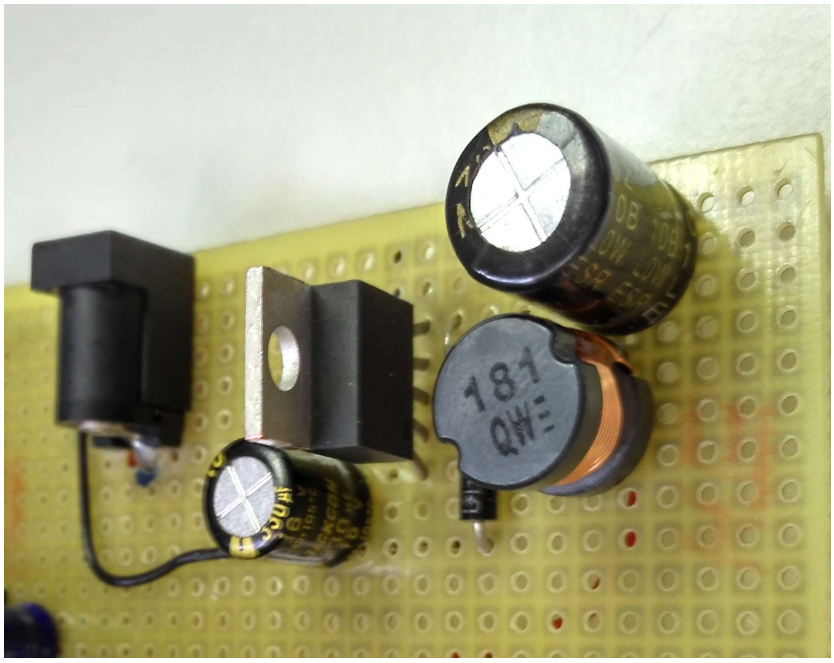


Como vemos, hay que proporcionar una tensión a a la entrada , que filtraremos con un condensador, y a su salida, se debe dotar de la estructura típica de los convertidores Buck que se ha explicado previamente.

Para facilitar más aún la tarea, en el datasheet se proporciona los valores necesarios para obtener una salida de 5 voltios, que es justo lo que se necesita.



Tras aplicar esto el resultado obtenido fue el siguiente:



He de destacar que yo no realicé este circuito ni los cálculos previos a la realización, pues ya se había encargado algún profesor de hacerlo. Sin embargo, mi tarea fue entender como funcionaba y porqué se había realizado de esa forma, para posteriormente realizar las mejoras que considerara oportunas. Pues, como se ha dicho, esta placa es un prototipo y no el diseño final.

4.8.4. Diseño PCB

Dado que el prototipo de la placa funcionaba correctamente y cumplía todos los aspectos que se requerían, la única mejora que cabía hacerle era diseñar una placa definitiva que fuese lo mas compacta posible. Para ello, debido a que cursé una optativa en la que aprendí a usar el paquete informático OrCAD, se ha realizado el diseño de una PCB usando dicho software de la siguiente manera:

El primer paso es realizar con OrCAD Capture el conexionado de todos los elementos. La finalidad con ello, ya que no podíamos hacer simulación del comportamiento del controlador del motor, de los GPIO de las Raspberry etc., era crear un netlist donde se estableciera las conexiones entre componentes. Para poder conectar los elementos entre sí, creé una nueva librería en la que añadí nuevos componentes con el número de entradas y salidas adecuado. Los componentes creados fueron: conector opto-interruptor, conector motor, conector de entrada, conector LIDAR-Lite, Pololu (controlador motor), Raspberry (simulación de los 16 GPIO que utilizamos) y regulador DC/DC. Creé un solo componente para el convertidor DC/DC. Éste englobará todo lo necesario para obtener los 5 voltios de salida. Para reducir lo máximo posible el tamaño de la PCB, sustituí el circuito del convertidor anteriormente nombrado por el de la siguiente imagen:

Este desempeñará la misma tarea, pero debido a su montaje con componentes SMD, las dimensiones son mucho más reducidas.

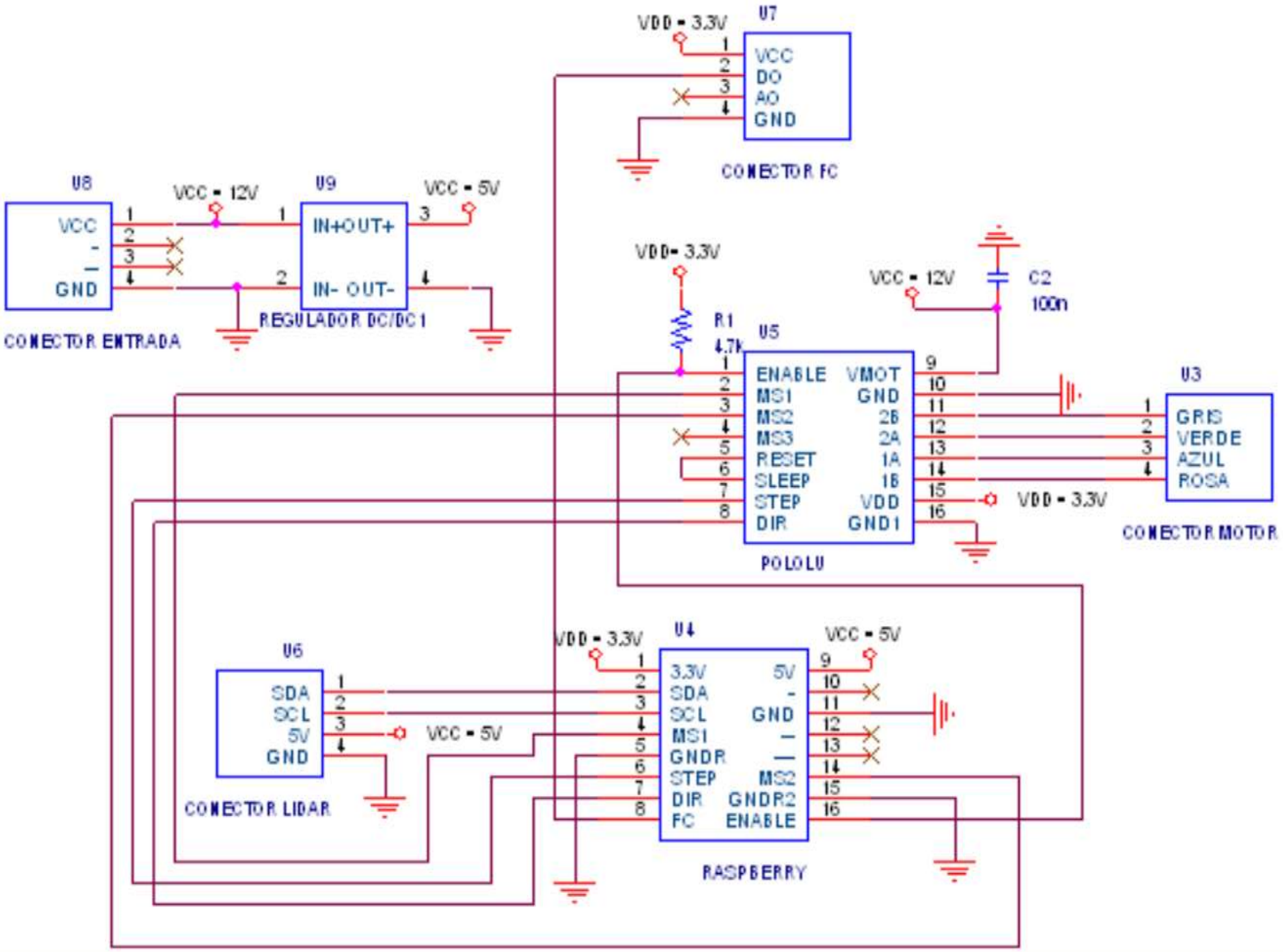
Una vez creado todos los componentes que se necesitan, se añade al circuito un condensador de 100 nanofaradios para filtrar los 12 voltios a la entrada del controlador del motor y una resistencia de 4700 ohmios, que actuará como resistencia Pull-Up y que permitirá activar o desactivar el controlador del motor. Las conexiones realizadas en todo el circuito son muy sencillas y se basan únicamente en la necesidad real de la unión de los componentes. Intentaremos explicar de forma general la conexión de los mismos:

- El conector de entrada se une al controlador del motor para darle 12 voltios y al convertidor DC/DC para poder obtener a su salida los 5 voltios.

- El regulador DC/DC reparte los 5 voltios para alimentar el conector del Lidar y el GPIO de la Raspberry.
- Controlador motor: el controlador del motor debe estar unido en su entrada a los GPIO de las Raspberry y a su salida al conector del motor.

- GPIO Raspberry: al ser la encargada de enviar las órdenes al controlador del motor, recibir la información del opto-interruptor y de comunicarse con el Lidar, deben ser estas las conexiones que se establezcan. Cabe mencionar que las Raspberry tras ser alimentada con 5 voltios, es capaz de proporcionar 3.3 voltios por otro de sus GPIO.

La siguiente imagen muestra de forma mas clara las conexiones realizadas:

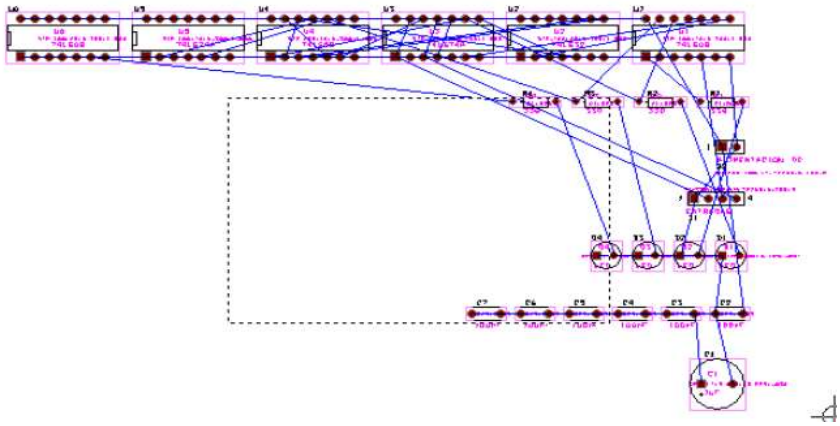

U8
VCC
-
—
GND
CONECTOR ENTRADA
VCC = 12V
U9
IN+ OUT+
IN- OUT-
REGULADOR DC/DC1
VCC = 5V
VDD = 3.3V
U7
VCC
DO
AO
GND
CONECTOR FC
VDD = 3.3V
R1
4.7k
U5
ENABLE
MS1
MS2
MS3
RESET
SLEEP
STEP
DIR
VMOT
GND
2B
2A
1A
1B
VDD
GND1
POLOLU
VCC = 12V
C2
100n
U3
GRIS
VERDE
AZUL
ROSA
CONECTOR MOTOR
VDD = 3.3V
U6
SDA
SCL
5V
GND
CONECTOR LIDAR
VCC = 5V
VDD = 3.3V
U4
3.3V
SDA
SCL
MS1
GNDR
STEP
DIR
FC
5V
GND
MS2
GNDR2
ENABLE
RASPBERRY
VCC = 5V

Tras aseguranos de que las conexiones están bien hechas, ya que no se puede simular nada y el software no informará de posibles errores, se obtiene el fichero .mnl.

Ahora se pasa al segundo software utilizado: Orcad Layout. En este programa se diseña el trazado de pistas que se imprimirán en la PCB. Para ello necesita el fichero .mnl. Éste describe cada una de las conexiones de nuestro circuito y los componentes que lo forman.

Otra información importante que necesita el Layout son los footprints asociados a cada uno de los componentes previamente creados en Capture. Por tanto, al ser componentes que no están en la librería de footprints de layout, tuve que crearlos mediante la herramienta correspondiente. Para ello, tomé las medidas de cada componente con un calibrador y además, comprobamos que todos los componentes tenían un paso estándar de 2,54 milímetros. Con estos datos se diseño los footprints de cada componente para así poder continuar con el software layout.

Tras la correcta asignación de footprints, la primera pantalla que aparece tiene la siguiente forma:



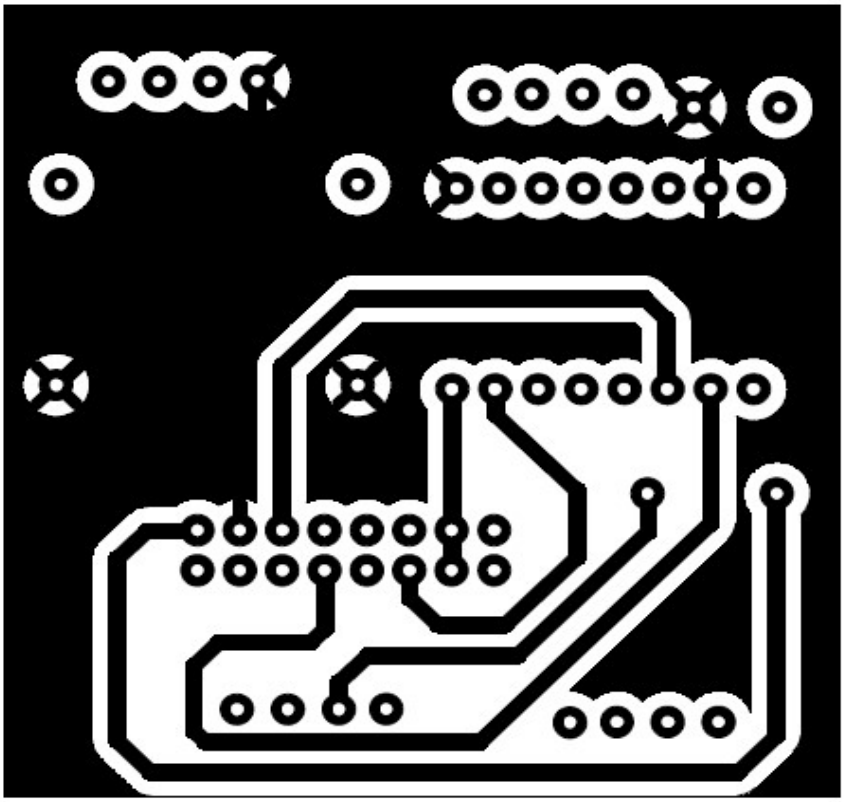
Como se ve en el ejemplo, se obtiene una imagen del footprint de cada componente y de las conexiones que se han definido en el OrCAD Capture. El primer paso es establecer la configuración inicial que implica fijar las unidades de medida en milímetros, crear el borde exterior de la placa, definir el espaciado que pretermitirá entre (pistas y pads) las capas en las que se posibilitará el routeo, el ancho de pistas mínimo, típico y máximo etc.

Desde el principio el objetivo era que el tamaño de la placa no excediera los 5x5 centímetros con la finalidad de poder introducir dicha placa en la estructura final del telémetro. Por consiguiente, se diseñó una PCB donde las pistas resultaron en su mayoría de 0.5 milímetros de grosor para poder condensarlo todo en una cara.

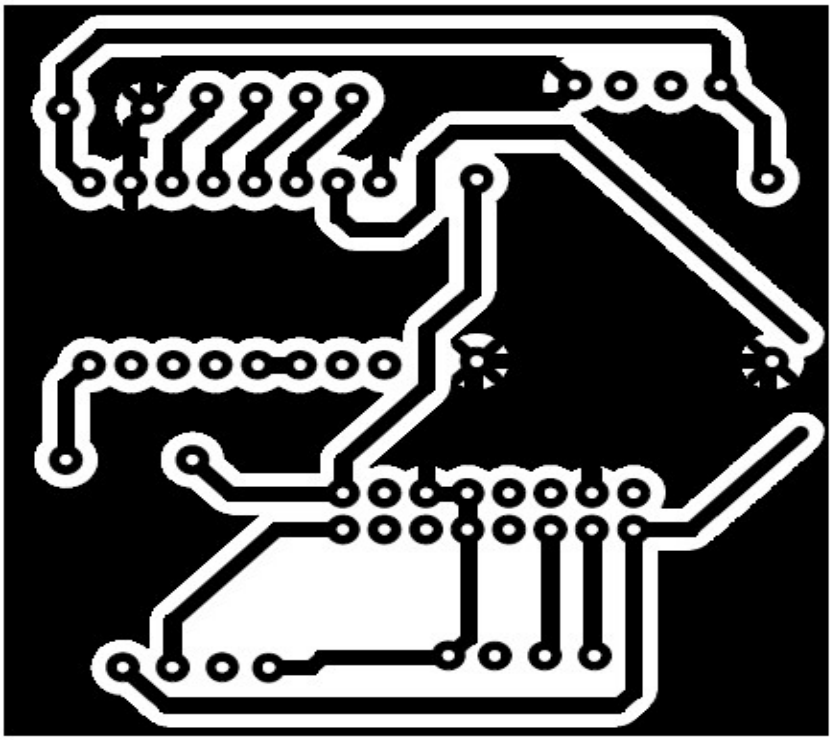
Sin embargo, debido a la dificultad de obtener una PCB mediante fabricación química con su mayoría de pistas a 0.5 mm, decidímos comenzar un nuevo diseño de una PCB a dos caras. Trazar pistas tanto en la cara top como en la cara botom permitió obtener un diseño mucho más limpio con todas las pistas a un grosor de 1 mm y todos los pads a 2 mm pensando en facilitar la soldadura.

La siguiente imagen muestra los fotolitos de las dos caras necesarias para su fabricación:

TOP LAYER



BOTOM LAYER



4.8.5. Fabricación PCB

Una vez obtenidos los fotolitos, podemos dar por finalizada la etapa de diseño de la PCB y pasamos a su fabricación. Previamente, en la optativa de Circuitos Impresos, había tenido la oportunidad de realizar todo el proceso de fabricación de una PCB. Este proceso lo podemos dividir en los siguientes pasos:

- Obtención del fotolito: una vez se obtienen los fotolitos mediante el software, se deben imprimir. Para esto necesitamos papel transparente y una impresora láser.

- Insolado de la placa: se realiza utilizando una insoladora de luz ultravioleta y colocando el fotolito sobre la placa virgen. Este proceso se encargará de debilitar las zonas donde se requiere eliminar el cobre. Es decir, las partes que no están en negro en el fotolito.

- Revelado de la PCB: en este paso se introduce la placa en una disolución de sosa caústica y agua. El objetivo es eliminar la película fotosensible que ha sido debilitada previamente en el insolado.

- Atacado de la PCB: introduciremos la placa en una solución ácida que destruirá el cobre en las zonas que no estén protegidas por la película fotosensible.

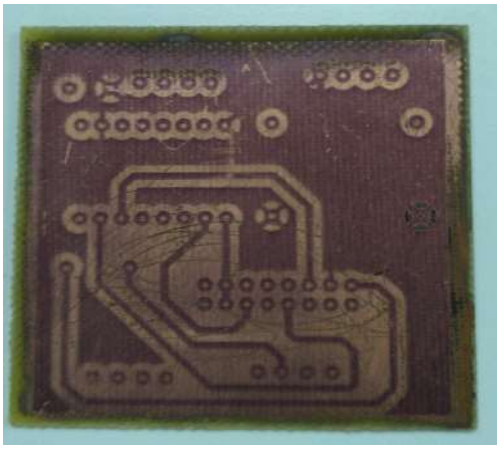
- Taladrado y montaje de componentes: es el paso final de la fabricación de la PCB. En él debemos taladrar los pads y soldar los componentes a través de los mismos.

Tras esta breve explicación de los pasos necesarios para obtener la PCB, debemos decir que a pesar de realizar reiterados intentos y de seguir estrictamente los pasos descritos, incluso con ayuda del técnico del laboratorio, no fuimos capaces de obtener una PCB viable.

Evidentemente, mi falta de experiencia, (ya que solo había hecho una durante el transcurso de la optativa), es uno de los factores determinantes para no conseguirlo. Sin embargo,en mi opinión, el punto que no se realizaba correctamente, era el revelado de la placa. A pesar de seguir las instrucciones del producto químico comprado para el revelado, el cual indicaba disolver en 2,5 litros de agua, en las primeras ocasiones que seguimos esto al pie de la letra obtuvimos resultados muy lejos de lo deseado. El producto destruía completamente la película fotosensible en esta concentración, no solo las partes debilitadas en la insoladora. Es por esto que, siguiendo un método de prueba y error, rebajamos dicha concentración con más agua intentando conseguir el punto adecuado sin ningún éxito.

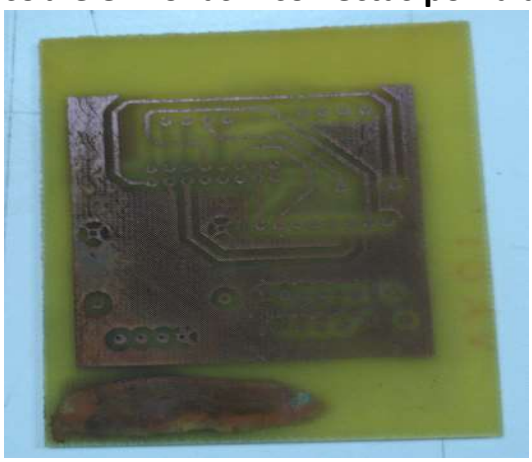
Hemos de destacar, que achacar esto al paso del revelado es una valoración personal, ya que entre tantos intentos, obtuvimos resultados muy diversos y probablemente no fuera el único fallo, concluyendo pues, que esto haya sido debido fundamentalmente a la falta de experiencia. Añadimos las siguientes imágenes en las que se puede observar tres intentos fallidos de la producción de la PCB.

Posible acción insuficiente de la sosa



Acción excesiva de la sosa

Destrucción del cobre en zonas incorrectas por la acción del ácido.



Para finalizar este capítulo, destacar el trabajo de diseño realizado, mediante el cual obtuve los fotolitos necesarios. Con ellos, creo firmemente que una persona con experiencia podría fabricar la placa. Además el software OrCAD Layout también permite obtener los ficheros Gerbers para poder producir la PCB mediante un fabricación en seco. Lamentablemente,el laboratorio no dispone de una fresadora por control numérico para fabricarla siguiendo este método.

5. Diseño Estructura 3D

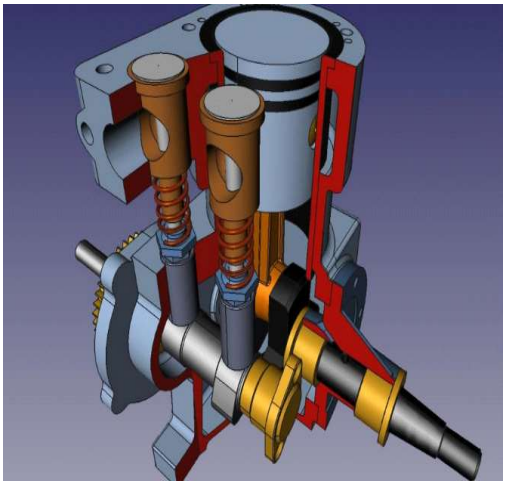
Este capítulo estará dedicado al trabajo realizado en lo referente al diseño y fabricación de las piezas necesarias para la unión del sistema. Para ellos comenzaremos explicando brevemente las herramientas utilizadas. Posteriormente describiremos las tareas llevadas a cabo en este campo.

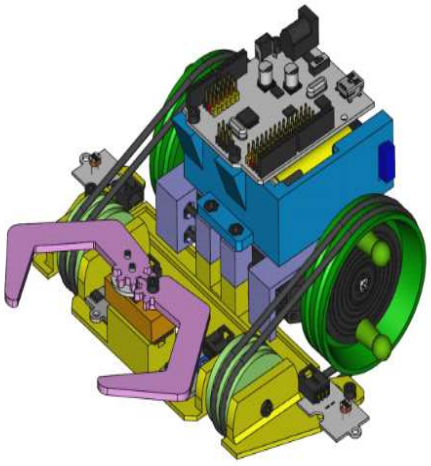
5.1. FreeCAD

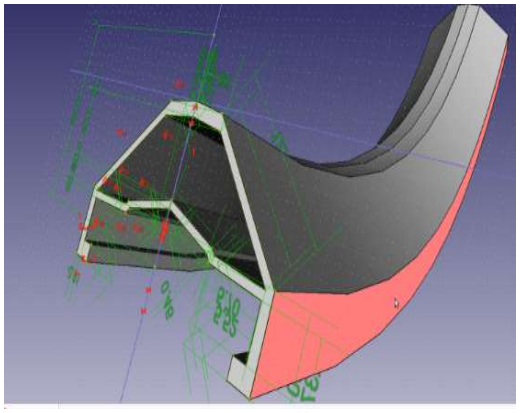
FreeCAD es una aplicación libre de diseño asistido por computador. Su funcionamiento se basa en un modelador 3D paramétrico que permite modificar fácilmente el diseño regresando dentro del historial del modelo y cambiando sus parámetros. Este software esta basado en Open Cascade y programado en C++ y Python. Sigue una filosofía basada en el desarrollo de la comunidad, es por esto que es multiplataforma (Windows, Mac y Linux), altamente personalizable, programable mediante scripts y soporta extensiones.

Este software esta dirigido a un amplio abanico de usuarios: a aquellos que no tienen ninguna experiencia previa en CAD, ya que debido a la amplia comunidad que posee, hay gran cantidad de tutoriales. Usuarios CAD avanzados, pues este software posee diferentes bancos de trabajo que son muy semejantes a las alternativas comerciales más populares. Los programadores encontrarán en este software una opción con la que se sentirán muy cómodos ya que casi la totalidad de funciones de FreeCAD son accesibles a través de Python. Aparte de eso,encuentra en la educación un gran interés debido a que permite enseñar a los estudiantes un programa CAD sin tener que preocuparse por la compra de licencias.

Para terminar, cabe remarcar que este software es plenamente capaz de competir con las alternativas comerciales más importantes. Para demostrar esto, podemos ver algunos ejemplos que se han diseñado con este programa. [3]







5.2. BQ Prusa i3 Hephestos

En la actualidad hay más de un tipo de impresora 3D, cada uno de ellos se basa en un principio de funcionamiento distinto. Podemos destacar los 4 tipos mas usados en la actualidad.[2]

- Basadas en estereolitografia (SLA): Es la mas antigua de todas y se basa en la aplicación de una luz ultravioleta a una resina líquida contenida en un recipiente sensible a la luz. La resina se va solidificando debido a la acción de la luz UV.
- Basadas en sinterización selectiva por Láser (SLS): comparte similitudes con la SLC, pero en este caso permite la utilización de materiales en polvo. Su principio de funcionamiento se basa en el impacto del Laser con el polvo del material, el cual funde dicho material y se solidifica con la forma deseada.
- Basadas en inyección: es el sistema que mas similitudes comparte con la impresoras de folios convencionales. En lugar de inyectar gotas de tinta en el papel, se inyectan capas de polímero líquido que se pueden curar en la bandeja de construcción.
- Basadas en deposición de material fundido: consiste en depositar polímero fundido sobre una base plana. El material que inicialmente se encuentra en estado sólido almacenado en rollos se funde y es expulsado por una boquilla en minúsculos hilos, los cuales se solidifican conforme van tomando la forma de cada capa. Este tipo de tecnología es la que ha permitido un abaratamiento de las impresoras 3D, permitiendo así, su alcance a usuarios domésticos y no solo a la industria.

En este proyecto se utilizado una impresora basada en deposición de material fundido. Concretamente, el modelo Prusa i3 Hephestos, fabricado por la empresa española BQ, ha sido la impresora utilizada para la fabricación de las piezas previamente diseñadas usando FreeCAD. Se presentó como una evolución de la Prusa i3. Esta es una de las impresoras 3D más populares de la comunidad RepRap.

El proyecto RepRap es una iniciativa con el propósito de crear una máquina de prototipado rápido, bajo la licencia GNU GPL, que sea capaz de replicarse a si misma. Una máquina de este tipo puede fabricar objetos físicos a partir de modelos generados por ordenador. De la misma manera que la impresora de un ordenador permite imprimir imágenes en dos 2D en papel, las impresoras 3D imprimen objetos a base de plástico .

Esta impresora es un claro ejemplo de la llegada de las impresoras 3D al gran público y del abaratamiento de estas máquinas en los últimos años. Se vende en forma de kit que incluye todas las piezas necesarias además de los pasos para su montaje explicados de forma fácil y eficaz.

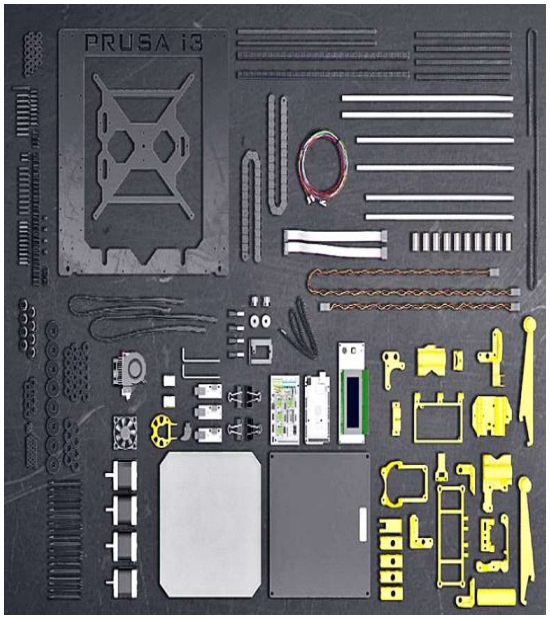
Una de las características más importantes de este modelo es la mayor protección frente a quemaduras que ofrece, pues elimina la cama caliente e incluye un protector sobre el extrusor para aumentar la seguridad del mismo. Además, eliminando la cama caliente se consigue que el consumo de la misma se vea notablemente reducido.

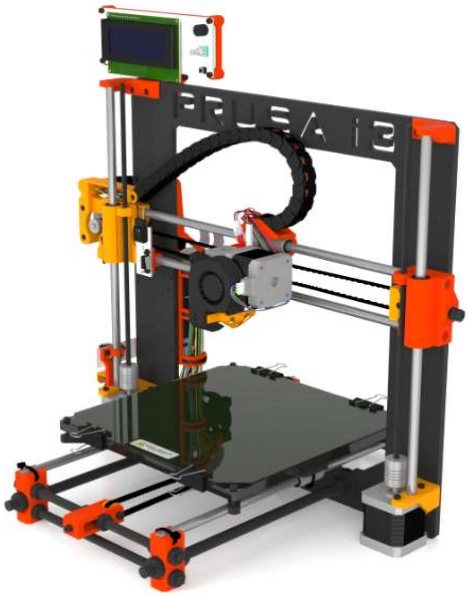
Este modelo viene diseñado para trabajar con PLA, uno de los materiales más usados en ambientes domésticos por no emitir fuertes olores como el ABS y por tener unos resultados mejores en menor tiempo. A pesar de ello, se puede modificar de una manera no excesivamente compleja para trabajar ABS y otros plásticos si así lo deseamos.

Otra característica destacable es que está realizada con hardware 100% libre, por lo que permite ser modificada y adaptada de una manera sencilla. Tanto es así que BQ libera todas las mejoras que realiza con la comunidad.

Para acabar,es importante mencionar sus especificaciones técnicas y una imagen de las mismas:

- Velocidad media de impresión: 40-60 mm/s.
- Resolución media de 200 micras.
- Impresora con bobina PLA(X,Y,Z): 460x383x580 mm.
- Volumen de impresión: 215x210x180 mm.
- Materiales soportados sin modificaciones: PLA, HIP y FilaFlex.
- Peso de la impresora 9 Kg.





5.3. Diseños necesarios.

Antes de explicar los diferentes diseños que realizamos, es imprescindible definir las piezas básicas necesarias para el ensamblaje de todo el sistema con el fin de facilitar su comprensión.

- Unión entre Lidar y motor: como ya se ha explicado, el Lidar irá unido al eje del motor paso a paso para que mientras se controla el giro del motor, se pueda controlar también la dirección en la que el telémetro realiza una medida.

- Soporte para que el opto-interruptor esté en la posición correcta para que, al girar el telémetro, éste interrumpa el haz de luz y así poder establecer un sistema de coordenadas.

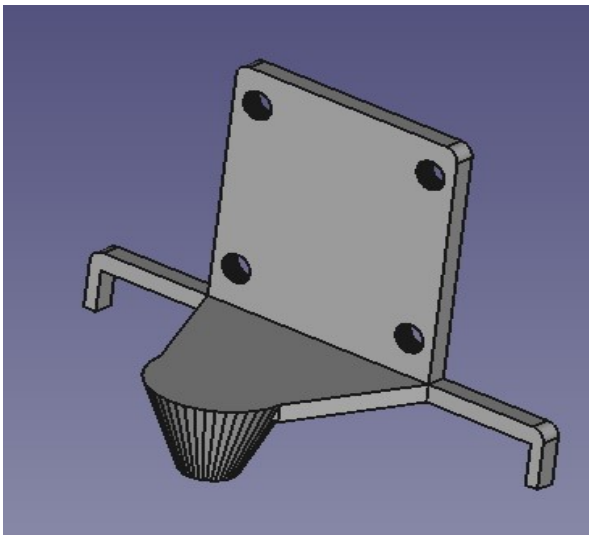
- Estructura que englobe, proteja todo el sistema y facilite su anclaje.

5.4. Trabajo realizado

Relataremos las soluciones iniciales con las que partimos y como modificamos estas propuestas para intentar obtener un diseño final óptimo.

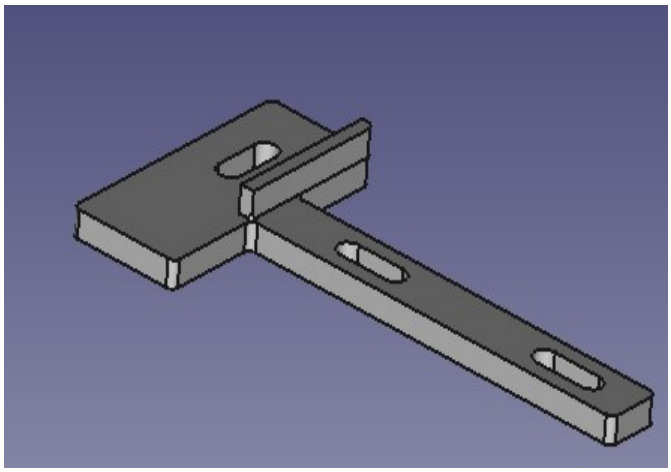
5.4.1. Prototipo inicial

Al comenzar esta etapa del proyecto lo primero fue aprender a utilizar el FreeCAD. Tras la curva de aprendizaje, resultó bastante intuitivo realizar las ideas que se me iban ocurriendo con este software, y posteriormente fabricarlas mediante la impresora 3D. El primer diseño con el que trabajé, ya que ya estaba realizado cuando comenzamos con el TFG, es el siguiente:

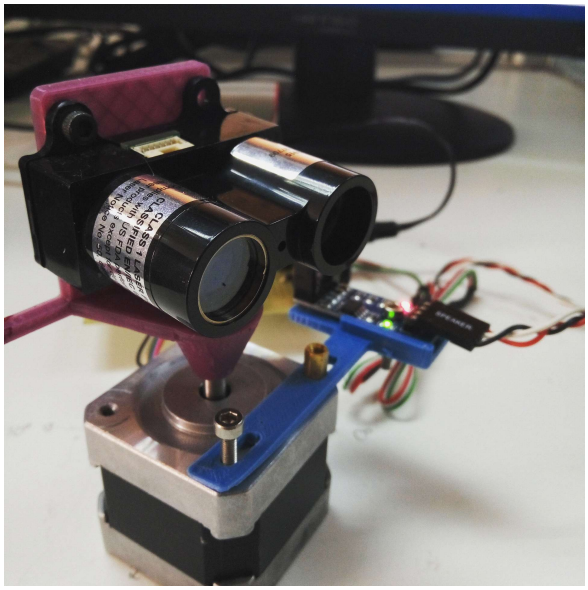


Su primera función es la unión del Lidar al motor mediante la base cónica que podemos apreciar en la imagen. Esa función la cumple perfectamente, no se puede mejorar nada más. La segunda es la de interactuar con el opto-interruptor. Es esta la funcionalidad de las dos patas laterales que se observan.

Junto con este soporte inicial nos encontramos con un soporte para el sensor ya diseñado y fabricado. Se muestra en la siguiente imagen:



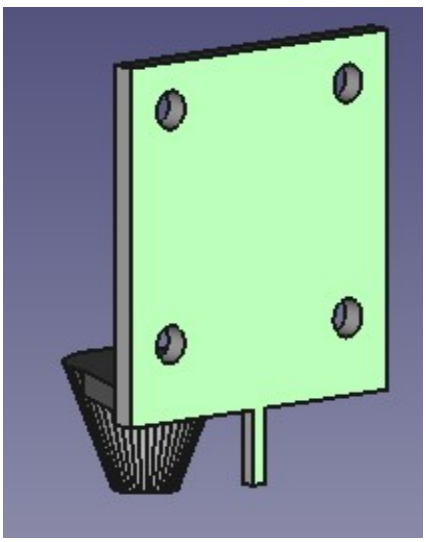
Su diseño era simple y eficaz; coloca el opto-interrupor en el lugar indicado para que las patas del soporte del telémetro interactuaran debidamente con el sensor. Estas dos piezas se ensamblaron y permitieron un prototipo funcional del sistema. A continuación se aprecia una fotografía de esta fase inicial.

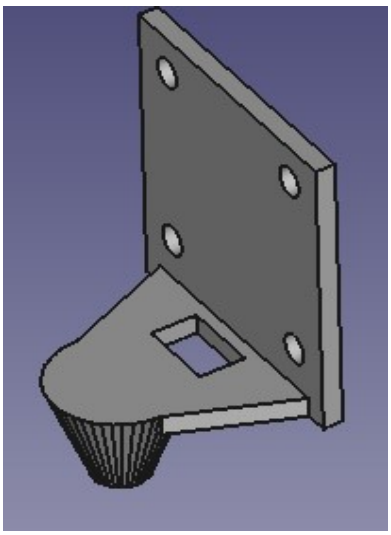


5.4.2. Mejoras realizadas en prototipo

Basándonos en el correcto funcionamiento de los diseños realizados hasta ese momento, nos propusimos mejorarlo añadiendo y quitando características para solucionar problemas que se hacían notar en el día a día.

La idea de contar con dos patas que interrumpieran el haz de luz del opto-interruptor tenía el fin de controlar mas fácilmente (ya que no tendríamos que esperar a realizar una vuelta completa) el error con respecto a los grados girados según el controlador y la realidad. Tras realizar varias pruebas con algoritmos de giro del motor que permitieran identificar cual de las dos patas era la detectada por el sensor, decidimos que era mejor optar por un diseño con una sola pata. Para tomar esta decisión, nos basamos en que, tras realizar varias pruebas, observamos que el error era despreciable debido a la precisión de los motores paso a paso híbridos y al control tan preciso del Pololu, el cual permite realizar giros de tan solo 0.0125 grados por paso.
Es por esto que el siguiente diseño para el soporte del Lidar fue el siguiente:



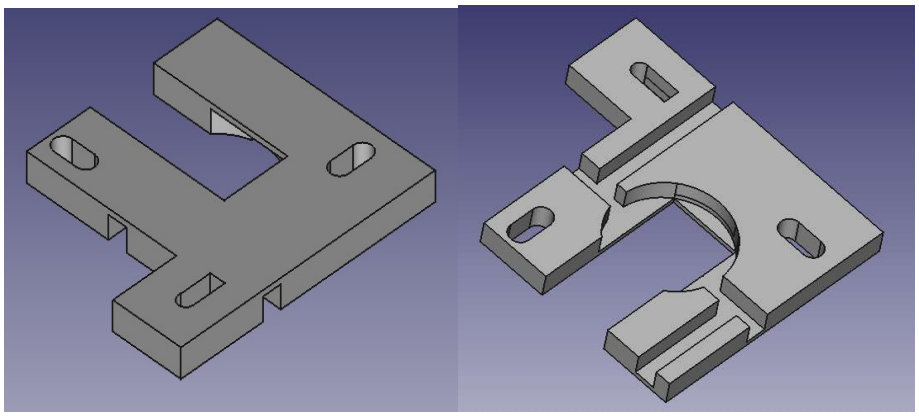


En este diseño las mejoras introducidas fueron dos:

- Colocar una sola pata para interaccionar con el opto-interruptor: ahora está en el centro del soporte y no saliendo por uno de los laterales. Esto permite reducir bastante las dimensiones del sistema, ya que anteriormente el radio del sistema venía determinado por la longitud de la misma. Ahora el radio de giro solo viene determinado por el tamaño del telémetro, pues el soporte se diseñó de manera que el giro del Lidar se desarrollara con respecto a su centro.

- Abertura para el paso del cable del Lidar: inicialmente se pensaba sacar este cable por la parte superior del soporte e inventar algún diseño que lo mantuviese alejado del giro del motor para evitar enredos. Tras realizar numerosas pruebas para determinar la forma en la que este cable limitara lo menos posible el giro del motor, observamos que trenzando los cables entre si no se veían excesivamente influenciados por el giro del motor. La tensión producida al trenzar los cables entre si hacia al conjunto resultó ser mucho mas rígida y los cables tendían a no entorpecer tanto el giro del motor.

Debido al cambio introducido en el soporte del Lidar diseñamos un nuevo soporte para el opto-interruptor:



Las necesidades que vino a cubrir este nuevo diseño fueron:

- La parte superior (imagen izquierda): esta diseñada para la colocación del sensor de forma que coincida con el radio de giro de la pata central del soporte del Lidar. Esta posición fue diseñada junto con la pata del soporte del Lidar para reducir las dimensiones del sistema lo máximo posible. Además, se consiguió que el sensor se colocara en una posición en la que si la pata interaccionaba con el sensor significaría que estábamos posicionados con el telémetro tomando una medida hacia la dirección frontal de todo el sistema. Esto facilita un poco el algoritmo diseñado posteriormente.

- La parte inferior, que inicialmente era totalmente lisa y sin utilidad, se diseñó con el objetivo de ayudar al trenzado de los cables a no interferir con el giro del motor. Los túneles que se observan cumplen el objetivo de guiar los cables y limitar su movimiento. Con esto estaba casi resuelto el enredo que se producía al girar. Para solucionarlo definitivamente, añadímos la circunferencia que se observa en la imagen. Esta actúa de techo para los cables, los cuales no encuentran otro camino que el salir muy pegados al eje del motor, prácticamente vertical a la abertura del soporte del telémetro.

Con todo lo hasta ahora nombrado, se consiguió que el telémetro pudiera realizar giros de 0 a 360 grados sin importar que deba tener un cable conectado.

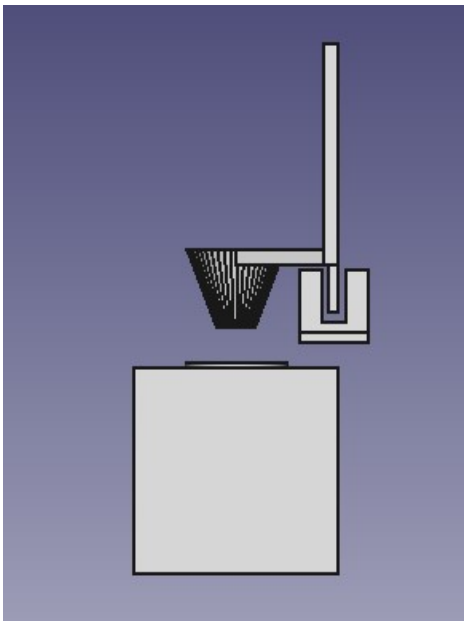
5.4.3. Diseño final

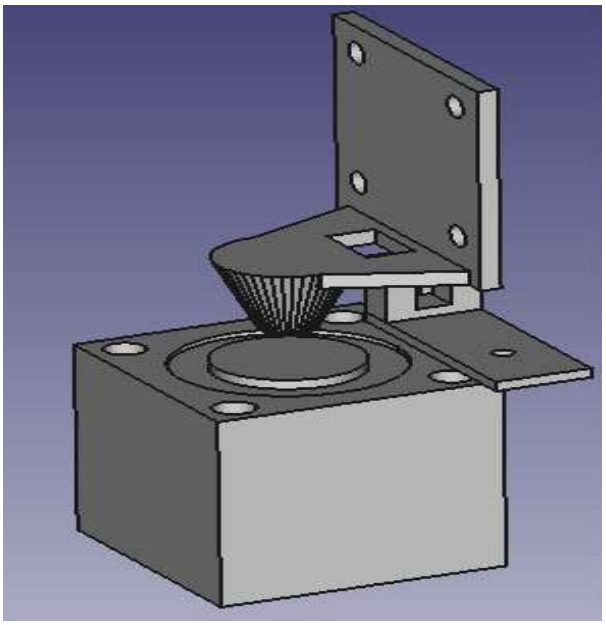
Una vez comprobado el buen funcionamiento de lo diseñado hasta ahora y de no tener mas sugerencias de mejora, llegó el momento de englobar este sistema en una 'caja' que sirviera para protegerlo y que no perjudicara las medidas que debe hacer el telémetro. Para enfrentarnos a este diseño, debíamos tener en cuenta dos premisas: que fuera lo más compacta posible y que pudiese ser fabricable mediante la impresora con la que contábamos. Como ya se ha mencionado, lo compacta que puede ser la caja viene limitado por el radio de giro que describe el Lidar al girar. Tras realizar unos sencillos cálculos en los que se supuso al Lidar como un cuadrado de 50x50 mm (para ser conservadores ya que es un poco menos), se concluyó que un cuadrado de 50 mm de lado que gire con respecto a su centro queda circunscrito en el interior de una circunferencia de diámetro 70 mm. Por lo tanto, este sería al espacio interior mínimo que debe tener nuestro diseño,añadiendo el grosor necesario para las paredes.

Una vez establecidas las bases de medidas mínimas necesarias, nos encontramos con la limitación de las impresoras 3D; no se puede llevar a cabo 'huecos' que unan una misma pieza en el eje vertical. Estos 'huecos' eran algo necesario ya que nuestro recipiente debe tener una gran abertura entre lo que lo proteja inferiormente y lo que lo protege superiormente que permita al telémetro tomar medidas hacia el exterior. Es por esto que se optó por crear un diseño modular; el recipiente estaría basado en 3 piezas que al ensamblarse protegerían en su totalidad el sistema. Esto aportaba una ventaja, la posibilidad de acceder mas fácilmente a los componentes del sistema.

Se ha intentado reducir al mínimo el número de prototipos necesarios para obtener un diseño final. Debido a esto, se simuló con FreeCAD todos los componentes que intervienen y así comprobar que todo se adaptaba correctamente a lo deseado.

Las siguientes imágenes que explican mucho mejor el desarrollo realizado.

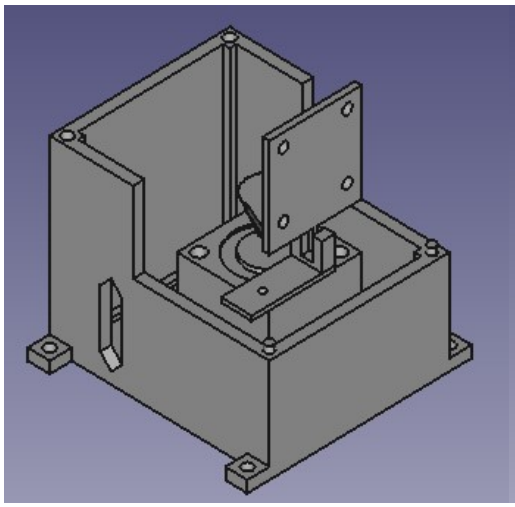




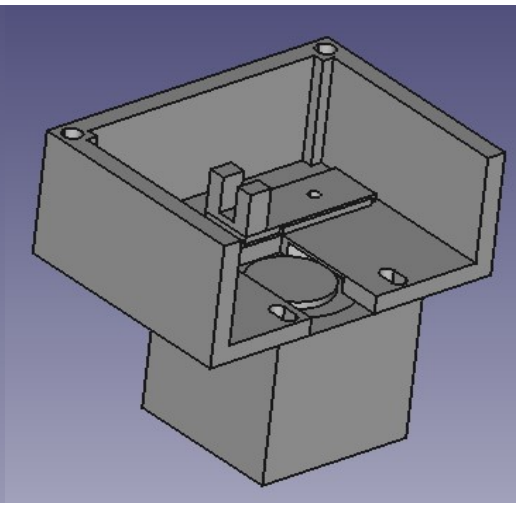
En estas dos imágenes se aprecian los tres elementos que irán dentro de la caja: motor, soporte del Lidar y opto-interruptor, el cual aparece sin su soporte diseñado, ya que para facilitar todo el sistema, se aprovechó el módulo central de la caja con el objetivo de realizar a su vez la tarea de soporte para este sensor.

A continuación se muestran las imágenes del diseño final primero para que los comentarios sean más fáciles de comprender posteriormente

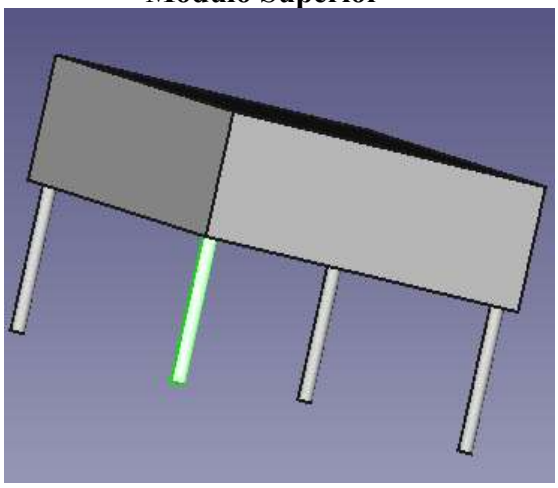
Módulo Inferior



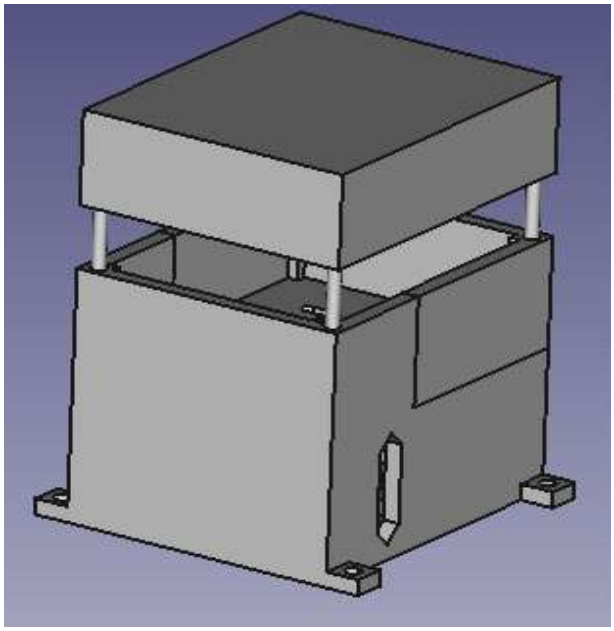
Módulo Central



Módulo Superior



Los 3 módulos ensamblados



Para acabar este capítulo solo queda mencionar algunas características del diseño final:

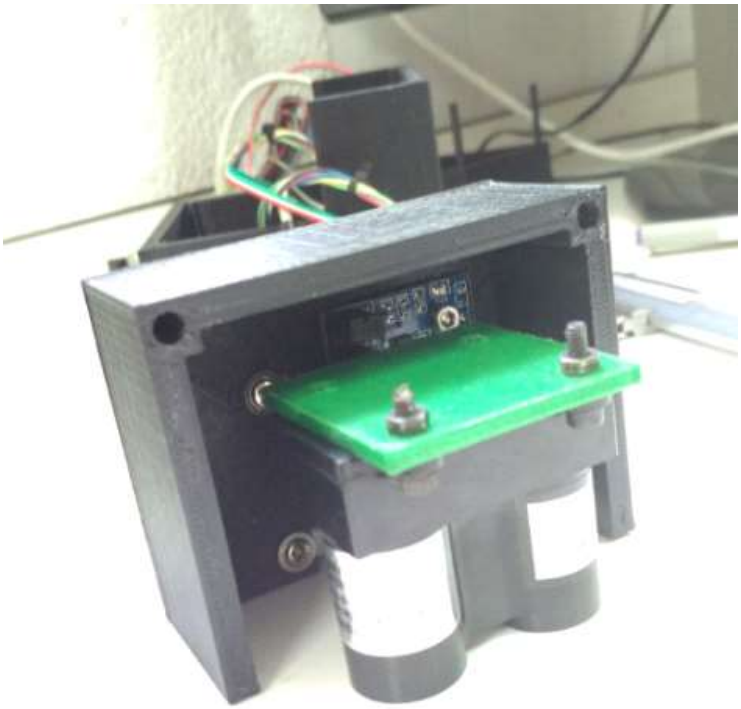
- El recipiente fabricado: este necesita de la sustitución del conector de fábrica del opto-interruptor que por defecto lo tiene sobresaliendo hacia un lado para colocar un conector en su parte inferior con la finalidad de no ampliar el ancho de la caja.
- El módulo central no es un cuadrado cerrado en todos sus lados: pues con esto se puede extraer dicho módulo sin la necesidad de tener que separar el soporte del Lidar del motor. Retirando los tornillos de fijación al motor y deslizando hacia el lado opuesto de la abertura se puede retirar.

- Las medidas exteriores y por tanto de todo el sistema son : 100 mm largo, 80 mm de ancho y 130 mm de alto. Se podría conseguir un recipiente cuadrado de 80 mm de lado, pero decidimos ampliar esos 20 mm para que la PCB diseñada, tras su fabricación, pueda incorporarse dentro. De esta forma, todo lo necesario para el funcionamiento queda en el interior. Por consiguiente, solo saldría al exterior el cable de alimentación junto con el cable de control que se necesita para las Raspberry, ahorrando así el tener que sacar el cable del motor, el cable del Lidar y el cable del opto-interruptor y de tener que buscar un lugar donde situar la PCB.

- Por último, debemos mencionar que las columnas de unión entre el módulo central y el superior son independientes a ambos módulos. Los módulos inferior y superior poseen unos agujeros para encajar dichas columnas. Esto es así para facilitar su sustitución en caso de rotura.

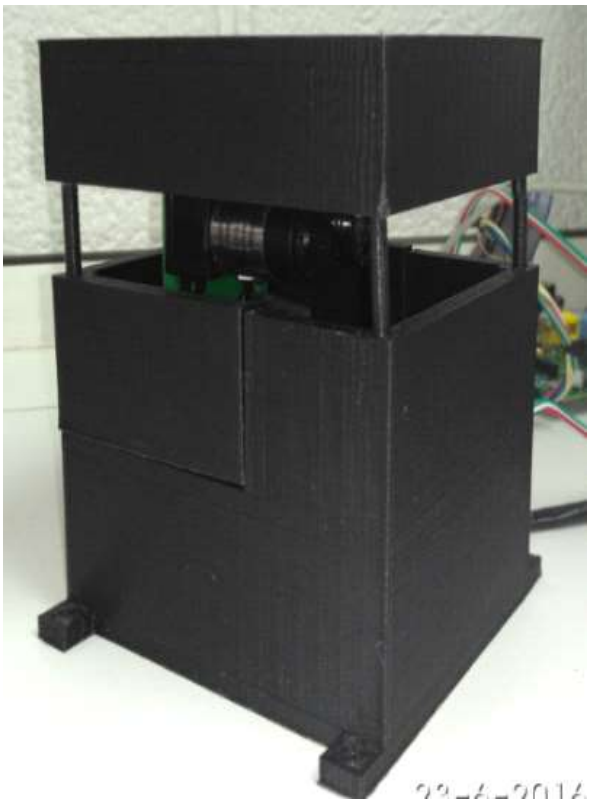
Cerramos este capítulo con unas imágenes reales del resultado final de esta parte del proyecto.

Coincidencia del cero establecido por el opto-interruptor con la orientación completamente frontal de la medida tomada por el telémetro:



Módulo central, el cual no solo realiza de la tarea de protección si no que sitúa el opto-interruptor y a su vez sigue cumpliendo las tareas descritas para que el cable no moleste durante el giro:

Diseño final listo para su funcionamiento:



6. Desarrollo Software

Hablaremos en este capítulo del desarrollo software que se ha realizado. Tanto de los códigos creados en el transcurso de este trabajo, como de códigos o herramientas que se han tomado de la comunidad. Para ellos inicialmente se hace una introducción al lenguaje de programación utilizado y al sistema operativo ROS, el cual resulta de una gran utilidad para procesar los los resultados obtenidos.

6.1. Python

Python es un lenguaje de programación interpretado. Este tipo de lenguajes no requieren de un código a ser compilado. Por lo general, los lenguajes interpretados suelen ser de alto nivel y se caracterizan por expresar el algoritmo diseñado de una manera que facilite su entendimiento por los humanos en lugar de facilitar su ejecución en la máquina. Este lenguaje, en especial, sigue una filosofía en la que intenta mantener una sintaxis que favorezca un código limpio y fácilmente legible. Python proporciona una serie de características que conviene nombrar brevemente:

Es un lenguaje de programación multiparadigma. Esto significa que permite varios estilos de programación: programación orientada a objetos, programación imperativa y programación funcional. Esto se considera una gran ventaja ya que no obliga al programador a adoptar un estilo particular de programación, como sí pasa con otros lenguajes.

Otra característica de Python es que utiliza tipado dinámico, esto quiere decir que no es necesario declarar el tipo de dato que contendrá nuestra variable. Dicha variable se adaptará al tipo que le asignemos en la ejecución del programa. A su vez esta denominado como fuertemente tipado, lo que significa que no permite violaciones en los tipos de datos. Es decir, no permitirá el uso de una variable de un tipo como si fuese de otro.

Además, posee una gran cantidad de librerías que permiten comunicarse con hardware de diferentes tipos, por ejemplo: *I2C*, el cual hemos utilizado en este trabajo, *serial*, *modbus* y otros tipos de comunicación. Para ello, en algunos casos, se aprovecha de interfaces de acceso a librerías de otros lenguajes.

Por último, debemos nombrar que es multiplataforma, es decir, lo podemos encontrar en distintos sistemas operativos como por ejemplo: Windows, Linux, Mac OS, Android etc. [4]

6.2. Git y Github

Git es un software de control de versiones. Su objetivo es la eficiencia y la confiabilidad del mantenimiento de versiones en aplicaciones con un gran número de archivos de código fuente. Este se basa en una serie de comandos *git add*, *git pull*, *git push*, *git commit* etc. Estos comandos son los básicos, ya que posee mas, con los cuales podemos añadir un nuevo fichero al que realizar control de versiones, subirlo o descargarlo del repositorio etc.

Github es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git, considerado como uno de los sistema de control de versiones más populares entre los desarrolladores. En esta plataforma el código se almacena de forma pública, aunque existe la posibilidad de hacerlo de forma privada. Es el servicio elegido por proyectos de software libre tan importantes como reddit, Ruby on Rails, node.js. Además, empresas de gran prestigio como facebook, Xiaomi y otras muchas más alojan aquí sus desarrollos públicos tales como SDK, librerías, etc.

Hoy en día Github es mucho más que un servicio de alojamiento de códigos; ofrece varias herramientas extras que facilitan el trabajo en equipo:

• Un sistema de seguimiento de problemas que permite a los miembros de un equipo (o cualquier usuario si su repositorio es público), abrir un ticket detallando un problema que tenga el software o sugerencias que se deseen hacer sobre al mismo.
• Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

Además, Github dispone de funcionalidades para hacer fork y solicitar pulls. Si por ejemplo, a un usuario se le ha ocurrido una funcionalidad, mejora o la corrección de un bug de un software que no es suyo, pero que está alojado en GitHub, se puede clonar el repositorio ajeno (hacer un fork) para que se copie a su cuenta, y así, efectuar en esta copia los cambios que se necesiten. Finalmente, se emite una solicitud de pull al dueño del repositorio original que podrá analizar fácilmente los cambios que se han realizado. Si este considera interesante la contribución del usuario, hará un merge de los mismos con el repositorio original sin que se pierda la autoría de sus commits.

6.3. Robot Operative System (ROS)

Es un framework para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo. ROS comenzó su desarrollo en 2007 por el laboratorio de inteligencia artificial de Stanford. A partir de 2008, su desarrollo continúa bajo el amparo de Willow Garage, un instituto de investigación robótica con más de veinte instituciones colaborando. Debido a esto en su mayor parte el código de ROS es software libre. [5]

ROS provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, paso de mensajes entre procesos y mantenimiento de paquetes, etc.. La librería esta orientada para un sistema UNIX aunque se esta trabajando en la adaptación a otros sistemas operativos como Mac OS, Windows y Fedora. Sin embargo, éstos todavía se encuentran en una fase experimental.

Muchos de los robots de la actualidad no existirían de no ser por ROS, el robot humanoide iraní 'Surena', el primer robot reponedor en supermercados 'Tally' son solo dos ejemplos de lo popular de este software. Gran parte de los robots de la actualidad se han servido, en mayor o menor medida, de este sistema operativo para dar vida a sus robots.

Para comprender mas fácilmente el trabajo realizado posteriormente debemos conocer algunas de las funciones y terminologías de ROS que hemos utilizado.

- Nodos: Los nodos son ejecutables que se comunican con otros procesos usando tópicos y servicios. Su finalidad es proporcionar tolerancia a fallos y separar las funcionalidades del sistema haciéndolo mas simple. Cada nodo debe tener un nombre único en el sistema para poder conectarse entre si sin ambigüedad. Contamos con una herramienta llamada *rosnode,* con ella podemos manejar y obtener información de los nodos creados: *rosnode list, rosnode kill, rosnode info* son los mas usados.
- Tópicos: son canales de comunicación para transmitir datos. Un tópico puede tener varios suscriptores que consulten dicha información. Cada tópico esta fuertemente pitado por un tipo de mensaje. Los nodos pueden subscribirse a un tópico solo si tienen el mismo tipo de mensaje. En este caso también contamos con una herramienta para trabajar con tópicos. Es una herramienta de línea de comandos que proporciona información sobre el tópico: *rostopic echo, rostopic list, rostopic find* etc.
- RVIZ:es un visualizador de datos muy completo y útil incluido en el paquete ROS. Su finalidad es ofrecer al usuario una forma fácil e intuitiva de visualizar los datos obtenidos. Estos datos deben proceder de sensores cuya finalidad sea el reconocimiento del entorno.

6.4. Trabajo realizado

Una vez más, se seguirá un orden cronológico para el relato de las tareas llevadas a cabo en este bloque.

Lo primero consiste en la preparación de las Raspberry, cerebro de el sistema, para que tenga todo lo necesario para cumplir el objetivo. Raspberry, utiliza como disco duro una tarjeta SD, es por ello que primero se instala un sistema operativo en dicha tarjeta. Se opta por Raspbian ya que como ya se ha comentado, es el más estable para este dispositivo. Unas vez instalado, siguiendo unos sencillos pasos descritos en su página oficial, comienza la configuración consistente en usar la línea de comandos para establecer idioma y teclado como español, configurar tanto I2C como SMBUS, ya que lo necesitaremos para establecer la comunicación con el LIDAR-Lite y finalmente instalar el paquete ROS necesario para la publicación y visualización de las medidas obtenidas con el telémetro.

Una vez superada la fase de configuración y con la intención de empezar a programar, resulta que la programación directamente en las Raspberry es algo incómoda. Es por ello que se realiza la programación en mi equipo personal. Debido a esto, se tuvo qué buscar que alternativas habían para conectarse a la Raspberry usando un ordenador con sistema operativo Windows. Para transferir los scripts realizados a la Raspberry, se optó por WinSCP, un cliente SFTP con entorno gráfico. Para ejecutar el código enviado mediante WinSCP, elegimos Putty, un software que permite crear un consola de comandos mediante una conexión ssh.

Para utilizar estos programas es necesario conocer la dirección IP de la Raspberry. Esta Raspberry la utilizaba en casa, universidad, bibliotecas etc. Por ello, no era buena idea asociarle una IP estática. Debido a que no en todos estos sitios se contaba con un monitor para poder conectar la Raspberry y ver su IP, decidimos conocer la dirección MAC del puerto ethernet. Mediante un software que escanea la red local y muestra las IP's asociadas a los dispositivos por medio de sus direcciones MAC, se resolvió el problema de llevar las Raspberry a cualquier lugar y poder acceder a ella con otro ordenador.

A lo largo de mi paso por el grado he aprendido diferentes lenguajes de programación, principalmente en C++, pero no Python. Es por esto que debí familiarizarme con el mismo. Debido a su similitud con C++, es cierto que no me resultó excesivamente difícil. Gracias al conocimiento que obtuve en la asignatura de Informática Industrial, opté por utilizar el paradigma de la programación orientada a objetos. Basándome en esto, realicé la división de los objetivos en tres clases. Cada una de ellas resolverán de manera progresiva el objetivo final.

6.4.1. Clase 1: motorPasoAPaso

Hacer girar el motor fue el primer objetivo que a cumplir. No obstante, para ello. primero hay que configurar los GPIO de la Raspberry de manera que cumplan su tarea según el conexionado previamente diseñado. Los GPIO, que se han configurado y utilizados en esta clase son:

- GPIO 13 → pinDir (controla la dirección del giro del motor).
- GPIO 11 → pinStep (controla la señal para dar un paso).
- GPIO 7 y 12 → pinMS1 y pinMS2 (controlan el tamaño de los pasos dados).
- GPIO 15 → pinFC (lectura del estado del opto-interruptor).
- GPIO 16 → pinEN (Activa/desactiva el controlador Pololu).

Una vez los GPIO están configurados, se comienza con la definición de los métodos.

Es conveniente ir explicando cada uno de los métodos que posee cada clase además de aclarar porqué creemos conveniente sus inclusiones.

- `PololuConfig`: este método nos permite activar y desactivar el controlador Pololu mediante la recepción de un True (ON) o un False (OFF)**.**

- `SetSizeStep`: con el seleccionaremos el tamaño en grados que queremos que tenga el motor. Para ello debe recibir el valor de MS1 y MS2.

- `girarUnPaso`: es el método con el que se consigue que el motor gire. Basándonos en el tamaño del paso, y habiendo previamente establecido el sistema de coordenadas mediante el método inicializar, se llevará la cuenta de la posición en grados del motor en incrementos y decrementos dados por la resolución del paso.

- `inicializar`: Su función es establecer el cero intentado encontrar la posición del opto-interruptor. Es un método muy útil y necesario, pero debe usarse con cuidado, debido a que el Lidar lleva conectado un cable. No pude diseñar, aunque se intentó repetidas veces, ningún algoritmo que permitiera encontrar el opto-interruptor sin importar cual fuera la posición de partida. Cualquier algoritmo que intente resolver esto, sin importar su posición de partida, intentará girar un determinado número de grados en una dirección, y si no se consigue el opto-interruptor, se deberá cambiar la dirección y probar hacia el otro lado. Sin embargo, esto plantea la siguiente cuestión: ¿Qué pasaría si el cable del Lidar está en su máxima extensión en una de las direcciones y se intenta girar hacia allí? la respuesta es que no tenemos forma de conocer si esta situación se da. Por esto, tuvimos que buscar una solución; decidimos que el método inicializar siempre buscará el opto-interruptor en el sentido de las agujas del reloj, así, el usuario solo asegurarse de que el cable permite el giro hacia la derecha. Además, con el objetivo de mejorar lo anterior, dispusimos que este método irá acompañado de otro, el de apagar**.**

- `apagar`: se debe llamar a este método siempre que se desee acabar la ejecución del programa. De esta forma, solo se necesita 'ayudar' al método inicializar con la posición del cable una vez. Si el método inicializar termina con éxito, se llevará un registro de donde estamos con respecto al opto-interruptor en todo momento. Basándonos en este registro el método apagar podrá colocar el motor justo en la posición del opto-interruptor. Así, la próxima vez que se ejecute el método inicializar, éste detectará que ya esta en la posición del opto interruptor y por tanto ya esta inicializado. Con la correcta combinación del método inicializar y el de apagar no es necesaria la supervisión de la posición del cable.

- `seleccionarGiro`: recibe una cadena de texto, 'Left' o 'Right' , con esto se modifica el valor del GPIO correspondiente para adaptarse a lo pedido.

- `invertirGiro`: Este método resulta imprescindible para la ejecución de bucles en los que queremos que el motor gire entre dos posiciones constantemente.

- `modificarPasos` y `modificarAngulo`: son los encargados, teniendo en cuenta la dirección de giro del motor y la resolución de paso actual, de modificar los atributos de la clase que guardan la posición del motor tanto en pasos como en grados.

- `calcAngle`: recibe un ángulo en grados y se calcula, teniendo en cuenta el tamaño del paso, el número de pasos necesarios para ir a ese ángulo.

- `GoToAngle`: este método recibe el ángulo en grados en el que se desea posicionar al motor. Utilizando el resto de métodos explicados hasta ahora, envía al motor al ángulo pasado. El posicionamiento en dicho angulo tendrá un error. Los ángulos que sean múltiplos de nueve, se alcanzarán sin ningún error. El resto de ángulos se alcanzarán, dependiendo del tamaño del paso con un error máximo de 0.1 grados (resolución de 0.225º/paso) o con un error máximo de 0.8 máximo grados (resolución de 1.8º/paso).

6.4.2. Clase 2: LIDAR-Lite

Esta clase había sido desarrollada por el profesor Alberto Hamilton antes de que yo comenzara el proyecto. Mi tarea ha sido entender su construcción, probarla y comprobar su correcto funcionamiento. El funcionamiento de esta clase se basa en la escritura y lectura del bus I2C. Esta es la manera en la que se establece comunicación con el Lidar-Lite. Escribiendo en el bus se mandan 'órdenes' al telémetro y mediante la lectura del bus obtenemos la 'respuesta' del telémetro. Las órdenes que entiende el Lidar-Lite y las respuestas que envía vienen descritas en el manual de usuario del mismo. Esta clase posee una gran variedad de métodos tales como el envió de órdenes o lecturas de respuestas, todas siguiendo lo descrito en el manual.

Aparte de los métodos nombrados, posee dos métodos importantes con los que se realiza la captura de medidas de distancia:

- `distance`: es el método principal para lanzar una captura de distancia con el Lidar. Este método devuelve la distancia medida en centímetros.

- `DistanceHealth`: al igual que el anterior método, devuelve la distancia medida en centímetros. La diferencia de este método con respecto al anterior, es que éste repite la medida siempre y cuando el Lidar-Lite no devuelva mediante el bus un 1 en el bit asociado al Health. Este 1 indicaría que el preamplificador del telémetro trabaja correctamente y que un pulso de referencia ha sido procesado correctamente.

- Debo decir que en la práctica, tras realizar repetidas pruebas, ambos métodos devuelven la misma distancia. Usando DistanceHealth, si el health no se devuelve como se espera, se realizará automáticamente otra medida, lo que genera una mayor lentitud en la adquisición de datos. Por ello, parecía que el segundo método era más fiable, pero revisando las medidas tomadas con el método `distance`, aún sin tener en cuenta el health, todas las medidas parecen correctas.

6.4.3. Clase 3: StepperAndLidarLite

Se basa en la combinación de la clase motorPasoAPaso y la clase LidarLite para conseguir el objetivo principal del proyecto: realizar medidas en distintas direcciones combinando el giro del motor con la capacidad del Lidar de tomar medidas.

Debido a que el trabajo de control del motor y de toma de medidas con el telémetro se ha realizado en los dos códigos previos, esta clase solo cuenta con dos métodos. Ambos sirven para realizar un barrido desde un ángulo inicial a un ángulo final. Este barrido consiste en dar un paso con el motor, tomar una medida, guardar dicha medida y ejecutar esto cíclicamente hasta que se llegue al ángulo superior del barrido.

Ambos métodos desarrollan la misma tarea, la diferencia entre ellos es que sweep toma las medidas sin tener en cuenta el bit asociado al health y `sweepWithHealth` sí lo hace, con las consecuencias ya explicadas.

6.4.4. Clase 4: TFG_CRESKO

Esta es la última clase y la que se ejecutará para poder obtener el mapa, por esto decidí llamarla TFG_CRESKO, ya que refleja el resultado final del trabajo.

Esta clase será la encargada de crear el mapa con las medidas realizadas utilizando la herramientas de ROS. Concretamente, se utiliza Laserscan, uno de los tópicos definidos en las librerías de ROS. Para utilizar la clase LaserScan, se necesita rellenar una serie de datos de forma correcta. Para ello,se consultaron los foros oficiales en los que hay ejemplos de cómo hacerlo. Algunos de estos parámetros son: el ángulo mínimo y máximo entre los que se realizará la medida, el incremento de ángulo entre medida y medida, en la siguiente imagen podemos ver la estructura típica que se proporciona en la página oficial de ROS para este tópico.

```
Header header               # timestamp in the header is the acquisition time of
                            # the first ray in the scan.
                            #
                            # in frame frame_id, angles are measured around
                            # the positive Z axis (counterclockwise, if Z is up)
                            # with zero angle being forward along the x axis

float32 angle_min           # start angle of the scan [rad]
float32 angle_max           # end angle of the scan [rad]
float32 angle_increment     # angular distance between measurements [rad]

float32 time_increment      # time between measurements [seconds] - if your scanner
                            # is moving, this will be used in interpolating position
                            # of 3d points
float32 scan_time           # time between scans [seconds]

float32 range_min           # minimum range value [m]
float32 range_max           # maximum range value [m]

float32[] ranges            # range data [m] (Note: values < range_min or > range_max
float32[] intensities       # intensity data [device-specific units].  If your
                            # device does not provide intensities, please leave
                            # the array empty.
```

Al ser este el código que da lugar al resultado final me parece conveniente mostrar una imagen del mismo y explicar su funcionamiento.

```
def lidar_loop(Lidar):

#Instanciamos un objeto de la clase LaserScan y definimos los parametros necesarios

    scan = LaserScan()
    lidar = stepperAndLidarLite.stepperAndLidarLite()

    size = int ( 360 / lidar.motor.stepSize ) + 1

    first = True

    scan.angle_min          = math.radians(0)
    scan.angle_max          = math.radians(360)
    scan.angle_increment    = math.radians(1.8)
    scan.time_increment     = 0.015
    scan.scan_time          = 0.015
    scan.range_min          = 0.03
    scan.range_max          = 10
    scan.ranges             = [0] * size
    scan.header.frame_id = 'map'

    #Inicializamos el nodo

    rospy.init_node('tfg_cresko', anonymous=True)
    update_rate = rospy.Rate(scan.scan_time)
    pub  = rospy.Publisher('LidarLite', LaserScan, queue_size=10)

    #bucle que se ejecutara mientras el nodo no se cierre

    while not rospy.is_shutdown():

        lidar.sweep(math.degrees(scan.angle_min), math.degrees(scan.angle_max) )

        for i in range(size):

            scan.ranges[i] = lidar.samples[i] / 100.0 #pasamos de centimetros a metros

        pub.publish(scan)

    lidar.motor.apagar()
```

La clase `tfg_cresko` consta de un único método en el que definimos todas las acciones necesarias. Inicialmente rellenamos los parámetros necesarios para el tópico LaserScan, tras esto con la instrucción `ropy.init_node` creamos un nodo cuyo nombre sera tfg_cresko. Con la ejecución de `rospy.Publisher` crearemos un tópico de nombre LidarLite, este tópico sera detectado por RVIZ para visualizar la nube de puntos.

Posteriormente entraremos en un bucle que solo terminará si el nodo se cierra. En este bucle llamamos al método `sweep` de la clase stepperAndLidarLite para realizar el barrido y la captura de medidas para posteriormente publicarlo y que RVIZ puede actualizar el mapa.

Si este bucle termina se ejecutará el método a`pagar`, para colocar el motor en la posición de inicio.

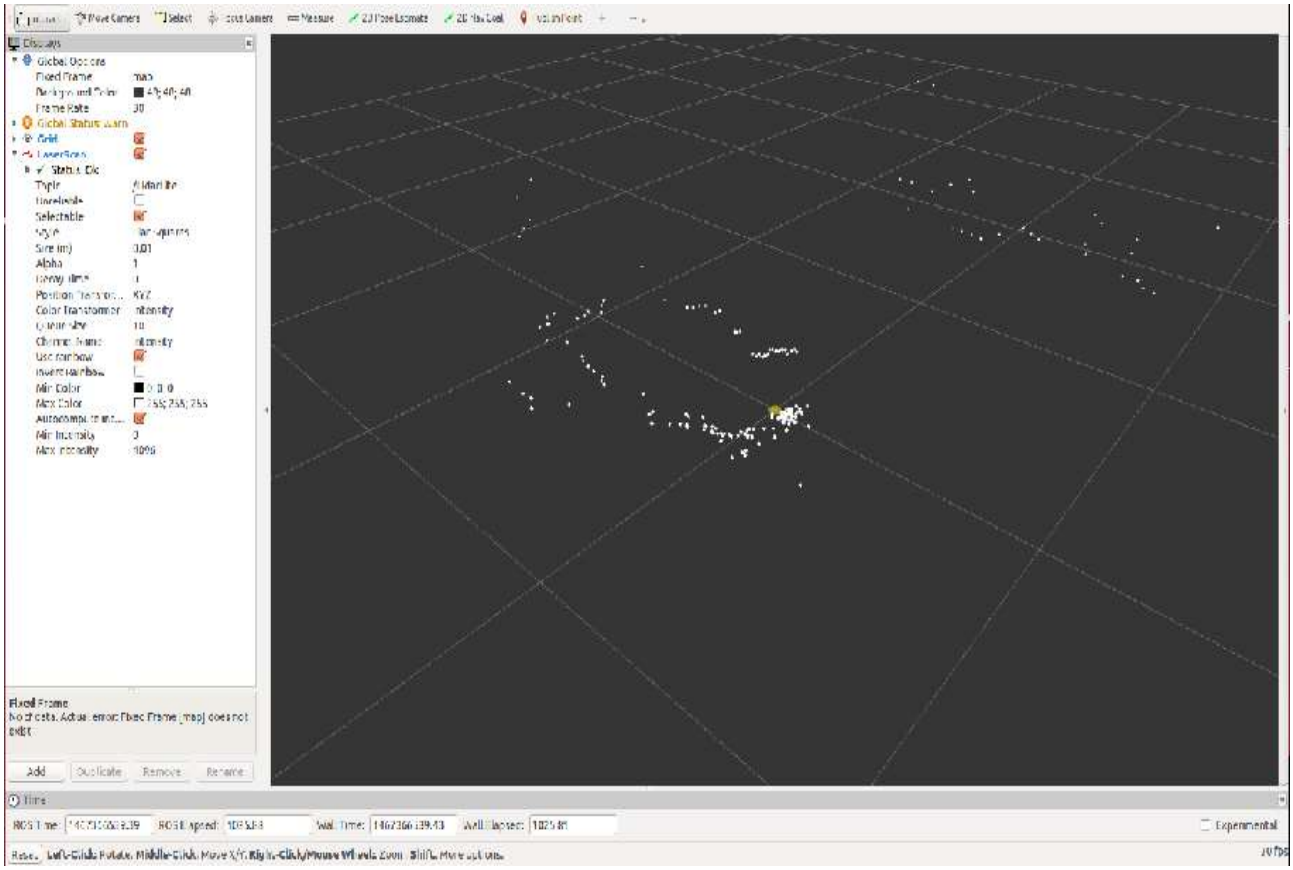
Una vez comprendido el funcionamiento del código creado es importante describir los pasos necesarios para ejecutarlo. Se ejecutará en la Raspberry los scripts dedicados al giro del motor y a la toma de medidas con el telémetro y en un ordenador con Linux, ejecutaremos ROS y RVIZ. Por tanto, el primer paso debe de ser ejecutar `roscore`, esto arrancará todo los servicios asociados a ROS. Esta ejecución nos devolverá una clave que deberemos introducir en la Raspberry, para ello ejecutaremos en la Raspberry la siguiente linea:

export ROS_MASTER_URI=http://XXX.XXX.X.X:YYYYY

Las X hacen referencia a la dirección IP del ordenador Linux y las Y a un código que nos devolverá la ejecución del comando `roscore` en el ordenador que ejecuta Linux.

Ahora podemos ejecutar el comando `rviz` en el ordenador Linux para abrir dicha aplicación. Por ultimo, y para comenzar la visualización de la nube de puntos, ejecutaremos en la Raspberry, desde el directorio /src/tfg_cresko/scripts el comando `rosrun tfg_cresko tfg_cresko.py`

Muestro una imagen del RVIZ en el que podemos observar una nube de puntos obtenida mediante las medidas realizadas por el LidarLite.



Hemos de mencionar que el código desarrollado ha seguido un control de versiones utilizando Github, pero no desde el comienzo del proyecto ya que es una herramienta a la que nos costó acostumbrarnos. Sin embargo, tras la fase de adaptación es una herramienta tremendamente útil. Ademas, este código estará disponible para compartir con la comunidad en dicha plataforma.

Para concluir este capítulo, destacar que a pesar de ser el capítulo más breve, sin duda es el que mas complicado ha resultado. Su brevedad es debido a que no vemos la necesidad de explicar línea a línea el código desarrollado. Consideramos más interesante explicar las bases de su funcionamiento para que todos puedan entender lo que se ha realizado y que aquellos que cuenten con los conocimientos de programación puedan consultar el código si así lo desean.

7. Líneas abiertas

En este apartado resumiremos las líneas que se deben de seguir tras la finalización de este proyecto. Hemos conseguido un dispositivo funcional, que cumple el objetivo que buscábamos, sin embargo queda un gran trabajo de pruebas, investigación y mejora para el que no hubo tiempo suficiente. Los aspectos en los que se debe continuar trabajando son:

- Fabricación de la PCB, con los ficheros obtenidos es posible obtener una PCB de un tamaño bastante reducido. La estructura tiene un espacio reservado para poder incluir el circuito dentro del sistema. Esto compactaría aun más el dispositivo final y evitaría desgastes y posible deterioros de los conectores, ya que lo único que habría que desconectar es alimentación y Raspberry.

- Las pruebas realizadas con ROS han sido suficientes y concluyentes, pero sin duda se necesita probar más situaciones y evaluar los resultados obtenidos para concluir que el dispositivo responde según lo esperado bajo cualquier circunstancia.

- Investigar alternativas en el modo de toma de medidas. Puede resultar interesante pensar formas diferentes de combinar el giro del motor y la toma de medidas. Una de ellas consiste en estudiar la viabilidad de mantener un giro constante del motor a tamaños de pasos muy pequeños e ir tomando medidas con el Lidar de forma continuada, sin que la toma de medidas tenga que esperar a dar un paso, ni que dar un paso deba esperar a la finalización de una captura de distancia. Esto podría, además de reducir vibraciones y ruidos en la estructura, acortar el tiempo de refresco de la nube de puntos obtenida.

- Los códigos creados en Python pueden ser, sin duda, mejorables y optimizables. Estos establecen una base para su funcionamiento principal. Por tanto se deberían perfeccionar y adaptar a los avances que se vayan logrando.

8. Presupuesto

El presupuesto asociado a este documento técnico se desarrolla a continuación y constara de dos partes básicas: costes materiales y costes de ejecución.

8.1. Material necesario

• Motor paso a paso	13€
• Controlador Pololu	2€
• LIDAR-Lite	90€
• Opto-Interruptor	2€
• Raspberry pi	35€
• Convertidor DC/DC	2€
• Impresion de Diseños 3D	80€
• Total en materiales	224€

8.2. Horas de trabajo realizadas

• Dedicadas a electrónica	80 Horas
• Dedicadas a diseño 3D	70 Horas
• Dedicadas a desarrollo Software	150 Horas
• Total Horas	360 Horas
• Precio por Hora Ingeniero Electrónico	30€
• Total mano de obra:	9000€

Total Proyecto: 9.224€

9. Conclusions

In this section, I would like to briefly summarize those objectives that have been achieved throughout the development of this project.

First, we have designed a PCB using the software OrCAD capture and Layout which makes possible the connection of all the electronics required for the engine, telemeter and opto-interupter proper function.

It has been created through the FreeCAD application, the different parts needed for the coupling and protection of the whole system and have been manufactured utilizing a 3D printer.

We have applied Python in order to produce the needed algorithms for the control of the engine rotation. Furthermore, we created a code that allows us merge the engine rotation of the motor with the measures taken by the Lidar.

Finally we have develop a Python code that leverages ROS tools in order to visualize the measures taken in a map.

Therefore, with all of these, we obtained a low cost telemeter able to taken measures in 360º. This device is considered as great utility for the mobile robotics, especially in interiors.

10. Valoración personal

Es necesario incluir una pequeña valoración personal acerca del proyecto que he desarrollado. Empiezo exponiendo que la realización de este proyecto me ha parecido muy interesante. Considero que he tenido mucha suerte al poder tener contacto real con el mundo de la robótica. A pesar de las numerosas complicaciones que han surgido durante el desarrollo del trabajo, he aprendido mucho; con él, he comprobado como sería el trabajo que me gustaría desempeñar durante mi carrera profesional.

Además he podido comprobar que he cursado la ingeniería adecuada, pues gracias a lo aprendido en el grado de Ingeniería Electrónica, he sido capaz de aplicar los conocimientos imprescindibles para fabricar la electrónica para el control de dispositivos. Además de esto,he obtenido conocimientos de programación suficientes para afrontar problemas que se me plantearán, como en este proyecto, a lo largo de mi vida profesional.

He de puntualizar que el proyecto tiene margen de mejora, sobre todo con respecto a la programación diseñada. Su funcionamiento es correcto, aunque se puede mejorar realizando un código mejor. Además, me hubiera gustado conseguir fabricar la PCB, algo que no pudo llevarse a cabo por falta de tiempo.

Sin embargo, considero que he conseguido un producto que cumple los objetivos que se planteaban con este proyecto. Es por ello. que me gustaría poder ver algún día que el trabajo que desarrolle ha sido aprovechado, aunque sea en parte, para su aplicación en un sistema real. Concretamente, sé que se plantea la posibilidad de que este dispositivo forme parte de un proyecto llevado a cabo por el Grupo de Robótica de la ULL (GRULL). Este proyecto consiste en conseguir una silla de ruedas autónoma, para ello necesita de sensores que le provean de información de su entorno como el que he desarrollado. No obstante, pienso que no tiene que limitarse a esta finalidad, ya que cualquier sistema que necesite de un reconocimiento de su entorno puede aprovecharse del sistema que he diseñado. Sobre todo en aquellos en los que sea importante mantener un precio reducido en los sensores utilizados.

11. Bibliografía

1. http://www.todorobot.com.ar/tutorial-sobre-motores-paso-a-paso-stepper-motors/
2. https://impresoras3d.com/blogs/noticias/102883975-tipos-de-impresoras-3d
3. http://www.freecadweb.org/?lang=es_ES
4. https://www.python.org/
5. http://wiki.ros.org/