UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO Y CONTROL DE UN ROBOT MÓVIL INESTABLE USANDO ARDUINO

TRABAJO FINAL DEL

GRADO EN INGENIERÍA ELECTRÓNICA INDIUSTRIAL Y AUTOMÁTICA

REALIZADO POR

JOSÉ ANTONIO MÉNDEZ IBARRA

TUTORIZADO POR

EDUARDO QUILES CUCARELLA

CURSO ACADÉMICO: 2020/2021

Índice del proyecto

DOCUMENTO Nº 1.

MEMORIA Y ANEXOS

Índice

Índice de figuras

1. Objeto

La finalidad de este proyecto es llevar a la práctica y combinar los distintos conocimientos adquiridos durante el Grado en Ingeniería Electrónica Industrial y Automática, tales como la electrónica, la robótica, la mecánica, el control o la programación. Esto se va a conseguir realizando un prototipo de un robot móvil de dos ruedas, un robot autobalanceado.

Este robot tendrá que mantenerse en posición vertical utilizando solamente sus dos ruedas, por lo que habrá que realizar un control de la estabilidad, siendo necesario obtener el ángulo de inclinación del robot en todo momento. Además, el usuario deberá ser capaz de controlar el robot, desplazándolo mediante un control remoto a través de la creación de una aplicación para móvil.

En este proyecto se va a abordar el diseño electrónico, el diseño mecánico, la programación, la creación de la aplicación y el control de la estabilidad del robot. Además, será necesaria la compra de los distintos componentes electrónicos y la impresión 3D de las piezas del robot.

2. Antecedentes

La robótica es un campo de estudio nuevo, pero muy avanzado en estos días y con capacidad de llegar aún más lejos en cualquier ámbito en el que se utiliza. Durante la realización del Grado de Ingeniería Electrónica Industrial y Automática se ha podido sentar unas bases en robótica y debido a esto se ha elegido este proyecto como Trabajo Fin de Grado, además del interés personal en esta materia.

Ya se cuenta con experiencia en el montaje, la programación y diseño de robots debido a distintos proyectos que se han realizado en las asignaturas de la carrera. Estos proyectos han sido el montaje y programación de un brazo robot y el diseño, fabricación y programación de un robot móvil de tres ruedas. En este proyecto se aumenta la dificultad realizando un robot que en un principio sería inestable sin un control de su estabilidad.

3. Estado del arte

3.1. Inicios de la robótica

Un robot es una máquina automática programable que puede realizar diversas actividades sin necesidad de intervención humana, sustituyendo a las personas en algunas tareas [1].

Los primeros robots de la historia fueron las tortugas de Bristol, desarrollados en el año 1948 por el neurofisiólogo estadounidense William Gray Walter [2]. Estos robots (Elmer y Elsie eran sus nombres) contaban, entre otras cosas, con dos sensores, uno capaz de detectar luz y el otro, tacto; y dos motores, uno para el desplazamiento y otro para la dirección. Estos robots se movían en dirección a un estímulo luminoso captado por el sensor.

En Figura 1 se puede ver uno de estos robots junto a su inventor.

Figura 1: Una de las tortugas robot con el inventor. Fuente: [3]

Por otro lado, el primer robot industrial programable fue Unimate [4], creado por George Devol, un inventor estadounidense, en el año 1954; pero no sería hasta el año 1961 cuando el primer Unimate sería instalado. Esto sucedió tras haberse asociado Devol con Joseph Engelberger, un ingeniero de la Universidad de Columbia en el año 1956, formando la primera empresa dedicada a la robótica llamada Unimation.

3.2. Robots autobalanceados

Los robots autobalanceados son robots capaces de mantener el equilibrio sobre dos ruedas aún en presencia de perturbaciones externas. El comportamiento de este tipo de robots se basa en el de un péndulo invertido, uno de los problemas más recurrentes de la teoría de control, pues implica el paso de un sistema que es inestable en lazo abierto a un sistema estable en lazo cerrado.

Un péndulo invertido consiste en una varilla vertical que puede girar libremente unida en uno de sus extremos a una banda conectada a un carro [5]. Con un motor y dicha banda, se puede hacer que el péndulo se deslice de forma horizontal, buscando así mantener el otro extremo de la varilla por encima del carro en una posición vertical tal y como se puede apreciar en Figura 2. Este concepto ha sido experimentado por muchas personas sin saberlo, pues algo que bastante gente ha hecho, ya sea siendo un niño o adulto, es coger una escoba, una fregona o cualquier cosa semejante a un palo con algo pegado en el extremo e intentar mantenerlo en equilibrio moviendo solamente la palma de la mano, en la que se encontraría el otro extremo del objeto.

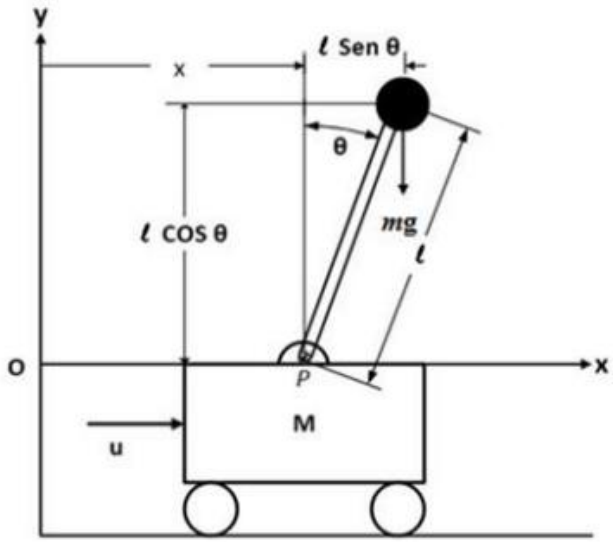


Figura 2: Sistema de un péndulo invertido sobre un carro. Fuente: [6]

Para mantener el robot autobalanceado en una posición vertical, los motores deben compensar la caída del robot accionando las ruedas en el sentido en el que el robot está cayendo (Figura 3). Para hacer esto es necesario conocer la inclinación del robot mediante el uso de sensores que permitan al robot conocer el ángulo en el que se encuentra y mediante un lazo cerrado de control se compara dicho ángulo con el ángulo de referencia que debe seguir el robot. En función del error que haya, los motores se accionarán con una velocidad mayor o menor para conseguir alcanzar la referencia.

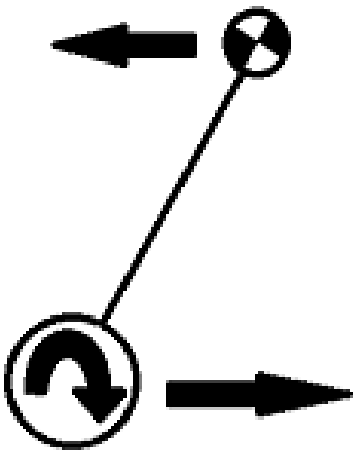


Figura 3: Esquema de funcionamiento de un robot autobalanceado.

En la actualidad, uno de los casos más representativos de este tipo de robots es el Segway (Figura 4), un vehículo de transporte ligero de dos ruedas capaz de medir su inclinación gracias a diversos sensores y cuyo balanceo es controlado por ordenador. Este producto fue inventado por Dean Kamen en el año 2001 y es comercializado por la compañía Segway Inc., vendiéndose las primeras unidades en el año 2002 [7].

Figura 4: Segway. Fuente: [8]

3.3. Aplicaciones

Las aplicaciones que se le pueden dar a sistemas basados en el péndulo invertido son muy diversas abarcando sectores como biomecánica, aeroespacial o transporte [9]; aunque también una de las aplicaciones más abundantes es el de la docencia y la investigación.

Como se mencionó anteriormente, en el campo del transporte un ejemplo de esto sería el vehículo Segway.

En el ámbito de la biomecánica el péndulo invertido es utilizado para modelar bípedos caminantes, concretamente cuando el robot está apoyado solo de una pierna, mientras que la pierna en movimiento se modela como un péndulo que oscila libremente suspendido de la cadera del robot. Un ejemplo de esto es el robot Asimo de Honda (Figura 5).

Figura 5: Robot Asimo. Fuente: [10]

En cuanto al sector aeroespacial, este problema se aplica en el control de la posición de un cohete durante el despegue para mantenerlo en una posición vertical (Figura 6). En este caso,

el ángulo de inclinación del cohete es controlado variando el ángulo de aplicación de la fuerza de empuje colocada en la base del cohete.

Además, recientemente se conoció que el cohete Starship de la empresa estadounidense de fabricación aeroespacial SpaceX aterrizó en la Tierra (Figura 7), lo que exigía llegar a la superficie en una posición vertical aplicando también el problema del péndulo invertido.

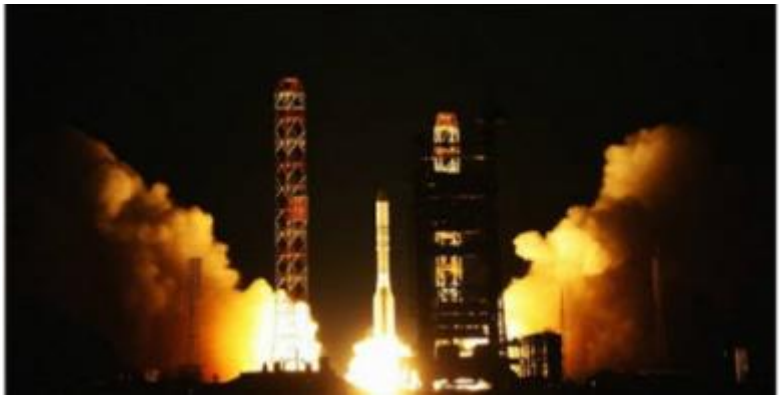


Figura 6: Cohete despegando. Fuente: [9]

Figura 7: Starship aterrizando. Fuente: [11]

4. Estudio de necesidades

A continuación se van a enumerar las especificaciones y las prestaciones que debe tener el robot realizado en este proyecto.

- Capacidad de mantener el equilibrio. Esto es lo más importante del proyecto ya que en caso de no cumplirse el robot caería al suelo siempre que se pusiera en marcha. Por tanto, se va a necesitar realizar un control de la estabilidad en lazo cerrado y va a ser necesario disponer de un microcontrolador, sensores capaces de medir la inclinación y motores para conseguir la estabilidad del robot.

- Control remoto del robot mediante una aplicación para teléfono móvil. Con esto se consigue que el usuario tenga control sobre el robot y que este no solo se mantenga

en equilibrio. Para conseguir esto se va a necesitar un módulo bluetooth que comunique el microcontrolador con el teléfono móvil y una aplicación para el teléfono del usuario.

- Alimentación del robot recargable. Las fuentes de alimentación utilizadas en el proyecto deben poder ser recargadas, evitando así la compra periódica de pilas o baterías para que el prototipo funcione.

- Interfaz de usuario de la aplicación sencilla. La aplicación debe ser lo suficientemente entendible por el usuario para que no haya problemas a la hora de manejar el robot.

- Diseño sencillo y fácil de montar y desmontar. Esto facilita el ensamblaje del robot y los cambios que se puedan realizar en el prototipo durante el desarrollo del proyecto, ya sea remplazar los componentes electrónicos, realizar el conexionado del circuito o realizar cambios en la estructura del mismo mediante la sustitución de piezas por otras distintas.

- Tamaño pequeño o mediano del robot. Al ser un prototipo cuya finalidad es el aprendizaje y debido a las limitaciones en la fabricación de las piezas que puedan surgir si se hace un robot muy grande, la altura no será superior a los 50 cm y el ancho no será mayor de 25 cm.

- Las ruedas deben presentar un buen agarre con el suelo evitando que el robot deslice.

- Este proyecto debe estar terminado, como muy tarde, antes de la última convocatoria para la exposición del Trabajo Fin de Grado de la Escuela Técnica Superior de Ingeniería del Diseño para el curso 2020/2021, por lo que a principios de septiembre de 2021 tanto el prototipo, como la memoria y la presentación de este trabajo deberán estar finalizados.

- Para este proyecto no se ha establecido un límite en el coste del prototipo, aunque se va a intentar hacer lo más barato posible.

5. Alternativas

En este apartado se presentan las diferentes alternativas que se han tenido en cuenta a la hora de elegir los componentes del robot. Se explicarán en mayor profundidad los componentes elegidos en el apartado de "Descripción detallada de la solución".

5.1. Actuadores

En este apartado se van a comentar las diferentes alternativas existentes a la hora de mover el robot.

5.1.1. Motor

A la hora de elegir el tipo de motor, se han valorado las siguientes opciones:

- Motor de corriente continua: Este motor presenta el comportamiento más sencillo. Al aplicarle tensión comienza a funcionar y puede alcanzar una alta velocidad de giro.

- Servomotor de giro completo: Presenta un alto par y el control de velocidad se puede realizar de forma sencilla pues tiene incorporado un circuito de control.
- Motor paso a paso: Convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, por lo que gira una cantidad de grados en función de sus entradas de control. Los ángulos de giro suelen ser pequeños, por lo que favorecería un control de velocidad en lazo abierto.

La solución elegida en este caso ha sido el **motor de corriente continua**, debido a la facilidad en el uso de este motor, el buen par que proporciona y la velocidad que puede alcanzar. Otro de los factores tenidos en cuenta ha sido el precio, pues es más barato que un motor paso a paso.

Si bien es cierto que estos motores son los que dan más dificultades a la hora de controlar su velocidad en lazo abierto (habría que variar la tensión de entrada), muchos de ellos se pueden comprar con sensores de velocidad incorporados para así conseguir un control en lazo cerrado.

5.1.2. Driver

El sistema de control debe utilizar un driver para controlar los motores. Las opciones que se han planteado han sido las siguientes:

- Controlador L289N: Este circuito cuenta con un driver L298N, el cual es capaz de controlar dos motores de corriente continua a la vez y es capaz de soportar una elevada potencia gracias a que en el circuito viene incorporado un disipador.
- Controlador TB6612FNG [12]: Al igual que el controlador L289N, puede controlar dos motores de corriente continua al mismo tiempo. Aunque no tiene un disipador de potencia, presenta una mayor eficiencia y tamaño más reducido.
- Controlador L293D [13]: Capaz de controlar cuatro motores, pero con unas dimensiones mayores al resto de opciones.

En este caso se ha elegido el **controlador L298N**. El principal motivo ha sido que este controlador puede funcionar como una fuente de alimentación de 5 V si es alimentado con una tensión menor de 12 V, gracias a que posee un regulador de tensión LM7805. Otro de los factores que han influido en la decisión ha sido el precio, pues es el más barato de los tres drivers planteados.

5.2. Control

En este apartado se van a exponer las diferentes alternativas que se han estudiado para controlar el robot, es decir, los componentes con los que se va a programar el robot, se va a medir la inclinación para su control de estabilidad y se va a realizar la comunicación con el dispositivo móvil.

5.2.1. Hardware

Para la programación del robot se han tenido en cuenta estas opciones:

- Raspberry Pi: Es una serie de ordenadores de placa reducida de bajo coste. La ventaja de esta placa son sus prestaciones: Tiene un buen procesador, posee WiFi y bluetooth y su memoria depende de la tarjeta microSD que se le introduzca.

- STM32 Discovery: Placa de alto rendimiento con muchos puertos de conexión. Presenta un acelerómetro de 3 ejes y un micrófono digital omnidireccional.
- Arduino Nano: Placa de uso libre y versátil. Se alimenta fácilmente, ya sea con baterías o con transformadores. Su lenguaje de programación es C ligeramente modificado.

Se ha decidido utilizar un **Arduino Nano** para el control del robot por diversos motivos: es bastante más barato y pequeño que las otras opciones, tiene pines suficientes para llevar a cabo el proyecto y durante la carrera se ha utilizado Arduino en numerosas ocasiones, por lo que hay experiencia previa. Además, actualmente se posee esta placa.

5.2.2. Bluetooth

Entre las opciones tenidas en cuenta para la conexión bluetooth se encuentran:

- SPP-C.
- HC-06.
- BLE 4.0 AT-09.

Se ha elegido el módulo **SPP-C**. Estos módulos funcionan de manera muy similar, alimentados con una tensión similar y son compatibles con Android. El único factor determinante en escoger uno u otro ha sido que ya se cuenta con un SPP-C de proyectos anteriores.

5.2.3. Sensor de inclinación

Para medir la inclinación de robot se ha elegido entre:

- MPU-6050 [14]: Es una unidad de medición inercial (IMU) con seis grados de libertad, pues consta de un acelerómetro de 3 ejes y un giroscopio de 3 ejes. La comunicación con el microcontrolador se puede realizar mediante bus SPI o bus I2C.
- ADXL345 [15]: Acelerómetro de 3 ejes independientes y de bajo consumo. Es un sensor micromecanizado (MEMS) que detecta aceleración en los ejes X, Y y Z. Se puede realizar la comunicación entre el sensor y el microcontrolador tanto por bus I2C como por bus SPI.
- MMA7455L [16]: Este sensor presenta prestaciones muy similares al ADXL345.

Para medir la inclinación se ha elegido el **MPU-6050** debido a que consta con un acelerómetro y un giroscopio, cuya combinación a la hora de obtener la inclinación del robot permite obtener medidas más fiables. Además, a la hora de buscar información sobre sensores para obtener la inclinación del prototipo, este sensor es uno de los que más aparecen, por lo que hay más facilidad para encontrar información sobre este.

5.3. Alimentación

La alimentación del robot es un aspecto importante para el proyecto, pues debe poder obtener la corriente necesaria para que todos sus componentes funcionen correctamente. Además, como se mencionó en "Estudio de necesidades", debe ser recargable. Se ha elegido cómo alimentar el robot entre las siguientes opciones:

- Batería Li-Ion [17]: Batería de iones de litio. Estas baterías son ligeras y utilizan como electrolito una sal de litio entre el electrodo positivo y el negativo.

- Batería LiPo [17]: Batería de polímero de iones de litio. En esta batería el electrolito no es un líquido como en el caso anterior sino que emplean un gel. Esta batería es más flexible que la anterior.
- Batería Ni-MH [18]: Batería níquel-metalhidruro. En esta batería hay un ánodo de oxihidróxido de níquel y en el cátodo una aleación de hidruro metálico.

Se ha escogido la batería **Li-Ion**. En este apartado ha habido muchas dudas, pues una buena opción habría sido también la batería LiPo, la cual es capaz de entregar más corriente que una batería Li-Ion. El factor determinante en la elección realizada ha sido el precio, pues es más barata, y la experiencia previa usando ese tipo de batería en otros proyectos.

5.4. Estructura del robot

El montaje del prototipo debe ser rápido y fácil y sus piezas no deben ser difícilmente realizables. Para la realización de la estructura se ha escogido entre las siguientes opciones:

- Impresión 3D: Las ventajas de la impresión 3D son la capacidad de realizar piezas con relieves, por lo que no hay necesidad de limitarse a diseñar piezas planas a la hora de crear la estructura del robot. El material en este caso sería derivado del plástico.
- Corte láser: En este caso las piezas son planas y con un grosor concreto. Al no poder añadirse relieves, a la hora del montaje es necesario el uso de piezas auxiliares y de pegamentos para mantener esas piezas pegadas. En este caso el material a usar en el montaje puede ser aluminio, metacrilato, policarbonato, etc.

Se ha elegido fabricar las piezas mediante **impresión 3D**, por lo que el material del que estará hecho el robot será **plástico**. Se ha elegido esta opción debido a que con la impresión 3D se pueden realizar piezas con relieve debido a que si se usara corte láser, se necesitaría unir varias piezas para conseguir un equivalente o, en algunos casos, no se podrían reproducir. En adición, se tiene en propiedad una máquina de impresión 3D junto con el material para realizar las piezas, además de conocer cómo utilizar dicha máquina.

6. Descripción detallada de la solución

En este apartado de la memoria se va a explicar con detalle los componentes elegidos para la realización de este proyecto, así como sus principios de funcionamiento.

6.1. Actuadores

6.1.1. Motor

Para la realización de este proyecto se ha usado un motor de corriente continua de 6 V, el cual se puede ver en Figura 8.

Figura 8: Motor CC. Fuente: [19]

Un motor de corriente continua (motor DC o motor CC) es una máquina que transforma la energía eléctrica en energía mecánica, basando su funcionamiento en la Ley de la Fuerza de Lorentz, la cual postula que una partícula con carga eléctrica en movimiento, al ser sometida a un campo magnético, experimenta una fuerza perpendicular a dicho movimiento y a la dirección del flujo del campo magnético [20].

6.1.1.1. Partes de un motor CC

Un motor de corriente continua está formado principalmente por las siguientes partes [21][22]:

- Estator: Es la parte fija del motor y uno de los elementos fundamentales para transmitir potencia en los motores eléctricos (Figura 9). Está formado por un material ferromagnético y en su interior se distribuyen un número par de polos inductores a cuyo alrededor hay unas bobinas que forman el devanado del inductor, que generan el campo magnético del motor, al ser alimentados con corriente continua, presentando polaridades norte y sur de forma alternativa.

Figura 9: Estator de un motor CC. Fuente: [23]

- Rotor: Es la parte móvil del motor y el otro elemento fundamental (Figura 10). Está formado por chapas de acero aisladas entre sí montadas sobre el eje del motor, el cual soporta bobinas arrolladas sobre un núcleo magnético, que gira dentro del campo magnético generado por el estator.

Figura 10: Rotor de un motor CC. Fuente: [24]

- Entrehierro: Es el espacio que hay entre el estator y el rotor cuya finalidad es evitar el rozamiento entre ambos, aunque debe ser mínimo para no debilitar el campo magnético.
- Colector de delgas: Esta parte permite que el rotor y el estator tengan la misma alimentación. Se sitúa sobre el eje de giro y consta de un anillo, aislado de dicho eje, formado por láminas aisladas unas de otras y conectadas a cada una de las bobinas giratorias (Figura 11). Para la conexión entre rotor y estator se hace uso de las escobillas.

Figura 11: Colector de un motor CC. Fuente: [24]

- Escobillas: Son unos bloques de grafito que hacen presión sobre los colectores que se encuentran en el eje de giro para que se produzca el contacto eléctrico (Figura 12).

Figura 12: Escobillas de un motor CC. Fuente: [24]

6.1.1.2. *Principio de funcionamiento*

Ambos extremos de la bobina del rotor (o electroimán) están conectados al colector y este hace contacto la mayor parte del tiempo con el estator, cuyos polos magnéticos son permanentes. Cuando circula corriente eléctrica, esta llega a la bobina del electroimán y sus extremos adquieren la polaridad de la parte del estator con la que se realiza la conexión eléctrica.

Cuando se establece la polaridad en cada extremo del rotor, estos inmediatamente son repelidos por los polos del estator, pues se enfrentan dos polos iguales. De esta forma se inicia el movimiento del motor.

Una vez el rotor ha girado, llega un momento en el que el colector no realiza conexión con el estator, por lo que el electroimán pierde su polaridad. Sin embargo, la fuerza de inercia que mantiene hace que la bobina siga en movimiento y llega el momento en el que el colector vuelve a conectarse con el estator, por lo que este vuelve a polarizarse y se repite el proceso.

En Figura 13 se representa de forma esquemática el funcionamiento del motor, siendo a y b los terminales del colector y 1 y 2, las mitades de la bobina del electroimán.

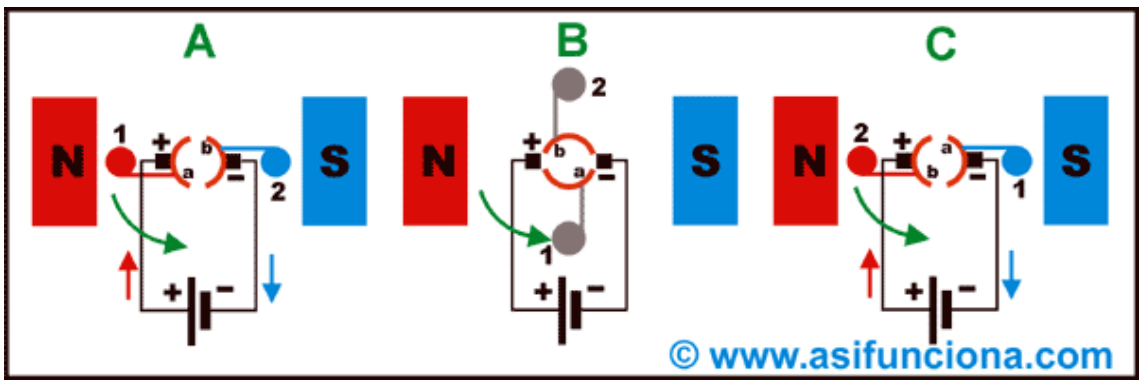


Figura 13: Esquema de funcionamiento de un motor CC. Fuente: [20]

Todo este proceso se debe a la mencionada Ley de la Fuerza de Lorentz, cuya fórmula es:

$$\vec{F} = q \cdot \vec{v} x \vec{B}$$

Siendo:

- $\vec{F}$: Fuerza producida
- q: Carga eléctrica
- $\vec{v}$: Velocidad de la carga
- $\vec{B}$: Campo magnético

Esta fórmula se corresponde con la fuerza que aparece en una carga puntual. Sin embargo, para aplicarla al motor, es decir, a las espiras que forman su bobinado es necesario modificarla. En el caso de un cable, la carga eléctrica se desplaza una distancia igual a la longitud de dicho cable y emplea un tiempo en realizar dicho desplazamiento. Teniendo esto en cuenta:

$$v = \frac{l}{t} ; I = \frac{q}{t}$$

$$\vec{F} = q \cdot \frac{\vec{l}}{t} x \vec{B} = \frac{q}{t} \cdot \vec{l} x \vec{B} = I \cdot \vec{l} x \vec{B}$$

Siendo:

- I: Intensidad de la corriente eléctrica
- $\vec{l}$: Longitud del cable
- t: Tiempo

Dado que en las espiras del motor, tanto el extremo inicial y final están conectados al colector, la corriente eléctrica circulará en sentidos distintos en función del tramo de cable en el que se encuentre, lo cual provocará dos fuerzas que produzcan rotación en esta. En Figura 14 se muestra un esquema simplificado de lo que se acaba de mencionar.

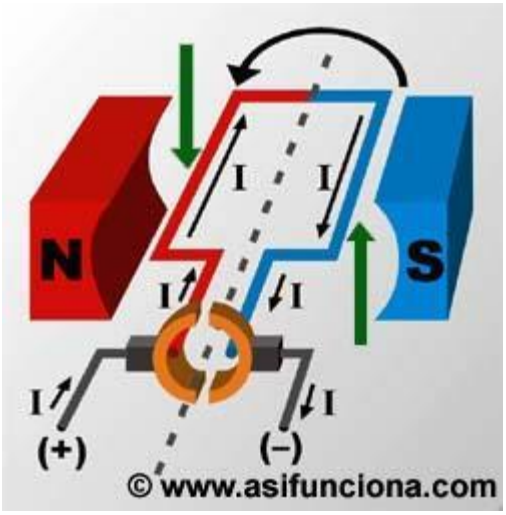


Figura 14: Esquema del efecto de un campo magnético en una espira. Fuente: [20]

6.1.2. Driver

El driver o controlador elegido para este proyecto es el L298N (Figura 15), el cual es capaz de controlar tanto el sentido como la velocidad de giro de dos motores de corriente continua, conectados a sus cuatro salidas, haciendo uso de seis entradas (cuatro digitales y dos analógicas), además de otras tres utilizadas para la alimentación del circuito.

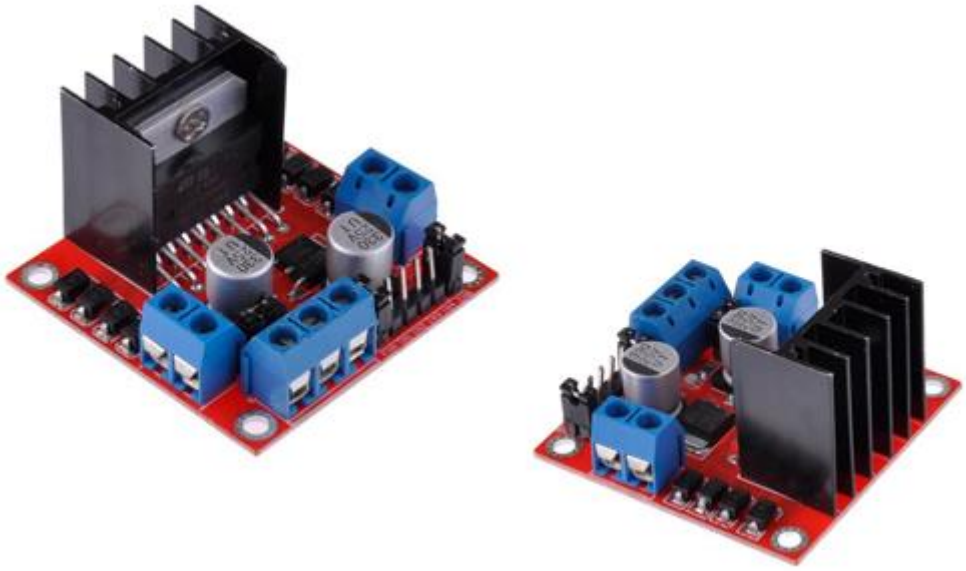


Figura 15: Driver L298N. Fuente: [25]

6.1.2.1. L298N

El L298N puede ser alimentado con tensiones entre 6 y 35 V y es capaz de proporcionar a cada salida unos 2 A. Además, cuenta con una entrada a la que hay que conectar 5 V para alimentar el circuito lógico del driver. Sin embargo, este controlador posee un regulador de tensión LM7805, cuya finalidad es la de alimentar con un voltaje estable de 5 V la parte lógica del circuito. Este regulador se puede utilizar si la alimentación de entrada (Vin) es menor de 12 V, en cuyo caso no hay necesidad de conectar 5 V a la entrada antes mencionada y esta pasa a convertirse en una salida que proporciona 5 V. En casa de tener una Vin mayor, hay que desconectar el jumper regulador y alimentarla.

En cuanto al control de los motores, este driver cuenta con cuatro salidas, dos para cada mator. A estas salidas llega la tensión que se proporciona a través de Vin, aunque con una bajada de unos 2 V debido al consumo del propio circuito. Estas salidas son controladas por las entradas ENA, ENB, IN1, IN2, IN3 e IN4; las cuales se encargan de controlar la velocidad (ENA y ENB al ser conectadas con una señal PWM) y el sentido de giro (IN1, IN2, IN3 e IN4) de cada motor (ENA, IN1 e IN2 controlan el motor A y el resto, motor B).

En Figura 16 se pueden apreciar las entradas y salidas que se han mencionado hasta ahora y en Figura 17, el circuito electrónico del driver.

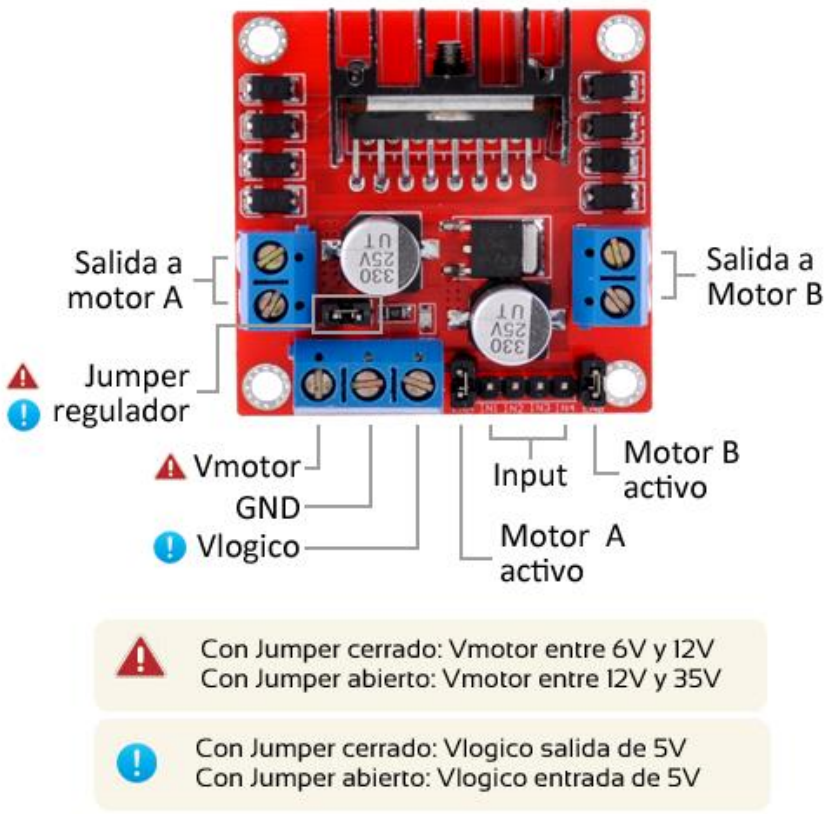


Figura 16: Vista con detalle de entradas y salidas del L298N. Fuente: [25]

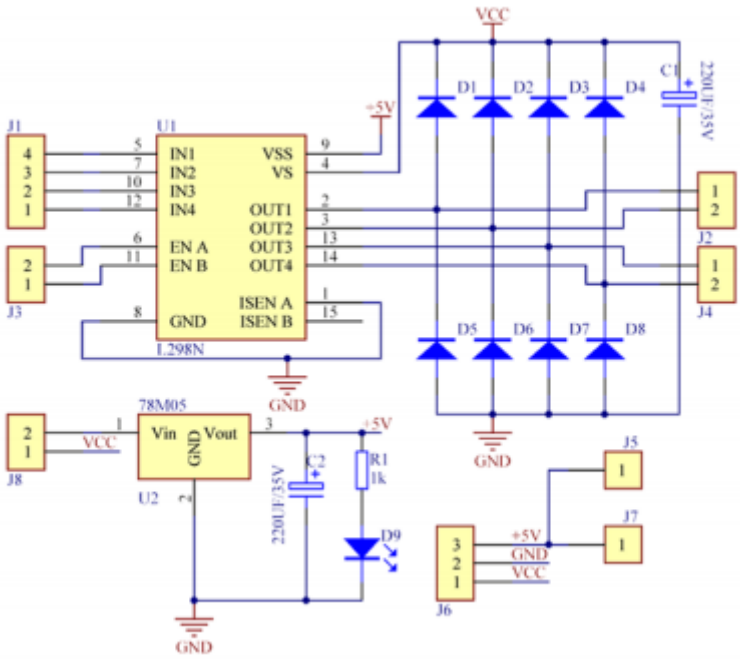


Figura 17: Circuito del driver L298N. Fuente: [26]

El driver además consta de protecciones contra sobre intensidad, sobre temperatura y ocho diodos de protección contra corrientes inducidas [25].

6.1.2.2. Principio de funcionamiento

El control de los motores con este driver se basa en un puente H, el cual consiste en un circuito con cuatro transistores, conectados como se puede ver en Figura 18, que controlan el sentido de la corriente que llega a una carga (Figura 19).

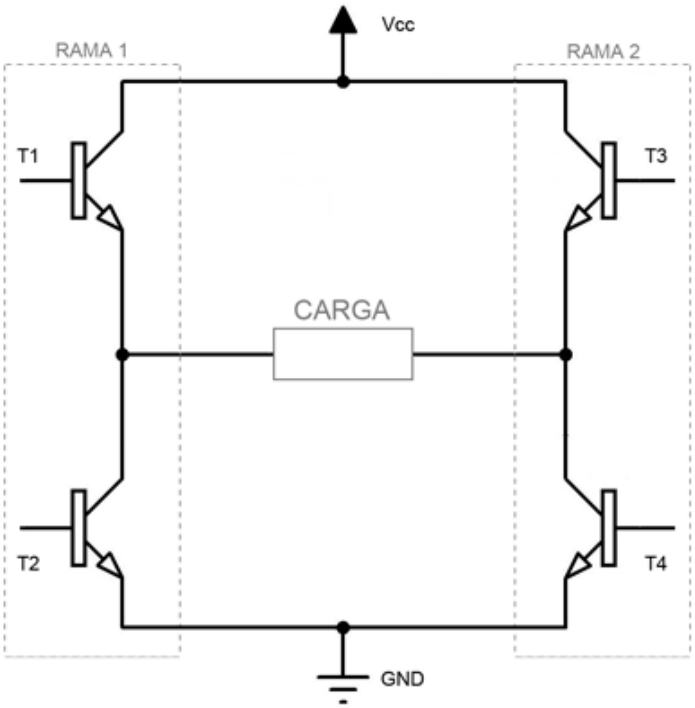


Figura 18: Esquema de un puente H. Fuente: [25]

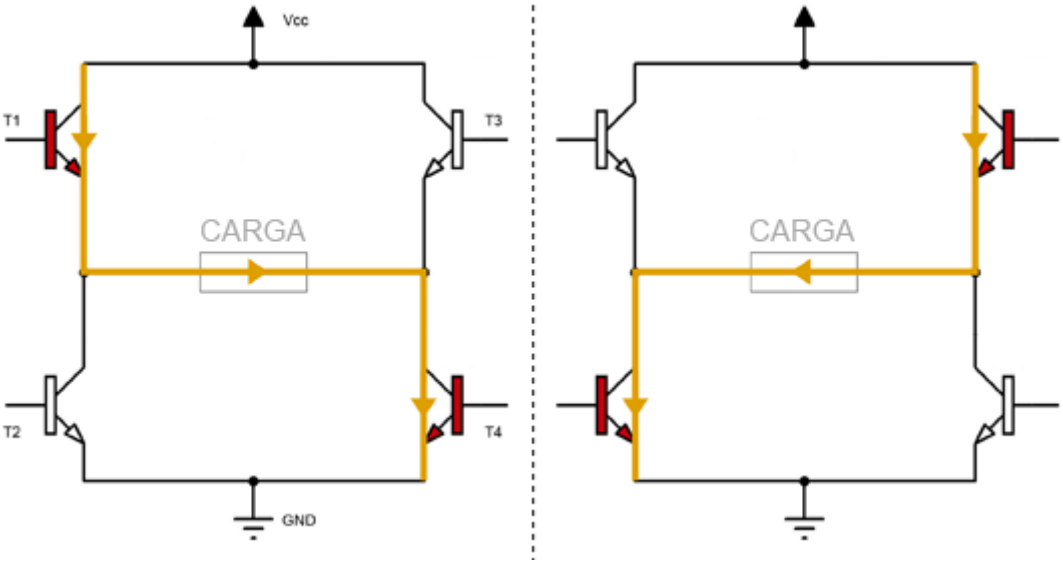


Figura 19: Funcionamiento de un puente H. Fuente: [25]

En cada rama se activa un transistor y los activados en cada rama deben ser opuestos para que así circule la corriente desde Vcc hasta GND pasando por la carga que haya conectada, pues si se activan los dos transistores de una misma rama se produciría un cortocircuito entre Vcc y

GND y si se activan los transistores de ramas distintas pero no son los opuestos, la diferencia de potencial en la carga es 0 V, por lo que no funcionaría.

El control de estos transistores se realiza con las entradas IN1, IN2, IN3 e IN4. Las entradas ENA y ENB, que controlan la velocidad de giro, son conectadas a una señal PWM (explicado en el apartado "Señal PWM") o a unos jumpers que establecen la tensión de estas entradas en 5 V, haciendo que los motores vayan a la velocidad máxima.

6.2. Control

6.2.1. Hardware

Para la realización de este proyecto se ha elegido para el control y la programación del robot la placa Arduino Nano. El motivo de la elección ha sido su bajo coste, su versatilidad, la facilidad para encontrar información y la posesión antes de iniciar el proyecto de una de estas placas.

6.2.1.1. Arduino

Arduino es una plataforma de electrónica de código abierto que se basa en hardware y software libre, es decir, que cualquier persona puede encontrar la información necesaria, como pueden ser el diseño de la PCB (Printed Circuit Board) o el esquema eléctrico, para replicar la placa y que su código es accesible para cualquiera ya sea para usarlo o modificarlo [27]. Además, ofrece la plataforma Arduino IDE (Integrated Develpment Enviroment), un entorno de programación gratuito en el que cualquier persona puede realizar su propio proyecto con las placas de Arduino mediante un lenguaje de programación adaptado de C. En Figura 20 se muestra el logo de Arduino.

Figura 20: Logo de Arduino. Fuente: [28]

Las placas de Arduino cuentan con todo lo necesario para conectar periféricos (componentes externos a la placa) a las entradas y a las salidas de un microcontrolador. En este caso, las placas de Arduino se basan en un microcontrolador de la serie ATmegaXXX, de la empresa ATMEL, los cuales presentan una arquitectura Harvard: una configuración en la que los datos y las instrucciones de un programa se pueden abordar independientemente debido a que están en celdas separadas de memoria [29].

6.2.1.2. Tipos de Arduino

Entre las placas de Arduino existe una gran variedad de ellas entre las cuales se puede elegir en función de las necesidades del proyecto que se vaya a realizar. Algunas de las placas son:

- Arduino UNO: Está basado en el ATmega328P y cuenta con 14 entradas y salidas digitales (6 de estas salidas se pueden usar como salida PWM) y con 6 entradas analógicas. La frecuencia de reloj es de 16 MHz y se puede alimentar con tensiones entre 6 V y 20 V [30]. En Figura 21 se muestra esta placa.

Figura 21: Arduino UNO. Fuente: [30]

- Arduino Leonardo: Está basado en el ATmega32u4 y cuenta con 20 pines de entrada y salida digital, de los cuales 7 pueden ser usados como salida PWM y 12 como entrada analógica. La frecuencia de reloj es de 16 MHz y puede ser alimentado con tensiones entre 6 V y 20 V [31]. En Figura 22 se muestra esta placa.

Figura 22: Arduino Leonardo. Fuente: [31]

- Arduino Nano: Está basado en ATmega328 y cuenta con 14 pines de entrada y salida digital, entre los cuales 6 pueden actuar como salida PWM, y con 8 pines de entrada

analógica. La frecuencia de reloj es de 16 MHz y puede ser alimentado con una tensión entre 7 V y 12 V [32]. Sus prestaciones son parecidas a las de Arduino UNO, pero su tamaño es más reducido. En Figura 23 se muestra esta placa.

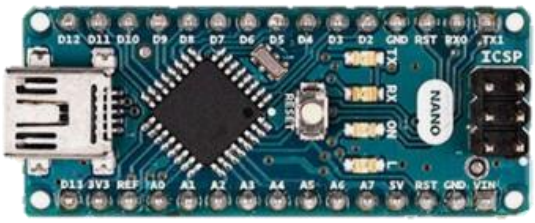


Figura 23: Arduino Nano. Fuente: [32]

- Arduino Micro: Está basado en ATmega32u4 y cuenta con 20 pines de entrada y salida digital (7 de estos pines pueden ser usados como salida PWM y 12, como entrada analógica). La frecuencia de reloj es de 16 MHz y puede ser alimentado con tensiones entre 6 V y 9 V [33]. En Figura 24 se muestra esta placa.

Figura 24: Arduino Micro. Fuente: [33]

- Arduino Yun: Está basado en ATmega32u4 y el Atheros AR9331 y cuenta con 20 pines de entrada y salida digital, de los cuales 7 pines pueden ser usadas como salida PWM y 12, como entrada analógica. La frecuencia de reloj es de 16 MHz y se alimenta con 5 V. Además, esta placa cuenta con soporte Ethernet, Wifi, USB y micro SD [34]. En Figura 25 se muestra esta placa.

Figura 25: Arduino Yun. Fuente: [34]

- Arduino Due: Está basado en el ATMEL SAM3X8E ARM Cortex-M3 CPU. Cuenta con 54 pines de entrada y salida digital (12 pueden ser usados como salida PWM), 12 pines de entrada analógica, 4 UARTs (puertos serie). La frecuencia de reloj es de 84 MHz y puede ser alimentado con tensiones entre 6 V y 16 V [35]. En Figura 26 se muestra esta placa.

Figura 26: Arduino Due. Fuente: [35]

6.2.1.3. Señal PWM

En el apartado anterior, "Tipos de Arduino", se ha mencionado la señal de salida PWM (modulación de ancho de pulso) y en este se va a explicar su funcionamiento y utilidad.

Esta salida presenta la forma de una onda cuadrada en la que se va variando el ciclo de trabajo, es decir, la relación entre el tiempo que la señal está a nivel alto (Ton) y el periodo de la señal (T) expresado en tanto por ciento, como se puede ver en Figura 27.

$$Ciclo\ de\ trabajo = \frac{T_{on}}{T} \cdot 100$$

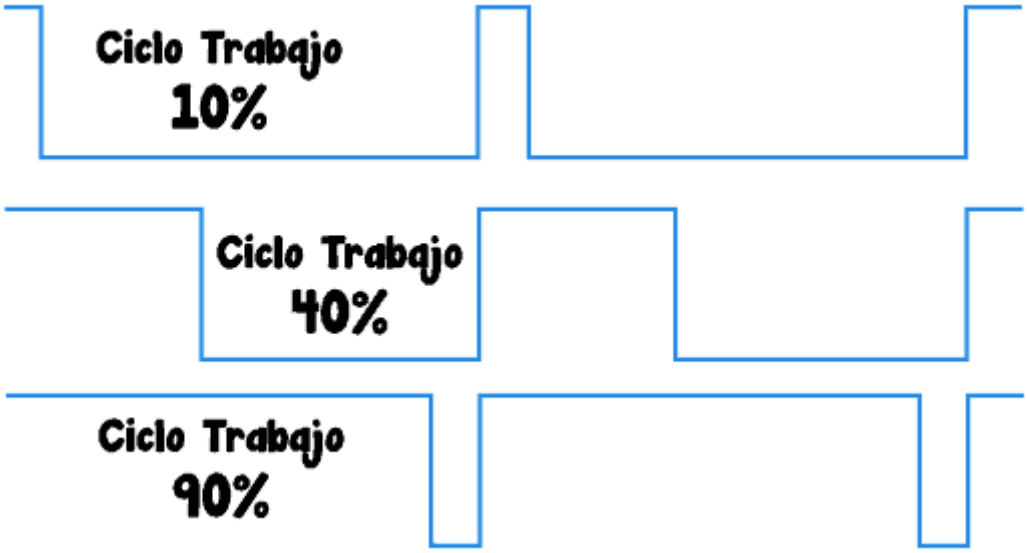


Figura 27: Señales PWM. Fuente: [36]

El objetivo de esto es emular, a partir de una salida digital, una señal analógica que varíe entre los valores máximo y mínimo de la señal digital. Al variar el ciclo de trabajo de la onda cuadrada el valor medio de la señal varía obteniendo valores medios más elevados cuanto mayor es el ciclo de trabajo y viceversa.

$$V_{medio} = \frac{V_{max} \cdot T_{on} + V_{min} \cdot (T - T_{on})}{T}$$

Algunos ejemplos de las aplicaciones de esta técnica usando Arduino son la variación de la intensidad de brillo de una bombilla LED, la variación de la velocidad de un motor de corriente continua, la variación del volumen de un altavoz o un zumbador, etc.

6.2.2. Módulo bluetooth SPP-C

Tal y como se ha mencionado en el apartado de "Estudio de necesidades", será necesario comunicar el robot con un dispositivo móvil. En este caso, se va a realizar la comunicación inalámbrica mediante un dispositivo bluetooth, el SPP-C (Figura 28).

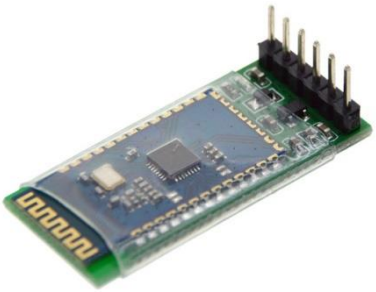


Figura 28: SPP-C. Fuente: [37]

Este módulo trabaja en un rango de frecuencias entre 2,4 GHz y 2,48 GHz, puede ser alimentado con una tensión entre 3,3 V y 5,6 V y la comunicación se realiza mediante puerto serie [37].

La comunicación serie consiste en la transmisión de datos de forma secuencial, de manera que el bit menos significativo es enviado al receptor en primer lugar y se van sucediendo los siguientes bits del dato. Para esto, es necesario temporizar la transmisión serie, identificando así los datos que se envían correctamente. Además, hay una serie de parámetros necesarios para establecer esta comunicación como son los bits por carácter, los bits por segundo, la velocidad en baudios, la paridad y los bits de inicio, detención y marca [38].

En este caso, tanto el módulo BT como el microcontrolador pueden enviar y recibir datos de forma simultánea, por lo que hay dos canales: el canal de recepción de datos (Rx) y el canal de transmisión de datos (Tx) en cada uno de los dispositivos. Para lograr una comunicación correcta, el canal Tx de un dispositivo debe ir conectado al canal Rx del otro, como se muestra en Figura 29.

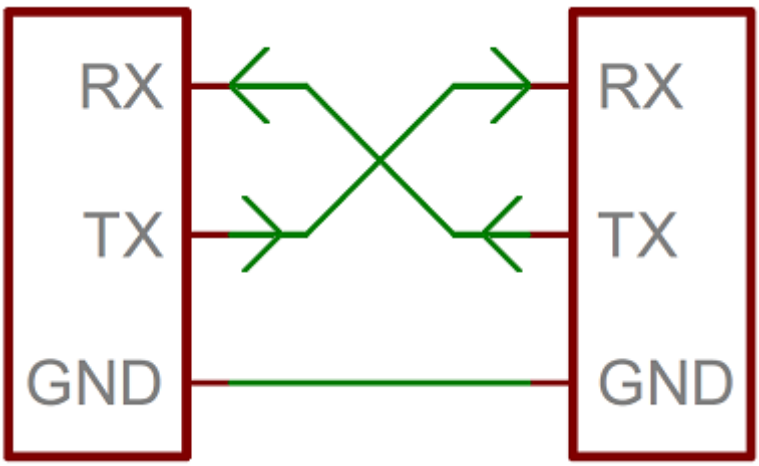


Figura 29: Comunicación serie full-duplex. Fuente: [39]

6.2.3. Sensor de inclinación MPU-6050

Para medir la inclinación del robot se ha elegido un MPU-6050, una IMU que combina acelerómetro y giróscopo. Gracias a este componente, el microcontrolador será capaz de conocer el ángulo de inclinación del robot en todo momento, siendo capaz de corregir la posición por medio de los actuadores.

6.2.3.1. IMU

Las unidades de medida inercial son dispositivos empleados para medir velocidad, orientación, aceleración y fuerza gravitacional. Estos dispositivos combinan diferentes sensores para conseguirlo [40]:

- Acelerómetros: Son usados para medir la aceleración inercial.
- Giroscopios: Son usados para medir la velocidad angular.
- Magnetómetros: Son usados para medir la dirección del rumbo magnético, permitiendo la mejora de la lectura del giroscopio.

En el caso de este proyecto, el IMU utilizado no cuenta con un magnetómetro, pero sí con un acelerómetro y un giroscopio con 3 grados de libertad cada uno. Acelerómetro y giroscopio presentan unas características distintas como puede ser la magnitud física que miden, pero ambos pueden proporcionar la inclinación. También presentan limitaciones distintas, pero mediante la combinación de los sensores se pueden solventar. Las limitaciones serían [41]:

- Los acelerómetros se ven influenciados por el movimiento del sensor y el ruido, por lo que no son fiables a corto plazo.
- Los giroscopios miden la velocidad angular y cuando se obtiene el ángulo de inclinación integrando la medición realizada se acumulan errores y ruido, generando una deriva, por lo que no son fiables a medio o largo plazo.

Mediante la combinación de estos sensores se pueden obtener mediciones más precisas que usándolos por separado. Para combinar dichas mediciones existen varios métodos que se pueden utilizar:

- Filtro de Kalman: Este filtro fue desarrollado por Rudolf E. Kalman en el año 1960. En este método se estiman los estados futuros de un sistema usando las medidas tomadas en el tiempo, incluyendo tanto ruido como imprecisiones, comparándolos luego con la medición [41][42]. Con este filtro se pueden obtener medidas más precisas, pero es un método complejo y costoso.
- Filtro complementario: Es un filtro sencillo que se comporta como un filtro paso alto para el giroscopio y como un filtro paso bajo para el acelerómetro, por lo que a corto plazo tiene más influencia en la medición el giroscopio y a largo plazo, el acelerómetro [41].

Con los datos filtrados, el sistema será capaz de conocer su orientación, representada como tres rotaciones ortogonales conocidas como roll, pictch y yaw en torno a los ejes X, Y y Z respectivamente (Figura 30).

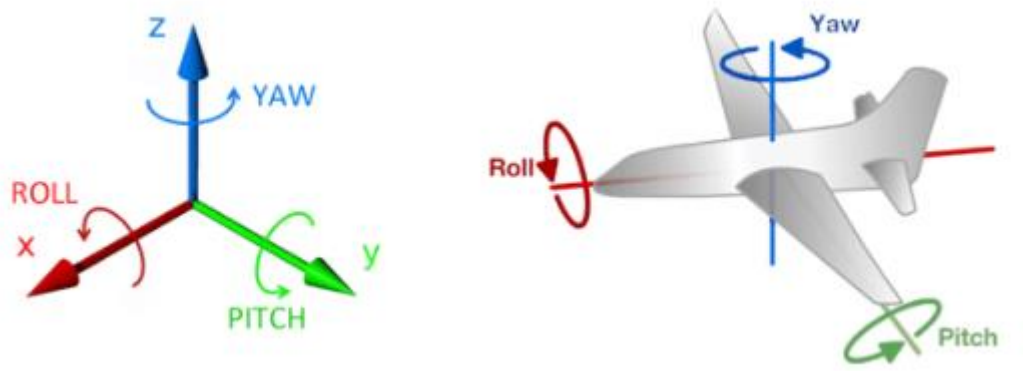


Figura 30: Roll, pitch y yaw. Fuente: [41]

6.2.3.2. Giroscopio

Como se ha mencionado en "Sensor de inclinación MPU-6050", el sensor MPU-6050 cuenta con un giroscopio, por lo que se en este apartado se va a explicar cómo funciona.

Un giroscopio permite medir el ángulo de giro de un cuerpo. Existen de muchos tipos, pero los más utilizados en los MEMS (sistemas micro-electro-mecánicos, una tecnología que logra micro-componentes que pueden servir como microsensores o microactuadores [43]) basan su funcionamiento en el efecto Coriolis, recibiendo el nombre de giroscopios vibratorios de efecto Coriolis (CVG) [44].

La fuerza de Coriolis se da sobre un cuerpo en movimiento en un sistema en rotación. El principio de funcionamiento de un CVG es que, aunque un objeto vibratorio esté rotando, este debe vibrar en el mismo plano (Figura 31). Esto provoca que aparezca dicha fuerza de Coriolis a partir de la cual se puede determinar la rotación del cuerpo.

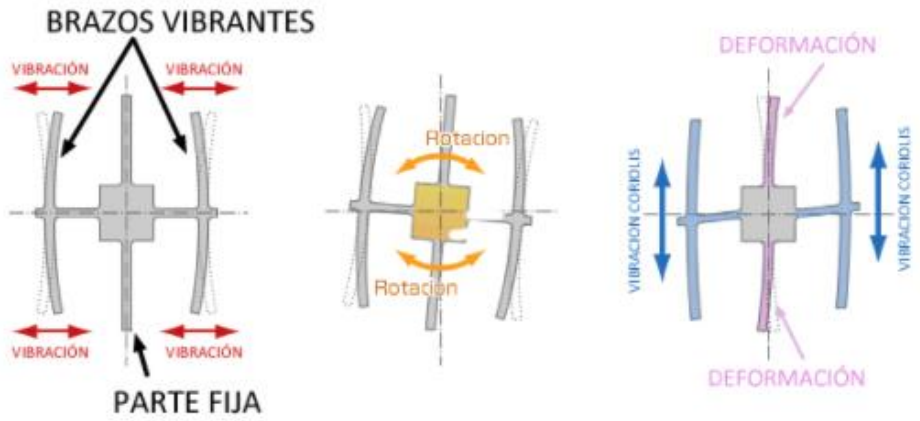


Figura 31: Funcionamiento del giroscopio. Fuente: [44]

En los MEMS, ciertas partes del cuerpo se someten a vibración por resonancia y la fuerza de Coriolis las deforma, lo cual puede ser medido mediante la variación de la capacitancia del sistema. De esta manera, se puede medir el efecto de la fuerza de Coriolis.

Debido a que se emplea este método, a partir de la fuerza de Coriolis lo que se obtiene es la velocidad angular del objeto, no el ángulo que ha girado, por lo que es necesario integrar el resultado obtenido con el fin de obtener el ángulo de giro:

$$\overrightarrow{F_c} = -2m(\vec{\omega} x \vec{v})$$

$$\omega = \frac{d\theta}{dt}$$

$$\theta = \omega \cdot \Delta t$$

Siendo:

- $\overrightarrow{F_c}$: Fuerza de Coriolis
- m: Masa del cuerpo
- $\vec{\omega}$: Velocidad angular del sistema de rotación
- $\vec{v}$: Velocidad del cuerpo en el sistema de rotación
- θ: Ángulo girado por el cuerpo
- t: Tiempo de giro

Una vez conocido el ángulo que ha girado el cuerpo, para conocer la orientación del objeto hay que conocer el ángulo en el que se encontraba antes de girar. Por tanto:

$$\theta = \theta_{anterior} + \omega \cdot \Delta t$$

Como se mencionó en el apartado anterior, "IMU", al integrar la medida se acumulan errores de medición y ruido generando la deriva. Sin embargo, estos sensores proporcionan una respuesta rápida y precisa en tiempos cortos, además de responder bien a cambios bruscos.

6.2.3.3. Acelerómetro

En este apartado se va a explicar el principio de funcionamiento de un acelerómetro, el otro sensor que existe en el MPU-6050.

Los acelerómetros miden la aceleración a la que se ven sometidos. A la hora de fabricar uno, se podría construir un sensor compuesto por un cuerpo sólido y en su interior suspender una masa sujeta al cuerpo a través de unos muelles (Figura 32) [45].

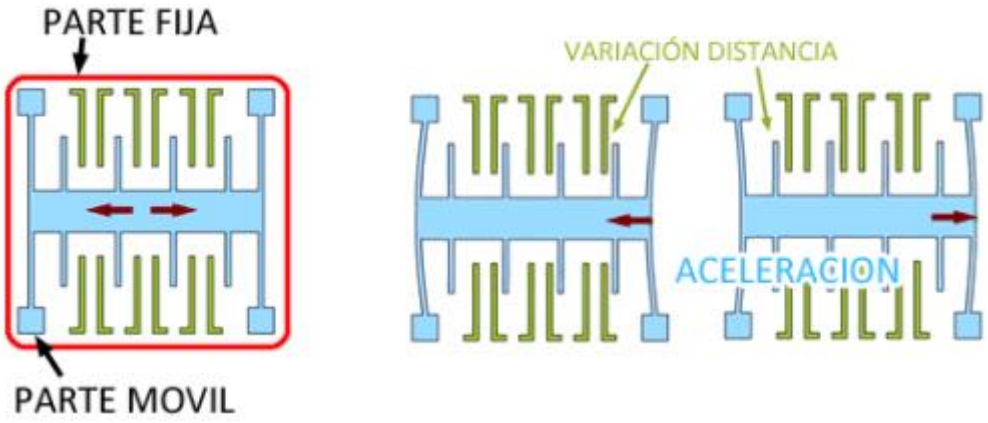


Figura 32: Funcionamiento del acelerómetro. Fuente: [45]

Cuando hay aceleración, la masa suspendida ejerce fuerza sobre los muelles variando su posición dentro del sensor. Este desplazamiento se puede medir para determinar la aceleración, ya que es proporcional, y es constante si la aceleración también se mantiene constante.

La orientación del sensor se puede conocer gracias a que el acelerómetro registra la aceleración de la gravedad y gracias a que este mide en tres ejes, mediante operaciones trigonométricas es posible conocer el ángulo de inclinación. En Figura 33 y las siguientes ecuaciones se explica cómo se obtendría el ángulo de inclinación de un objeto 2D a partir de la aceleración:

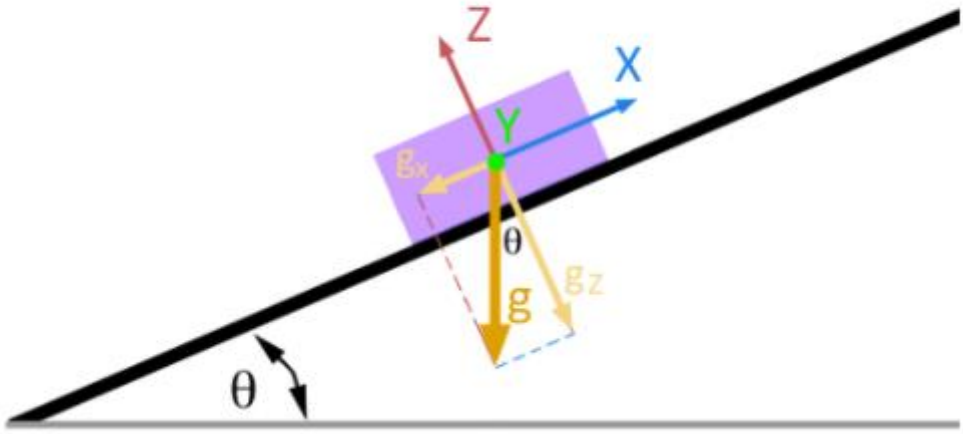


Figura 33: Esquema 2D de un cuerpo afectado por la gravedad en un plano inclinado. Fuente: [45]

$$g_z = -g \cdot cos(\theta)$$

$$g_x = -g \cdot \sin(\theta)$$

$$\frac{g_x}{g_z} = \frac{sen(\theta)}{\cos(\theta)} = \tan(\theta)$$

$$\theta = \arctan\left(\frac{g_x}{g_z}\right)$$

Siendo:

- g: Módulo de la aceleración de la gravedad
- g_x: Valor de la aceleración de la gravedad en el eje x
- g_z: Valor de la aceleración de la gravedad en el eje z
- θ: Ángulo de inclinación

De la misma forma, en un modelo 3D como el de Figura 34:

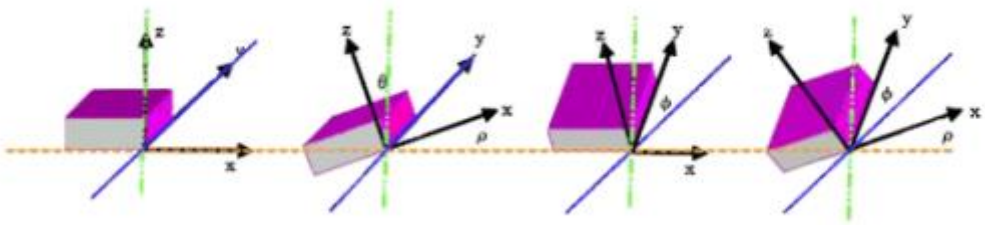


Figura 34: Esquema 3D de un cuerpo afectado por una aceleración en un plano inclinado. Fuente: [45]

$$\theta_x = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right)$$

$$\theta_y = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right)$$

$$\theta_z = \arctan\left(\frac{\sqrt{A_y^2 + A_x^2}}{A_z}\right)$$

Siendo θ_x, θ_y y θz las inclinaciones del cuerpo en los tres ejes y A_x, A_y y A_z, las aceleraciones.

Como se mencionó en el apartado "IMU", estos sensores son muy sensibles a las vibraciones, por lo que presentan mucho ruido de alta frecuencia, pero no presenta una deriva.

6.2.3.4. Filtro complementario

Para combinar las mediciones de los dos sensores, es necesario el uso de un filtro. En este apartado se va a explicar cómo se realiza un filtro complementario.

Este filtro es una simplificación del filtro de Kalman, aunque no tiene en cuenta el análisis estadístico [41]. En este filtro, la medida obtenida con el acelerómetro pasa por un filtro paso bajo, deshaciéndose de los ruidos de alta frecuencia; y la medida del giroscopio, por un paso alto quitando así la deriva, tal y como se muestra en Figura 35, Figura 36 y Figura 37:

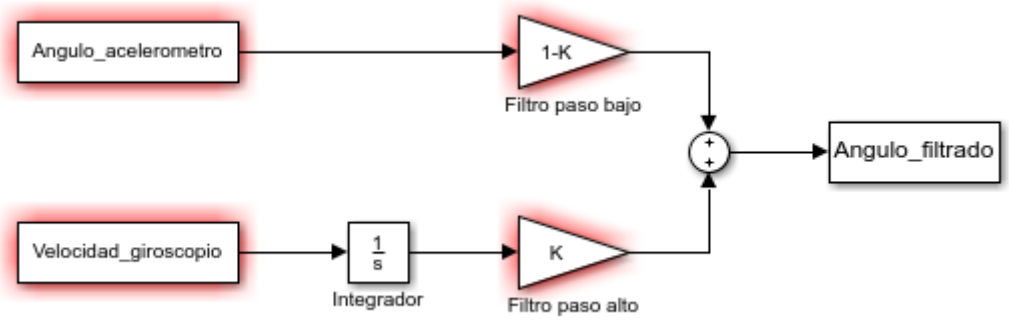


Figura 35: Diagrama de bloques del filtro complementario

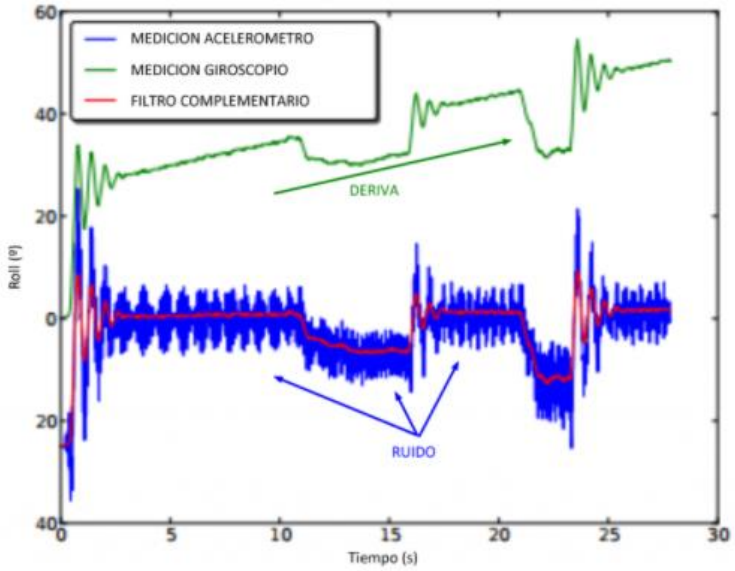


Figura 36: Medidas individuales y filtrada. Fuente: [41]

Figura 37: Medidas individuales y filtrada del MPU-6050 para K = 0.9

Para conseguir esto, hay que aplicar la siguiente ecuación:

$$\theta = K \cdot \left(\theta_{anterior} + \omega_{giroscopio} \cdot \Delta t\right) + (1 - K) \cdot \theta_{acelerómetro}$$

Siendo:

- θ: Ángulo filtrado
- $\omega_{giroscopio}$: Velocidad angular medida por el giroscopio
- t: Tiempo
- $\theta_{acelerómetro}$: Ángulo obtenido con el acelerómetro
- K: Constante cuyo valor está entre 0 y 1

6.2.3.5. MPU-6050

En este apartado se va a explicar cómo funciona el componente (Figura 38), qué pines se van a utilizar y para qué sirven.

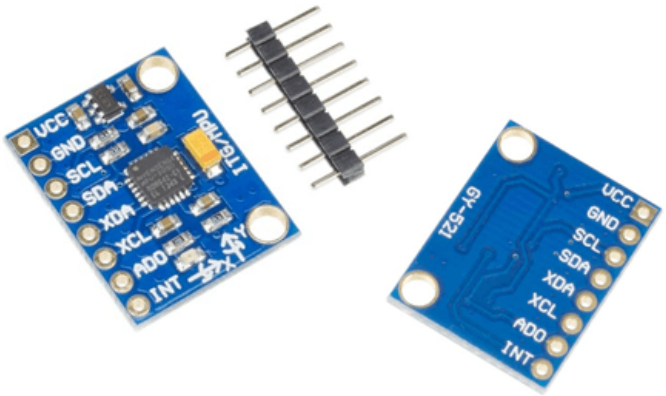


Figura 38: MPU-6050. Fuente: [14]

Este sensor puede ser alimentado con 5 V y tiene una serie de conversores analógicos digitales (ADC) que pasarán el valor analógico medido a bits para entregar la información al

microcontrolador a través de comunicación serial. Estos ADC son de 16 bits y el rango de valores que puede devolver el dispositivo se puede ajustar, aunque se van a utilizar los rangos que vienen por defecto: ±2 g (g = 9,81 m/s²) para el acelerómetro y ±250°/s para el giroscopio [14]. Con esos valores límite y el número de bits de los ADC, se puede pasar dentro del código del valor digital a un valor analógico para obtener la inclinación.

En el caso del acelerómetro, no es necesario realizar ninguna conversión dado que para obtener la inclinación se está realizando una relación entre varias variables. Dado que a todas las variables se les estará aplicando el mismo factor de conversión, el resultado será el mismo convirtiendo la aceleración a un valor analógico que usando su valor digital.

Siendo el factor de conversión de digital a analógico X y aplicando la fórmula que se ha visto en el apartado "Acelerómetro":

$$\theta_x = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right)$$

Se prueba que se puede obtener el ángulo de la misma manera usando los valores en forma de bits que proporcione el sensor:

$$\theta_x = \arctan\left(\frac{A_{xDigital} \cdot X}{\sqrt{\left(A_{yDigital} \cdot X\right)^2 + \left(A_{zDigital} \cdot X\right)^2}}\right)$$

$$\theta_x = \arctan\left(\frac{A_{xDigital} \cdot X}{\sqrt{A_{yDigital}^2 \cdot X^2 + A_{zDigital}^2 \cdot X^2}}\right)$$

$$\theta_x = \arctan\left(\frac{A_{xDigital} \cdot X}{\sqrt{X^2 \cdot \left(A_{yDigital}^2 + A_{zDigital}^2\right)}}\right)$$

$$\theta_x = \arctan\left(\frac{A_{xDigital} \cdot X}{X \cdot \sqrt{A_{yDigital}^2 + A_{zDigital}^2}}\right)$$

$$\theta_x = \arctan\left(\frac{A_{xDigital}}{\sqrt{A_{yDigital}^2 + A_{zDigital}^2}}\right)$$

Por otro lado, para obtener el ángulo a partir de la medida del giroscopio, si será necesario obtener el valor de la velocidad a partir de lo que devuelva el MPU-6050. El ángulo se obtendría de la siguiente forma:

- Para un valor de ω = 250°, el sensor devuelve $\omega_{MPU-6050}$ = 32767 y para ω = -250°, $\omega_{MPU-6050}$ = -32768.
- El factor de conversión es: $X = \frac{\Delta\omega}{\Delta\omega_{MPU-6050}} = \frac{500}{65535} \approx 0{,}00763$ ó $Y = \frac{1}{X} \approx 131$.

Por tanto:

$$\theta = \theta_{anterior} + \frac{\omega_{MPU-6050}}{131} \cdot \Delta t$$

Con esta información, ya se puede obtener correctamente la medida en el código.

Los pines que se van a utilizar para este proyecto, además de los pines de alimentación VCC y GND, son los pines SCL y SDA. Estos pines son las líneas serie de tiempo (SCL, Serial Clock Line) y de datos (SDA, Serial Data Line) para establecer una comunicación conectando dichos pines con los propios pines SCL y SDA del microcontrolador.

El protocolo I2C es una forma de comunicación serie con dos elementos básicos: un maestro y un esclavo. El maestro controla el reloj, con el que se marcan los tiempos de lectura y escritura, e inicia y para la comunicación; mientras que el esclavo recibe o transmite información [46].

6.3. Alimentación

Para la alimentación del robot autobalanceado se van a utilizar cuatro baterías Li-Ion de 3,7 V y 2500 mAh. La elección de esta cantidad de baterías se debe a que los motores necesitan como máximo 6 V y el driver consume algo menos de 2 V, por lo que a estos les llegará una tensión algo menor a la de alimentación, además de consumir mucha corriente.

Por tanto, para conseguir una alimentación correcta y un suministro de corriente suficiente, se han utilizado cuatro baterías que se dispondrán como se muestra en Figura 39:

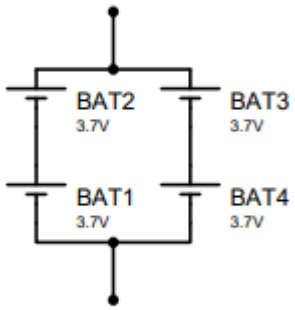


Figura 39: Disposición de las baterías en el robot

Como se puede observar, hay dos ramas en paralelo con dos baterías en serie cada una. El motivo de esta disposición es conseguir una tensión de alimentación de 7,4 V y una corriente de 5000 mAh, pues:

- Cuando varias baterías con la misma tensión son colocadas en paralelo, la tensión se mantiene igual y sus corrientes se suman. De esta forma, se han obtenido los 5000 mAh mencionados.
- Cuando varias baterías son colocadas en serie, la tensión entre sus bornes es la suma de la de las baterías. De esta forma se han obtenido los 7,4 V mencionados.

7. Diseño

En este apartado se aborda cómo se han realizado los diseños del circuito electrónico, de las piezas del robot, de la aplicación para el móvil y del código utilizado en el robot.

7.1. Diseño electrónico

Para la realización del diseño del esquema eléctrico del robot se han tenido en cuenta los componentes elegidos en el apartado "Descripción detallada de la solución".

Este apartado se va a estructurar en dos subapartados donde se explica detalladamente el circuito correspondiente a los actuadores y al control. En Figura 40 se muestra una imagen del circuito completo.

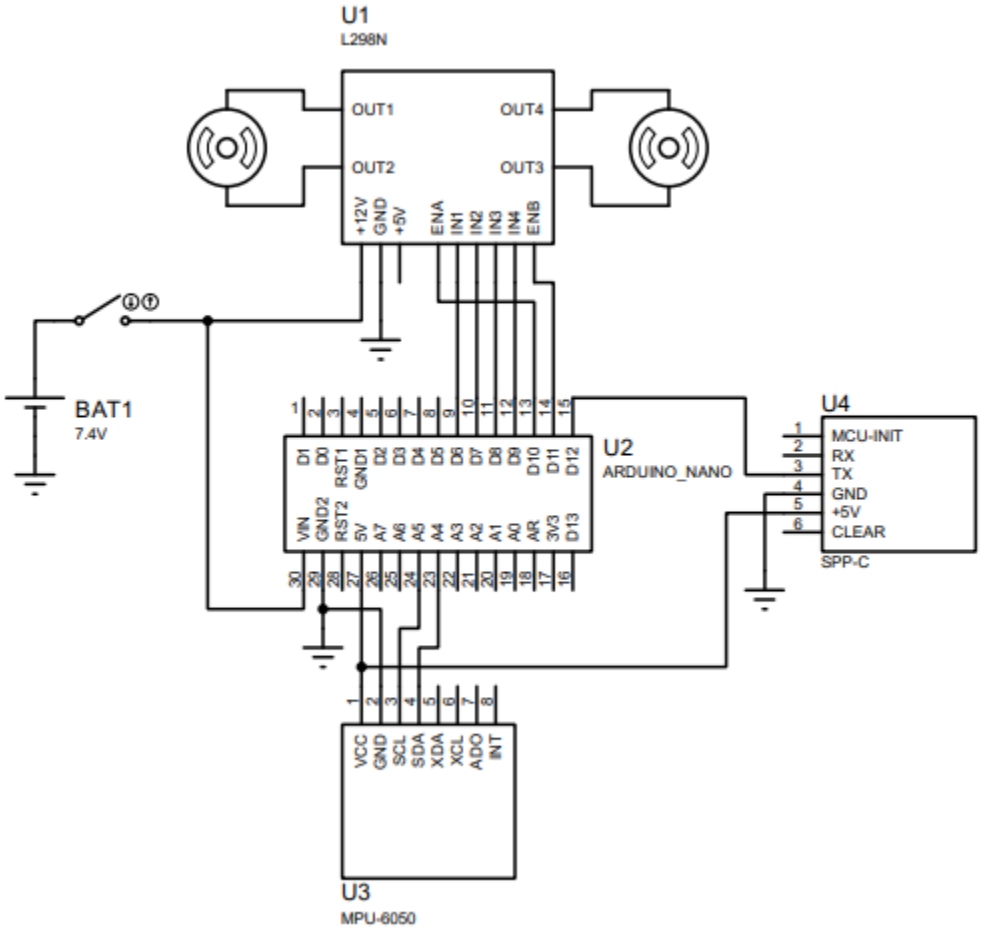


Figura 40: Circuito realizado.

Se puede observar en Figura 40 que se ha añadido un interruptor entre la alimentación y los componentes.

7.1.1. Actuadores

En Figura 41 se muestra el conexionado entre el Arduino Nano, el driver L298N y los motores de corriente continua.

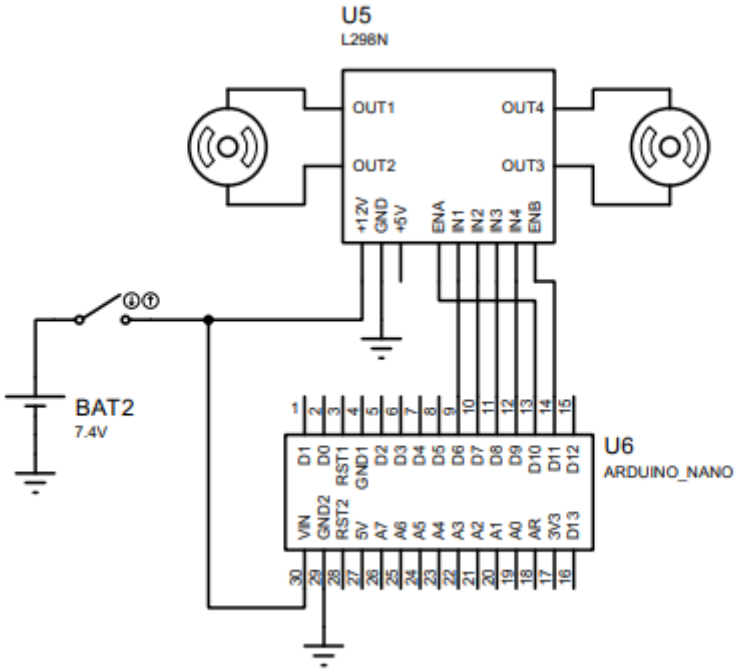


Figura 41: Conexionado del Arduino Nano y los actuadores.

La placa de Arduino Nano cuenta con 14 pines digitales (D0 – D13) de los cuales, 6 pueden funcionar como salida PWM (D3, D5, D6, D9, D10 y D11). Como se mencionó en "Funcionamiento", a las entradas ENA y ENB del driver se les va a proporcionar una entrada PWM para controlar la velocidad de los motores y al resto de entradas, valores digitales para controlar el sentido de giro. Teniendo esto en cuenta, se han elegido dos de los pines capaces de proporcionar una salida PWM, los pines D10 y D11, para conectarlos a ENA y ENB. Las otras cuatro entradas del driver se han conectado con cuatro pines digitales cualquiera del Arduino.

Dado que la alimentación es de 7,4 V, para que el circuito lógico del driver funcione hay que conectar el jumper regulador y como es una tensión dentro del rango que admite el Arduino Nano (ver "Tipos de Arduino"), se puede conectar a su entrada Vin.

En cuanto a la conexión de los motores con el driver, no importa cómo se conecten siempre y cuando los cables de un mismo motor estén conectados en los pares OUT1 y OUT2 o OUT3 y OUT4, pero hay que conseguir que, a la hora de mover el robot hacia delante o atrás, ambos vayan en el mismo sentido. Si esto no ocurre, se puede solucionar fácilmente cambiando la conexión de uno de los motores (el cable que antes estuviera en OUTX pasa a estar en OUTY y viceversa) o cambiando la conexión de sus entradas correspondientes (el cable que antes estuviera conectado en INX pasa a estar en INY y viceversa).

7.1.2. Control

En Figura 42 se muestra el conexionado entre el Arduino Nano, el SPP-C y el MPU-6050.

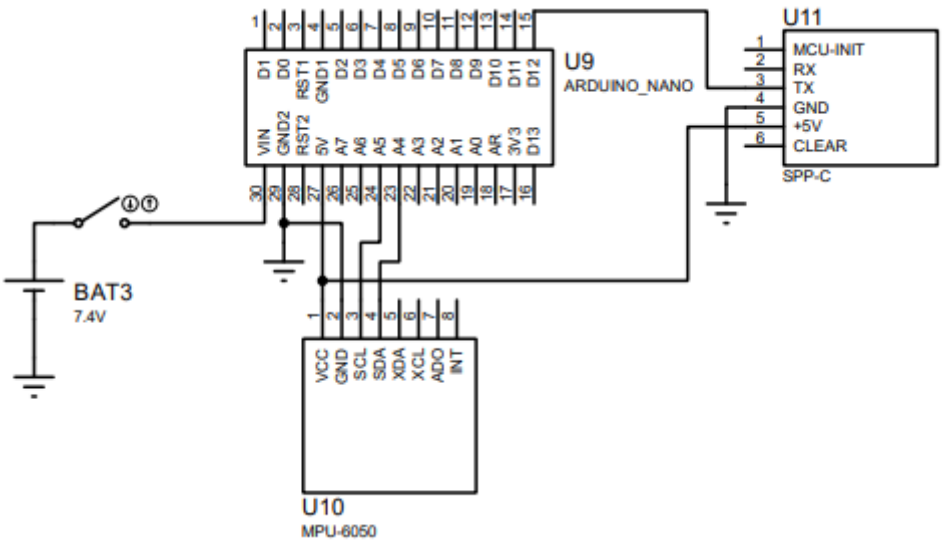


Figura 42: Conexionado del circuito de control.

Para la conexión entre estos componentes hay que tener en cuenta lo mencionado en "MPU-6050" y "Módulo Bluetooth SPP-C":

- El pin Rx del SPP-C debe estar conectado al pin Tx del Arduino Nano y viceversa.
- Los pines SCL y SDA del MPU-6050 deben ir conectados a los pines SCL y SDA del Arduino Nano.

En el caso del módulo Bluetooth, los pines Rx y Tx de Arduino Nano deben ser establecidos mediante programación. En este caso, se ha decidido que el pin Rx sea el D12 y el Tx, el D13. En Figura 42 se puede ver como el pin Rx del SPP-C no está conectado con el D13 del Arduino Nano. Esto se debe a que en este proyecto solo es necesario que el módulo Bluetooth transmita información, por lo que no es necesaria la conexión para la recepción de datos.

En cuanto al sensor, los pines SCL y SDA del Arduino Nano se corresponden con las entradas analógicas A5 y A4 respectivamente.

Los dos componentes necesitan ser alimentados con 5 V, por lo que para alimentarlos se han conectado sus entradas de 5 V al pin 5V del Arduino Nano, aunque también se podrían haber conectado a la salida +5V del driver L298N.

7.2. Diseño mecánico

En este apartado se va a relatar el proceso de diseño de las piezas que conforman el robot y los motivos por los que se han realizado esos diseños. Sin embargo, no se van a detallar las medidas de dichas piezas pues dicha información se encuentra en el "Documento Nº 2: Planos".

7.2.1. Diseño con SolidWorks

Para realizar el diseño se ha utilizado SolidWorks, un software de diseño CAD (diseño asistido por ordenador) 3D que permite modelar piezas y ensamblajes en 3D y realizar planos en 2D. En Figura 43 se muestra una imagen de su logo.

Figura 43: Logo de SolidWorks. Fuente: [48]

En cuanto a la historia de este programa, en el año 1993 Jon Hirschtick fundó SOLIDWORKS Corp. y en el año 1995 se lanzó una primera versión del programa de CAD 3D [47].

El manejo del programa no es complicado y para realizar este proyecto no ha sido necesario conocer todo lo que se puede hacer con el programa. A continuación, se van a comentar los diferentes tipos de operaciones que se han utilizado para el diseño de las piezas.

En Figura 44 se muestra como es la interfaz de usuario del programa y se pueden apreciar los distintos tipos de líneas, figuras y operaciones que se pueden realizar en un croquis.

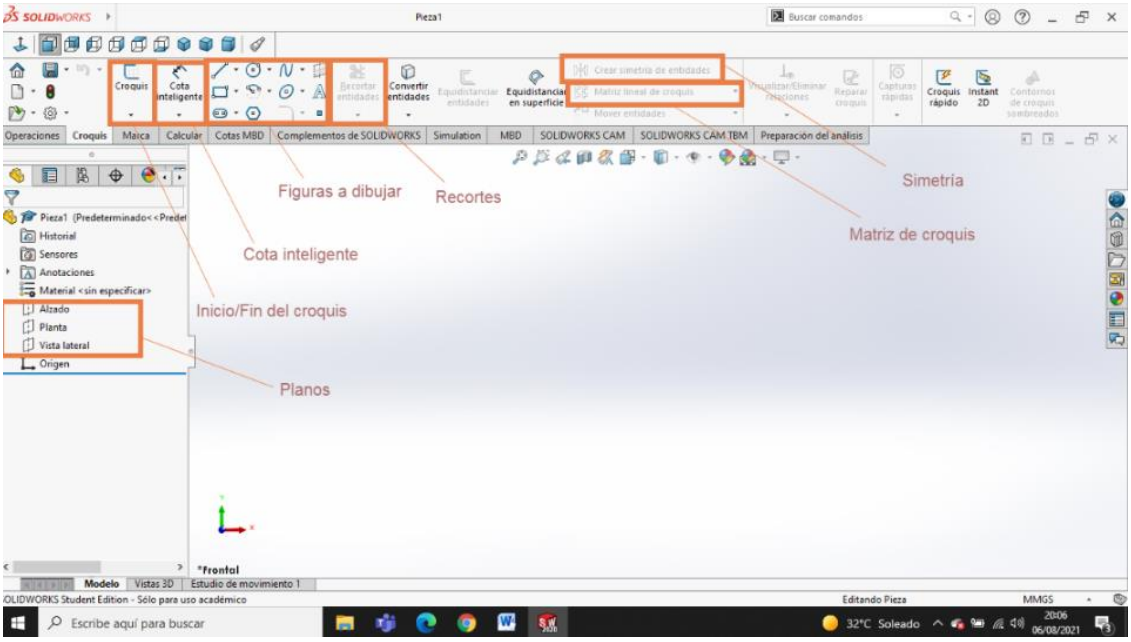


Figura 44: Interfaz de SolidWorks en la pestaña "Croquis" con los elementos importantes señalados

En Figura 44 está seleccionada la pestaña "Croquis" de la barra de herramientas, la cual es la sección donde se realiza el croquis 2D de una de las partes de la pieza. Para hacer esto hay que seleccionar uno de los planos o una de las caras de la pieza si aún se está diseñando e iniciar el croquis.

Una vez iniciado el croquis, hay que empezar a dibujar empleando las figuras que se proporcionan. Las que se han utilizado en este proyecto han sido líneas, líneas constructivas (sirven como referencia, pero no forman parte de la pieza), círculos, rectángulos y polígonos. Para establecer las medidas del croquis se debe utilizar la herramienta “Cota inteligente”.

Otras de las operaciones que se han necesitado han sido el “Recorte de entidades”, el cual permite eliminar partes de una figura que se intersectan con otras, y “Crear simetría de entidades” para simplificar el proceso de diseño cuando el croquis es simétrico.

Una vez realizado el diseño se finaliza el croquis y se realiza la operación que se desee. En Figura 45 se puede observar el croquis finalizado de una de las partes de la pieza “Acople para rueda 1” (plano “P-06”).

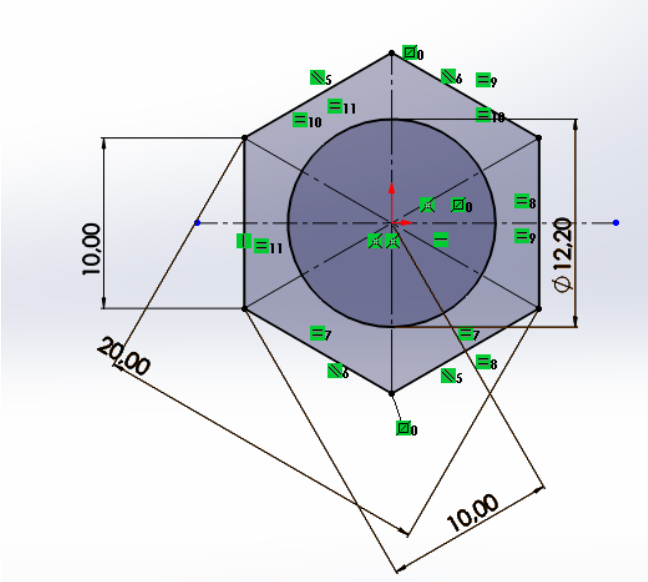


Figura 45: Croquis de una de las partes de una pieza

En cuanto a las operaciones, se han utilizado las que se destacan en Figura 46.

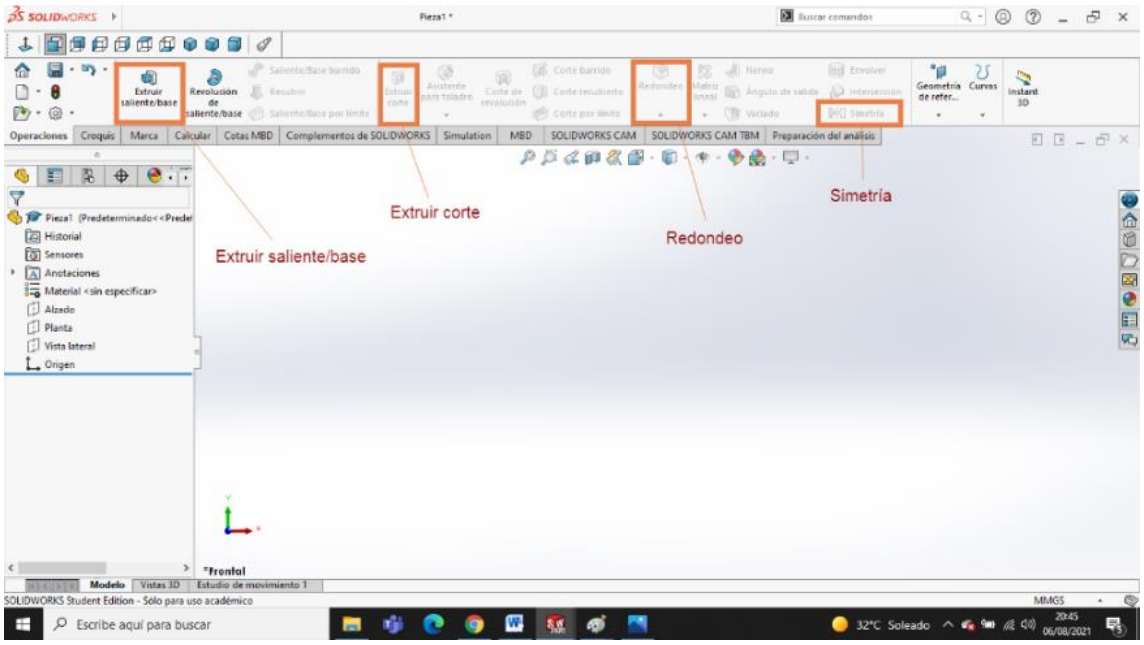


Figura 46: Interfaz de Solidworks en la pestaña "Operaciones" con las operaciones usadas señaladas

Para operar con los croquis diseñados, hay que ir a la pestaña “Operaciones”. Las operaciones que se han utilizado para este trabajo han sido “Extruir saliente/base”, la cual permite generar una pieza con la forma del croquis y con la longitud que se elija; “Extruir corte”, que permite cortar la forma del croquis en la pieza con la longitud que se elija; “Redondeo” para hacer las esquinas circulares determinando el radio; y “Simetría” para simplificar la creación de piezas simétricas.

En Figura 47 se muestra cómo queda el croquis de Figura 45 una vez se ha utilizado la operación “Extruir saliente/base”.

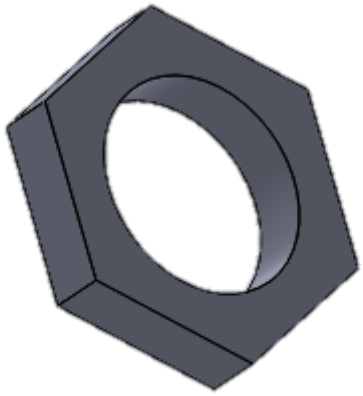


Figura 47: Parte de la pieza creada a partir del croquis de Figura 45

7.2.2. Piezas

En este apartado se van a mostrar las piezas que se han diseñado y se van a comentar los motivos para la realización de sus diseños.

Para el diseño de estas piezas se han tenido en cuenta las necesidades listadas en el apartado "Estudio de necesidades": diseño fácil de montar y desmontar y de pequeño tamaño. Además, hay que tener en cuenta los distintos componentes que se van a colocar en el robot. Hay que atender a las dimensiones de estos componentes, si necesitan cableado o como se pueden fijar en las piezas.

Los distintos componentes y sus características son las siguientes:

- Protoboard: Es una placa con agujeros conectados eléctricamente entre sí de froma interna siguiendo patrones de líneas. Este es el componente más ancho y largo que se va a colocar. En esta placa se conectan el Arduino Nano, el módulo bluetooth SPP-C y el sensor MPU-6050. Para fijarla al robot no se necesita nada, pues cuenta con un material adhesivo en su base.
- Arduino Nano: Este componente se debe conectar con el ordenador a través de un cable, por lo que es necesario hacer un agujero en uno de los laterales.
- Driver L298N: Este es el componente más alto, por lo que determinará la altura mínima entre las bases del robot. Este componente se conecta tanto al Arduino Nano como a los motores, por lo que se ha decidido colocarlo junto al microcontrolador, pues los motores estarán en un lugar fijo mientras que el resto de componentes se pueden colocar en distintos lugares. Este controlador de motores tiene cuatro agujeros para fijarlo al robot, por lo que en el diseño de las piezas se han añadido esos agujeros para poder atornillar el componente.
- Portapilas: Se van a utilizar dos portapilas, pues este accesorio permite colocar en un lugar fijo dos baterías conectadas en paralelo pues, como se ha comentado en el apartado "Alimentación", se van a usar cuatro baterías. Cuenta con dos agujeros en su parte central, por lo que se han realizado dos agujeros en las piezas diseñadas para poder atornillarlo.
- Motores: Ambos motores cuentan con dos agujeros en su parte frontal para ser atornillados, por lo que el soporte diseñado presenta dos agujeros, además del necesario para que el eje del motor atraviese la pieza.
- Ruedas: El agujero de las ruedas no coincide con el tamaño del eje de los motores, por lo que se debe diseñar alguna pieza que permite su acople. Para acoplarlo se ha decidido diseñar dos piezas, una que se acople en la cara interior de la rueda y la otra, en la exterior; siendo fijadas por varios tornillos.
- Interruptor: Cuenta con un saliente en su parte frontal, lo que permite se puede fijar si se diseña un agujero por el que pase su parte posterior, pero no la parte frontal.

7.2.2.1. Base con soportes para motor

Esta pieza se corresponde con el plano "P-01". En ella se acoplan los motores en la parte inferior y se deben colocar o las baterías o la electrónica. En Figura 48 se muestra la pieza.

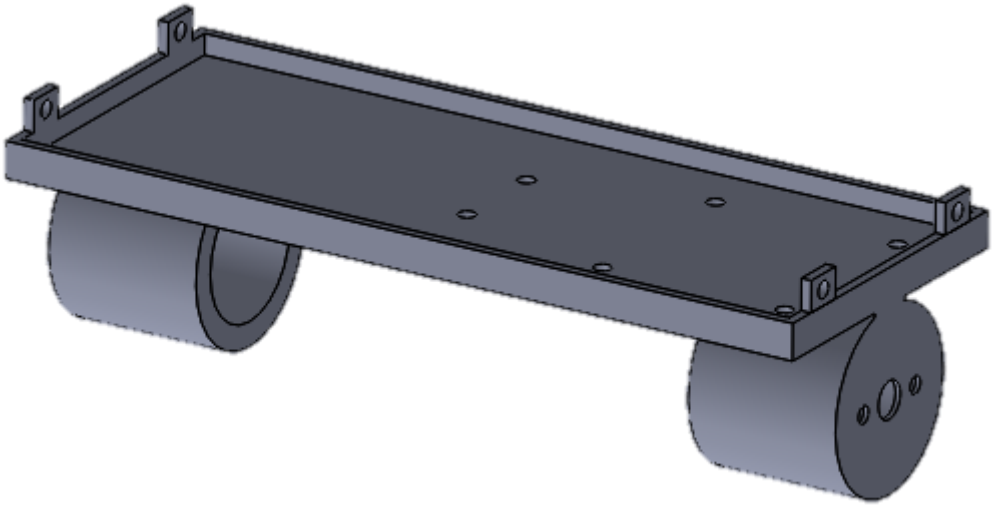


Figura 48: Base con soportes para motor

Las partes y decisiones de este diseño son los siguientes:

- En los soportes se encuentran dos agujeros pequeños en los que atornillar los motores y un agujero grande que permita que el eje salga de la pieza.
- Los soportes circulares de la parte inferior se han realizado para acoplar los motores y que permitan que estos no caigan en caso de que los tornillos se suelten.
- En los laterales de la pieza se encuentran unos salientes rectangulares con agujeros para atornillar la base a los laterales del robot (igual en todas las bases).
- En la base hay dos agujeros en el centro y cuatro en el lateral formando un cuadrado. Estos agujeros permiten diferentes configuraciones a la hora de colocar la electrónica en el robot. En una de ellas se utilizan os agujeros centrales, permitiendo atornillar el soporte para las baterías; mientras que en la otra se usan los laterales, atornillando el driver dejando el resto del espacio para la protoboard con los componentes electrónicos.
- El saliente rectangular en el borde de la base tiene como objetivo ayudar a mantener la protoboard en su sitio. Este saliente se encuentra en todas las bases que se han realizado.
- Las dimensiones de todas las bases se han establecido teniendo en cuenta la anchura de la protoboard y su longitud más la del driver.

7.2.2.2. Base con agujeros

Esta pieza se corresponde con el plano “P-02”. Esta pieza es la base en la que se coloca el segundo par de baterías. En Figura 49 se muestra la pieza.

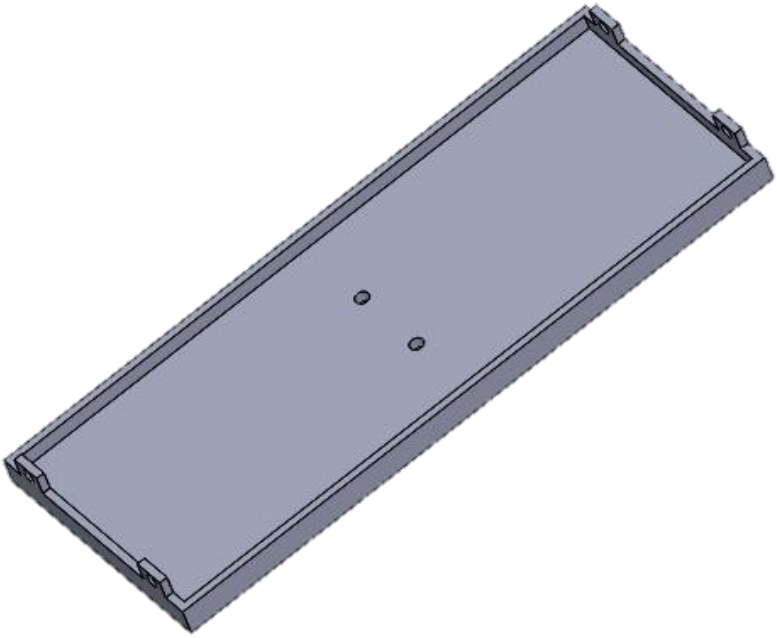


Figura 49: Base con agujeros

Esta pieza sería una versión simplificada de la "Base con soportes para motor" en la que se han eliminado los soportes y los agujeros correspondientes al driver.

7.2.2.3. Base con agujeros para Driver

Esta pieza se corresponde con el plano "P-03". En esta base se coloca lo que no se ha colocado en "Base con soportes para motor". En Figura 50 se muestra la pieza.

Figura 50: Base con agujeros para Driver

En este caso, se han mantenido los agujeros del driver para así poder colocar cualquiera de las cosas que no se hayan colocado en la base con soportes.

7.2.2.4. Lateral sin agujeros

Esta pieza se corresponde con el plano "P-04". Esta pieza es uno de los laterales del robot, la cual se puede observar en Figura 51.

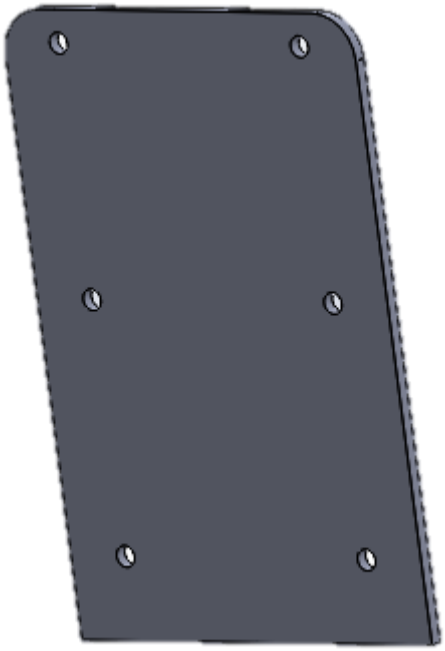


Figura 51: Lateral sin agujeros

Decisiones del diseño:

- La anchura es la misma que las de las bases.
- Los agujeros permiten atornillar las bases al lateral.
- La separación entre bases se ha establecido teniendo en cuenta la altura del driver, que es el componente más alto del robot, haciéndola mayor.
- El redondeo en las esquinas superiores se debe a razones estéticas.

7.2.2.5. Lateral con agujeros

Esta pieza se corresponde con el plano "P-05". Este es el otro lateral del robot, el cual cuenta con dos agujeros rectangulares que permiten acoplar un interruptor y realizar la conexión entre el Arduino y el ordenador. La pieza se puede observar en Figura 52.

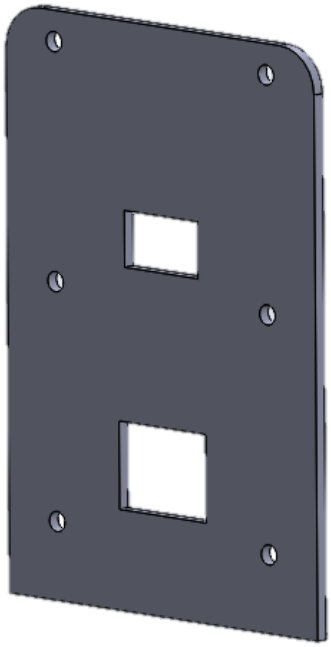


Figura 52: Lateral con agujeros

Decisiones del diseño:

- Las dimensiones de esta pieza deben ser iguales que las de "Lateral sin agujeros".
- El agujero superior se ha realizado para poder acoplar un interruptor, por lo que se han tenido en cuenta sus medidas para realizar dicho agujero.
- El agujero inferior permite la entrada del cable de conexión entre el Arduino y el ordenador en el caso de que se coloque en esa base. En el caso de que el Arduino se colocara en la parte superior del robot, el cable no se encontraría con ningún obstáculo.
- El agujero para el interruptor impide que el driver y la protoboard se puedan colocar en la parte central.
- El agujero para el cable obliga a colocar este lateral en el lado opuesto al que se encuentran los agujeros para el driver en "Base con soportes para motor" en el caso de colocar dicho driver en esa base.

7.2.2.6. Acople para rueda 1

Esta pieza se corresponde con el plano "P-06". Esta pieza se ha diseñado, junto con "Acople para rueda 2", para poder colocar la rueda en el eje del motor, pues su agujero es mucho mayor que dicho eje. La pieza se muestra en Figura 53.

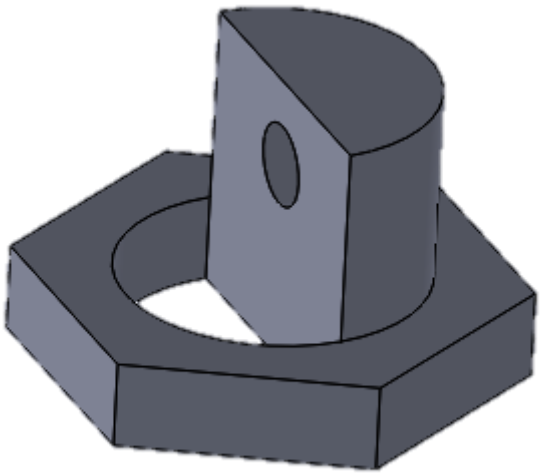


Figura 53: Acople para rueda 1

Las partes y decisiones de este diseño son los siguientes:

- La forma hexagonal de la base se debe a que en la parte interior de la rueda hay un hueco hexagonal, por lo que las dimensiones elegidas son las de ese hueco.
- El semicírculo tiene el mismo radio que el agujero que presenta la rueda para ser acoplada.
- El saliente se ha realizado para poder atornillar esta pieza con "Acople para rueda 2" mediante el agujero que tiene.
- La altura del saliente permite que haya un hueco entre las dos piezas que debe ocupar la rueda.

7.2.2.7. Acople para rueda 2

Esta pieza se corresponde con el plano "P-07". Como se mencionó anteriormente, esta pieza forma parte de un conjunto de dos piezas que permiten acoplar las ruedas al motor. En Figura 54 se muestra esta pieza.

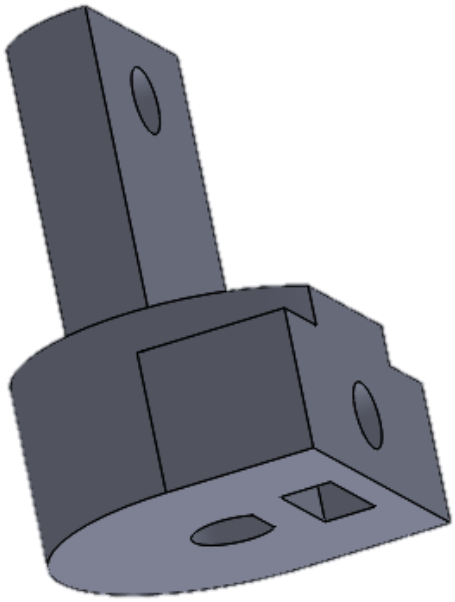


Figura 54: Acople para rueda 2

Las partes y decisiones de este diseño son los siguientes:

- El agujero en forma de D de la parte inferior permite el acoplamiento del eje del motor en la pieza.
- El agujero rectangular se ha realizado para colocar una tuerca en su interior.
- El agujero en la parte vertical de la base permite introducir un tornillo que aprieta el eje del motor para impedir que se salga de la pieza. La tuerca mencionada anteriormente impide que el tornillo se caiga.
- El saliente se ha colocado para introducirlo por el agujero que hay en la parte exterior de la rueda. Su radio es similar al del agujero y cabe por el semicírculo de "Acople para rueda 1".
- El agujero del saliente se ha colocado en un lugar en el que sea concéntrico con el del acople 1.

7.2.3. Ensamblaje

En este apartado se van a mostrar los ensamblajes realizados en SolidWorks para comprobar que todas las piezas se pueden acoplar y que no ha habido fallos en el diseño y una fotografía del robot ya montado con todos sus componentes.

7.2.3.1. Acople de las ruedas

En Figura 55 se muestra como queda el ensamblaje de las piezas diseñadas para acoplar las ruedas al eje del motor, donde se puede apreciar un espacio en el centro del ensamblaje, el cual es el lugar en el que se encontraría la llanta de la rueda, como se muestra en Figura 56 ya con la rueda puesta.

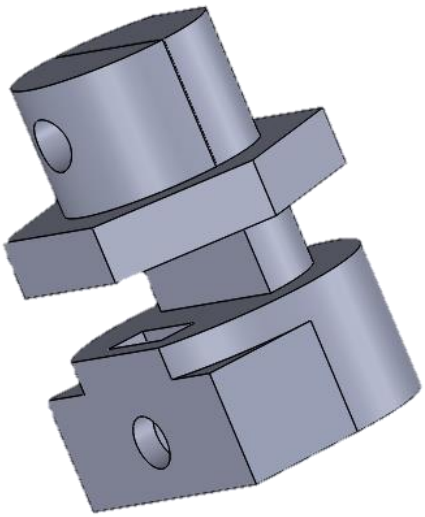


Figura 55: Ensamblaje de las piezas para acoplar las ruedas

Figura 56: Ruedas con las piezas de acople

El agujero circular superior mantiene las piezas juntas mediante un tornillo y los agujeros inferiores mantienen fijo el ensamblaje en el eje del motor.

7.2.3.2. Robot completo

En Figura 57 se muestra el ensamblaje completo del robot en SolidWorks y en Figura 58, una fotografía del robot montado y con todos los componentes colocados.

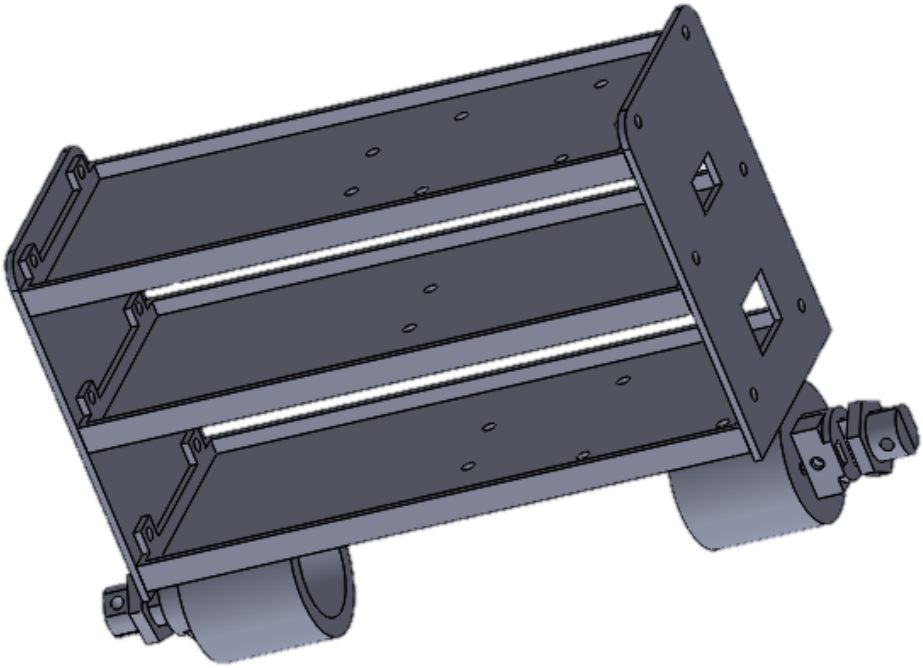


Figura 57: Ensamblaje del robot en SolidWorks

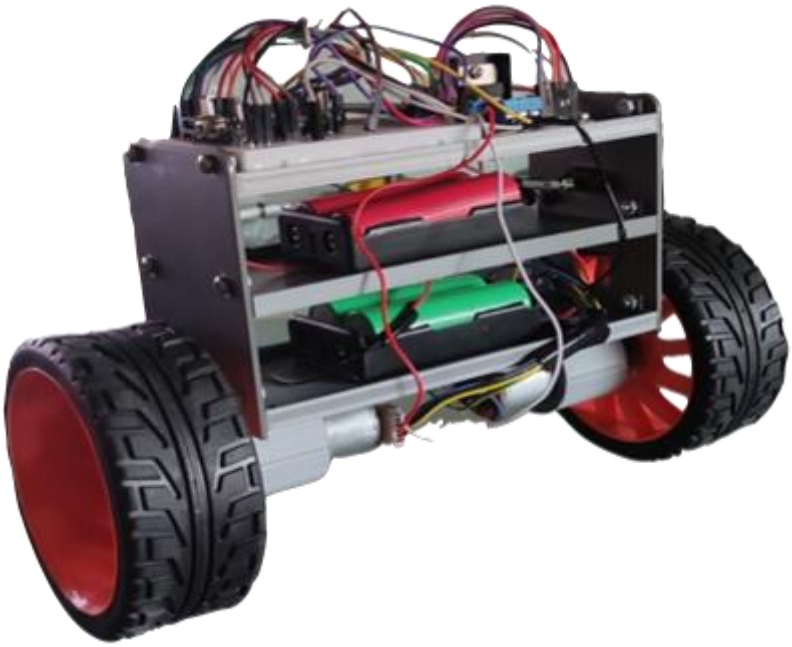


Figura 58: Robot

7.3. Programación de la app

El control del robot se ha realizado mediante una aplicación de teléfono móvil que permite la conexión con el robot mediante bluetooth. Para ello, se ha hecho una app mediante MIT App Inventor.

MIT App Inventor (ver logo en Figura 59) es un entorno de programación visual e intuitivo, desarrollado por Google Labs y lanzado en 2010, que permite crear aplicaciones de una forma simple, pues la programación se realiza mediante bloques [49][50].

Figura 59: Logo de App Inventor

El uso de esta página web es muy sencillo. Cuenta con dos ventanas: "Blocks" y "Designer" en las cuales se realiza la programación y el diseño de la interfaz de usuario respectivamente.

La aplicación creada es muy sencilla: conecta o desconecta el robot con el móvil y envía datos que se corresponden con el movimiento que debe hacer el robot.

7.3.1. Diseño de la interfaz de usuario

En esta ventana se colocan los diferentes elementos que va a tener la aplicación y se establecen las características de los diferentes elementos que se han colocado (color, texto, tamaño…). En Figura 60 se muestra una captura de pantalla de la ventana "Designer" con los elementos que se han utilizado señalados.

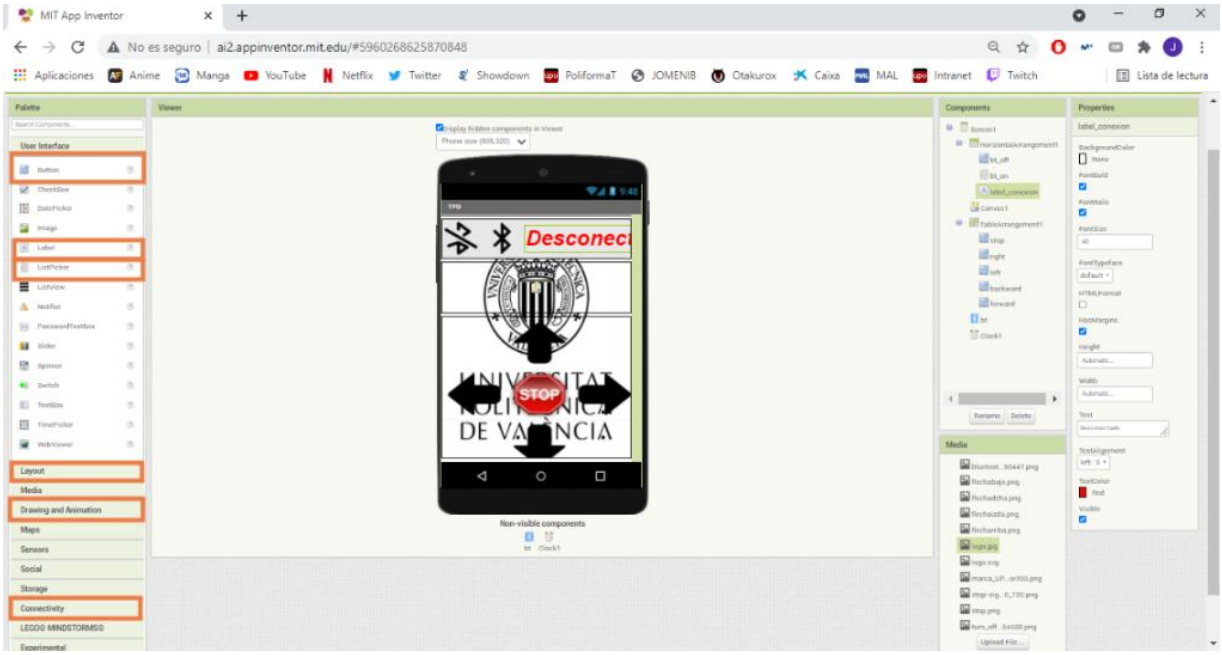


Figura 60: Captura de pantalla de App Inventor con los elementos utilizados señalados

Como se observa, en el centro de la pantalla ("Viewer") hay un dibujo de un móvil, el cual permite visualizar de una forma aproximada como queda la interfaz de usuario que se está realizando.

En la parte izquierda, "Palette", se encuentran los elementos que se pueden incorporar a la aplicación y se han señalado los que se han utilizado:

- "Button": Permite añadir botones. Los botones utilizados son los de movimiento del robot y el botón de desconexión del bluetooth.
- "Label": Permite mostrar por pantalla un texto.
- "ListPicker": Al seleccionarlo aparece un listado con diferentes elementos. Se ha utilizado para la conexión bluetooth, pues al pulsar aparecerán los dispositivos disponibles para conectarse.
- "Layout": Dentro de esta pestaña se han utilizado las distribuciones "HorzontalArrangement" para colocar los botones de conexión y desconexión del bluetooth en horizontal y la etiqueta con el texto y "TableArrangement" para colocar los botones que permiten mover el robot en una matriz 3x3.
- "Drawing and Animation": En esta pestaña se ha usado el elemento "Canvas" para separar los botones. Esto se ha incluido por un motivo estético.
- "Connectivity": En esta pestaña se ha incluido la función "BluetoothClient", el cual permite utilizar bluetooth en la aplicación.

A todos estos elementos se les han establecido diferentes propiedades, en la sección "Properties" en la parte derecha de la imagen. Con esto se ha determinado el tamaño de los elementos, el texto o la imagen que presentan los elementos, si son visibles en la app, el color del fondo o del texto, etc.

En Figura 61 se muestra una captura de pantalla de la aplicación en un teléfono móvil.

Figura 61: Captura de pantalla de la aplicación

7.3.2. Programación de la aplicación

Para programar la aplicación hay que situarse en la ventana “Blocks”. En esta ventana hay que colocar los bloques en función de lo que se quiera realizar. En Figura 62 y Figura 63 se muestra el código realizado mediante bloques.

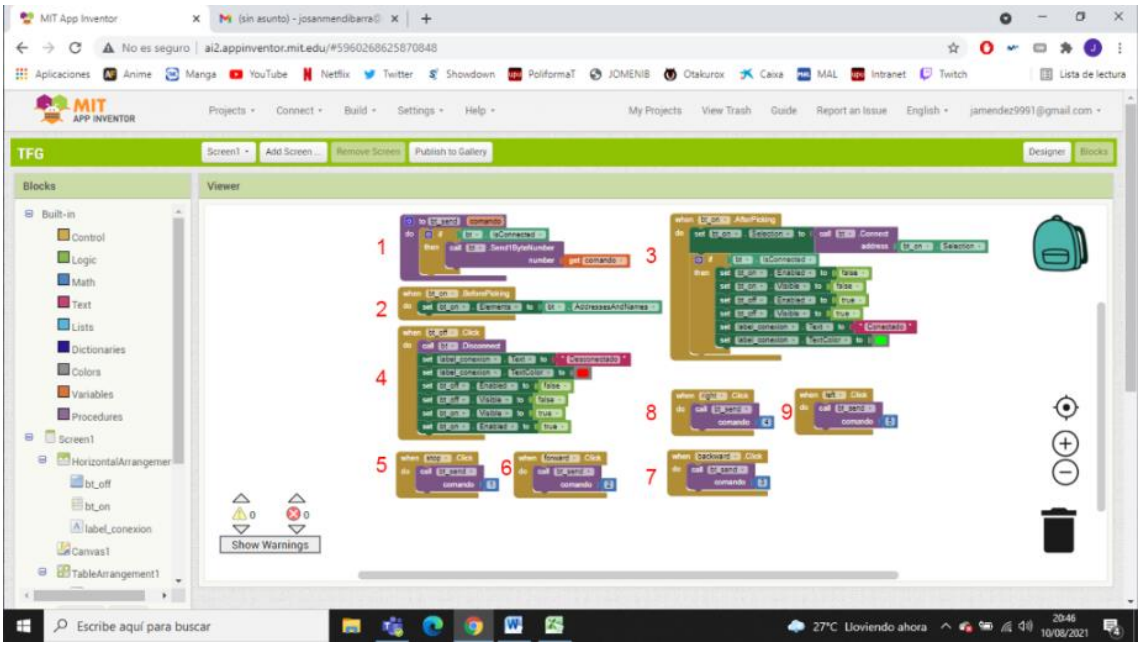


Figura 62: Captura de pantalla de la ventana "Blocks" de la aplicación

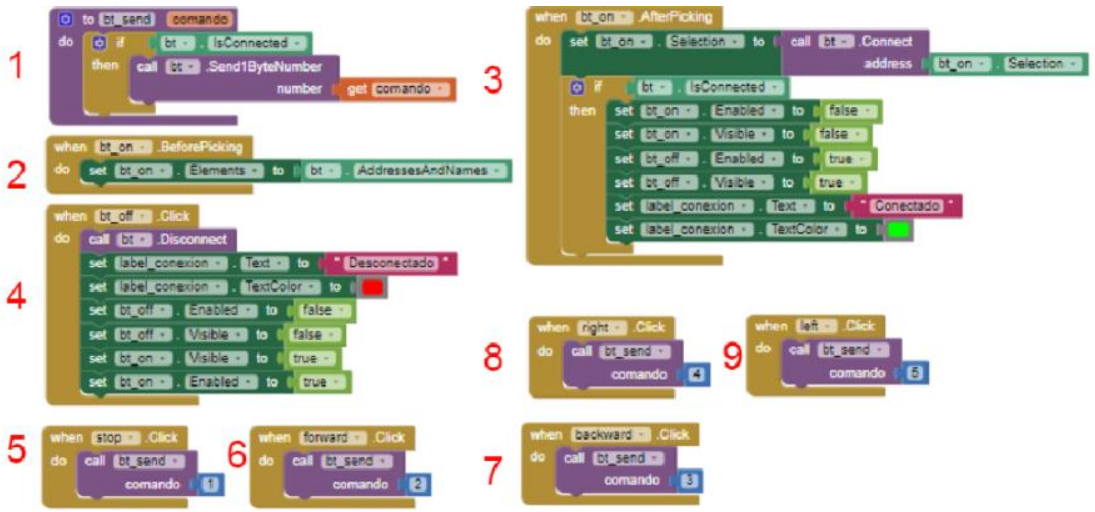


Figura 63: Código de la aplicación

En la imagen se ha numerado los bloques para explicarlos con mayor facilidad a continuación:

- Bloque 1: En este bloque se ha realizado la función "bt_send" a la que se le manda la variable "comando". Cuando se llama a la función, se comprueba que existe conexión con un dispositivo bluetooth y en caso de ser así, se le envía la variable con la que se ha llamado a la función.
- Bloque 2: Este bloque determina el comportamiento del "ListPicker". Al tocarlo, aparece un listado con todos los dispositivos bluetooth disponibles.
- Bloque 3: Este bloque se encarga de lo que ocurre una vez se ha elegido un dispositivo. En primer lugar, realiza la conexión con el dispositivo y, una vez conectado, desactiva el "ListPicker" para que no aparezca en pantalla (pues ya hay conexión con un dispositivo) y habilita el botón de desconexión (el cual inicialmente está desactivado).

Además, el texto pasa de "Desconectado" (texto inicial) a "Conectado" y cambia el color del texto de rojo a verde.

- Bloque 4: Este se encarga de la desconexión con el dispositivo que esté conectado. Al pulsar el botón se realiza la desconexión, se cambian el texto y su color a los iniciales y se vuelve a inhabilitar el botón de desconexión habilitando el de conexión.
- Bloques 5, 6, 7, 8 y 9: Estos se encargan de enviar las órdenes al robot. Al pulsar cualquiera de estos botones, se envía un número entero al robot utilizando la función "bt_send" y ya será el microcontrolador el que interprete dicho número.

7.4. Programación

Para la realización del software del este proyecto se ha utilizado el entorno de desarrollo gratuito Arduino IDE, cuyo lenguaje de programación es una adaptación de lenguaje C.

Las diferentes partes que existen en un código de Arduino son las siguientes:

- Setup: Una vez cargado el código en la placa de Arduino, todo lo que contenga la función "setup()" es ejecutado una sola vez al inicio del programa. Forma parte del bloque principal del código junto al loop.
- Loop: Esta función se repite en un bucle hasta que el Arduino es desconectado.
- Funciones: Fuera del bloque principal se pueden crear subrutinas las cuales se ejecutan solamente cuando son llamadas por el setup o el loop o por otras funciones llamadas con anterioridad por el bloque principal.
- Variables globales: Las variables se pueden declarar tanto dentro como fuera de una función, pero si se declaran dentro estas solo pueden ser usadas en dicha función. Las variables globales se declaran fuera de las funciones, pero pueden ser usadas por todas ellas.

En el código realizado se han tenido en cuenta estas partes para optimizarlo lo máximo posible. Por ejemplo, la inicialización de componentes o pines, dado que solo es necesario realizarla una vez, se ha incluido en el setup. Por otro lado, todo el control del robot se ha programado en el loop, pues debe hacerse de forma continua, y se ha hecho uso de funciones encargadas de distinta tareas para que sea más fácil el desarrollo del software. En cuanto a las variables, si estas deben ser usadas en diferentes partes del código o no deben perder su valor entonces se han declarado como variables globales.

En Figura 64 se muestra el diagrama de flujo que sigue el programa realizado.

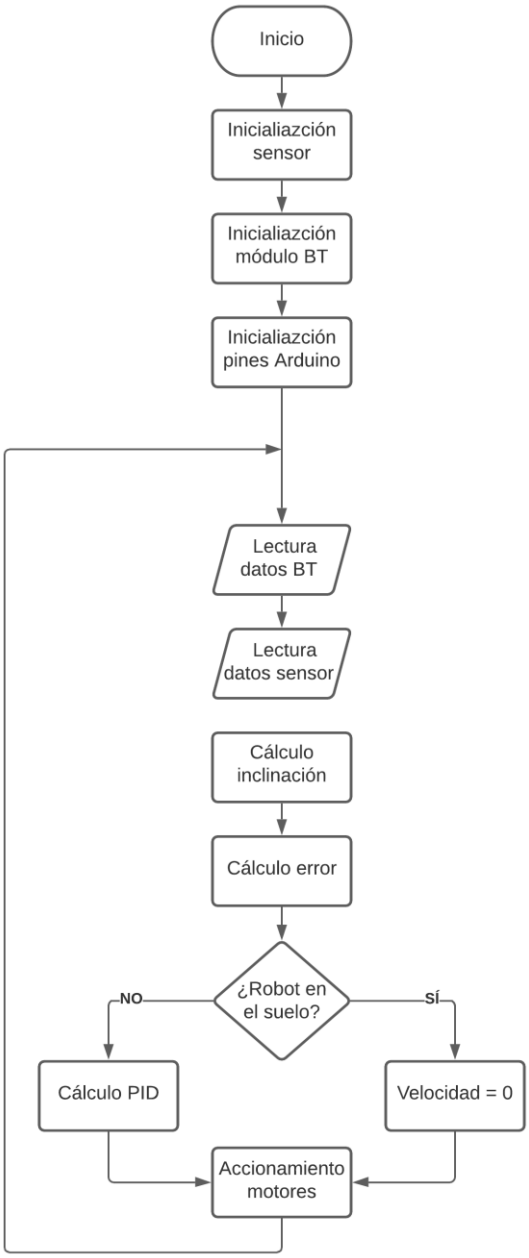


Figura 64: Diagrama de flujo del programa

A continuación se van a explicar los diferentes bloques que constituyen el diagrama de flujo. El código completo se encuentra en el Anexo: Software.

7.4.1. Inicialización

Lo primero que debe suceder cuando se enciende el robot es comenzar la comunicación entre los diferentes componentes que lo forman. En consecuencia, el microcontrolador, el sensor, el módulo bluetooth, el driver y los motores deben poder comunicarse entre sí.

Para configurar los pines de Arduino se utiliza la función “pinMode(pin, mode)”, cuya finalidad es establecer un pin de Arduino en modo entrada o salida.

Para establecer la comunicación con el módulo bluetooth se ha utilizado la librería "SoftwareSerial". Esta librería permite establecer una comunicación serie utilizando cualquier pin de la placa de Arduino, evitando tener que utilizar los pines Rx y Tx de la placa (pines D0 y D1). Las funciones utilizadas de esta librería son:

- Objeto SoftwareSerial: Escribiendo "SoftwareSerial nombre(pinRx,pinTx)" se crea una variable y se determinan los pines Rx y Tx.
- SoftwareSerial.begin(speed): Con esta función se inicializa la comunicación serie a la velocidad en baudios que se le pase a la función.

Para la comunicación con el sensor, se ha hecho uso de tres librerías: "MPU6050", "Wire" e "I2Cdev". La librería "Wire" permite la comunicación I2C con el sensor, mientras que la librería "MPU6050" hace uso de "I2Cdev" para facilitar la programación a la hora de obtener las lecturas del sensor. Las funciones utilizadas para inicializar el sensor son:

- Objeto MPU6050: Escribiendo "MPU6050 nombre(registro)" se crea una variable cuya dirección de registro es la que se le pasa a la función.
- Wire.begin(speed): Inicializa la comunicación I2C a la velocidad en baudios que se le pase a la función.
- MPU6050.initialize(): Inicializa el sensor.

La inicialización de los distintos componentes solo es necesario que suceda una vez, por lo que todo esto se realiza en el Setup.

```
void setup() {
  //DECLARACIÓN DE LOS PINES DEL DRIVER
  pinMode(In1, OUTPUT);
  pinMode(In2, OUTPUT);
  pinMode(In3, OUTPUT);
  pinMode(In4, OUTPUT);
  pinMode(EnA, OUTPUT);
  pinMode(EnB, OUTPUT);

  //DECLARACION DE LOS PINES DEL BT
  pinMode(Rx, INPUT);
  pinMode(Tx, OUTPUT);
  //INICIALIZACION DE LA COMIUNICACION CON EL MODULO BT
  bt.begin(9600);

  //INICIALIZACIÓN DE LA COMUNICACION CON EL PUERTO SERIE
  Serial.begin(57600);

  //INICIALIZACION DE LA COMUNICACION CON EL SENSOR
  Wire.begin();
  sensor.initialize();
}
```

7.4.2. Lectura de datos

7.4.2.1. Lectura de datos del módulo BT

Para leer los datos que lleguen desde el teléfono se ha realizado una función que se ejecuta en el loop. Esta función comprueba que han llegado datos procedentes del móvil y si han llegado,

los lee y los devuelve en una variable entera. Se muestra el diagrama de flujo de la función el Figura 65.

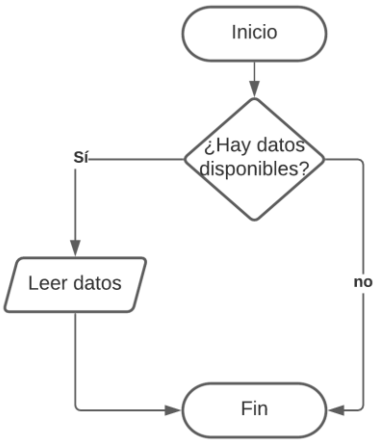


Figura 65: Diagrama de flujo de la función "leerBT"

Para esto se han utilizado dos funciones de la librería “SoftwareSerial”:

- SoftwareSerial.available(): Comprueba si ha llegado información, devolviendo el número de bytes disponibles para leer.
- SoftwareSerial.read(): Lee la información.

La función queda de la siguiente forma:

```
uint8_t leerBT()
{
  if(bt.available() > 0) //COMPRUEBA QUE SE HAN RECIBIDO DATOS
  {
    return bt.read();
    //SI SE HAN RECIBIDO, SE DEVUELVE LO QUE HAYA LLEGADO
  }
}
```

7.4.2.2. Lectura del sensor y cálculo del ángulo

Se ha realizado una función que lee las medidas en los tres ejes (X, Y y Z) tanto del giroscopio como del acelerómetro y que calcula el ángulo de inclinación mediante un filtro complementario. Para esta función se han utilizado las siguientes funciones de la librería “MPU6050”:

- MPU6050.getAcceleration(&accX,&accY,&accZ): Esta función lee los valores de la aceleración en los tres ejes y los almacena en las variables que se le pasan a la función.
- MPU.getRotation(&gX,&gY,&gZ): Mismo funcionamiento, pero lee la medición del giroscopio.

Para realizar el cálculo del ángulo se han utilizado las fórmulas ya mencionadas en “Filtro complementario”, “Acelerómetro” y “Giroscopio”.

En Figura 66 se muestra el diagrama de flujo y a continuación, el código.

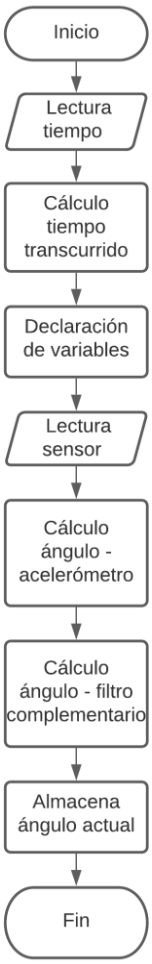


Figura 66: Diagrama de flujo de la función "obtenerAngulos"

```
float obtenerAngulos()
{
  dt = (millis()-tAnterior);
  //TIEMPO TRANSCURRIDO DESDE EL ULTIMO CALCULO DEL ANGULO
  tAnterior = millis();
  //ALMACENAMIENTO DEL TIEMPO ACTUAL PARA EL PROXIMO BUCLE

  //DECLARACION DE VARIABLES LOCALES
  float accelX;
  float K = 0.98;

  //MEDICION DEL SENSOR
  sensor.getAcceleration(&ax,&ay,&az);
  sensor.getRotation(&gx,&gy,&gz);


  //CALCULO DE LA INCLINACION A PARTIR DEL ACELEROMETRO
  accelX = atan(ax/sqrt(pow(ay, 2) + pow(az, 2)))*(180.0/3.14159265);

  //CALCULO DE LA INCLINACION MEDIANTE EL FILTRO COMPLEMENTARIO
  angX = K*(angAnteriorX - gy*dt/131000) + (1-K)*accelX;

  angAnteriorX = angX;
  //ALMACENAMIENTO DEL ANGULO ACTUAL PARA EL PROXIMO BUCLE
}
```

En el cálculo del ángulo de inclinación en el eje X, se ha utilizado el valor del giroscopio para el eje Y, pues el eje que realiza un giro en la dirección del eje X, como se mostró en Figura 30.

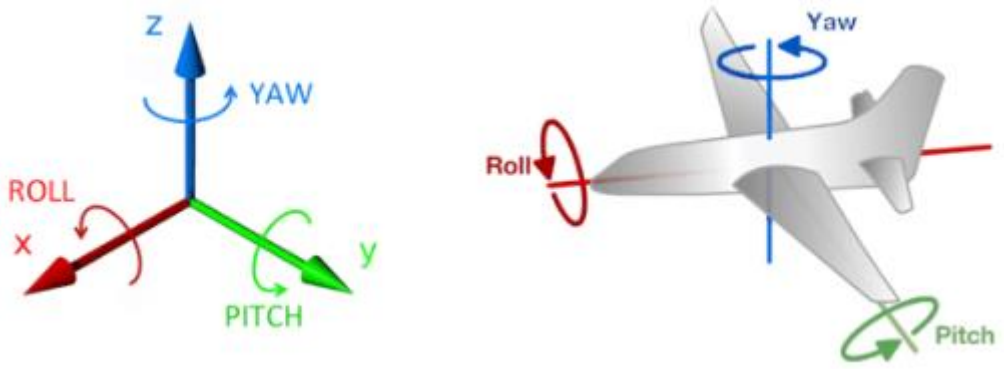


Figura 30: Roll, pitch y yaw. Fuente: [41]

Respecto a la constante utilizada en el filtro complementario (K = 0,98), se ha elegido este valor debido al buen rechazo del ruido en la medida y a la rápida respuesta ante un giro. Aun así, otros valores habrían sido igualmente válidos, pues en las pruebas realizadas se han obtenido resultados satisfactorios con valores de K superiores a 0,8 (en Figura 37 ya se mostró el comportamiento del filtro para un valor de K = 0,9).

Figura 37: Medidas individuales y filtrada del MPU-6050 para K = 0.9

Para comprobar el correcto funcionamiento del algoritmo se han realizado pruebas dejando el robot en una posición vertical y, con la ayuda de una escuadra y un cartabón, se han hecho mediciones con el robot girado 30°, 45° y 60°; las cuales fueron medidas de forma precisa.

7.4.3. Control PID

7.4.3.1. Introducción

Un control PID es un dispositivo o un algoritmo que permite controlar un sistema en bucle cerrado para obtener una salida deseada y está formado por tres elementos: Proporcional, Integral y Derivativo. En Figura 67 se muestra como es el diagrama de bloques de un control PID en un proceso cualquiera.

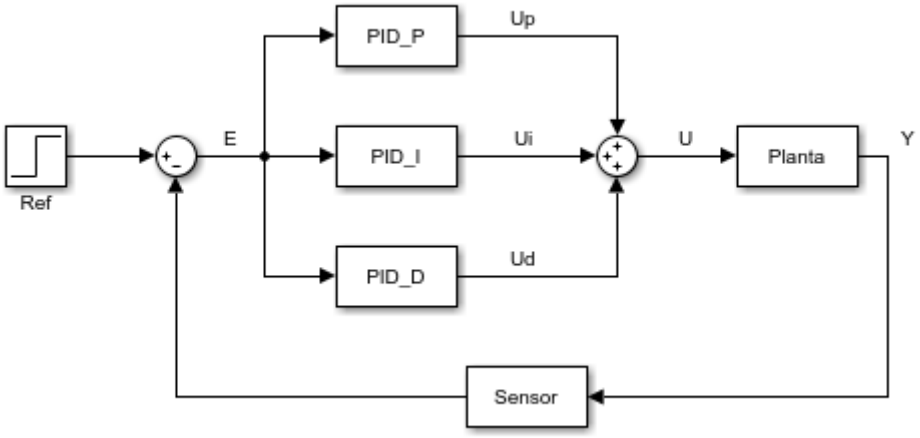


Figura 67: Diagrama de bloques de un control PID

El funcionamiento de un control PID consiste en comparar la referencia (Ref) con el valor medido a la salida del proceso (Y) y con el error obtenido (E) se obtiene una acción de control (U) que hace que el proceso alcance la salida deseada.

Los tres elementos de un control PID presentan diferentes características, las cuales los hacen adecuados para algunos tipos de controles, pero no para otros.

- Proporcional: Como su nombre indica, proporciona una acción de control proporcional al error obtenido. Esto sería el "presente" del control, es decir, no tiene en cuenta errores o acciones de control anteriores y no intenta predecir futuros comportamientos. La acción de control proporcional aumenta la velocidad de respuesta, disminuye el error en régimen permanente y aumenta la inestabilidad del sistema. La ecuación de un control proporcional es:

$$u_P(t) = K_P \cdot e(t)$$

- Integral: Esta acción de control realiza una integral de la señal del error, es decir, una suma o acumulación de errores. El objetivo de esto es reducir el error en régimen permanente, pero hace que la inestabilidad y la velocidad del sistema aumenten. Esto sería la parte "pasada" del control, pues tiene en cuenta como se ha comportado el sistema hasta conseguir eliminar el error. La ecuación de un control integral es la siguiente:

$$u_I(t) = K_I \cdot \int_0^t e(\tau) \cdot d\tau$$

- Derivativo: Esta acción de control deriva el error para así poder predecir la salida de la planta en un futuro, por lo que esto sería la parte "futura" del control. Entre sus características se encuentran el aumento de la estabilidad y la disminución de la velocidad del sistema. Su ecuación es la siguiente:

$$u_D(t) = K_D \cdot \frac{de(t)}{dt}$$

La acción total de un controlador PID es la suma de los controles de sus diferentes elementos:

$$u(t) = u_P(t) + u_D(t) + u_I(t)$$

En este proyecto, la acción de control se realiza en tiempo discreto, mientras que las fórmulas que se han mostrado son para tiempo continuo, por lo que hay que obtener su equivalente. Al ser un control el tiempo discreto, el tiempo que transcurre entre dos muestras, es decir, el tiempo de muestreo (T), influye significativamente en el comportamiento del sistema.

Las acciones de control en tiempo discreto quedan de la siguiente forma:

$$U_{P_k} = K_P \cdot E_k$$

$$U_{I_k} = U_{1_{k-1}} + K_I \cdot T \cdot E_k$$

$$U_{D_k} = K_D \cdot \frac{E_k - E_{k-1}}{T}$$

$$U_k = U_{P_k} + U_{D_k} + U_{I_k}$$

Siendo k el momento actual.

Respecto al método utilizado para realizar el controlador PID en este proyecto, se ha decidido emplear el método empírico.

La forma óptima de obtener un buen control del robot habría sido realizando un análisis matemático del prototipo, obteniendo las ecuaciones que rigen su comportamiento, y con esa información hacer un controlador a partir del lugar de las raíces obtenido, es decir, utilizando un método analítico que habría permitido predecir el comportamiento del robot antes de ponerlo en marcha.

Sin embargo, el método analítico se ha descartado para este proyecto debido a la complejidad del mismo. Durante el grado se ha realizado muchos controladores PID a partir de la función de transferencia de un proceso, por lo que esa parte no ha sido un problema, sino que lo complejo llega a la hora de obtener el modelo matemático del sistema. Existen numerosas variables que hay que tener en cuenta a la hora de hacerlo, como son el peso, las dimensiones o el lugar de colocación de los componentes y piezas o las características de los motores, las cuales no han sido proporcionadas cuando se han comprado y se habría necesitado obtenerlas mediante pruebas. Por este motivo, se ha considerado que este método es lo suficientemente difícil y extenso para ser un Trabajo Fin de Grado en sí mismo y no parte de otro.

Por tanto, se ha empleado un método empírico para realizar el control. En este caso, los parámetros del controlador se ajustan mediante prueba y error. Se establecen los valores de las constantes K_P, K_D y K_I en ese orden (cuando se considera que una constante tiene un valor adecuado, se pasa a ajustar el siguiente) y en función del comportamiento se van aumentando o reduciendo sus valores.

Cabe destacar que la realización de un control PID mediante el método empírico en este caso particular, en el que se tiene un sistema inestable, conlleva riegos pues es posible que nunca se llegue a obtener un control lo suficientemente bueno para estabilizar el sistema.

7.4.3.2. Código

La función realizada para realizar el control se basa en lo mencionado en el apartado anterior. A la función se le pasa el error obtenido a partir de la referencia y el ángulo medido por el sensor y, mediante las fórmulas mostradas en el apartado anterior, se obtienen los valores de los diferentes elementos del control, se suman obteniendo la acción de control total y se almacenan en una variable global.

En Figura 68 se muestra el diagrama de flujo de la función y a continuación, el código.

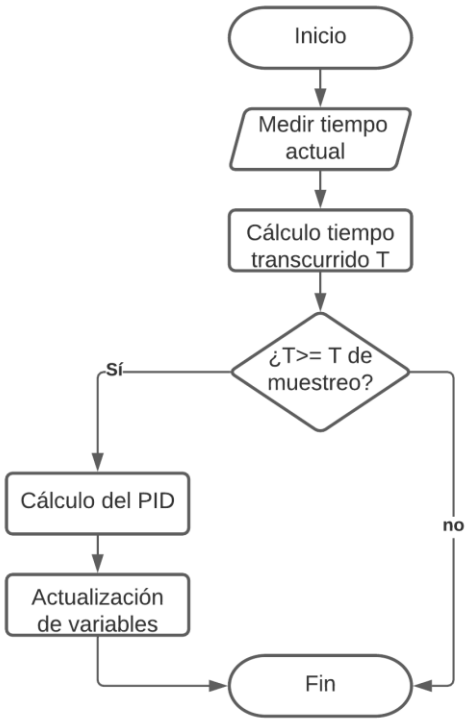


Figura 68: Diagrama de flujo de la función "posicionPID"

```
void posicionPID(float error)
{
  //DECLARACION Y CALCULO DE VARIABLES LOCALES
  long tInicioPos = millis(); //OBTENCION DEL TIEMPO ACTUAL
  long T = tInicioPos - tAnteriorPos;
  //TIEMPO TRANSCURRIDO DESDE EL ULTIMO CALCULO

  if (T >= tMuestreoPos)
```

```
  { //SI EL TIEMPO TRANSCURRIDO ES MAYOR O IGUAL AL TIEMPO DE MUESTREO
    tAnteriorPos = tInicioPos;
    //ALMACENAMIENTO DEL TIEMPO ACTUAL PARA EL PROXIMO BUCLE

    float PID_P = posKp*error; //CALCULO PARTE PROPORCIONAL

    float PID_D = posKd*(error - errorPosAnterior)*1000/T;
    //CALCULO PARTE DERIVATIVA
    errorPosAnterior = error;
    //ALMACENAMIENTO DEL ERROR PARA EL PROXIMO BUCLE

    float PID_I = posKi*(sumaErrorPos + error)*(T/1000);
    //CALCULO PARTE INTEGRAL
    sumaErrorPos = sumaErrorPos + error;
    //ALMACENAIENTO DEL ERROR ACUMULADO

    posPID = PID_P+PID_D+PID_I; //CALCULO DEL CONTROL PID
  }
}
```

7.4.4. Accionamiento de los motores

La función que se ha realizado se encarga de interpretar los datos que le llegan, que son la velocidad y si hay que girar, y se encarga de establecer las salidas adecuadas en los pines conectados al driver tal y como se explicó en "Principio de funcionamiento" cuando se explicó este componente. Además, se encarga de limitar el valor de la velocidad, pues los pines encargados de la salida PWM son de 8 bits, por lo que el valor máximo es 255.

En Figura 69 se muestra un diagrama de flujo que explica la función realizada y a continuación, el código.

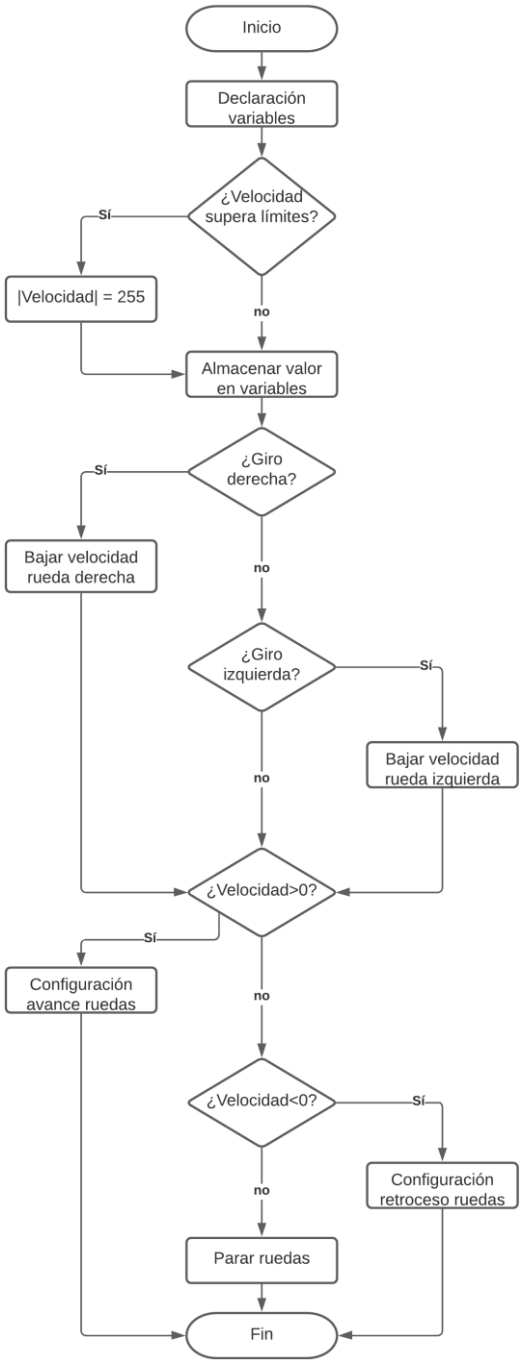


Figura 69: Diagrama de flujo de la función "mueveRuedas"

```
void mueveRuedas(int velocidad, uint8_t giro)
{
  int velocidadDerecha,velocidadIzquierda;
  //DECLARACION DE VARIABLES LOCALES

  //SI LA VELOCIDAD SUPERA EL LIMITE, SE ESTABLECE LA VELOCIDAD MAXIMA
DECLARADA
  if (velocidad > salidaMax) velocidad = salidaMax;
  else if (velocidad < -salidaMax) velocidad = -salidaMax;

  //SE ESTABLECE LA VELOCIDAD PARA CADA RUEDA
  velocidadDerecha = velocidad;
```

```
  velocidadIzquierda = velocidad;

  if (giro == giroDerecha)
  {
    //SI HAY QUE GIRAR A LA DERECHA, SE VARIA LA VELOCIDAD DE LA RUEDA
DERECHA
    velocidadDerecha = velocidadDerecha - 30;
    if (velocidadDerecha < -salidaMax) velocidadDerecha = -salidaMax;
  }
  else if (giro == giroIzquierda)
  //SI HAY QUE GIRAR A LA IZQUIERDA, SE VARIA LA VELOCIDAD DE LA RUEDA
IZQUIERDA
    velocidadIzquierda = velocidadIzquierda - 30;
    if (velocidadIzquierda<-salidaMax) velocidadIzquierda=-salidaMax;
  }

  //ACIONAMIENTO DE LOS MOTORES
  if (velocidad < 0) //SI LA VELOCIDAD ES NEGATIVA
  {
    //CONFIGURACION PARA QUE AMBOS MOTORES RETROCEDAN
    digitalWrite(In1, LOW);
    digitalWrite(In2, HIGH);
    analogWrite(EnA, -velocidadDerecha);
    digitalWrite(In3, LOW);
    digitalWrite(In4, HIGH);
    analogWrite(EnB, -velocidadIzquierda);
  }
  else if (velocidad > 0) //SI LA VELOCIDAD ES POSITIVA
  {
    //CONFIGURACION PARA QUE AMBOS MOTORES AVANCEN
    digitalWrite(In2, LOW);
    digitalWrite(In1, HIGH);
    analogWrite(EnA, velocidadDerecha);
    digitalWrite(In4, LOW);
    digitalWrite(In3, HIGH);
    analogWrite(EnB, velocidadIzquierda);
  }
  else //SI VELOCIDAD = 0, MOTORES SE PARAN
  {
    analogWrite(EnA, velocidad);
    analogWrite(EnB,velocidad);
  }
}
```

Como se muestra en el código, las salidas de los pines IN1-IN4 son contrarias en función de si el motor debe avanzar o retroceder y la velocidad de las ruedas siempre se manda al pin con un valor positivo.

Cuando el usuario está controlando el robot y decide realizar un giro, se baja la velocidad de la rueda derecha si se quiere girar a la derecha o de la izquierda si se quiere girar al otro lado. Para explicar esto, hay que tener en cuenta la cinemática de un robot móvil.

En este caso, la configuración del robot es una configuración diferencial, es decir, posee dos ruedas de tracción y ninguna orientable y el eje de rotación se encuentra en el centro del eje de tracción. En estos casos, la velocidad de las ruedas determina si el móvil avanza, retrocede, gira o va recto.

En Figura 70 se muestra un esquema de la cinemática de un robot móvil diferencial.

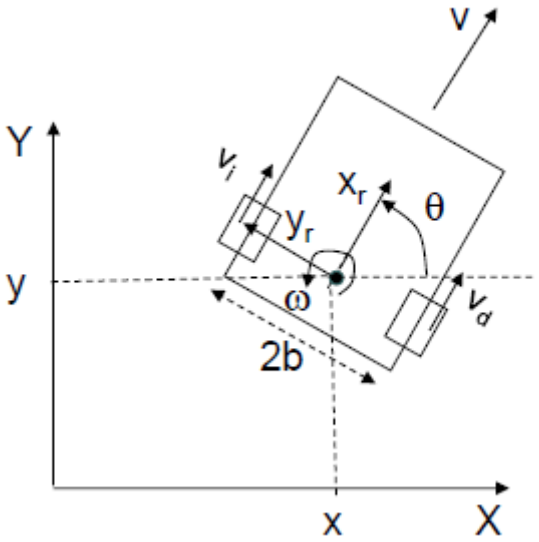


Figura 70: Cinemática de un robot móvil diferencial

En un robot diferencial, la velocidad lineal de su centro de gravedad, V, es igual al promedio de las velocidades lineales de cada una de sus ruedas.

$$V = \frac{v_d + v_i}{2}$$

Por otro lado, su velocidad angular se obtiene mediante las relaciones geométricas del movimiento de cada rueda [51], como se muestra en Figura 71.

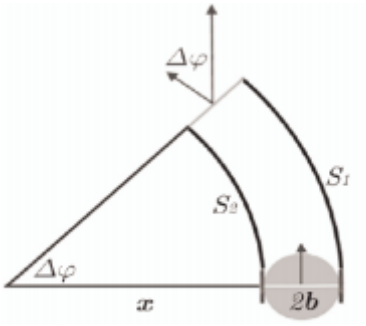


Figura 71: Esquema movimiento de un robot diferencial. Fuente: [51]

En Figura 70 se muestra como el arco producido por la rueda izquierda tiene un radio x, mientras que la rueda derecha, un radio x+2b. Dado que ambos arcos tienen el mismo ángulo φ, la velocidad angular de la rueda derecha tiene que ser mayor, describiendo así una trayectoria S_1 más larga que la S_2.

$$S_1 = \Delta\varphi \cdot (x + 2b) = r \cdot \Delta\theta_1$$

$$S_2 = \Delta\varphi \cdot x = r \cdot \Delta\theta_2$$

Calculando la diferencia de los dos desplazamientos y derivando el ángulo girado respecto del tiempo, se obtiene la velocidad angular del móvil en relación a la velocidad de giro de sus ruedas.

$$S_1 - S_2 = \Delta\varphi \cdot 2b = r \cdot (\Delta\theta_1 - \Delta\theta_2)$$

$$\Delta\varphi = \frac{r \cdot (\Delta\theta_1 - \Delta\theta_2)}{2b}$$

$$\frac{\Delta\varphi}{\Delta t} = \omega = \frac{r \cdot (\Delta\theta_1 - \Delta\theta_2)}{\Delta t \cdot 2b} \quad ; \quad \frac{r \cdot \Delta\theta}{\Delta t} = v$$

$$\omega = \frac{v_1 - v_2}{2b}$$

Siendo:

- $\Delta\theta$: Aumento del ángulo girado por la rueda
- ω: Velocidad angular del robot
- Δt: Tiempo
- r: Radio de la rueda
- $2b$: Distancia entre las dos ruedas

A partir de esta fórmula se deduce que para girar a un lado, la rueda más alejada debe ir a mayor velocidad, tal y como se ha realizado en el código mostrado.

8. Resultados y conclusiones

Este proyecto ha puesto a prueba todo lo que se ha aprendido durante el Grado en Ingeniería Electrónica Industrial y Automática, obligando por primera vez a pasar meses realizando investigación, pruebas y diseños.

En los siguientes apartados se comentan los resultados y conclusiones que se sacan de este proyecto.

8.1. Resultados

La realización del robot móvil inestable ha sido una tarea larga y complicada que ha requerido mucho esfuerzo durante estos meses de trabajo. Se ha construido un robot con una estructura mecánica y un circuito electrónico funcionales para el cumplimiento de los objetivos planteados para este proyecto. Sin embargo, aunque también se ha podido conocer el ángulo de inclinación del robot en todo momento (ver Figura 37), el control de la estabilidad no ha funcionado de forma óptima.

Durante la realización del trabajo se han realizado numerosas pruebas y modificaciones en el prototipo con el objetivo de que este se mantuviera en pie sin ayuda, pero no se ha podido conseguir. El comportamiento del robot, en las diversas pruebas realizadas ha sido siempre muy similar: cuando este se inclina hacia atrás o existe una perturbación que lo empuja hacia atrás, en muchas ocasiones es capaz de corregir bastante bien evitando la caída en ese

momento; sin embargo, al caer hacia el otro lado, este no es capaz de evitar la caída en la mayoría de ocasiones.

8.1.1. Incidencias y pruebas realizadas

En este apartado se enumeran todas los imprevistos que han sucedido y todas las pruebas que se han realizado sin éxito para que el robot se estabilizara.

Las incidencias durante la realización de este trabajo han sido:

- **Reseteo**
  En un primer momento, este prototipo solo contaba con dos baterías puestas en serie a modo de alimentación. Durante las pruebas, el robot realizaba un reinicio debido a que no obtenía suficiente corriente para mantenerse en funcionamiento. Por este motivo, el prototipo final cuenta con cuatro baterías, como se mencionó en el apartado "Alimentación".

- **Drivers en mal estado**
  A lo largo del proyecto ha sido necesario comprar tres controladores para los motores debido a fallos en su funcionamiento. Estos drivers comenzaron a suministrar menos corriente a uno de los motores, provocando que las ruedas giraran a distinta velocidad, lo que hacía que el robot girara. El motivo de estos fallos es desconocido, pues las conexiones dentro de las placas están correctas y no se ha visto ningún componente quemado. En Figura 72 se muestra una fotografía del robot junto a los dos drivers defectuosos.

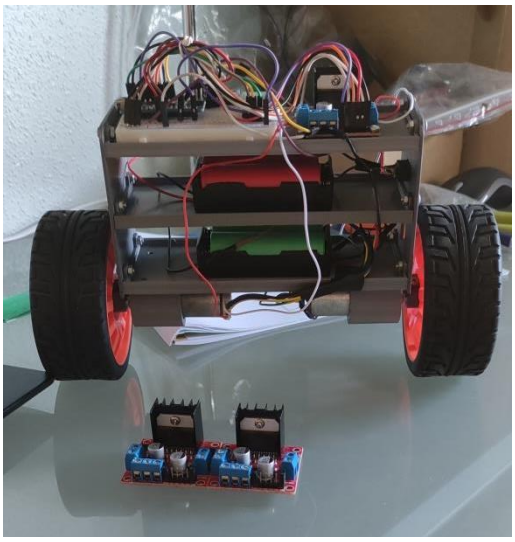


Figura 72: Robot junto a los drivers defectuosos

- **Rotura de piezas**
  Este caso afortunadamente solo se ha dado en una ocasión. Esta pieza era la base con los soportes para los motores, en la que se partieron dos de las piezas rectangulares en los que se colocan los tornillos, tal y como se muestra en Figura 73.

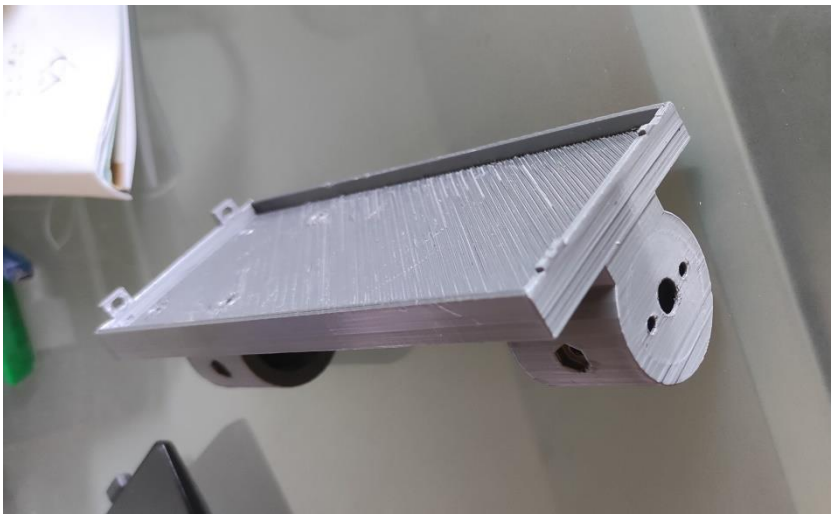


Figura 73: Base con soportes rota

- **Margen de error para la impresora 3D**
  En numerosas ocasiones, se ha necesitado reimprimir una pieza debido a que las dimensiones eran demasiado ajustadas, por lo que, por ejemplo, no cabían los tornillos en los agujeros.

- **Caída de las ruedas**
  En las primeras etapas del prototipo, las ruedas se realizaron mediante diseño e impresión 3D y en estas no se añadió ningún tipo de elemento que las mantuviera fijas en el eje del motor, lo cual provocaba que durante las pruebas el robot cayera debido a que le faltaba alguna de ellas. En diseños posteriores se realizaron dichos elementos, aunque al final se acabó comprando ruedas ya fabricadas.

- **Banda muerta en los motores**
  Este puede ser el inconveniente más importante de este proyecto. Los motores de corriente continua no son capaces de trabajar en todo el rango de voltaje que el driver les suministra (0 V a 6 V).
  Durante las pruebas, se ha notado que era necesaria una acción de control de al menos 60 (aproximadamente 1,5 V en los motores), siendo la máxima acción de control 255, para que los motores comenzaran a moverse. Esto implica que las ruedas del robot no se muevan cuando se presentan acciones de control entre -60 y 60, o sea, errores de entre -2° y 2° aproximadamente. Esto puede suponer que el robot comience a moverse cuando ya es demasiado tarde, pues es más fácil estabilizarlo cuando está más cerca de la referencia.

Referente a las pruebas realizadas sin éxito para que el robot no se cayera, estas se enumeran a continuación:

- **Modificación del centro de gravedad**
  Como se ha comentado en "Piezas", dos de las bases realizadas permitían colocar o dos baterías o la protoboard más el driver. El motivo de esto es manipular la altura del centro de gravedad del robot, pues las baterías son más pesadas que los otros dos

elementos, por lo que si están más abajo, bajan el centro de gravedad; y si están más elevadas, lo suben.
La localización del centro de gravedad en el robot afecta principalmente a la acción de control, siendo que la acción de control tiene que ser más grande cuanto más elevado se encuentre.

- **Prototipo "largo"**
  En relación a lo comentado anteriormente, también se ha realizado un prototipo alto del robot con el objetivo de obtener un centro de gravedad muy alto. En este caso le costaba bastante corregir el error en la inclinación, pues lo hacía más despacio que el prototipo final. En Figura 74 se muestra una imagen del prototipo "largo" (en el caso de la imagen, esta está tomada una vez se entendió que no se debía elevar mucho el centro de gravedad, por lo que los componentes se encuentran en la parte baja).

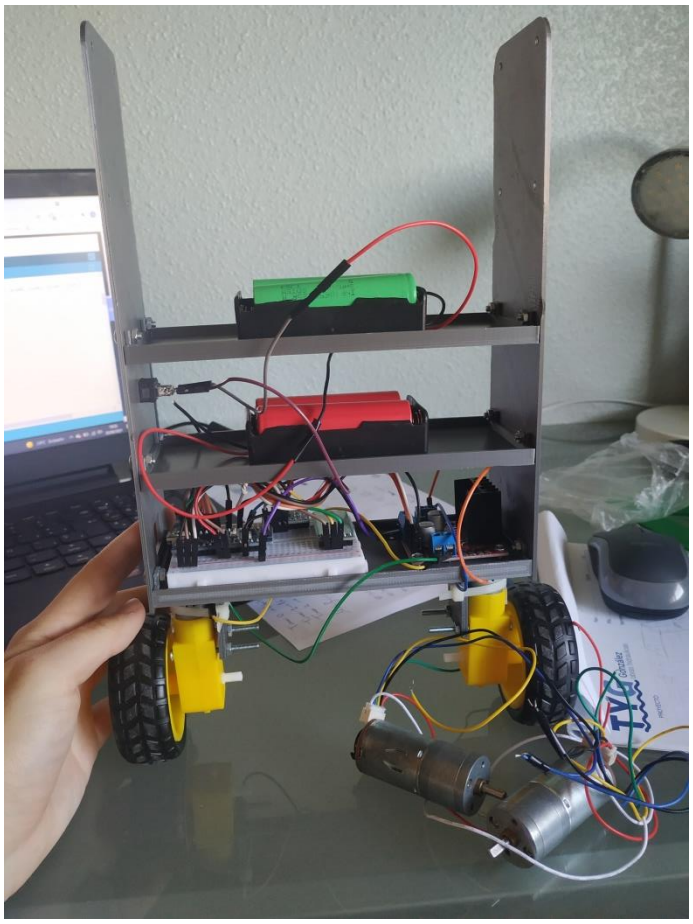


Figura 74: Prototipo "largo"

- **Cambio de motores**
  En Figura 74 se puede apreciar como en la parte inferior, en la mesa, se encuentran los motores utilizados en el modelo final, mientras que hay otros en el robot. Esto se debe a que a lo largo del proyecto se han probado dos tipos de motores de corriente continua y un par de motores paso a paso.
  Los motores que se ven en la imagen son de corriente continua y se decidió comprarlos debido a su precio y a que su velocidad era mayor que los que se han

usado al final, pues durante las pruebas se pensó que el motivo por el que el robot no se estabilizaba era por falta de velocidad. Esto no resultó eficaz, pues la calidad de estos tampoco era la ideal. En Figura 75 se muestra una imagen de los motores y la base creada para estos.

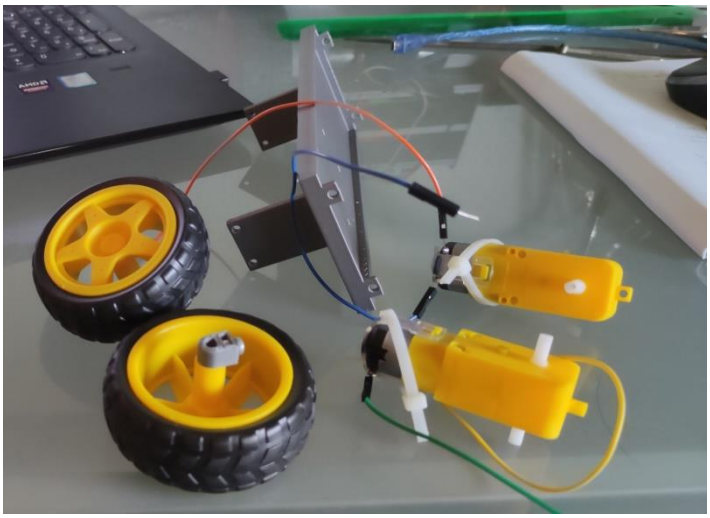


Figura 75: Motores CC descartados, ruedas y soporte

- **Motores paso a paso**

  Se decidió comprar un par de motores NEMA 17 para ver si con estos motores se conseguía la estabilidad, pues estos podían ir más rápido y a la hora de cambiar de sentido de giro pueden realizarlo de forma inmediata, pues no tienen la inercia de los motores de corriente continua.

  Esta opción se desestimó debido a que el principio de funcionamiento de estos motores es muy diferente, lo que complicaba la programación. Se probó programando interrupciones en el microcontrolador cuando pasara el tiempo establecido para obtener una determinada velocidad y cuando ocurría dicha interrupción, los motores realizaban un paso.

  Uno de los problemas principales es que el driver utilizado no es el idóneo para realizar un control de esas características, por lo que habría sido buena idea comprar dos drivers enfocados a este tipo de control. Además, las interrupciones comenzaron a dar fallos una vez se unieron al programa principal (cuando se programaron en un código a parte funcionaban correctamente). En Figura 76 se muestra una imagen de los motores y la base creada para estos.

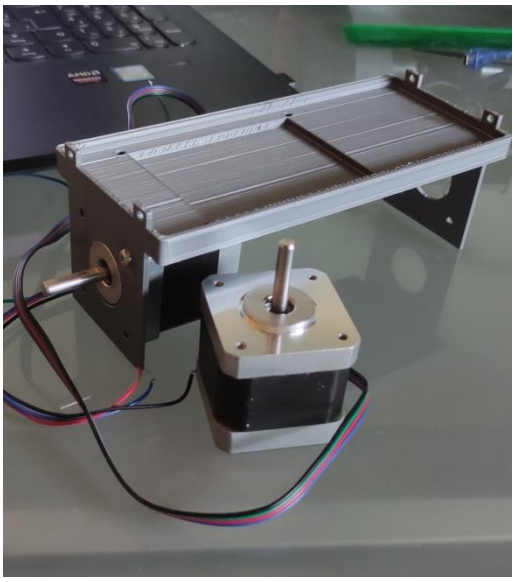


Figura 76: Motores paso a paso y su soporte

- **Cambio de ruedas**

  Durante todo el proyecto, se han utilizado diferentes ruedas que se han diferenciado en su radio y en su anchura. Con la variación del radio se pretendía variar la velocidad del robot pues esta es proporcional al radio de las ruedas. En cuanto al ancho, se quería mejorar la estabilidad al darle más superficie de contacto a la rueda.

  En el prototipo final se ha acabado utilizando un par de ruedas compradas que tienen un diámetro de 10 cm, pero al inicio del proyecto, las ruedas impresas tenían un diámetro de 6 cm. En Figura 77 se muestra el prototipo final junta a las ruedas que se han utilizado durante el proyecto.

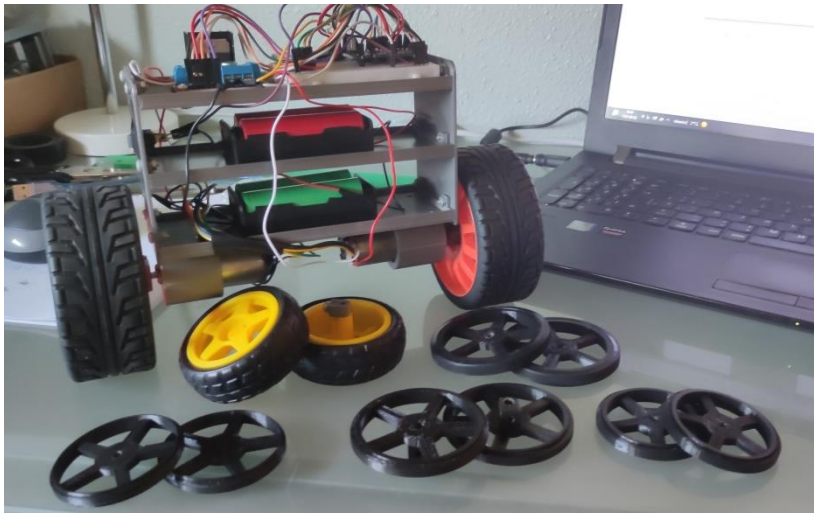


Figura 77: Prototipo junto a las ruedas probadas durante el proyecto

- **Librería "PID_v1"**

  Tras numerosas pruebas, cabía la posibilidad de que el problema no fuera del hardware, sino que fuera el software. El código para la realización del PID podía contener errores, por lo que se decidió utilizar la librería "PID_v1". Esta librería contiene todo lo necesario para realizar un control PID, en el código solo es necesario

emplear las funciones que proporciona pasándole los parámetros del control. Sin embargo, el resultado de las pruebas fue muy similar al código que se ha realizado, por lo que se ha continuado con el código original.

Tanto con la librería como con el código original, se han realizado pruebas primero con un controlador P y cuando este otorgaba un buen comportamiento se realizaba un control PD o PID (no se ha realizado un PI, pues tanto la componente proporcional como la integral estarían incrementando la inestabilidad, como se menciona en el apartado "Control PID", de un sistema ya inestable de inicio).

El controlador P provocaba que las ruedas avanzaran con una velocidad proporcional al error en la inclinación y variaran su sentido de giro en el momento que la inclinación cambiara de estar hacia delante a atrás y viceversa, provocando que los movimientos fueran bruscos imposibilitando la estabilización solo con un control P.

Cuando el control P era capaz de efectuar un control rápido y menos brusco (pero aun inestable), se comenzaba a variar el parámetro K_D, el cual hacía que el movimiento fuera más suave favoreciendo la estabilidad. Si se decidía implementar un PID, los movimientos volvían a ser un poco más bruscos, pero más veloces.

- **Variación de la referencia y del tiempo de muestreo**

  La forma de sintonizar el controlador PID ha sido mediante prueba y error, es decir, dando valores a los parámetros K_P, K_D y K_I hasta que el comportamiento fuera el deseado. Sin embargo, existen otros dos parámetros muy importantes para este control, el tiempo de muestreo y la referencia.

  La variación del tiempo de muestreo no ha supuesto cambios muy significativos durante las pruebas, mientras que los cambios en la referencia han sido más notables. El objetivo era evitar que el robot solo corrigiera bien uno de los lados, ya que esto podía ser síntoma de que la referencia no estaba exactamente en el centro, sino que estaba desplazada hacia delante o atrás. Sin embargo, los numerosos valores probados durante el proyecto (se ha probado con referencias de -2° a 2° tomando decimales por medio) nunca llegaron a corregir la inclinación en ambos lados, pues la mayoría de veces siempre se conseguía corregir decentemente uno de los dos, pero el otro seguía cayendo).

- **Intento de solución de la banda muerta**

  El problema de la banda muerta ha sido uno de los principales inconveniente para el control del prototipo. Por este motivo, se ha intentado solucionar mediante código. Lo que se ha intentado es reducirla, limitando esta banda a una acción de control entre ± 15, haciendo que para acciones de control entre ese valor límite y ± 60, el valor absoluto de la acción de control pasara a ser 60. Esto no mejoró el comportamiento, al revés, lo empeoró, pues el movimiento a veces era muy brusco para lo que necesitaba el robot en ese momento.

8.1.2. Hipótesis y posibles soluciones

Una vez conocido como se ha comportado el robot durante todas la pruebas realizadas, se plantean una serie de hipótesis con sus posibles soluciones del porqué no ha funcionado correctamente el robot.

- **Error en el método**
  El método empírico conllevaba riesgos, siendo posible no alcanzar nunca una solución óptima para el control realizado, por lo que de haberse hecho mediante un método analítico se podría haber llegado a una solución.

- **Banda muerta**
  El hecho de reaccionar tarde para conseguir la estabilidad puede haber sido uno de los principales motivos por lo que el robot no se ha mantenido en pie. Del rango de 0 V a 6 V que le llegaba a los motores, estos comenzaban a moverse cuando le llegaban 1,5 V aproximadamente; y no era un problema exclusivo de estos, el otro par de motores de corriente continua también contaba con esta característica.
  Si extrapolamos esta característica a todos los motores de corriente continua y suponemos que la banda muerta permanece en el mismo rango de tensión, una solución podría ser utilizar motores con una tensión máxima mayor, necesitando también cambiar la fuente de alimentación para poder suministrar ese voltaje. Por ejemplo, si se utilizaran motores de 12 V y la banda muerta permaneciese igual, lo que una salida PWM de 60 en el Arduino antes eran 1,5 V, ahora sería el doble, por lo que el robot tardaría la mitad de tiempo en reaccionar. Pero como se ha mencionado, esto podría funcionar si se mantiene el rango de tensión de la banda muerta.
  En una futura continuación del proyecto se podrían utilizar otras estrategias de control para sistemas no lineales con compensación de la banda muerta, pues ya no queda tiempo suficiente para realizarlo en este proyecto.

- **Velocidad del robot**
  Una vez la inclinación se alejaba bastante de la referencia, el robot avanzaba a su máxima velocidad, pero no era capaz de estabilizarse, solo prolongaba el tiempo que tardaba en caerse. Esto se puede deber a que la velocidad del robot no era suficiente y se puede deber a los dos principales factores que determinan la velocidad del robot: la velocidad de giro de los motores y el diámetro de las ruedas.
  Para aumentar la velocidad del prototipo se podría cambiar el par de motores usado por otro con una velocidad máxima mayor y conseguir ruedas con mayor diámetro. Sin embargo, durante el proyecto se han cambiado las ruedas en diversas ocasiones y no ha sido suficiente, por lo que el problema principal podría estar en los motores.

- **Control PID**
  No se puede descartar que el control realizado sea erróneo o que en ninguna prueba se haya llegado al ajuste necesario para estabilizar el robot, solo se tiene constancia de que la librería “PID_v1” otorgaba un comportamiento similar al prototipo. Por tanto, una forma de solucionar esto sería continuar probando ajustes siempre que se siga enfocando el control desde un método empírico o emplear el método analítico.

- **Código**
  Es posible que el propio algoritmo en su conjunto no sea el óptimo para optimizar el control del robot. El robot no realizaba movimientos suaves a la hora de intentar mantenerse erguido cuando estaba en torno a la referencia, por lo que los cambios bruscos podían provocar que no se consiguiera estabilizar. Esto también se puede

deber al problema mencionado de la banda muerta, pero no se puede descartar que se pueda optimizar el código de forma que se consigan movimientos más suaves que mejoren el control.

8.2. Conclusiones y líneas futuras

Este trabajo ha requerido mucho tiempo y dedicación y ha puesto a prueba muchas facetas importantes en un ingeniero que no se pueden examinar en un examen. Aspectos tales como la investigación propia para realizar un proyecto, la capacidad de estar enfocado en un mismo tema durante meses buscando llevar adelante un trabajo o la búsqueda de soluciones para los distintos problemas que han surgido durante el desarrollo del prototipo eran cosas que han sucedido durante la carrera en menor medida, pero el Trabajo Fin de Grado las ha magnificado.

En cuanto a los aspectos teóricos para realizarlo, diversas asignaturas cursadas durante el Grado han otorgado los conocimientos necesarios para llevarlo a cabo:

- Electrónica: Electricidad, Tecnología Electrónica, Electrónica Analógica, Electrónica Digital, Electrónica Orgánica y Procesos en el Diseño Electrónico; y Laboratorio de Circuitos.
- Informática: Informática, Informática Industrial I y II; e Informática Aplicada.
- Control: Automática Básica, Técnicas de Control, Control Avanzado por Computador, Ingeniería de Control y Control de Sistemas Mecatrónicos.
- Robótica: Robótica Móvil y Sistemas Robotizados.
- Redacción de los documentos: Oficina Técnica.

Sin embargo, con todo el trabajo y esfuerzo puesto por hacerlo bien, el prototipo no ha funcionado correctamente, lo que demuestra cuán difícil es llevar a cabo un trabajo de ingeniería y la responsabilidad que conlleva ser el responsable de este para con un cliente o una empresa.

No obstante, el hecho de haber fallado no significa en este caso que todo lo que se ha hecho no merezca la pena; al contrario, se ha llevado a cabo investigación, se han realizado diversas pruebas y cambios empleando los conocimientos adquiridos durante la carrera y la realización del proyecto y se ha aprendido sobre el comportamiento de un robot autobalanceado.

Se ha aprendido la importancia de una medición precisa del ángulo de inclinación del robot, pues el robot debe conocer con exactitud su situación para realizar un buen control; la relación entre la altura del centro de gravedad y la acción de control, ya que se ha visto que son proporcionales y es importante no presentar acciones de control muy elevadas ya que siempre hay un valor máximo que no conviene superar muy rápido, para así proporcionar al prototipo durante más tiempo el control que necesita; y el riesgo de realizar un control sin un modelo matemático que defina su comportamiento.

En definitiva, este proyecto no representa un “fracaso”, más bien un aliciente para mejorar como ingeniero y volver a plantarle cara a este proyecto, pues “de los errores se aprende” y, aunque este Trabajo Fin de Grado haya finalizado, el robot algún día correrá por los pasillos de

casa controlado por un teléfono móvil. Y en ese futuro, puede que el prototipo cuente con los siguientes elementos que se pueden añadir para mejorarlo:

- **PCB**: Durante este proyecto se ha utilizado una protoboard para conectar los componentes electrónicos. En un primer momento, no se planteó realizar una PCB, aunque con el paso del tiempo la idea llegó, pero cuando se quiso realizar una, las páginas web encontradas establecían plazos de entrega que no ofrecían garantías de llegar a tiempo para la finalización de este trabajo.

- **Sensor de proximidad**: Lo más importante en este robot es que pueda mantenerse en pie y la aparición de un obstáculo provocaría la caída de este, por lo que la instalación de uno o varios sensores de proximidad en el prototipo evitaría que un obstáculo tirara el robot. Durante la carrera se han empleado sensores ultrasonidos (HC-SR04) y han funcionado bastante bien.

- **Baterías LiPo**: El hecho de haber necesitado cuatro baterías Li-Ion ha sido algo que no se esperaba al iniciar el proyecto. De haber sabido esto, se habría utilizado una batería LiPo de 7,4 V, la cual se descartó principalmente por el precio de la misma.

- **Cambio de motores**: Con más tiempo para pruebas, una opción sería utilizar los motores paso a paso que se han comprado y obtener un par de drivers más compatibles con el método de control, evitando así bandas muertas e inercias debidas a los motores CC.

- **Conexión WiFi**: El robot podría ser utilizado mediante bluetooth o WiFi. Esto ampliaría el tipo de controladores que se podrían utilizar, dejando de estar limitado a un móvil con la aplicación realizada.

- **Mejora de la aplicación móvil**: La app realizada ha sido algo muy sencillo cuyo objetivo era realizar un control básico del robot. Sin embargo, esta se podría mejorar añadiendo controles de velocidad o aparición de datos en pantalla como la inclinación actual y el estado de la batería.

- **Nuevo diseño mecánico**: Tanto si se mantienen las cuatro baterías o se cambian por una LiPo, una buena idea sería diseñar una base capaz de albergar toda la alimentación, lo cual permitiría que el robot solo tenga dos alturas y ayudaría a subir o bajar el centro de gravedad con mayor eficacia.

- **Cerrar el robot**: Actualmente, todos los componentes están a la vista, lo cual puede hacer que cualquier contacto desconecte un cable o mueva una pieza, por lo que se podría diseñar una carcasa para proteger los circuitos, además de mejorar la estética del robot.

- **Inclusión de LEDs**: Los componentes electrónicos cuentan con LEDs que permiten al usuario conocer el estado del dispositivo que se está usando: si está encendido, si hay un error, si la batería está baja… El prototipo actual no cuenta con ninguno, pues los componentes electrónicos utilizados cuentan con los suyos propios que han permitido conocer que no había ningún error en la alimentación. Si el robot se cerrara, la

inclusión de LEDs permitiría conocer cualquier tipo de información, pues se podría codificar ya sea por el número de LEDs encendidos o por el color de estos.

9. Bibliografía

[1] *¿Qué es la robótica?* (s.f.). Recuperado el 1 de junio de 2021, de EDS Robotics: https://www.edsrobotics.com/blog/que-es-la-robotica/

[2] Walter, W. G. (1950). An imitation of life. *Scientific American, 182* (5), 42-45.

[3] Polanco, A. (s.f.). *Elmer y Elsie, las tortugas robot de 1948*. Recuperado el 1 de junio de 2021, de Alpoma: https://alpoma.net/tecob/?p=11359

[4] Malone, B. (26 de septiembre de 2011). George Devol: A Life Devoted to Invention, and Robots. *IEEE Spectrum*.

[5] Castaños, F. (3 de febrero de 2003). *Levantamiento y Estabilización del Péndulo Invertido.* Recuperado el 1 de Junio de 2021, de https://www.ctrl.cinvestav.mx/~fcastanos/mios/bachelorCastanos.pdf

[6] García, J., Ramírez, L., Siordia, X., & Martínez, T. (2016). Las leyes de Newton en el modelado y control del péndulo invertido sobre un carro. *Revista Tecnología e Inovación*, *3* (9), 11-19.

[7] *Nuestra historia*. (s.f.). Recuperado el 1 de junio de 2021, de Segway: https://www.segway.es/es/comunicacion/nuestra-historia/

[8] *Segway PT x2 SE*. (s.f.). Recuperado el 1 de junio de 2021, de Segway: https://www.segway.es/es/catalogo-de-productos/estilo-de-vida/segway-pt-x2-se/

[9] Antonio-Cruz, M., Márquez-Serrano, C., Silva-Ortigoza, R., & Merlo-Zapata, C. (1 de enero de 2014). Sistemas Mecánicos Subactuados: Péndulos Invertidos. *Boletín UPIITA, 41*.

[10] *Asimo El robot humanoide más avanzado del mundo*. (s.f.). Recuperado el 1 de junio de 2021, de Honda: https://www.honda.mx/asimo

[11] *El prototipo de cohete "Starship" de SpaceX aterriza con éxito después de cuatro fracasos*. (6 de mayo de 2021). Recuperado el 1 de junio de 2021, de DW: https://www.dw.com/es/el-prototipo-de-cohete-starship-de-spacex-aterriza-con-%C3%A9xito-despu%C3%A9s-de-cuatro-fracasos/a-57450073

[12] (s.f.). Recuperado el 3 de junio de 2021, de Solectroshop: https://solectroshop.com/es/controladores-de-motores/40-controlador-tb6612fng-motor-dc-pap.html

[13] (s.f.). Recuperado el 3 de junio de 2021, de Solectroshop: https://solectroshop.com/es/shields-arduino/116-controlador-l293d-de-motor-y-servo-placa-de-expansion.html

[14] Llamas, L. (22 de septiembre de 2016). *Determinar la orientación con Arduino y el IMU MPU-6050*. Recuperado el 3 de junio de 2021, de Luis Llamas: https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/

[15] Llamas, L. (17 de septiembre de 2016). *Usar un acelerómetro ADXL345 con Arduino*. Recuperado el 3 de junio de 2021, de Luis Llamas: https://www.luisllamas.es/arduino-acelerometro-adxl345/

[16] Llamas, L. (20 de septiembre de 2016). *Usar un acelerómetro MMA7455 con Arduino*. Recuperado el 3 de junio de 2021, de Luis Llamas: https://www.luisllamas.es/arduino-acelerometro-mma7455/

[17] *¿Qué diferencias hay entre una Li-Po y una Li-Ion?* (s.f.). Recuperado el 3 de junio de 2021, de 330ohms: https://blog.330ohms.com/2020/06/22/que-diferencias-hay-entre-una-li-po-y-una-li-ion/

[18] *Batería de níquel-metalhidruro*. (s.f.). Recuperado el 3 de junio de 2021, de Wikipedia: https://es.wikipedia.org/wiki/Bater%C3%ADa_de_n%C3%ADquel-metalhidruro

[19] *Mini Metall DC Motor Gleichstrom Getriebemotor 25D 37D 6V 12V 76-1700RPM QITA*. (s.f.). Recuperado el 20 de junio de 2021, de Ebay: https://www.ebay.es/itm/353130720536?var=622328240494

[20] García, J. A. (s.f.). *Así funciona el motor de corriente continua o directa*. Recuperado el 20 de junio de 2021, de ¡Así funciona!: http://www.asifunciona.com/electrotecnia/af_motor_cd/af_motor_cd_3.htm

[21] *Constitución de un motor de corriente continua*. (s.f.). Recuperado el 20 de junio de 2021, de Automatismo industrial: https://automatismoindustrial.com/curso-carnet-instalador-baja-tension/motores/1-3-5-motores-de-corriente-continua/constitucion-motor-corriente-continua/

[22] paradacreativa. (13 de mayo de 2019). *Motor de corriente continua; tipos y partes*. Recuperado el 20 de junio de 2021, de Tercesa: https://tercesa.com/noticias/motor-de-corriente-continua-tipos-y-partes/

[23] *Bobinado de estator de Motor DC EC, hecho en China*. (s.f.). Recuperado el 20 de junio de 2021, de Alibaba.com: https://spanish.alibaba.com/product-detail/dc-ec-motor-stator-winding-made-in-china-60551109943.html

[24] *Motor de corriente continua (CD)*. (17 de abri de 2017). Recuperado el 20 de junio de 2021, de Ingeniería Mecafenix: https://www.ingmecafenix.com/electricidad-industrial/motor-corriente-continua/

[25] Llamas, L. (16 de mayo de 2016). *Controlar motores de corriente continua con Arduino y L298N*. Recuperado el 21 de junio de 2021, de Luis Llamas: https://www.luisllamas.es/arduino-motor-corriente-continua-l298n/

[26] *L298N Dual H-Bridge Motor Driver*. (s.f.). Recuperado el 21 de junio de 2021, de Handson Technology: http://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf

[27] Fernández, Y. (3 de agosto de 2020). *Qué es Arduino, cómo funciona y qué puedes hacer con uno*. Recuperado el 23 de junio de 2021, de Xataka: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

[28] (s.f.). Recuperado el 23 de junio de 2021, de Wikipedia: https://es.m.wikipedia.org/wiki/Archivo:Arduino_Logo.svg

[29] Sy Corvo, H. (23 de octubre de 2019). *Arquitectura Harvard: origen, modelo, cómo funciona*. Recuperado el 23 de junio de 2021, de lifeder: https://www.lifeder.com/arquitectura-harvard/

[30] *Arduino UNO REV3*. (s.f.). Recuperado el 24 de junio de 2021, de Arduino Official Store: https://store.arduino.cc/arduino-uno-rev3

[31] *Arduino Leonardo with headers*. (s.f.). Recuperado el 24 de junio de 2021, de Arduino Official Store: https://store.arduino.cc/arduino-leonardo-with-headers

[32] *Arduino Nano*. (s.f.). Recuperado el 24 de junio de 2021, de Arduino Official Store: https://store.arduino.cc/arduino-nano

[33] *Arduino Micro*. (s.f.). Recuperado el 24 de junio de 2021, de Arduino Official Store: https://store.arduino.cc/arduino-micro

[34] *Arduino Yún*. (s.f.). Recuperado el 24 de junio de 2021, de Arduino Official Store: https://store.arduino.cc/arduino-yun

[35] *Arduino Due*. (s.f.). Recuperado el 24 de junio de 2021, de Arduino Official Store: https://store.arduino.cc/arduino-due

[36] Gómez, E. (19 de diciembre de 2017). *Qué es PWM y para qué sirve*. Recuperado el 24 de junio de 2021, de Rincon ingenieril: https://www.rinconingenieril.es/que-es-pwm-y-para-que-sirve/

[37] *SPP-C Bluetooth (compatible HC-06)*. (s.f.). Recuperado el 24 de junio de 2021, de Robotica Fácil: https://roboticafacil.es/prod/spp-c-bluetooth/

[38] *Comunicación serie*. (s.f.). Recuperado el 24 de junio de 2021, de IBM: https://www.ibm.com/docs/es/aix/7.1?topic=communications-serial-communication

[39] *Serial Communication*. (s.f.). Recuperado el 24 de junio de 2021, de sparkfun: https://learn.sparkfun.com/tutorials/serial-communication/wiri

[40] Ahmad, N., Khairi, N., & Raja Ghazilla, R. A. (diciembre de 2013). Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications. *International Journal of Signal Processing Systems, 1* (2), 256-262.

[41] Llamas, L. (7 de septiembre de 2016). *Medir la inclinación con IMU, Arduino y filtro complementario*. Recuperado el 24 de junio de 2021, de Luis Llamas: https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/

[42] Qiang, L., Li, Q., Li, R., Dai, W., & Ji, K. (2015). Kalman Filter and Its Application. *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, (págs. 74-77).

[43] Estrada, H. (2009). *MEMS - Aplicaciones en Metrología*. Recuperado el 25 de junio de 2021, de https://www.cenam.mx/dme/pdf/PRE_E-Mie-2.pdf

[44] Llamas, L. (4 de septiembre de 2016). *Cómo usar un giroscopio en nuestros proyectos de Arduino*. Recuperado el 25 de junio de 2021, de Luis Llamas: https://www.luisllamas.es/como-usar-un-giroscopio-arduino/

[45] Llamas, L. (1 de septiembre de 2016). *Cómo usar un acelerómetro en nuestros proyectos de Arduino*. Recuperado el 25 de junio de 2021, de Luis Llamas: https://www.luisllamas.es/como-usar-un-acelerometro-arduino/

[46] *I2C – Puerto, Introducción, trama y protocolo*. (s.f.). Recuperado el 26 de junio de 2021, de Hetpro: https://hetpro-store.com/TUTORIALES/i2c/

[47] *SOLIDWORKS. Qué es y para qué sirve*. (s.f.). Recuperado el 6 de agosto de 2021, de Solid Bi: https://solid-bi.es/solidworks/

[48] *SolidWorks logo*. (s.f.). Recuperado el 6 de agosto de 2021, de 1000marcas: https://1000marcas.net/solidworks-logo/

[49] *App Inventor*. (30 de junio de 2021). Recuperado el 10 de agosto de 2021, de Wikipedia: https://es.wikipedia.org/wiki/App_Inventor

[50] *About Us*. (s.f.). Recuperado el 10 de agosto de 2021, de MIT App Inventor: https://appinventor.mit.edu/about-us

[51] Collazo Cuevas, J. I., Gorrostieta Hurtado, E., Pedraza Ortega, J. C., Villaseñor Carrillo, U. G., Romero Torrez, R. A., & González Aguirre, M. A. (26 y 27 de noviembre de 2009). Modelación de un Robot Móvil de Dos Ruedas con Tracción Diferencial. *8º Congreso Nacional de Mecatrónica*, (págs. 306-309). Veracruz.

ANEXO: CÓDIGO

```
////////INICIALIZACION////////

//INCLUSIÓN DE LIBRERIAS
#include <MPU6050.h>
#include <Wire.h>
#include <I2Cdev.h>
#include <SoftwareSerial.h>

//DEFINICIÓN DE LOS PINES A UTILIZAR DEL ARDUINO
#define In1 6
#define In2 7
#define In3 9
#define In4 8
#define EnA 10
#define EnB 11
#define Rx 12
#define Tx 13

//DIRECCIÓN I2C DEL MPU-6050
#define mpuAddress 0x68

//DEFINICIÓN DE LAS ACCIONES DEL ROBOT EN FUNCIÓN DE LA APP
#define arriba 2
#define paro 1
#define abajo 3
#define giroDerecha 4
#define giroIzquierda 5

//VARIABLES GLOBALES PARA EL CALCULO DEL PID DE POSICIÓN
float inclinacionRef = 0.6;
double posKp = 26;
double posKd = 0.8;
double posKi = 0;
uint8_t tMuestreoPos = 3;

long tAnteriorPos = 0;
float errorPosAnterior = 0;
float sumaErrorPos = 0;
float posPID = 0;

//VALOR MÁXIMO DEL PID
uint8_t salidaMax = 255;

//ERROR LIMITE DEL ROBOT PARA DEJAR DE FUNCIONAR
uint8_t errorLimite = 30;

//VARIABLES GLOBALES PARA EL CALCULO DEL ANGULO DE INCLINACION
int16_t ax,ay,az,gx,gy,gz;
float angX;
float dt;

float angAnteriorX = 0;
long tAnterior=0;

//long movAnteriorT = 0;

//DECLARACIÓN DE LAS VARIABLES DEL SENSOR Y EL MODULO BT
MPU6050 sensor(mpuAddress);
SoftwareSerial bt(Rx,Tx);

////////SETUP////////
```

```
void setup() {
  //DECLARACIÓN DE LOS PINES DEL DRIVER
  pinMode(In1, OUTPUT);
  pinMode(In2, OUTPUT);
  pinMode(In3, OUTPUT);
  pinMode(In4, OUTPUT);
  pinMode(EnA, OUTPUT);
  pinMode(EnB, OUTPUT);

  //DECLARACION DE LOS PINES DEL BT
  pinMode(Rx, INPUT);
  pinMode(Tx, OUTPUT);
  //INICIALIZACION DE LA COMIUNICACION CON EL MODULO BT
  bt.begin(9600);

  //INICIALIZACIÓN DE LA COMUNICACION CON EL PUERTO SERIE
  Serial.begin(57600);

  //INICIALIZACION DE LA COMUNICACION CON EL SENSOR
  Wire.begin();
  sensor.initialize();
}

////////LOOP////////
void loop() {
  uint8_t comando = leerBT(); //LECTURA DEL MODULO BT
  obtenerAngulos(); //OBTENCION DEL ANGULO DE INCLINACION
  float error = inclinacionRef-angX; //CALCULO DEL ERROR DE POSICION

//  long movT = millis() - movAnteriorT;

  //SE COMPRUEBA QUE LA INCLINACION SE ENCUENTRA DENTRO DE LOS LIMITES
  if (error < errorLimite && error > -errorLimite) //DENTRO DE LOS
LIMITES
  {
    if(movT >= 3*tMuestreoPos)
    {
      movAnteriorT = millis();
      mueveRobot(comando);
    }
//    else
//    {

      posicionPID(error); //CALCULO DEL CONTROL PID
      mueveRuedas(posPID,0); //ACCIONAMIENTO DE LOS MOTORES
//    }
  }
  else mueveRuedas(0,0); //FUERA DE LOS LIMITES SE PARAN LOS MOTORES
}

////////FUNCIONES////////

//LECTURA DE LA INFORMACION DEL SENSOR Y CALCULO DEL ANGULO DE
INCLINACION
float obtenerAngulos()
{
  dt = (millis()-tAnterior); //TIEMPO TRANSCURRIDO DESDE EL ULTIMO
CALCULO DEL ANGULO
  tAnterior = millis(); //ALMACENAMIENTO DEL TIEMPO ACTUAL PARA EL
PROXIMO BUCLE
```

```
  //DECLARACION DE VARIABLES LOCALES
  float accelX;
  sensor.getAcceleration(&ax,&ay,&az);
  sensor.getRotation(&gx,&gy,&gz);
  float K = 0.98;

  //CALCULO DE LA INCLINACION A PARTIR DEL ACELEROMETRO
  accelX = atan(ax / sqrt(pow(ay, 2) + pow(az, 2)))*(180.0 /
3.14159265);

  //CALCULO DE LA INCLINACION MEDIANTE EL FILTRO COMPLEMENTARIO
  angX = K*(angAnteriorX - gy*dt/131000) + (1-K)*accelX;

  angAnteriorX = angX; //ALMACENAMIENTO DEL ANGULO ACTUAL PARA EL
PROXIMO BUCLE
}

//LECTURA DEL MODULO BT
uint8_t leerBT()
{
  if(bt.available() > 0) //COMPRUEBA QUE SE HAN RECIBIDO DATOS
  {
    return bt.read(); //SI SE HAN RECIBIDO, SE DEVUELVE LO QUE HAYA
LLEGADO
  }
}

//ACCIONAMIENTO DE LAS RUEDAS DEL MOTOR
void mueveRuedas(int velocidad, uint8_t giro)
{
  int velocidadDerecha,velocidadIzquierda; //DECLARACION DE VARIABLES
LOCALES

  //SI LA VELOCIDAD SUPERA EL LIMITE, SE ESTABLECE LA VELOCIDAD MAXIMA
DECLARADA
  if (velocidad > salidaMax) velocidad = salidaMax;
  else if (velocidad < -salidaMax) velocidad = -salidaMax;

  //SE ESTABLECE LA VELOCIDAD PARA CADA RUEDA
  velocidadDerecha = velocidad;
  velocidadIzquierda = velocidad;

  if (giro == giroDerecha)
  {
    //SI HAY QUE GIRAR A LA DERECHA, SE VARIA LA VELOCIDAD DE LA RUEDA
DERECHA
    velocidadDerecha = velocidadDerecha - 30;
    if (velocidadDerecha < -salidaMax) velocidadDerecha = -salidaMax;
  }
  else if (giro == giroIzquierda)//Comprobar cuando retrocede a maxima
velocidad
  //SI HAY QUE GIRAR A LA IZQUIERDA, SE VARIA LA VELOCIDAD DE LA RUEDA
IZQUIERDA
    velocidadIzquierda = velocidadIzquierda - 30;
    if (velocidadIzquierda < -salidaMax) velocidadIzquierda = -
salidaMax;
  }

  //ACIONAMIENTO DE LOS MOTORES
  if (velocidad < 0) //SI LA VELOCIDAD ES NEGATIVA
  {
```

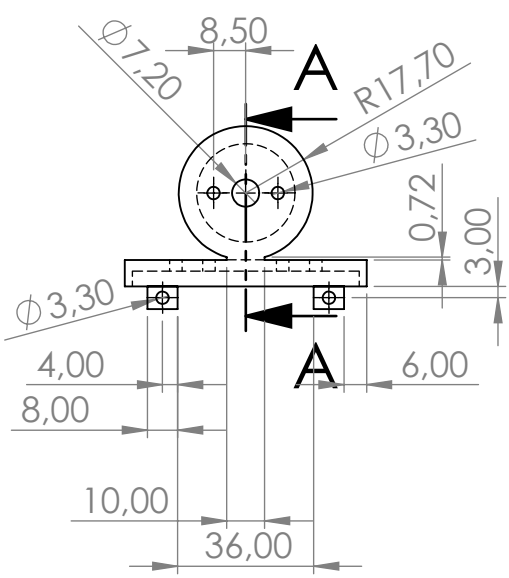
```
    //CONFIGURACION PARA QUE AMBOS MOTORES RETROCEDAN
    digitalWrite(In1, LOW);
    digitalWrite(In2, HIGH);
    analogWrite(EnA, -velocidadDerecha);
    digitalWrite(In3, LOW);
    digitalWrite(In4, HIGH);
    analogWrite(EnB, -velocidadIzquierda);
  }
  else if (velocidad > 0) //SI LA VELOCIDAD ES POSITIVA
  {
    //CONFIGURACION PARA QUE AMBOS MOTORES AVANCEN
    digitalWrite(In2, LOW);
    digitalWrite(In1, HIGH);
    analogWrite(EnA, velocidadDerecha);
    digitalWrite(In4, LOW);
    digitalWrite(In3, HIGH);
    analogWrite(EnB, velocidadIzquierda);
  }
  else //SI VELOCIDAD = 0, MOTORES SE PARAN
  {
    analogWrite(EnA, velocidad);
    analogWrite(EnB,velocidad);
  }
}

//CALCULO DEL CONTROL PID
void posicionPID(float error)
{
  //DECLARACION Y CALCULO DE VARIABLES LOCALES
  long tInicioPos = millis(); //OBTENCION DEL TIEMPO ACTUAL
  long T = tInicioPos - tAnteriorPos; //TIEMPO TRANSCURRIDO DESDE EL
ULTIMO CALCULO

  if (T >= tMuestreoPos)
  { //SI EL TIEMPO TRANSCURRIDO ES MAYOR O IGUAL AL TIEMPO DE MUESTREO
    tAnteriorPos = tInicioPos; //ALMACENAMIENTO DEL TIEMPO ACTUAL PARA
EL PROXIMO BUCLE

    float PID_P = posKp*error; //CALCULO PARTE PROPORCIONAL

    float PID_D = posKd*(error - errorPosAnterior)*1000/T; //CALCULO
PARTE DERIVATIVA
    errorPosAnterior = error; //ALMACENAMIENTO DEL ERROR PARA EL
PROXIMO BUCLE

    float PID_I = posKi*(sumaErrorPos + error)*(T/1000); //CALCULO
PARTE INTEGRAL
    sumaErrorPos = sumaErrorPos + error; //ALMACENAIENTO DEL ERROR
ACUMULADO

    posPID = PID_P+PID_D+PID_I; //CALCULO DEL CONTROL PID
  }
}

//MOVER EL ROBOT
void mueveRobot(uint8_t comando)
{

  if (comando < 6 && comando > 1) //SI SE HA ORDENADO MOVER EL ROBOT
  {
    uint8_t velMovimiento = 90; //ESTABLECER VELOCIDAD DE LAS RUEDAS
```

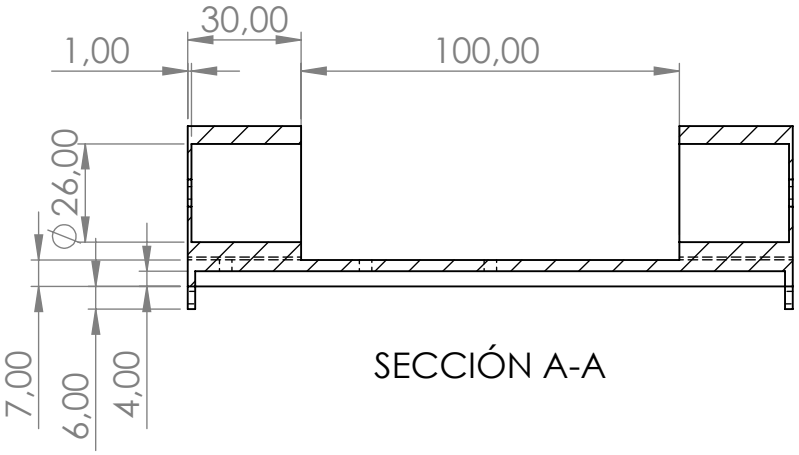
```
    if(comando = abajo) velMovimiento = -velMovimiento; //VELOCIDAD DE
LAS RUEDAS SI HAY QUE RETROCEDER
    mueveRuedas(velMovimiento, comando); //LLAMADA A LA FUNCION
MUEVERUEDAS
  }
  else if(comando == 1) mueveRuedas(0,comando); //SI SE HA ORDENADO
PARAR EL MOTOR, SE PARA
}
```

DOCUMENTO Nº 2.

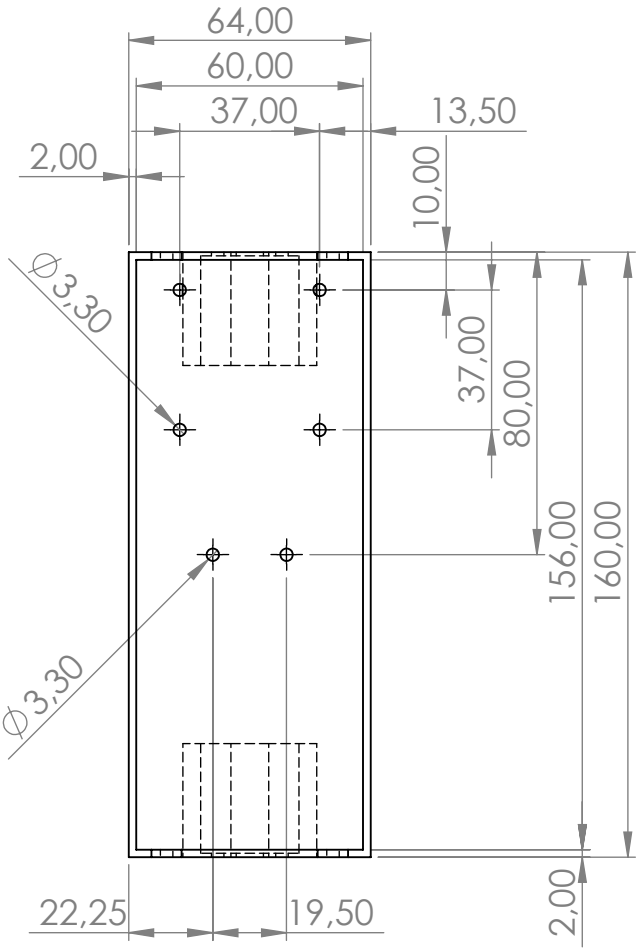
PLANOS

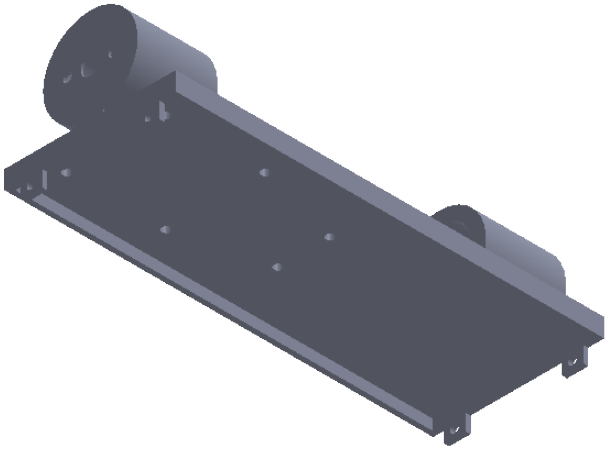
Índice



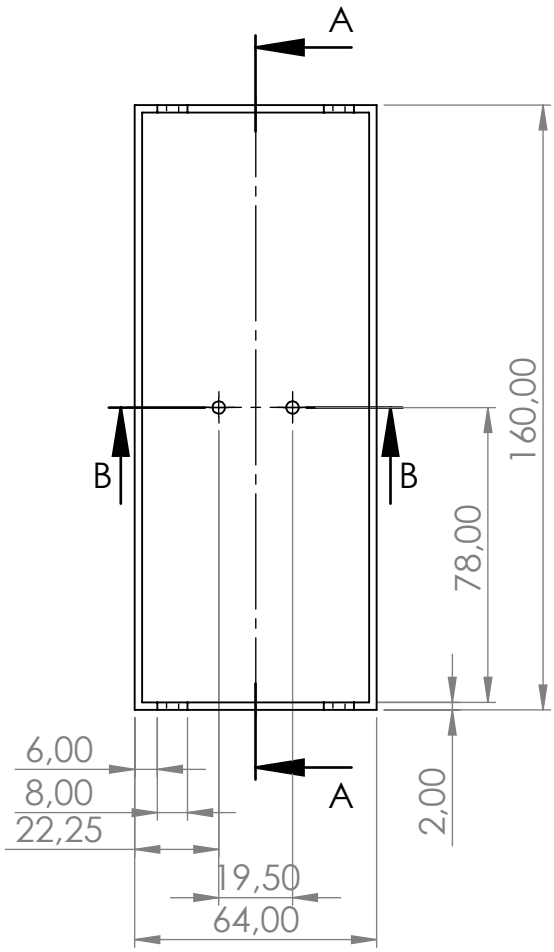


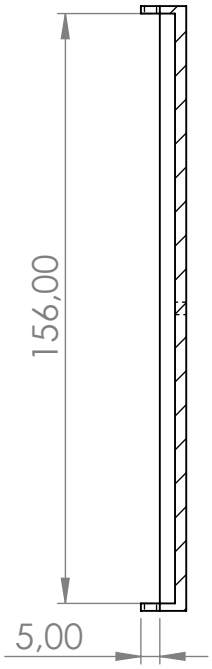
SECCIÓN A-A



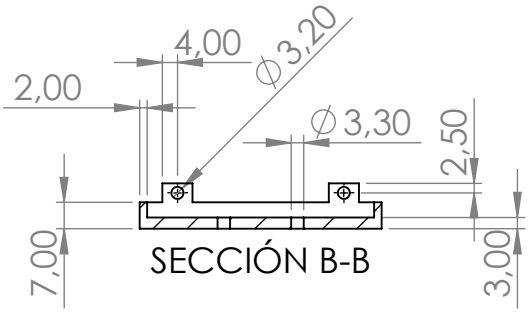


UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Base con soportes para motor	
	Escala: 1:2	Referencia: P-01
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 04/06/2021	Plano nº: 1	

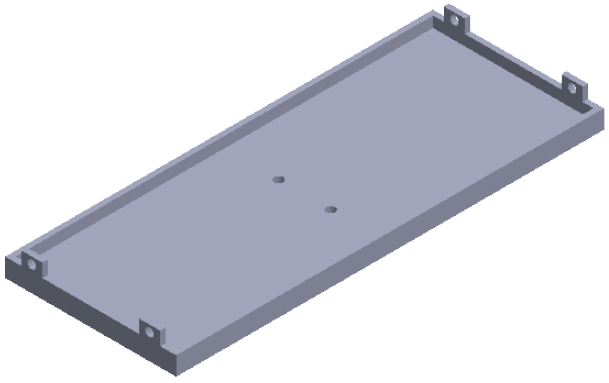




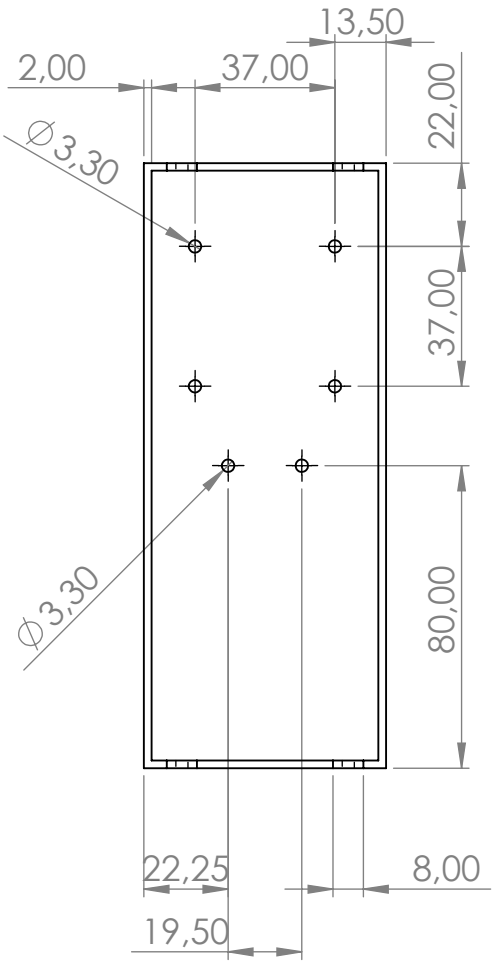
SECCIÓN A-A

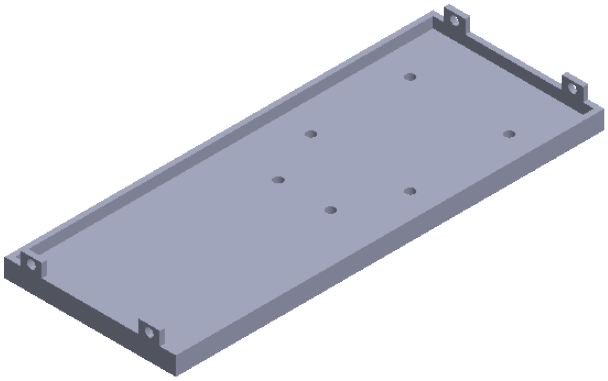


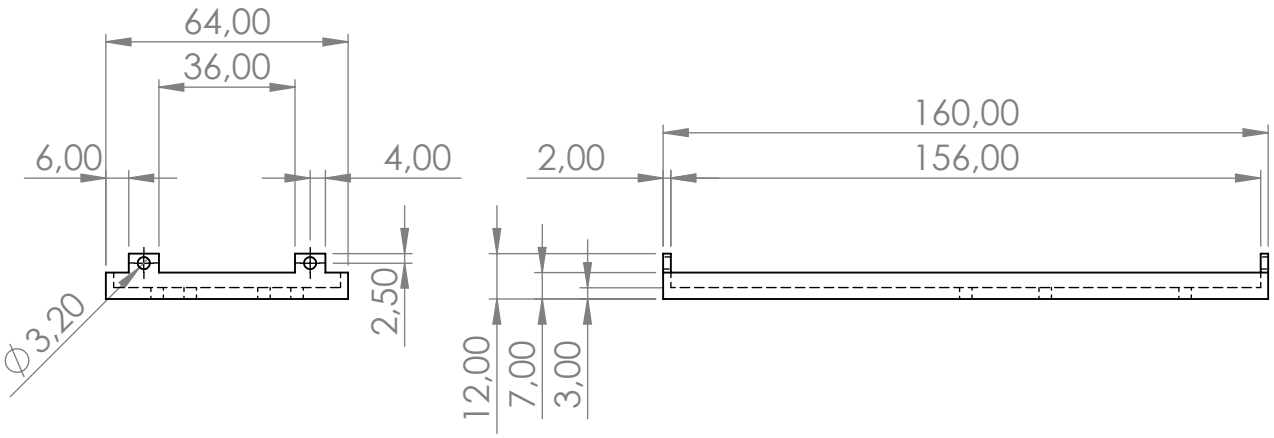
SECCIÓN B-B



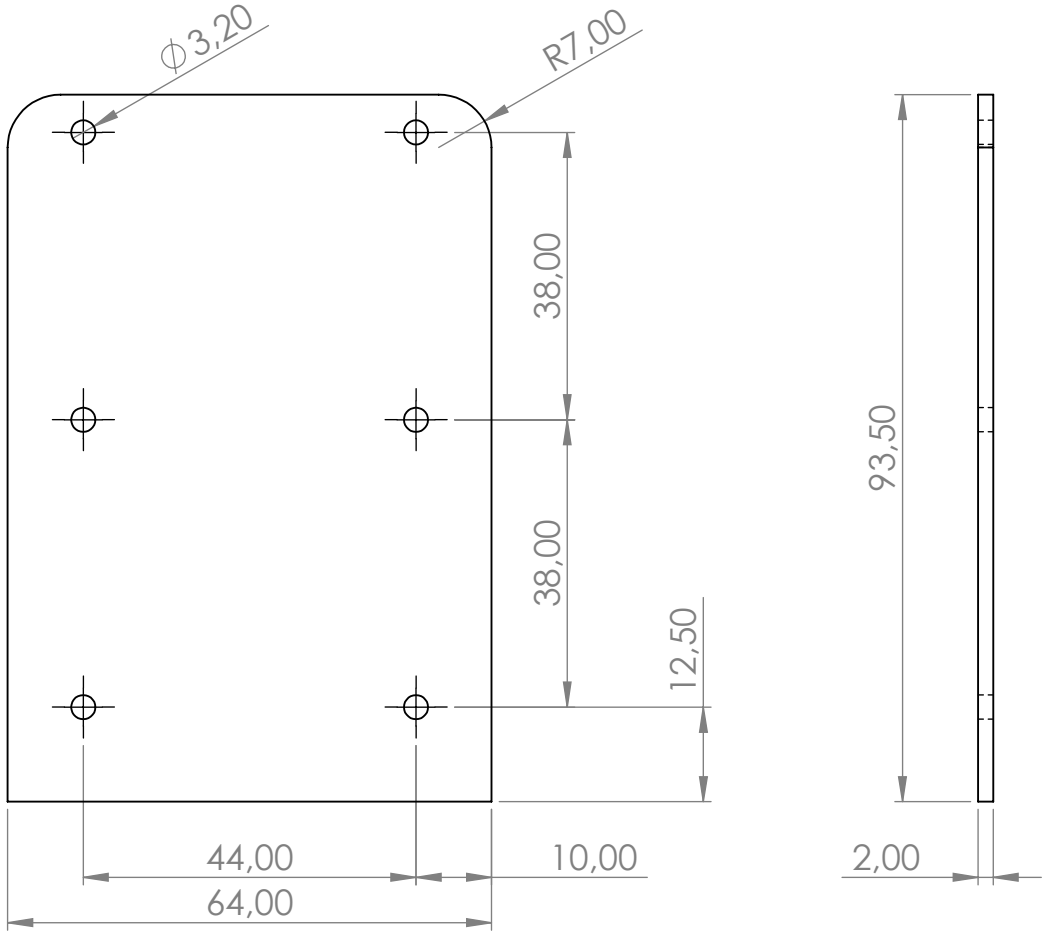
UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Base con agujeros	
	Escala: 1:2	Referencia: P-02
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 05/06/2021	Plano nº: 2	



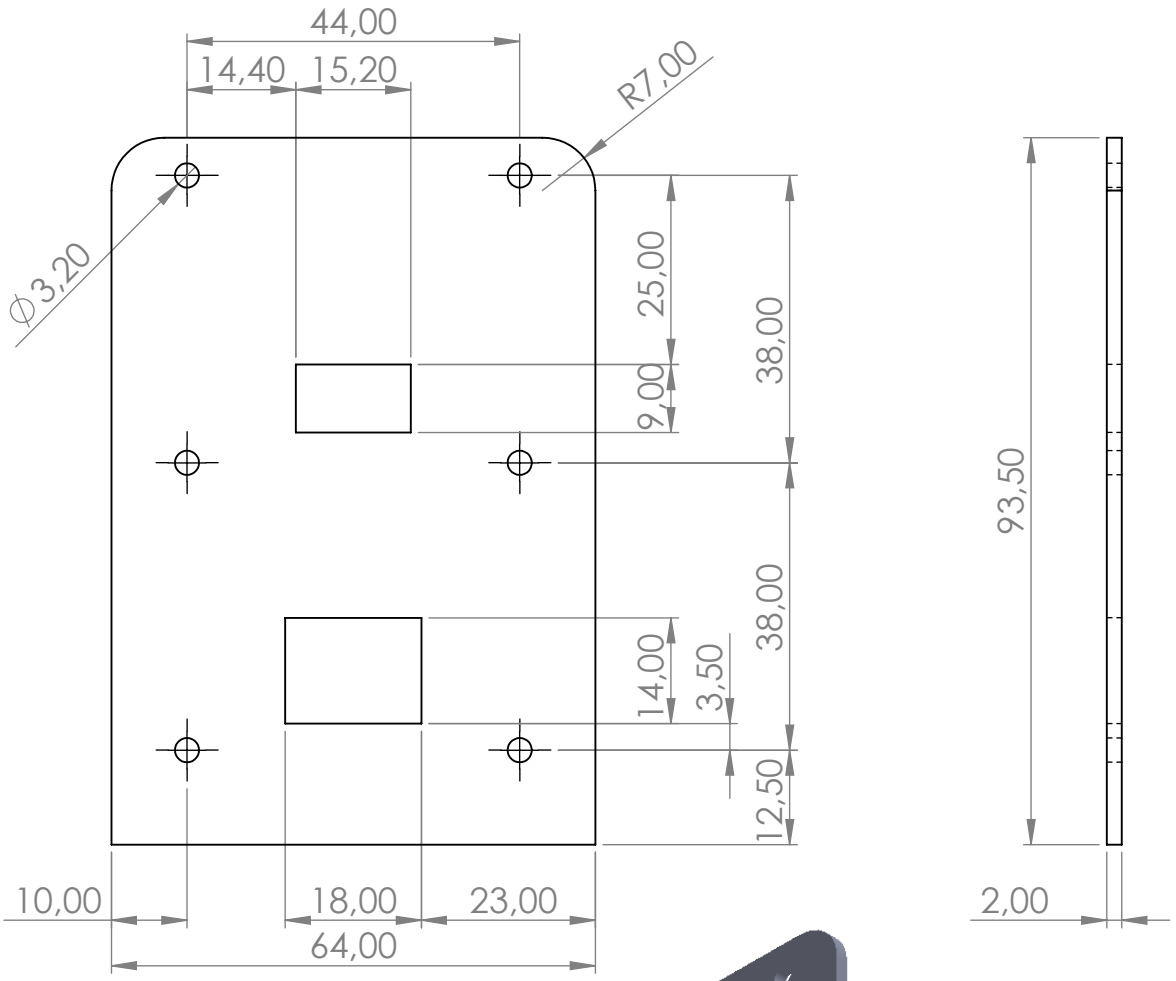




UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Base con agujeros para Driver	
	Escala: 1:2	Referencia: P-03
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 04/06/2021	Plano nº: 3	

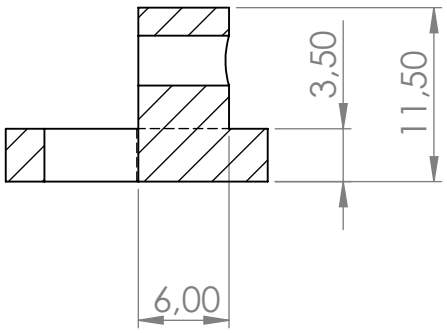


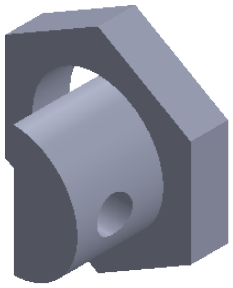
UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Lateral sin agujeros	
	Escala: 1:1	Referencia: P-04
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 05/06/2021	Plano nº: 4	

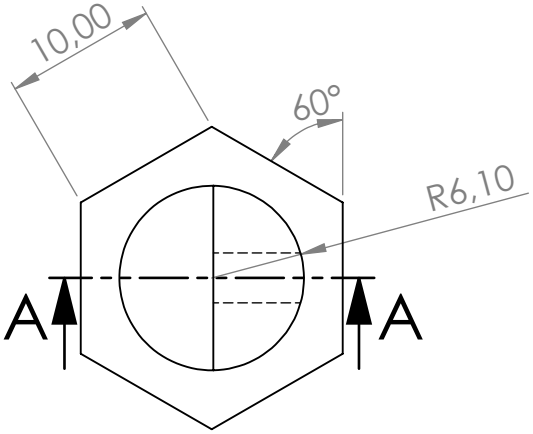


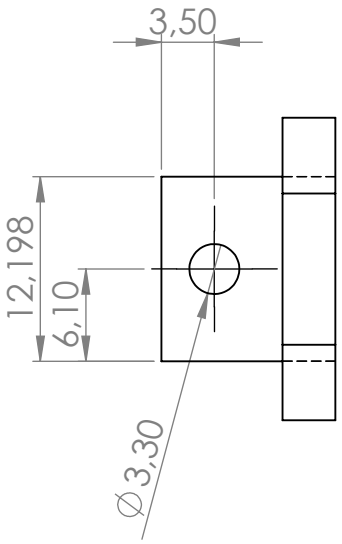
UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Lateral con agujeros	
	Escala: 1:1	Referencia: P-05
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 05/06/2021	Plano nº: 5	

SECCIÓN A-A



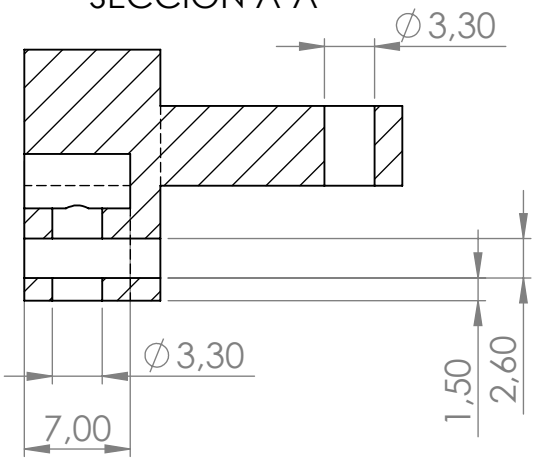


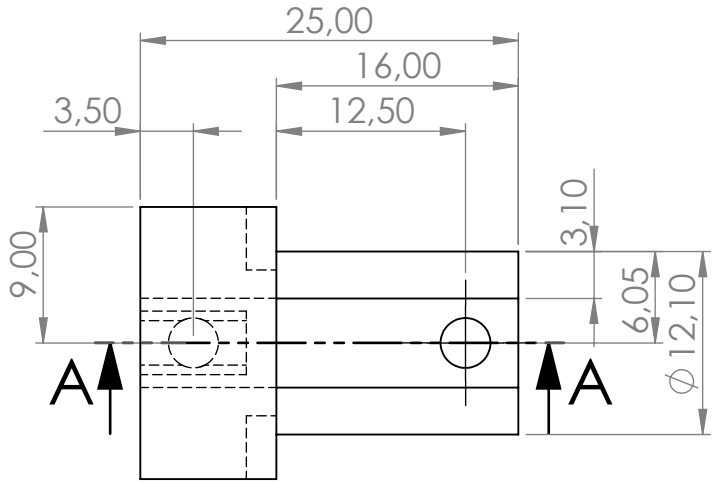


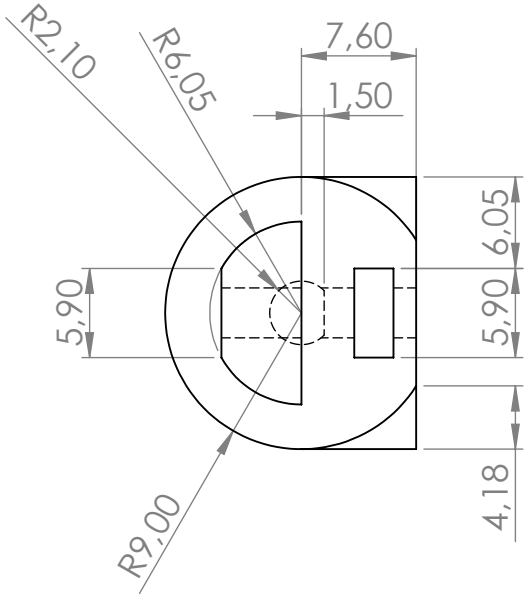


UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Acople para rueda 1	
	Escala: 2:1	Referencia: P-06
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 05/08/2021	Plano nº: 6	

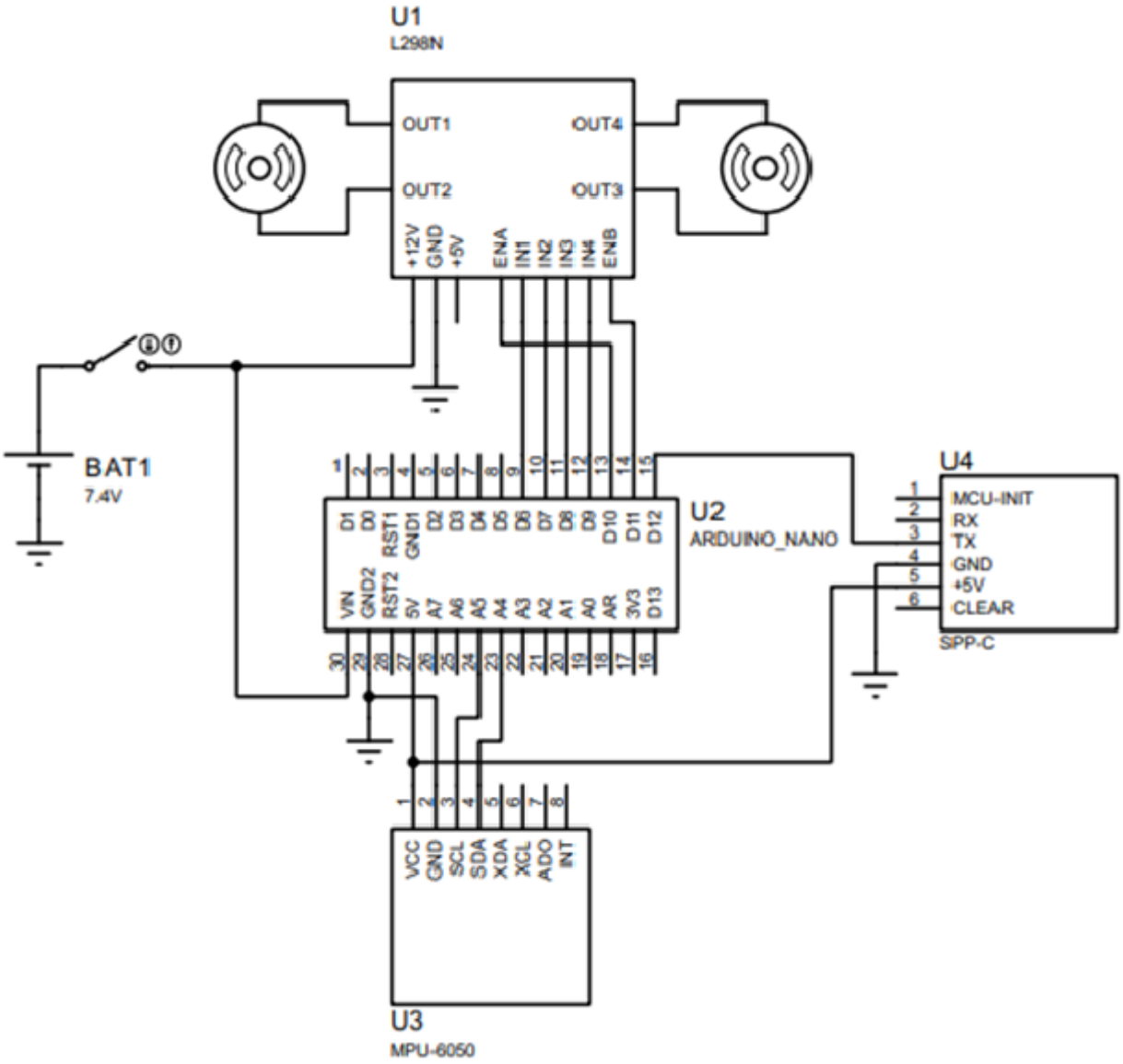
SECCIÓN A-A







UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Acople para rueda 2	
	Escala: 2:1	Referencia: P-07
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 05/08/2021	Plano nº: 7	



UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Nombre del plano: Esquema electrónico	
	Escala:	Referencia: EE-01
Proyecto: Diseño y control de un robot móvil inestable usando Arduino		
Autor: José Antonio Méndez Ibarra	Promotor: José Antonio Méndez Ibarra	
Fecha: 21/06/2021	Plano nº: 8	

DOCUMENTO Nº 3.

PLIEGO DE CONDICIONES

Índice

1. Objeto

En este documento se detallan las especificaciones técnicas del robot móvil inestable realizado en este Trabajo Fin de Grado.

Se incluye la parte mecánica y electrónica necesaria para el funcionamiento del prototipo (hardware) estableciendo las características y condiciones necesarias para el funcionamiento del prototipo, además del proceso de control de calidad de dichas partes.

2. Condiciones de los materiales

Los materiales mencionados a continuación han sido comprados a una empresa externa, en el caso de los componentes electrónicos y las ruedas, o fabricados mediante una impresora 3D en propiedad, en el caso de las piezas del robot. Con los elementos comprados a terceros se han realizado comprobaciones en cuanto a medidas y especificaciones y que se encuentran en buen estado, mientras que los fabricados se han sometido a medidas asegurando la posibilidad de ensamblar el prototipo.

2.1. Descripción

2.1.1. Actuadores

- Motores
  Motores de corriente continua con tensión de alimentación máxima de 6 V y velocidad máxima de giro sin carga del eje de 177 rpm. Tiene un diámetro externo de 25 mm y el eje, de 4 mm. Cuenta con dos agujeros para tornillos a los lados del eje.

- Driver L298N
  Compatible con fuentes de alimentación de hasta 35 V, capaz de alimentar el circuito digital a 5 V cuando es alimentado a menos de 12 V sin necesidad de alimentación externa. Posee 9 entradas, 3 de alimentación y 6 para el control de motores, y 4 salidas que permiten el control de dos motores CC o uno paso a paso con una corriente máxima de 2 A. Cuenta con 4 agujeros para tornillos en las esquinas.

2.1.2. Control

- Placa de Arduino NANO
  Microcontrolador ATMega328 con un voltaje operativo de 5 V, con 14 entradas/salidas digitales (6 salidas PWM) y 8 entradas analógicas. Posee un conector mini-USB hembra y un botón reset. Cuenta con una velocidad de reloj de 16 MHz y una memoria Flash de 32 Kb.

- Sensor MPU-6050
  IMU con acelerómetro y giroscopio con valores máximos de medición programables y conversores ADC de 16 bits. La tensión de alimentación es de 5 V y se comunica mediante el protocolo I2C.

- Módulo bluetooth
  Módulo bluetooth SPP-C con tensión de alimentación entre 3,3 V y 5,6 V. Trabaja en frecuencias entre 2,4 GHz y 2,48 GHz y se comunica mediante puerto serie.

2.1.3. Alimentación

- Baterías Li-Ion
  Baterías de ion de litio con tensión de 3,7 V y consumo de corriente de 2500 mAh. Tienen un diámetro de 18 mm y una longitud de 65 mm.

- Interruptor
  Interruptor basculante de 3 A y 250 V de corriente alterna con dos posiciones: abierto y cerrado. Tiene 15 mm de longitud, 10 mm de anchura y 17 mm de profundidad.

2.1.4. Ruedas

- Ruedas
  Ruedas de escala 1/8 (10 cm de diámetro) y ancho de 41 mm con neumáticos de goma. Agujero para acople circular en la cara externa de 12,5 mm y agujero hexagonal en la cara interna de 10 mm de lado.

2.1.5. Piezas

Para más información de los materiales que se muestran a continuación, complementar con el "Documento Nº 2: Planos".

- Base con soportes para motor
  Pieza de plástico rectangular de 160x64x3 mm con salientes en los bordes de 4 mm de altura y 2 mm de grosor, además de 4 salientes rectangulares de 8x6 mm con agujeros para tornillo de 3 mm. Cuenta con 6 agujeros para tornillos en la base. En los extremos de la parte inferior cuenta con dos salientes cilíndricos de diámetro interior de 26 mm. Información complementaria en el plano "P-01".

- Base con agujeros
  Pieza de plástico rectangular de 160x64x3 mm con salientes en los bordes de 4 mm de altura y 2 mm de grosor, además de 4 salientes rectangulares de 8x5 mm con agujeros para tornillo de 3 mm. Cuenta con 2 agujeros para tornillos en la base. Información complementaria en el plano "P-02".

- Base con agujeros para Driver
  Pieza de plástico rectangular de 160x64x3 mm con salientes en los bordes de 4 mm de altura y 2 mm de grosor, además de 4 salientes rectangulares de 8x5 mm con agujeros para tornillo de 3 mm. Cuenta con 6 agujeros para tornillos en la base. Información complementaria en el plano "P-03".

- Lateral sin agujeros
  Pieza de plástico rectangular de 93,5x64x2 mm con redondeo en las esquinas superiores de radio 7 mm. Cuenta con 6 agujeros para tornillos. Información complementaria en el plano "P-04".

- Lateral con agujeros
  Pieza de plástico rectangular de 93,5x64x2 mm con redondeo en las esquinas superiores de radio 7 mm. Cuenta con 6 agujeros para tornillos y 2 rectangulares para cable mini-USB e interruptor. Información complementaria en el plano "P-05".

- Acople para rueda 1
  Pieza de plástico con base hexagonal de 10 mm de lado, agujero central semicircular de 6,1 mm y saliente semicircular, con agujero para tornillo, de 6,1 mm y longitud 8 mm. Información complementaria en el plano "P-06".

- Acople para rueda 2
  Pieza de plástico con circular de 18 mm de diámetro y 9 mm de altura con agujeros para entrada de tornillo, tuerca y eje de motor. Cuenta con un saliente semicircular con, con agujero para tornillo, de radio 6,05 mm y longitud 16 mm. Información complementaria en el plano "P-07".

2.2. Control de calidad

Se detalla el proceso realizado en cada componente para comprobar que todos los componentes electrónicos cumplen con las especificaciones y las piezas, con las dimensiones de los planos.

2.2.1. Actuadores

- Motores
  Se ha alimentado con una batería de 5 V en ambas polaridades comprobando que gira en ambos sentidos a una velocidad de entre 2 o 3 revoluciones por segundo.

- Driver L298N
  Se ha alimentado con 7,4 V y se ha comprobado que se puede realizar el control de sentido de giro y velocidad a través de sus pines. Las pruebas se han realizado con dos motores CC para comprobar que la velocidad es igual en ambos y no hay fallos en el suministro de corriente.

2.2.2. Control

- Placa de Arduino NANO
  Se ha puesto a prueba el correcto funcionamiento de los pines a utilizar poniéndolos a nivel alto o con salida PWM y midiendo con un voltímetro. Se ha alimentado la placa con un cable mini-USB conectado a un ordenador y con una alimentación de 7,4 V en el pin "Vin" comprobando la correcta alimentación del componente.

- Sensor MPU-6050
  Se ha alimentado con 5 V comprobando que se alimenta correctamente y se han realizado pruebas junto al microcontrolador para ver si la comunicación se establece correctamente.

- Módulo bluetooth
  Se ha alimentado con 5 V comprobando que se alimenta correctamente y se han realizado pruebas junto al microcontrolador y un teléfono móvil para ver si se establece comunicación bluetooth y los datos se transmiten correctamente.

2.2.3. Alimentación

- Baterías Li-Ion
  Se han medido con un voltímetro para asegurar que la tensión es de 3,7 V.

- Interruptor
  Se ha comprobado con un voltímetro que no existe continuidad entre los bornes cuando está abierto y sí la hay cuando está cerrado.

2.2.4. Ruedas

- Ruedas
  Se ha comprobado que el diámetro es de 10 cm y se ha observado si había roturas en la pieza.

2.2.5. Piezas

El control de calidad de las piezas se ha basado en realizar diversas medidas para comprobar que coinciden con los planos realizados y revisar que no hay grietas o roturas en ellas.

3. Condiciones de la ejecución

3.1. Descripción

Se procede a enumerar los pasos que se deben seguir para realizar el montaje del robot.

Todos los tornillos y tuercas mencionados en los siguientes pasos tienen un diámetro de 3 mm. Para más información sobre las piezas que se van a nombrar, consultar “Documento Nº 2: Planos”.

1. Se introducirán los motores en los cilindros semicirculares de la pieza “Base con soportes para motor” hasta que el eje haya pasado correctamente.

2. Se atornillarán los motores a los soportes utilizando tornillos de 6 mm de longitud.

3. Se deberá atornillar un soporte de baterías a la pieza con tornillos de 6 mm de longitud y tuercas.

4. Se pondrá un conector macho en el cable negativo del soporte de baterías con una crimpadora.

5. Se colocarán dos baterías en el soporte.

6. Se atornillará la parte inferior de los dos laterales (“Lateral sin agujeros” y “Lateral con agujeros”) a la base con tornillos de 6 mm de longitud y tuercas.

7. Se atornillará la pieza “Base con agujeros” en la parte central de los dos laterales mencionados con tornillos de 6 mm de longitud y tuercas.
8. Se colocará el interruptor en el agujero rectangular superior del “Lateral con agujeros”.
9. Se soldarán sus terminales a dos cables. Uno de ellos de 20 cm macho-macho y el otro, 10 cm macho-hembra.
10. Se deberá atornillar un soporte de baterías a la base anterior con tornillos de 6 mm de longitud y tuercas.
11. Se soldará el conector negativo del soporte de baterías con el conector positivo del primer soporte colocado.
12. Se pondrá un conector macho en el cable positivo del soporte de baterías con una crimpadora.
13. Se colocarán dos baterías en el soporte.
14. Se atornillará la pieza “Base con agujeros para Driver” a la parte superior de los dos laterales mencionados anteriormente con tornillos de 6 mm de longitud y tuercas.
15. Se atornillará el driver a la base anterior con tornillos de 8 mm de longitud y tuercas.
16. Se pegará la protoboard en la superficie restante de la base lo más alejado posible del driver.
17. Se colocarán el Arduino NANO, el SPP-C y el MPU-6050 en la protoboard.
18. Se realizará la conexión de todos los componentes siguiendo el esquema eléctrico (para conectar cables en las entradas de alimentación del driver se necesitará un destornillador). Más información en “Documento Nº 2: Planos”, en el plano “EE-01”.

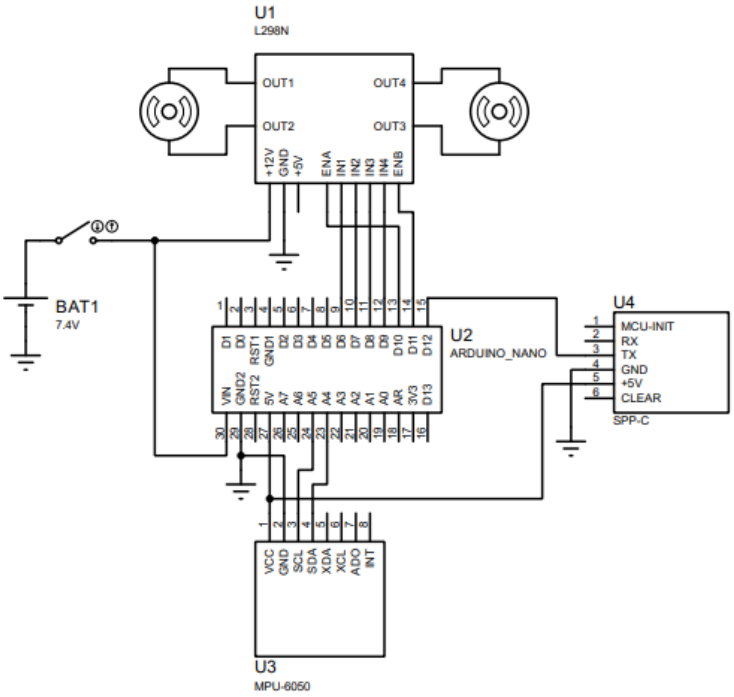


Figura 1: Esquema eléctrico

19. Se introducirá una tuerca en las piezas “Acople para rueda 2” hasta que su agujero sea concéntrico al de la pieza.

20. Se introducirán tornillos de 6 mm de longitud en las piezas anteriores hasta que se hayan enroscado un poco en las tuercas.

21. Se acoplarán las piezas “Acople para rueda 1” en las ruedas.

22. Se introducirán las piezas “Acople para rueda 2” en las ruedas, atravesando el semicírculo de “Acople para rueda 1” hasta que los agujeros de los salientes sean concéntricos.

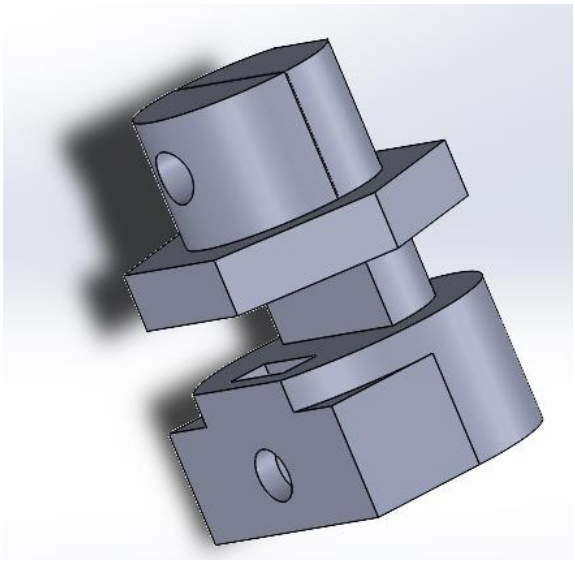


Figura 2: Ensamblaje sin rueda

23. Se atornillarán las dos piezas anteriores con tornillos de 16 mm de longitud y tuercas.
24. Se introducirán los ejes de los motores en los ensamblajes realizados con las ruedas.
25. Se atornillarán los tornillos de los ensamblajes de las ruedas hasta que hagan contacto con los ejes de los motores.
26. Se conectará el cable hembra del interruptor con el cable positivo del soporte de baterías.

3.2. Control de calidad

- Comprobación del conexionado del circuito, verificando que todos los componentes están siendo alimentados y que existe continuidad entre los distintos pines que deben estar conectados utilizando un voltímetro.
- Comprobación del correcto ensamblaje de los motores y las ruedas, verificando que están bien sujetas y, por tanto, no se caen.
- Medición de la tensión de alimentación, comprobando que la conexión entra baterías es correcta y proporciona la tensión deseada.
- Comprobación del ensamblaje completo, verificando que todos los componentes están bien sujetos y no hay peligro de que se separen o caigan.
- Verificación del movimiento de los motores, cerrando el interruptor de alimentación, pero sosteniendo el prototipo en la mano.
- Comprobación de la conexión bluetooth, cerrando el interruptor y conectando el teléfono móvil al módulo bluetooth.
- Comprobación de la medición de inclinación, conectando el robot al ordenador e inclinándolo manualmente para ver si la medición es correcta.

4. Pruebas y ajustes finales o de servicio

La prueba final realizada consiste en la puesta en marcha del robot sin control del movimiento mediante la aplicación de móvil, únicamente buscando la estabilidad del mismo en una posición vertical.

Para hacer esto se coloca el prototipo en el suelo en una posición vertical y se cierra el interruptor para que comience a funcionar. Seguidamente se comprueba si el robot es capaz de mantenerse en pie por sí mismo.

En caso de lograrlo, se procede a conectar el teléfono al robot para realizar el control remoto. Si en alguna de estas pruebas no se consigue el objetivo, significa que el robot es defectuoso y se debe buscar una solución.

Durante estas pruebas, logre o no funcionar correctamente, el robot se mantiene en funcionamiento todo el tiempo posible con el objetivo de determinar la duración de las

baterías y, por tanto, el tiempo que puede estar en funcionamiento el robot. Si este dejara de funcionar en poco tiempo, se buscaría una solución analizando si las baterías tienen algún problema y reemplazándolas en caso afirmativo.

DOCUMENTO Nº 4.

PRESUPUESTO

Índice

1. Presupuesto del prototipo

1.1. Coste de los materiales

1.1.1. Componentes electrónicos

	Elemento	Cantidad	Precio unitario (€)	Total (€)
Control	Arduino Nano + Cable mini-USB	1	7,49	7,49
	SPP-C Bluetooth	1	5,50	5,50
	MPU-6050 Acelerómetro + Giroscopio	1	1,98	1,98
Actuadores	Motor DC + Encoders	2	9,92	19,84
	Driver L298N	1	2,49	2,49
Alimentación	Batería Li-Ion 3,7 V	4	4,59	18,36
				55,66

El coste final de los componentes electrónicos es de **cincuenta y cinco euros con sesenta y seis céntimos**.

1.1.2. Piezas

Elemento	Cantidad	Peso (g)	Tiempo Impresión (h)	Kilo de hilo (€/kg)	Tiempo de impresión (€/h)	Total (€)
Lateral sin agujero	1	15	2	19,50	0,80	1,89
Lateral con agujero	1	15	2	19,50	0,80	1,89
Base	1	70	12	19,50	0,80	10,97
Medio para pila	1	35	5	19,50	0,80	4,68
Medio para Driver	1	35	5	19,50	0,80	4,68
Acople rueda 1	2	2	1	19,50	0,80	1,68
Acople rueda 2	2	3	1	19,50	0,80	1,72
						27,51

El coste final de las piezas es de **veintisiete euros con cincuenta y un céntimos**.

1.1.3. Gasto adicional en ruedas, cableado y cargador

	Elemento	Cantidad	Precio unitario (€)	Total (€)
Conexionado	Placa prototipo Protoboard	1	1,80	1,80
	Cable Hembra-Macho 20 cm	6	0,17	1,02
	Cable Macho-Macho 10 cm	12	0,05	0,60
	Cable Macho-Macho 20 cm	1	0,06	0,06
	Interruptor	1	0,95	0,95
	Conector Dupont Macho	2	0,02	0,04
	Conector BLS01 Hembra/Hembra 1 Pin	2	0,09	0,18
	Portapilas x2 Paralelo 18650 3,7 V	2	1,39	2,78
Recarga	Cargador 4 Pilas	1	8,99	8,99
Ruedas	Rueda de 10 cm diámetro	2	7,5	15,00
				31,42

El coste final de los componentes adicionales para conexión y recarga del prototipo es de **treinta y un euros con cuarenta y dos céntimos**

1.1.4. Tornillería

Elemento	Cantidad	Precio unitario (€)	Total (€)
Tornillo M3 6 mm	22	0,02	0,44
Tornillo M3 8 mm	4	0,02	0,08
Tuerca M3	24	0,01	0,24
Tornillo M3 16 mm	2	0,03	0,06
			0,82

El coste final de la tornillería es de **ochenta y dos céntimos**.

1.2. Coste de la mano de obra

Tarea	Cantidad (h)	Precio (€/h)	Total (€)
Montaje del prototipo	3	8	24,00
Conexionado del circuito	1	8	8,00
Soldadura	1	8	8,00
Crimpado cables	1	8	8,00
			48,00

El coste final de la mano de obra es de **cuarenta y ocho euros**.

1.3. Medios auxiliares

Descripción	Precio (%)	Cantidad (€)	Total (€)
10% sobre costes de materiales y mano de obra	10,00%	163,41	**16,34**

El coste final de los medios auxiliares es de **dieciséis euros con treinta y cuatro céntimos**.

1.4. Total

Concepto	Precio (€)
Materiales	115,41
Mano de obra	48,00
Medios auxiliares	16,34
	179,75

El presupuesto total del prototipo es de **ciento setenta y nueve euros con setenta y cinco céntimos**.

2. Presupuesto de ingeniería

2.1. Costes de ingeniería

Tarea	Cantidad (h)	Precio (€/h)	Total (€)
Redacción memoria	70	8	560,00
Estudio necesidades	2	8	16,00
Diseño mecánico	20	8	160,00
Diseño electrónico	2	8	16,00
Programación	10	8	80,00
Cálculos	1	8	8,00
Revisiones	3	8	24,00
Pruebas de funcionamiento	125	8	1000,00
Investigación	50	8	400,00
Redacción presupuesto	2	8	16,00
Redacción Pliego de Condiciones	3	8	24,00
Realización planos	6	8	48,00
			2352,00

El coste total de ingeniería es de **dos mil trescientos cincuenta y dos euros**.

2.2. Medios auxiliares

Descripción	Precio	Cantidad (€)	Total (€)
10% sobre costes de ingeniería	10,00%	2352,00	**235,20**

El coste total de los medios auxiliares es de **doscientos treinta y cinco euros con veinte céntimos**.

2.3. Total

Concepto	Precio (€)
Coste ingeniería	2352,00
Medios auxiliares	235,20
	2587,20

El presupuesto total de ingeniería es de **dos mil quinientos ochenta y siete euros con veinte céntimos**.

3. Presupuesto total del proyecto

Concepto	Precio (€)
Presupuesto del prototipo	179,75
Presupuesto de ingeniería	2587,20
	2766,95

El presupuesto total del proyecto es de **dos mil setecientos sesenta y seis euros con noventa y cinco céntimos**.