Fabricación e implementación de BiJuBot (robot bioinspirado con mecanismo de salto y control de orientación por medio de palanca de inercia)

Jose Ignacio Choriego Rizzo

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Fabricación e implementación de BiJuBot (robot bioinspirado con mecanismo de salto y control de orientación por medio de palanca de inercia)

Trabajo de graduación presentado por Jose Ignacio Choriego Rizzo para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Fabricación e implementación de BiJuBot (robot bioinspirado con mecanismo de salto y control de orientación por medio de palanca de inercia)

Trabajo de graduación presentado por Jose Ignacio Choriego Rizzo para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

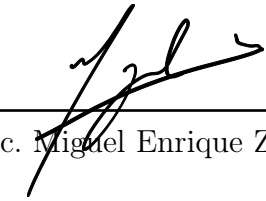
2022

Vo.Bo.:

(f) 

MSc. Miguel Enrique Zea Arenales

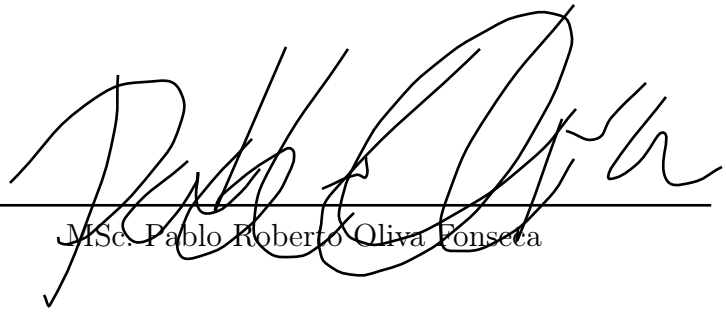
Tribunal Examinador:

(f) 

MSc. Miguel Enrique Zea Arenales

(f) 

Dr. Luis Alberto Rivera Estrada

(f) 

MSc. Pablo Roberto Oliva Fonseca

Fecha de aprobación: Guatemala, 07 de Enero de 2022.

Prefacio

Agradezco la oportunidad que me ha dado la Universidad de trabajar en este proyecto y darle el seguimiento necesario para que se forme como solución a un problema real en el futuro. Estoy contento de presentar los avances y los resultados de dicho proyecto y mostrar un robot en su fase 1, funcionando de manera real con avances significativos.

Me siento muy orgulloso de finalizar esta etapa de mi vida como estudiante y sueño con mucho anhelo y convicción que me espera un gran futuro como profesional. La universidad a sido una gran herramienta en esta etapa de mi vida y estoy feliz de poder ser un ingeniero egresado de la Universidad del Valle de Guatemala. He tenido muchos tropiezos como victorias y la experiencia que me llevo es única. Tuve un momento de incertidumbre durante el tercer año donde quería cambiarme de carrera, esta no fue la decisión, aunque el camino fue largo y más difícil, tengo la satisfacción y el objetivo que quería cumplir desde que empecé la universidad.

Agradezco personalmente a mis papás por todo el amor y cariño, los quiero mucho y son las personas más importantes en mí vida. Agradezco a mí catedrático también, Miguel Zea, por todo el apoyo y la asesoría en este proyecto, sin él no hubiera podido alcanzar este objetivo final. También agradezco a mis compañeros/amigos por el apoyo en las temas académicos, la compañía y amistad que me han brindado a lo largo de la carrera.

Índice

Lista de figuras

Lista de cuadros

Resumen

El presente trabajo tiene como desarrollo principal la implementación física de un robot bio inspirado con mecanismo de salto y con control de orientación por medio de una palanca de inercia. Este trabajo constituye una continuación al trabajo de tesis realizado por Jerry Rivera, la cual tuvo como objetivo el diseño y simulación del mismo Robot. En esta se empezó desde su fase inicial, donde se definieron características mecánicas, estructurales y se realizó un modelado con sistemas dinámicos para poder obtener resultados de trayectorias y simulaciones que se acercaran al comportamiento del robot.

En este trabajo se muestra el desarrollo del sistema que le da control al robot saltarín en el aire. Se mostrarán los resultados del movimiento de la palanca que utiliza y también cierta instancia de las pruebas realizadas en el aire. De igual manera se presenta el desarrollo del programa creado para su funcionamiento y el hardware diseñado para este.

Para alcanzar los objetivos se utilizó el entorno de Code Composer de Texas Instruments para poder desarrollar el código a implementar en el controlador del robot. También se emplearon programas de diseño CAD, como es el caso de Autodesk Fusion 360, Inventor y KiCad para poder diseñar piezas mecánicas, esquemáticos y PCBs. Para la la implementación del robot fue necesario del conocimiento de los componentes, como es el caso de los motorreductores, un driver para los motores y una unidad de medición inercial (MPU6050).

Abstract

The main purpose of this work is in the physical implementation of a bio-inspired robot with a jump mechanism and orientation control by means of an inertia lever. This work constitutes a continuation of the thesis work carried out by Jerry Rivera, which aimed to design and simulate the Robot itself. In his thesis, he began from its initial phase, where mechanical and structural characteristics were defined and a modeling was carried out with dynamic systems in order to obtain results of trajectories and simulations that were close to the behavior of the robot.

In this work the development of the system that gives control to the jumping robot in the air will be shown. It will show the results of the movement of the lever that it uses and also a certain instance of the tests carried out in the air. In the same way, the development of the program created for its operation and the hardware designed for it, will be shown.

To achieve the objectives, the Texas Instruments Code Composer environment was used to develop the code to be implemented in the robot controller. CAD design programs, such as Autodesk Fusion 360, Inventor and KiCad, were also used to design mechanical parts, schematics and PCBs. For the implementation of the robot, it was necessary to know the components, such as the gearmotors, a driver for the motors and an inertial measurement unit (MPU6050).

CAPÍTULO 1

Introducción

"Esta investigación gira en torno a uno de los temas de mayor auge en la actualidad: la robótica bio inspirada, la cual busca aprender o inspirarse en la naturaleza para desarrollar mecanismos que realicen tareas específicas de forma eficiente. Este trabajo tiene como objetivo desarrollar la simulación de un robot con salto, inspirado en el salto de animales como las langostas y que controle su orientación mediante una barra de inercia, tal y como lo hacen animales como las lagartijas al utilizar sus colas." - Jerry Rivera [1]

El propósito principal de este trabajo de graduación es la implementación física del robot saltarín propuesto por Jerry Rivera[1], Se busca poder obtener datos de este a través de pruebas y simulaciones reales, tal que el mecanismo de salto pueda servir como un método de movilización para robots en el futuro.

Para llevar a cabo este proyecto fue necesario la toma de datos y la validación del robot por medio del trabajo de graduación trabajado por Rivera [1]. En este trabajo se consideraron factores como el modelado matemático, el análisis estructural y mecánico, los cálculos de cinética, la simulación en software y el diseño en CAD. Muchas de estos resultados son esenciales para el planteamiento de este nuevo problema que es lo que sucedería si el robot se fuese a fabricar.

Durante el desarrollo de este trabajo de gradación se trabajó en temas como es el caso de programación de microcontroladores, diseño de PCBs y fabricación por medio de tecnologías aditivas. El proyecto que se desarrolla se secciona en módulos implementados por software, como es el caso de los filtros complementarios y los sistemas de control. A continuación se muestra el trabajo realizado.

CAPÍTULO 2

Antecedentes

A continuación se describen las generalidades del trabajo planificado y desarrollado por Rivera [1]. El trabajo en su mayoría se basa en el diseño y simulación del robot bio-inspirado en una langosta saltarina, y tenía como objetivo principal el diseño e implementación del mismo robot bio-inspirado con mecanismo de salto y con control de orientación por medio de una palanca de inercia. Este no pudo cumplirse en su mayoría debido a las implicaciones de la pandemia por el COVID-19. Dado esto, se desarrollaron varias partes del robot, estas para poderle dar seguimiento a la construcción e implementación real del mismo. El trabajo también se basó en un estudio previo, realizado acerca de las distancias y los métodos de salto hechos por distintos experimentos de robots similares hechos en distintas instituciones. Al final se escogió como inspiración el robot: "MSU Jumper: A Single-Motor-Actuated Miniature Steerable Jumping Robot" [2], el cual consigue tener un buen alcance horizontal con pocos actuadores.

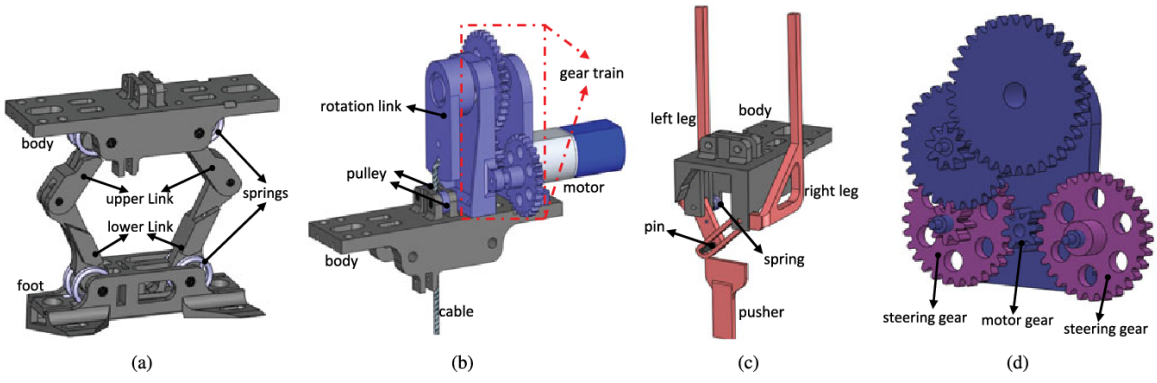


Figura 1: Modelo del MSU Jumper [2].

Luego, como objetivo específicos se trabajó en la implementación del mecanismo de salto con un juego de engranajes y resortes torsiónales en la implementación de

un sistema de control para la orientación del robot y por último, en la validación del robot en conjunto en un software de simulación (We-bots). El trabajo y su metodología como tal se dividió en tres secciones importantes las cuales fueron: Diseño mecánico, modelado del sistema y control de sistema.

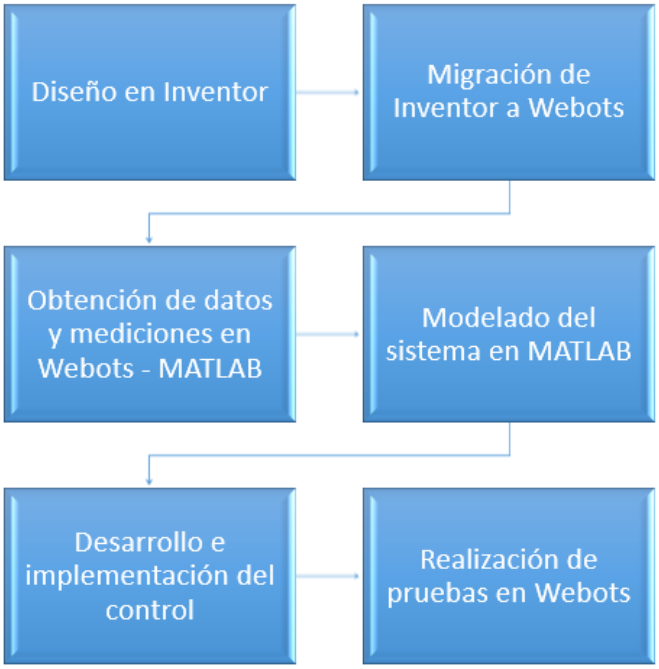


Figura 2: Diagrama de metodología [1].

En el diseño mecánico se desarrollaron las especificaciones requeridas para la selección del motor, la definición de parámetros para el mecanismo y se hizo un análisis para seleccionar resorte. Se evaluaron componentes físicos y se realizaron pruebas en Matlab para comparar distintas características de torque en los motores y en el resorte. Con los resultados obtenidos se consideraron varias elecciones para el mecanismo donde la longitud de cada eslabón fue de 2 cm. La altura del robot fue de 8 cm. De los resultados finales la mejor opción en motores es el "RBPol-448" y el motor "RBPol-452" [3] ya que con estos es posible alcanzar la razón de salto r mínima, caracterizada dentro de los cálculos.

Modelo	**Reducción**	**Torque (Nm)**	**Diámetro del eje (mm)**	**Peso (g)**
RB-Pol-448	100:1	0.0726	3.0	10
RB-Pol-452	298:1	0.1961	3.0	10

Cuadro 1: Características de los motores recomendados [1].

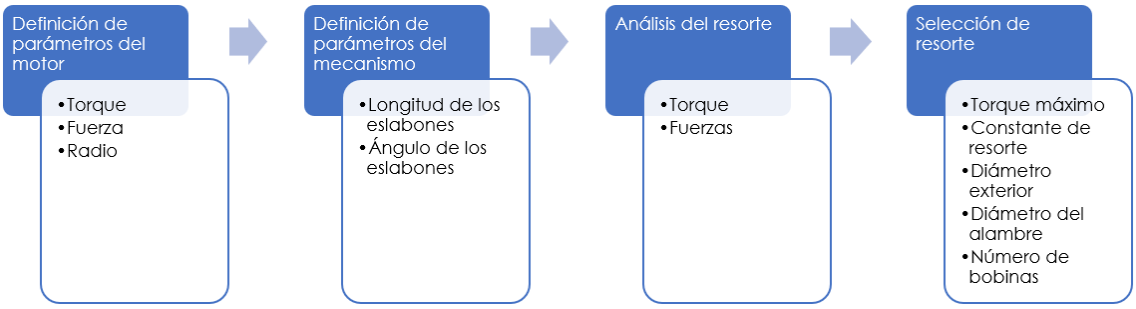


Figura 3: Diagrama del diseño mecánico [1].

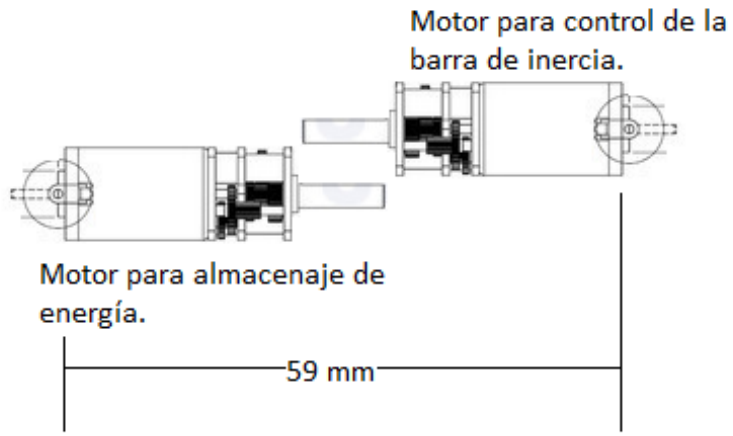


Figura 4: Posición y forma de los motores [1].

En el diseño CAD se consideraron distintas proposiciones, como la definición de la disposición de los motores, la definición del ancho y profundidad del mecanismo, el diseño de la base superior, el diseño de los eslabones y el diseño de la barra de inercia. Cada una de estas piezas fue trabajada en el software de Autodesk Inventor, considerando que después se tuvieron que realizar distintas exportaciones a diferentes formatos para después poder ser trabajado en como un archivo ".obj" en el entorno de Webots. En Autodesk Inventor se hizo un análisis estático de los esfuerzos aplicados en la estructura del robot, donde la sección crítica fue el resorte ya que era la parte donde se ejercían los esfuerzos. El esfuerzo máximo de deformación es mucho mayor al que se presenta en el análisis, por lo cual la estructura es estable y segura.

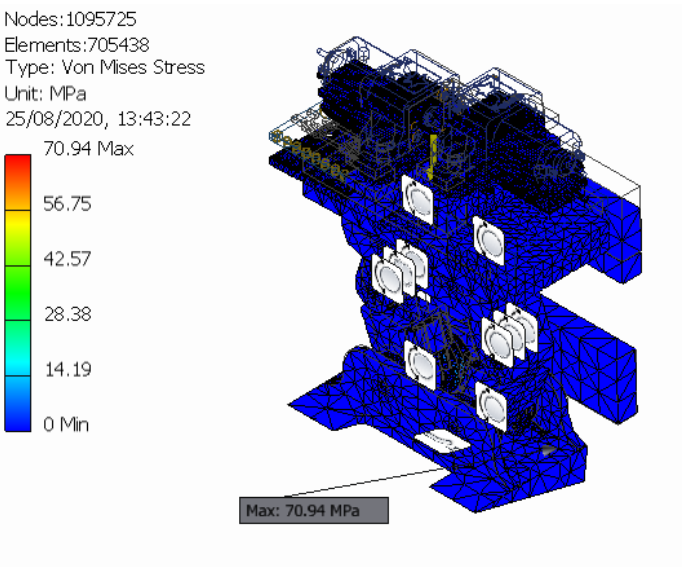


Figura 5: Esfuerzos analizados en el robot [1].

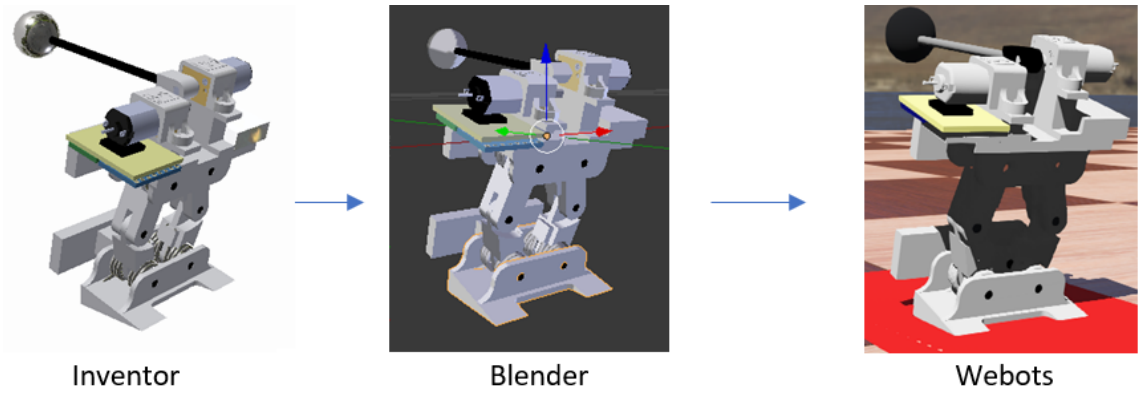


Figura 6: Migración de distintos formatos de archivos 3D [1].

En la siguiente sección se presentó una manera en la que el sistema se modela como un sistema dinámico no lineal, el cual presenta 6 variables de estado. Con este sistema dinámico fue posible aproximar el comportamiento del robot para luego realizar una intuición acerca del posible movimiento y las trayectorias de salto que realiza el robot. Para poder modelar el sistema dinámico no lineal se utilizó Matlab como plataforma. Las primeras tres variables de estado, representan la posición en x y en y del robot, junto al ángulo θ que era la orientación del robot. Una forma de modelar el robot fue colocándolo en un espacio planar. Las últimas tres variables, representan la velocidad en x, y y θ. Teniendo el modelado matemático y analítico se creó un escenario de prueba en el entorno de Webots. Utilizando el physics engine de Webots se pudo llegar a buenos resultados. para demostrar el efectivo funcionamiento del robot.

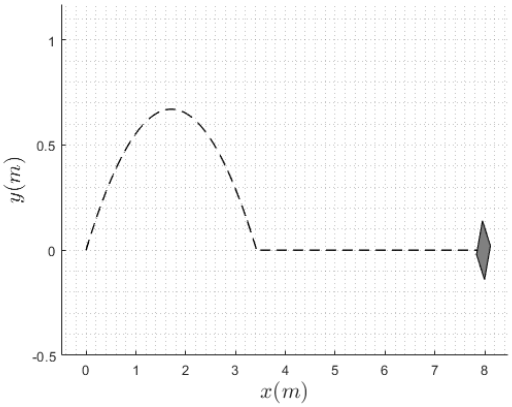


Figura 7: Parábola de salto simulada a través de Matlab [1].

El control y comportamiento que el se le asignó al robot fue por los distintos nodos de Webots. Uno de los nodos importantes a considerar fue el del uso de la Unidad de Medición Inercial (IMU:) para poder ubicar y tener la orientación del robot. Con la IMU se pudo dar una retroalimentación para el sistema de control efectuado. La señal de entrada se consideró como el torque de entrada y la señal de salida fue la velocidad angular. En Webots se obtuvieron gráficas de estas variables especificas. Luego, en Matlab se utilizó el system identification toolbox [4], para identificar el sistema y poder realizar control clásico sobre este. Se llegó a utilizar un controlador PD, ya que es suficiente para controlar y rastrear la señal a una posición deseada. En la simulación en Webots se realizaron dos pruebas importantes, en la primera prueba el robot logra brincar pero aterriza mal, por la colisión elástica del piso del entorno. En la segunda prueba se muestra cómo el robot aterriza mejor, esto porque se cambió la rigidez de las patas y se colocaron con un material más elástico. Al final se concluyó en utilizar un alambre galvanizado como conexión de las patas y la estructura del robot.

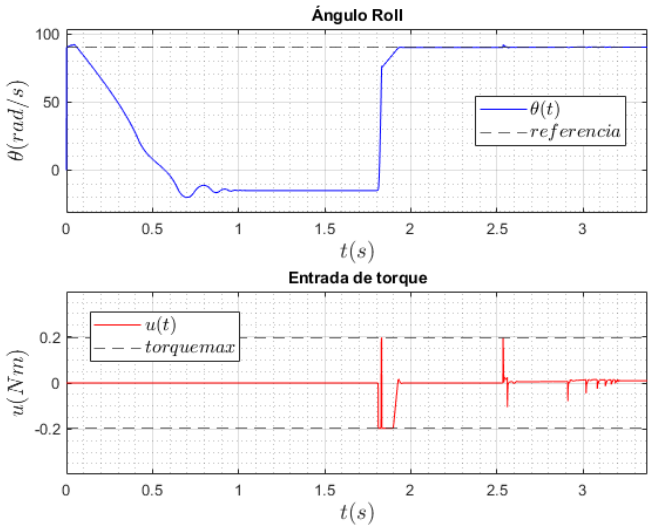


Figura 8: Comportamiento del control PD [1].

Al final se concluyó que el mecanismo de salto diseñado cuenta con un motor el cual incorpora un juego de engranajes en su caja reductora y ocho resortes torsionales que, en conjunto, almacenan y liberan energía potencial para generar un salto que permitió al robot alcanzar una altura de al menos 88 cm. Se implementó un controlador proporcional derivativo PD en el robot, haciendo uso de un motor para suministrar la entrada de torque requerido y una unidad de medición inercial (IMU) para tomar mediciones del ángulo roll y de la aceleración en el eje vertical este permitirá el control de la orientación del mismo durante el salto, posibilitando así un correcto aterrizaje.

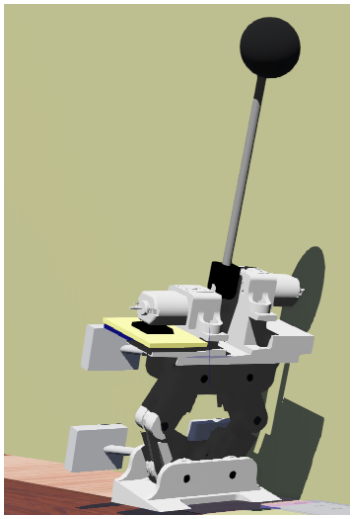


Figura 9: Diseño final del robot (BiJuBot) [1].

CAPÍTULO 3

Justificación

Los mecanismos antiguos, al principio de la segunda revolución industrial, presentaban sistemas ingeniosos aunque muy laboriosos, complejos y no siempre eficientes, como es el caso de los primeros motores y todos los sistemas mecánicos de transmisión o movimiento. Por esto mismo, dentro del proceso de diseño se ha llegado a la realización de observar a la naturaleza y la belleza que esta presenta, pudiendo apreciar cómo la fauna y flora han evolucionado para adaptarse a su entorno. Tratando de imitar a la naturaleza y los frutos de su diseño, se crea el concepto de la robótica bio inspirada, tratando de capturar el movimiento natural y el diseño de los organismos vivos. En este proyecto de graduación se busca implementar este concepto para la realización del robot saltarín diseñado por Rivera [1].

El siguiente trabajo consiste en la continuación y experimentación física con el mecanismo de salto, ya que se desea probar qué tan eficientes fueron los cálculos y las predicciones teóricas realizadas, todo esto para poder plasmarlo en un sistema real. Para esto será necesario programar un código para el microcontrolador a utilizar al igual que sera necesario ensamblar cada una de las piezas trabajadas en el proyecto anterior y diseñar un PCB para el orden y la modularidad de los componentes en el robot. Teniendo el robot ensamblado con el código cargado será el momento de probarlo haciendo pruebas de salto e iterando en el proceso de prueba. Por este motivo, el mecanismo de salto será indispensable y justificable en esta fase del proyecto. El robot bio-inspirado en el salto de las langostas, será llamado que es un acrónimo al nombre en inglés "Bio-Inspired Jumping Robot".

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Manufacturar y evaluar una plataforma robótica bio-inspirada en el comportamiento de salto de pulgas y langostas capaz de realizar saltos y ajustar su orientación en la transición de vuelo.

4.2. Objetivos específicos

- Manufacturar,ensamblar y validar las piezas y sistemas de transmisión de potencia mecánicas, diseñadas en la primera fase del proyecto.
- Diseñar y manufacturar una placa de circuitos impresos que contenga los componentes necesarios para la acción y el control de la plataforma robótica.
- Implementar y validar los algoritmos de control desarrollados con programas de simulación durante la primera fase del proyecto.

CAPÍTULO 5

Alcance

Debido a la dificultad presentada por la pandemia del COVID-19, se tuvo cierto retraso en la adquisición de componentes y con el trabajo realizado en el laboratorio. Se tuvieron ciertas limitaciones de tiempo y también de acceso al campus de la universidad. Por esto mismo motivo, no se pudo probar ciertas piezas indispensables para el funcionamiento completo.

En esta entrega se realizará el desarrollo principal del código a implementar en el robot. También se especificarán los resultados del mecanismo de auto balance que es una de las características más importantes del robot aparte de su mecanismo de salto. Además se describirá el mecanismo principal que se utilizará para que el robot se comprima y luego, a través de la energía potencial acumulada, logre saltar. Para la plataforma de auto balance fue necesario el diseño de un PCB con el cual se ordenaran y se colocaran de manera modular los componentes. Para las pruebas de auto balance se armó el sistema en un prototipo inicial con protoboard para poder verificar inicialmente el diseño y su funcionamiento.

Como contenido posterior se estará desarrollando el ensamblaje y funcionamiento del mecanismo de salto, terminando con pruebas de salto donde se pueda mostrar la implementación. En la fase anterior de este proyecto se logró dimensionar una implementación que consistía en la simulación y modelado de la plataforma robótica. En esta misma se realizará una justificación para lograr observar si las medidas y resultados planteados fueron los correctos.

CAPÍTULO 6

Marco teórico

6.1. Especificación de motor RBPol 448 y 452

Los motores RBPol, consisten en moto-reductores con un motor de corriente continua (DC) de 6 Volts de baja potencia. Estos contienen una caja de reducción metálica de 100:1 o 298: 1 . En sus dimensiones tienen una sección transversal de 10 × 12 mm, y el eje de salida de la caja de cambios en forma de D tiene 9 mm de largo y 3 mm de diámetro [3].

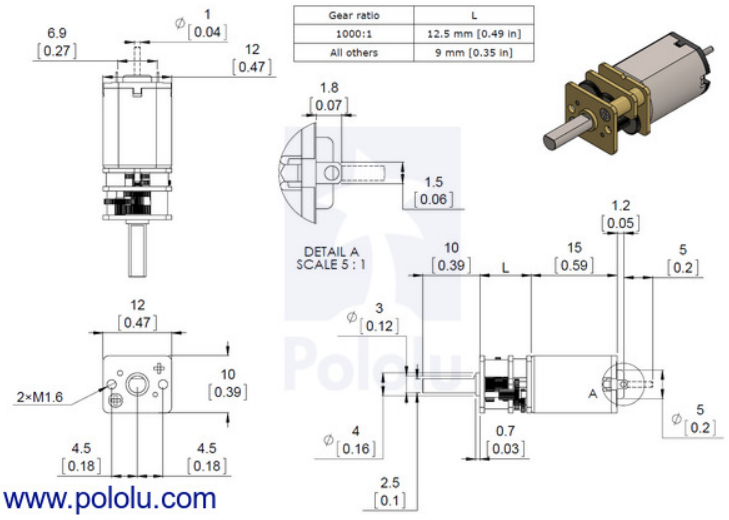


Figura 10: Planos y forma de motor [3].

Estos motores presentan escobillas de carbón. Las escobillas son pequeños contactos dentro del motor con el eje de rotor. A través de estas se transmite la corriente directa, para generar un campo electromagnético alrededor del rotor del motor. Este

campo produce movimiento en el eje del motor ya que se induce una fuerza electo motriz. El movimiento del motor se da principalmente por la repulsión y atracción de los polos magnéticos del rotor y el estator. En este caso por ser un motor simple de tamaño relativamente pequeño, el estator se compone principalmente de imanes permanentes.

Los motores de corriente continua o DC, son de los motores más básicos y se encuentran en casi cualquier dispositivo eléctrico o electrónico. En este caso, el motor RBpol tiene un set de engranajes que le permiten aumentar el torque dado, con el compromiso de la reducción de velocidad del motor. Cuando se tiene un mayor torque en el motor, este puede ejercer más fuerza. En el caso de este proyecto se utilizan dos reducciones de velocidades, una de 100:1 y la otra de 298:1. El que tiene mayor reducción se utiliza para la compresión del robot saltarín y el otro se utiliza para el movimiento de la palanca de inercia. A continuación se muestra la relación que se aplica para la reducción de velocidad para el motor con la reducción de 100:1. Cabe mencionar que esta ecuación solo es un modelo de la relación de cada diente de engranajes . Por el momento no se cuenta con el número de cada uno de estos dientes ya que es un dato que no se necesitara para las especificaciones requeridas.

$$\frac{35 \times 37 \times 35 \times 38}{12 \times 11 \times 13 \times 10} \approx 100.37 \tag{1}$$

Los detalles requeridos para estos motores son los siguientes:

Modelo	**Reducción**	**Torque Nominal (Nm)**	**Torque de Paro (Nm)**	**Potencia Max(W)**
RB-Pol-448	100:1	0.0726	0.1569064	1.3
RB-Pol-452	298:1	0.1961	0.3334261	1.0

Cuadro 2: Especificaciones de los motores empleados [3].

Cabe mencionar que los dos motores operan de la misma manera a 6V, ambos tienen una corriente de paro y sin carga de 1.5 y 0.1 Amps respectivamente. La característica peculiar de estos modelos es que pueden ser modulares y ambos cuentan con pines para ser soldados a pequeños "codificadores magnéticos", los cuales se explicaran a continuación. Estos a su vez se controlan con un driver externo el cual permite controlar al motor con una señal de pulsos modulado, similar a un servomotor.

6.2. Encoders magnéticos

Un encoder se refiere a un dispositivo que codifica cierto comportamiento de un dispositivo eléctrico o mecánico. Esto quiere decir que lo convierte en una señal eléctrica digital, para poder ser reconocido y procesado por otro dispositivo.

En este caso el encoder sirve para poder digitalizar el movimiento del eje del motor.

Esto lo hace por medio de un disco magnético y unos pequeños sensores de efecto hall.

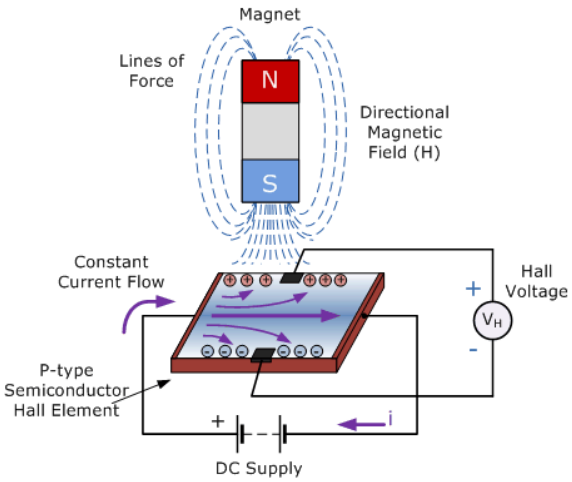


Figura 11: Sensor de efecto hall [5].

Los sensores de efecto hall consisten básicamente en una pieza delgada de material semiconductor rectangular tipo P,donde pasa una corriente continua a través de si mismo. Cuando el dispositivo se coloca dentro de un campo magnético, las líneas del campo magnético ejercen una fuerza sobre el material semiconductor que desvía a los electrones a ambos lados de la placa semiconductora. Este movimiento de los electrones de carga es el resultado de la fuerza magnética que experimentan al pasar a través del material semiconductor. Debido a esto los sensores son capaces de detectar magnetismo y transmitirlo como una impulso eléctrico [5].

El encoder magnético debe su nombre a este comportamiento que realiza el pequeño disco y los sensores. El pequeño disco se monta en un pequeño eje trasero del motor junto al modulo con la placa electrónica, donde esta placa tiene soldadas los pequeños sensores y estos leen continuamente el movimiento del disco generando así señales discretas para la lectura digital del motor. Al encoder también se le denomina un encoder giratorio debido al comportamiento de disco giratorio en este. El encoder que se estará utilizando se muestra en la Figura 12. El encoder en este caso cuenta con dos sensores de efecto hall, uno para el lado derecho y el otro para el izquierdo. Tienen tres pines, dos corresponden a la alimentación y el otro es el pulso de salida.

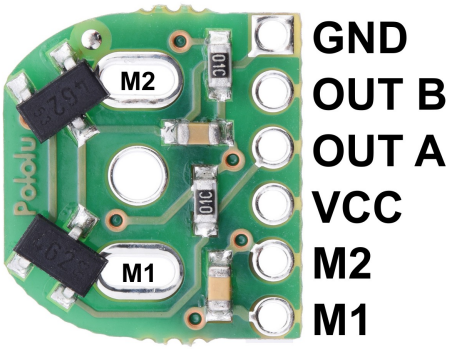


Figura 12: Encoder magnético de Polulu 12 CPR [6].

Este encoder cuenta con doce lecturas por una revolución completa del eje, las cuales se muestran en señales de pulsos cuadrados. En la Figura 12 se muestra el pinout (pines de entrada y salida) del encoder. Los pines “Out A” y “Out B”, son los pines donde se obtienen las lecturas del encoder. Estas se utilizan en el encoder para tener una referencia del motor y poder ser utilizada como retroalimentación del sistema. En las siguientes figuras se muestra el montaje del encoder y las lecturas de las señales eléctricas.

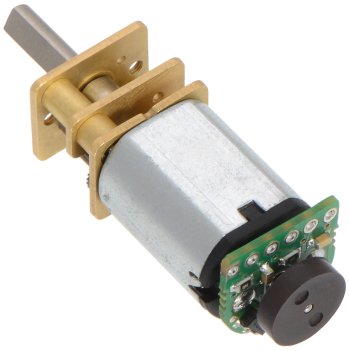


Figura 13: Encoder magnético montado sobre motor [6].

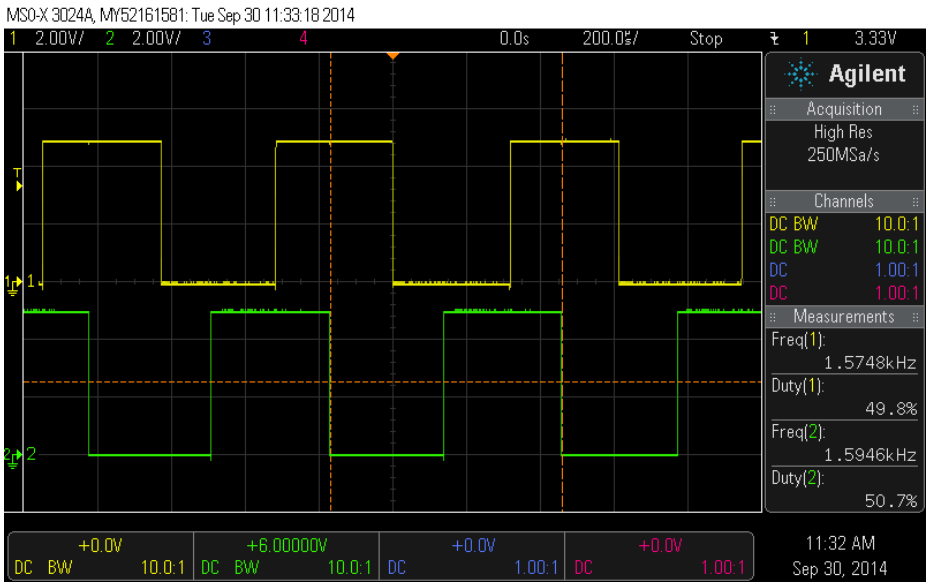


Figura 14: Lecturas digitales de los pines A y B del encoder magnético [6].

Con el encoder se puede determinar la posición del eje del motor por medio de las señales de los sensores. Aun sí no es posible controlar el movimiento del motor en su totalidad, para esto se requiere de un control de velocidad que lo garantiza un "driver".

6.2.1. Driver de motor

Un driver es un dispositivo que maneja o controla otro dispositivo por medio de un accionamiento o control completamente distinto al del dispositivo manejado. Los drivers para motores se componen de circuitos integrados y circuitos de potencia que regulan la corriente y voltaje aplicados. Estos drivers se componen de un elemento fundamental para el movimiento del motor, conocido como "puente H".

Un puente H consiste en un circuito con cuatro transistores funcionando como interruptores. Estos conmutan su estado dejando pasar o bloquear la señal eléctrica. De esta manera el motor que se encuentra conectado puede cambiar su sentido de giro y modular su velocidad. Existen diferentes modos para armar un punte H y hay unos que integran más componentes que otros. En este caso el modelo del circuito integrado que tiene el driver a utilizar es el FreeScale 33926 [7]. En la Figura 15 se aprecia el esquemático de este modulo integrado. Siendo un esquemático complejo, el puente H se encuentra en la parte derecha del esquemático, el cual contiene 4 transistores MOSFET, 2 de tipo N a la izquierda y 2 de tipo P a la derecha. Este orden en los transistores proporciona la conmutación adecuada para poder manejar los cambios de sentido de giro en el motor.

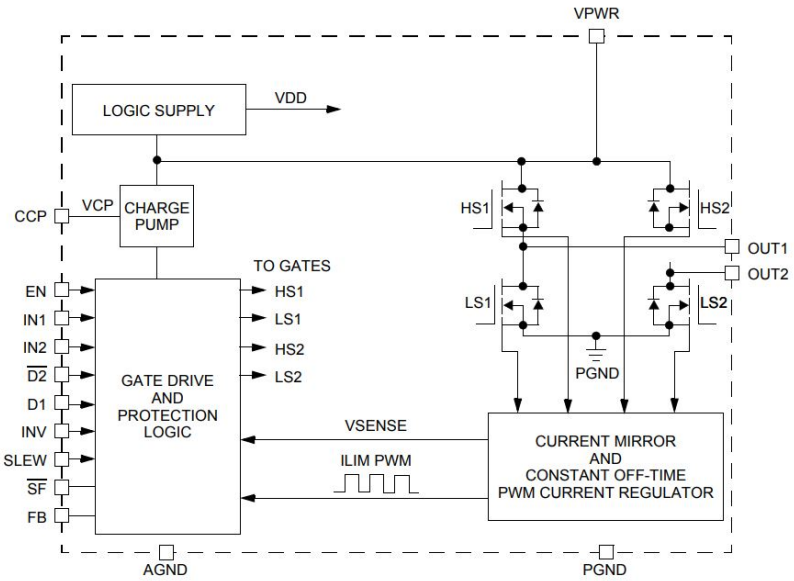


Figura 15: Esquemático del modulo "FreeScale 33926" [7].

En la Figura 15 también se pueden apreciar distintos módulos. Estos son componentes integrados que realizan distintas funciones. No se entrará en detalle, ya que cada una de estas representa un comportamiento complejo de explicar. Todos los módulos son importantes pero el de "lógica de protección y conductor de puerta" puede ser el más importante en este sistema ya que este es el encargado de recibir la retroalimentación del motor, leer y convertir la señal de pulsos proporcionada al driver. Este módulo es un microcontrolador y la tarea que ejecuta es necesaria para poder utilizar la señal modulada por pulsos o Pulse Width Modulation (PWM:) comúnmente conocida [8].

Finalmente el driver que se utiliza es el DRV8833 de Texas Instruments [8], el cual consiste en un portador de 2 motores. Este driver permite capacidades de corriente y voltaje moderadas alrededor de 2.7 a 10V y entrega una corriente máxima de 2A en alta capacidad. En la Figura 16 se aprecia el pinout (pines de entrada y salida) y la forma del componente.

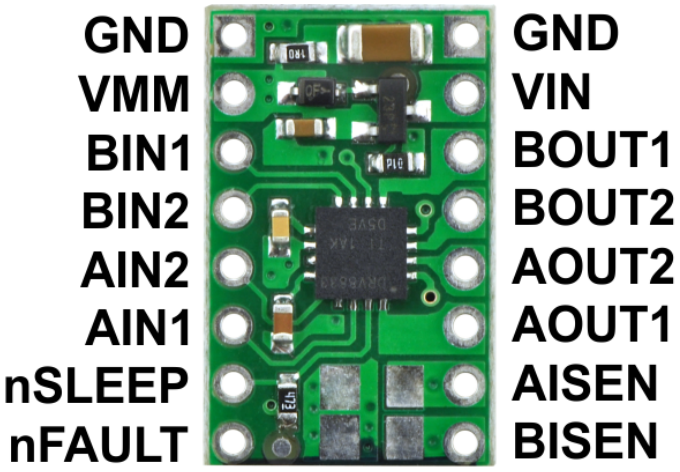


Figura 16: Pinout del driver DRV8833 [8]

6.3. Sistemas de control digital

La arquitectura de un sistema de control típico empleado en la mayoria modelos consiste en una planta, un sensor y un controlador. La planta, en muchos casos es el componente original, el cual emite su señal como en el caso del motor eléctrico. Luego el controlador procesa la señal retroalimentada para estabilizar la planta. Todo esto se tiene que conectar con una conexión de retroalimentación negativa, donde la salida del sistema se resta a la referencia o señal de entrada. En este modelo el sensor está conectado dentro de esta retroalimentación. En la Figura 17 se muestra un diagrama que ilustra este tipo de conexión para un sistema de control.

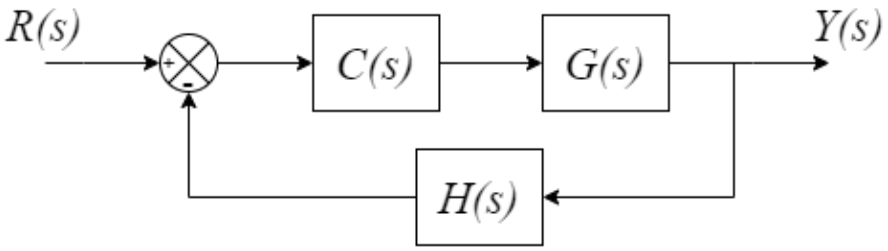


Figura 17: Diagrama de bloques de un sistema básico de control [9].

En la Figura 17 los bloques dados están en función de s (dominio de la frecuencia). El bloque C es la planta el bloque G es el controlador y el bloque H es el sensor. Las señales R(s) y Y(s) son la entrada y salida del sistema respectivamente. El círculo con los signos de suma y resta es un bloque que suma la señal de referencia entrada y resta la señal de salida, todo esto para generar la mencionada retroalimentación negativa la cual eliminara el error y la diferencia de comportamiento entre la señal de salida y entrada.En este tipo de sistemas la señal de referencia tiene que parecerse

lo más posible a la señal de salida [9].

Como se había mencionado antes el motor corresponde a la planta, dado que es un sistema real este presenta imperfecciones en su comportamiento. El controlador busca regular estas imperfecciones. El controlador PID, es el controlador más utilizado y el que se emplea para poder regular y controlar de mejor manera el comportamiento del motor. Dado que los drivers funcionan para usar una señal modulada por pulsos, el controlador PID controla los comportamientos bruscos y repentinos que el motor pueda tener a la hora de actuar. Para eso se necesita implementar el controlador dentro de un microcontrolador (MCU) el cual maneje todas las señales tanto de entrada como de salidas del robot BiJuBot completo. Para esto se necesitara manejar al robot con una implementación digital del PID.

Básicamente, lo que realiza el controlador PID es la superposición de tres distintos controladores. Estos son: el controlador Proporcional, el Integral y el Diferencial. Los tres tratan de corregir el error. El erro básicamente es la diferencia entre la señal de entrada y salida [9].

- Parte proporcional: corresponde al error presente, trata de alcanzar la referencia y hacer el error cero lo más rápido posible.
- Parte integral: corresponde al error pasado o acumulado, corrige el error en estado estable y permite que la señal de salida se parezca a la de entrada.
- Parte derivativa: error futuro, trata de predecir el error y actuar evitar que incremente. Este ayuda a la estabilidad y usualmente se emplea en menor medida que los otros controladores.

Dado que cada parte del controlador contribuye de forma más marcada a ciertas características de la respuesta del sistema a controlar, existen algunas variantes del control PID como lo son el control PI el cual es el más popular para rastreo de señales y el control PD para estabilización de señales. El control PD fue el recomendado en la tesis trabajada por Rivera [1]. Las ecuaciones (2) y (3) representan al PID , uno en el dominio del tiempo y el otro en el dominio de la frecuencia. Las constantes k_P, k_I y k_D son constantes que sirven tanto para aumentar y atenuar el peso de cada control sobre el sistema.

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \dot{e}(t). \tag{2}$$

$$U(s) = k_P E(s) + k_I \frac{E(s)}{s} + k_D s E(s). \tag{3}$$

En la implementación digital se considera que en lugar de una evolución continua en el tiempo como en el caso del modelo analógico, solo se tiene ciertos instantes de tiempo equidistantes de forma discreta o finita en forma de Δt. Esta constante es la

frecuencia de muestreo y es una relación de tiempo en la cual el sistema de computo procesa información. Como se definió antes al error como una noción de la diferencia de entrada y salida se tienen las siguientes ecuaciones a partir del modelo analógico del control P, I y D:

$$e(k\Delta t) = e_k \equiv \text{error, actual,} \tag{4}$$

$$e((k-1)\Delta t) = e_{k-1} \equiv \text{error, anterior/pasado,} \tag{5}$$

$$e_D = e_k - e_{k-1} \equiv \text{error, futuro,} \tag{6}$$

$$E_{k-1} = \sum_{i=0}^{k-1} e(i\Delta t) \equiv \text{error,acumulado,pasado,} \tag{7}$$

$$E_k = \sum_{i=0}^{k} e(i\Delta t) \equiv \text{error,acumulado,actual,} \tag{8}$$

De las ecuaciones (4) a (8) se obtiene:

$$u(t) \approx uk = k_P e_k + k_I E_k + k_D e_D, \tag{9}$$

donde la ecuación 9 presta cierta similitud a las ecuaciones (2) y (3) y es la que se implementa en la secuencia de iniciación de variables para la implementación del controlador PID digital. La variable u_k es la salida y esta es una aproximación discreta de la señal $u(t)$ en el dominio del tiempo.

La implementación completa es la siguiente, esto representa el pseudocodigo que sera implementado directamente (dependiendo del lenguaje de programación) en el microcontrolador como tal.

```
/* Estas variables se definen fuera del ciclo principal del programa*/

e_k_1 = 0;   //Se inicializa el error pasado como cero
E_k_1 = 0;   // Se Inicializa el error acumulado pasado como
                cero

/* Estas variables se inicializan en el ciclo principal del programa*/

e_k = r - y; // el error presente es la diferencia de la
                referencia (r) con la salida(y)
eD = e_k - e_k_1;
```

```
            // Se calcula el error diferencial como
               ecuación(6)
E_k = E_k + e_k;
            // Se calcula el error acumulado
u_k = kP*e_k + kI*E_k + kD*eD;
            // Se calcula la salida del sistema se utiliza la ecuación(9)
e_k_1 = e_k;
            // Para terminar se iguala el error pasado al  actual.
```

Esta es la implementación digital del controlador PID que se utiliza al final [9].

6.4. MPU 6050 (Unidad de Medición Inercial)

El MPU 6050 [10] es un modulo electrónico, que se utiliza principalmente como "IMU:" o Inertial Measurment Unit. Los IMUs son sensores inerciales con los cuales se puede determinar, aceleración, aceleración angular, velocidad angular y posición angular. Estos dispositivos trabajan con sensores microscópicos llamados "MEMS" (Micro-Electrical-Mechanical-System). Estos sensores microscópicos tiene pequeñas partes mecánicas las cuales detectan movimiento, como es el caso de los sensores hall estos también tienen pequeñas bobinas que detectan cambios magnéticos en la parte mecánica del sensor. En la Figura 18 se ejemplifica la membrana y el bloque que se utiliza en un sensor MEMS.

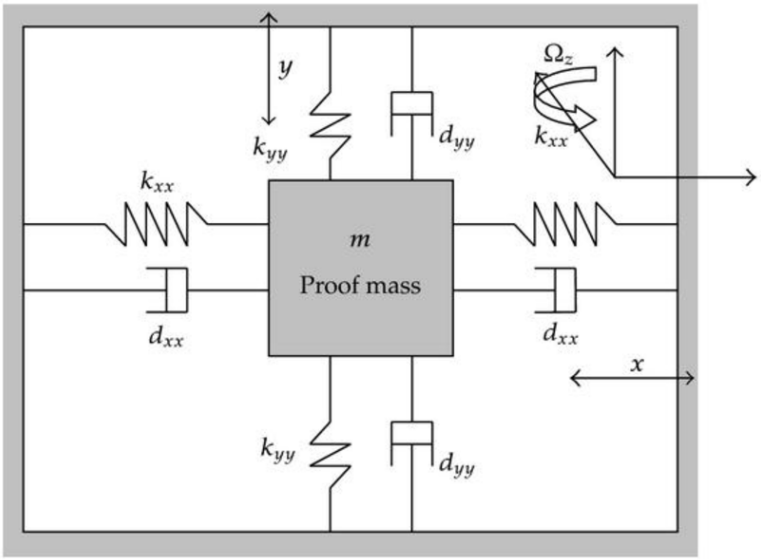


Figura 18: Sensor MEMS dentro del IMU [11]

El MPU 6050 cuenta con el protocolo de comunicación I2C, de esta manera, se transmite y se recibe señales del maestro al esclavo. El maestro sera el microcontro-

lador y el esclavo es la IMU como tal. En la comunicación I2C sólo existe un bus de datos, en este bus de datos se transmite y se recibe la señal de la IMU. La IMU cuenta también con un pequeño modulo ADC, para convertir las señales analógicas del MEMS a señales digitales del microcontrolador interno de este. La inicialización del IMU es compleja ya que se necesitan enviar ciertos registros hacia este para que comience. Aun así los datos necesarios para iniciarla que caben destacar son los dps (degree per second) para el giroscopio y las g (gravedades) para el acelerómetro. En la figura se muestra el modelo de la IMU (MPU6050) empleado.

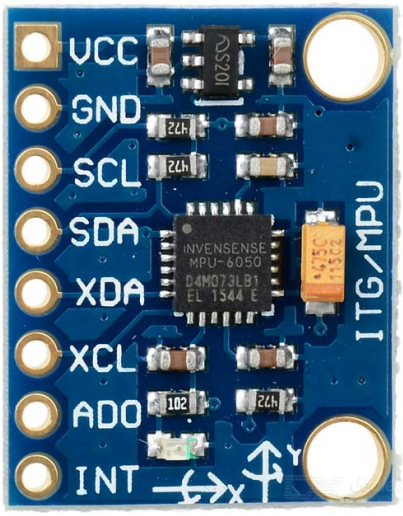


Figura 19: MPU 6050 [10]

Para la obtención de datos en bruto de la IMU se implementa la siguiente ecuación.

$$ACCEL\ X = ACCEL\ XOUTH << 8 + ACCEL\ XOUT, \tag{10}$$

donde también se puede obtener tanto información en y como en z. Esta ecuación hace un corrimiento hacia la izquierda para los bits de la aceleración alta y los suma con los bits bajos de la aceleración, todo esto para obtener la aceleración final. También se pueden obtener los valores en bruto del giroscopio de la misma manera. Luego se implementa la siguiente conversión.

$$a_x = \frac{ACCELX}{4096}. \quad \omega_x = \frac{GYROX}{65.5}. \tag{11}$$

Los denominadores de estas ecuaciones son constantes de resolución y estas se dan dependiendo del dato de dps (degrees per second) o g (gravedades). Luego se pueden emplear las siguientes ecuaciones para poder obtener la posición angular en "x y y" (roll, pitch) [12].

$$\varphi_a = -\frac{180}{\pi} \arctan 2 \left(\frac{a_x}{a_z} \right). \quad \theta_a = \frac{180}{\pi} \arctan 2 \left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}} \right). \tag{12}$$

Con todos estos datos y filtros pasa alta y bajas se puede diseñar un filtro complementario para una mayor estabilización del sistema inercial [12].

CAPÍTULO 7

Firmware e implementación de software

7.1. Desarrollo y algoritmo de implementación

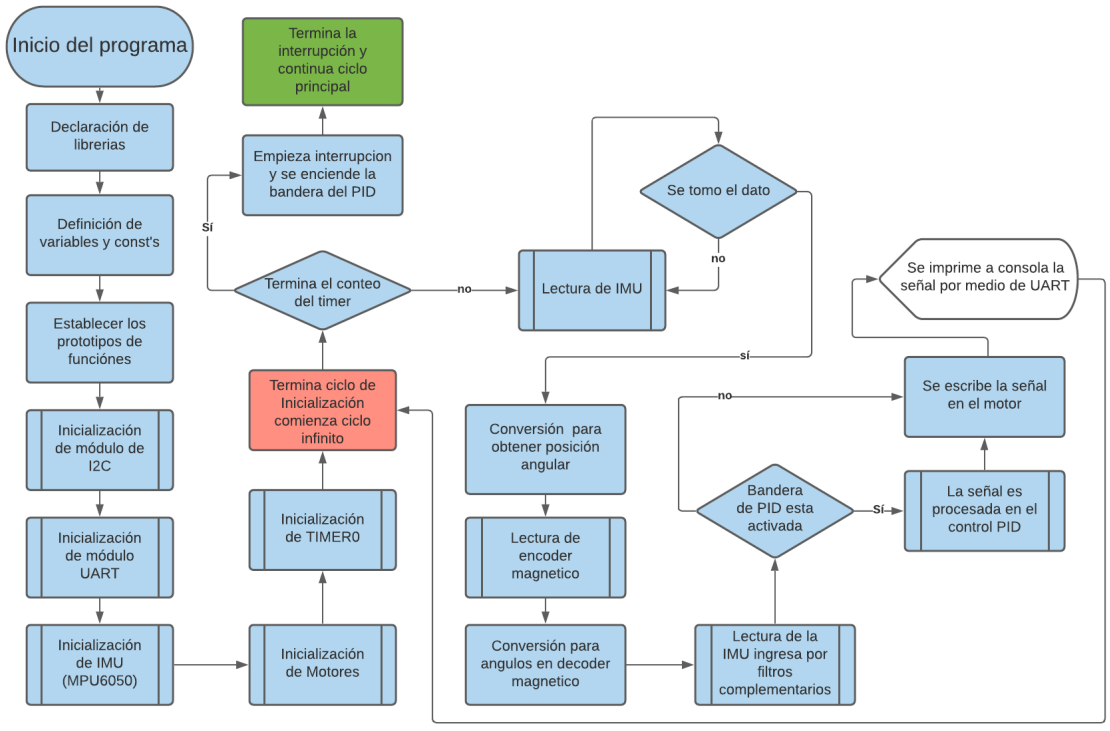


Figura 20: Diagrama de flujo general.

Para la implementación del código es importante clasificar cada paso del proceso y la rutina del algoritmo que estará funcionando. Para este funcionamiento se tiene un programa básico, con el cual se ingresa un valor a uno de los componentes y el controlador realiza distintas conversiones y filtrado de la señales, para que estas puedan ser procesada por el componente de salida. En este caso el componente de entrada es la Unidad de Medición Inercial y la de salida es el motor reductor.El sistema completo básicamente consiste en una caja negra donde hay una entrada(input) y una salida(output). En la Figura 20 se establece el diagrama de flujo general del programa del sistema de auto corrección, este sistema físicamente consiste en la palanca de inercia, un motor, la IMU, el driver para el motor y el microcontrolador con el cual se controla. El microcontrolador Tiva C: Launchpad, tiene los módulos y librearías requeridas para esta aplicación, como el caso de la librería para el MPU 6050 y el modulo de PWM utilizado para los dos motores en uso.

En el diagrama de flujo se expresa al inicio cómo se desarrollan e inicializan los distintos módulos. Luego inicia el ciclo principal el cual consiste en la adquisición de datos del encoder magnético y de la IMU. Estos datos son procesados por el filtro complementario en el caso de la IMU y luego se realiza un lazo cerrado con un sistema de control por medio del control PID. Este lazo sirve para que exista una retroalimentación negativa entre el encoder magnético y la señal de la IMU, y de esta manera el movimiento del rotor del motor sea exactamente al del ángulo de la IMU. Esto es lo que se busca en el programa de la palanca de inercia, poder controlar el giro del eje del motor a trevés de la rotación de la IMU.

En el diagrama también se muestra una interrupción que ocurre cuando hay un "time out" del TIMER0. En esta interrupción se activa una bandera, la cual activa el bloque PID, de esta manera generando una frecuencia de muestreo para que el procesamiento digital sea efectivo a cierto periodo de tiempo. A continuación se especifica cada modulo implementado.

7.2. Implementación de timer

El TIMER0, como se había mencionado anteriormente, tiene como objetivo principal proporcionar una interrupción en el programa periódicamente y de esta manera crear una frecuencia de muestreo para el sistema de control implementado. Para ver el código de inicialización ver sección de Anexos 12.1.

Como se muestra en los comentarios del código se tiene que realizar una secuencia para poder llegar a activar el temporizador. En los comentarios del código se tiene una enumeración, para dictar los pasos de esta secuencia. En el segundo paso se coloca una instrucción que permite que el conteo del timer sea periódico, esto quiere decir que al terminar el tiempo habilita la interrupción y empieza de nuevo el conteo del temporizador. La frecuencia con la cual funciona el reloj microcontrolador es de 40 MHz y es importante para la conversión que se desea hacer ya que la frecuencia del timer es resultado de la división de la variable de conteo con esta frecuencia del reloj

del microcontrolador. En la ecuación (13), se muestra esta operación, de la misma manera se puede realizar el recíproco (14) y obtener el periodo del timer el cual es el tiempo del temporizador.

$$f_{timer} = \frac{f_{clk}}{Cont}, \tag{13}$$

$$T_{timer} = \frac{Cont}{f_{clk}}. \tag{14}$$

El tiempo que se empleó en el timer es de 0.001 segundos, lo que implica una frecuencia de 1kHz. El valor que se utilizó en la variable de conteo fue cuarenta mil.

Para el caso de las interrupciones se tiene que habilitar un "handler" en uno de los archivos de configuración de la Tiva C. El handler es una linea de instrucción con la cual se avisa al sistema la interrupción del modulo especifico que estará activada. En el caso de otros módulos, como lo es para la IMU, este necesitará de su propio handler para activar una interrupción en el programa con la cual podrá funcionar el módulo del protocolo I2C. El timer en el programa principal tiene una subrutina, dentro de esta se activa la bandera del PID y se coloca una instrucción para reiniciar la interrupción.

7.3. Librería de motor y encoder

Para la implementación del motor con el encoder magnético se desarrollaron librerías respectivas. En el caso de la librería de los motores se desarrolló una librería que implementa el módulo PWM de la Tiva C. Para el caso del encoder magnético, se implementó una librería que utiliza un módulo del "driver lib". Este módulo sirve para leer señales de pulsos o cuadraturas, es justamente ideal para la función que cumple el encoder magnético.

7.3.1. Función de motor

En esta librería hay tres funciones importantes. Dos son para inicialización y configuración del motor 1 y 2 respectivamente y la otra es para escribir el ciclo de trabajo o velocidad en los motores correspondientes.

En la implementación completa del robot BiJuBot se utilizan dos motores en total. Un motor es el responsable de accionar y generar el movimiento de la palanca de inercia, mientras que el otro motor es el encargado de accionar el salto del robot para que este pueda despegar del suelo. En esta aplicación, mostrada en la Figura 20, Se hace una generalización para el uso de los dos motores. En el caso del motor que acciona el salto no se requiere de una IMU para funcionar, es más simple. También

se utiliza un sistema de control de lazo cerrado y un controlador PID dedicado para controlar motor. Para este caso solo se carga un ángulo a una variable y esta cuenta como la señal de referencia del motor. Otra forma de controlar a este motor es por medio de un GPIO general que activa y desactiva el movimiento del motor y este, por conveniencia, sólo girara en un sentido.

El microcontrolador cuenta con 14 módulos distintos de generación de señales moduladas por pulsos. Estos módulos se dividen en 2 bases (0 y 1), PWM0 y PWM1, el PWM0 es la base que se utiliza para los dos motores. Cada motor necesita dos pines con PWM. Estos pines irán conectados al driver y cada uno conmuta la activación de la señal de pulsos. Esta conmutación funciona para cambiar el sentido de giro del motor. Para la instrucción del período se cargó un valor que es la división entre el reloj del microcontrolador con la frecuencia deseada del PWM (ecuación (15)). Cabe mencionar que no se abarca todo el código de la inicialización en esta sección, para mayor detalle del código ver la sección de Anexos 12.1

$$PWM_{freq} = \frac{f_{clk}}{f_{deseada}} \tag{15}$$

Empezando la subrutina de inicialización de los motores, se habilita el PWM en los pines respectivos GPIOS para cada motor y al mismo tiempo se definía que se quería usar la base del PWM0 y los 2 módulos correspondientes a esta base. Los módulos, también nombrados "canales" son: el canal 0 y 1 para el primer motor y el canal 4 y 5 del segundo motor. Ya seleccionando la base y los canales, se carga la frecuencia del PWM al microcontrolador. De ultimo se apagan los bits de los GPIOS correspondientes a los canales.

Para poder regular la velocidad de los motores su utiliza la función `motor-velocity-write` con la cual se puede ajustar el ancho de banda del PWM y de esta manera poder variar la velocidad en el motor. El encoder solo se encarga de procesar las señales del PWM en el puente H y realiza la regulación de potencia correcta. En esta función descrita en el código se puede observar que en una de las instrucciones se tiene como límite máximo la variable de ancho con el valor de cien. Esto quiere decir que si se sobrepasa este valor en la instrucción, tomara el valor y lo convertirá en cien. Es como tener una instrucción que pide un ciclo de trabajo de entre 0 a 100 por ciento. Para el código, ver Anexos 12.1. En el siguiente fragmento se describe el algoritmo para regular la velocidad en el motor.

```
// 1.) Si el ancho de pulsos es menor a cero el signo es menor a uno.
    sign = (width < 0) ? -1 : 1;

// 2.) Si el ancho de pulsos es menor a cero el ancho de pulsos es negativo.
    width = (width < 0) ? -width : width;

// 3.) Si el ancho de pulsos es mayor a 100 el ancho de pulsos es igual a 100.
    width = (width > 100) ? 100 : width;
```

```
// 4.) Se define el ciclo de trabajo con el que trabajara el motor.
    duty = (width) * (SysCtlClockGet()/(pwm_period * 100));

/* Si el signo no es negativo*/
    {
    //5.) Se activan los pines para el movimiento del lado derecho.
    }

/* Por otra parte*/
    {
    //6.) Se activan los pines para el movimiento del lado izquierdo.
    }
```

Como se puede observar la variable de width es donde se carga la velocidad y esta en el rango de menos cien a cien, si se presenta un número negativo en la función el motor cambia de sentido de giro. En el código también se muestra las instrucciones que se usan para conmutar los pines donde están los canales del PWM.

7.3.2. Función de encoder magnético

El encoder magnético por otra parte también, requiere de una función de inicialización y la otra para obtener los datos y guardarlos en una variable. Se tienen dos funciones de inicialización, cada una para el motor respectivo, parecido a las funciones de la librería del motor. Al principio fue importante tener en cuenta cual es la reducción del motor. En el caso del motor que se implementa para la palanca de inercia, se tiene una reducción de 100 a 1. Es importante ya que las sumas del encoder tienen que ser precisas y con esta reducción se puede establecer ángulos exactos con relación al eje del motor. Dentro de la configuración del encoder de cuadratura se establecen los límites de giro del motor dada por la siguiente relación (16)

$$\textbf{Límite} = \textbf{Relación del motor} \times \textbf{Pulsos por vuelta} \ , \tag{16}$$

donde el límite es el numero mayor que se leerá dentro de la instrucción de la cuadratura. Este valor es necesario para contar el numero de vueltas y la cantidad de grados que se desea. En este caso se tiene una relación de motor igual a 100 y los pulsos por vuelta son 12, ya que estos son los que leen los sensores de efecto hall, por lo tanto el límite es de mil doscientos. En un encoder se suma o se resta a la variable con la información del ángulo dependiendo de cuál cuadratura inicie antes. Si inicia la salida A antes que la salida B gira en un sentido y si la salida B ocurre antes, este gira en sentido contrario. Al momento de solicitar la lectura del encoder es importante tener en cuenta la ecuación (17). En esta ecuación se realiza una conversión que convierte en radianes al valor obtenido por la instrucción `QEIPositionGet`. Los valores que se obtienen con esta instrucción van de 0 a 1200 que es el limite descrito anteriormente.

$$\textbf{EncoderPos} = \textbf{QEIPositionGet} \times \left(\frac{2\pi}{\textbf{Relación del motor} \times \textbf{Pulsos por vuelta}} \right). \tag{17}$$

Una vez se obtiene el valor del ángulo en radianes, se realiza otra conversión de los ángulos para que estos presenten un rango de 0 a 180 y de -180 a 0 grados. Este rango es importante ya que la lectura que se realiza de la posición del ángulo de roll con la IMU utiliza este mismo patrón. En la siguiente ecuación se muestra la conversión.

$$\textbf{EncoderPosGo} = \arctan 2 \left(\frac{\sin\left(\textbf{Encoder Pos}\right)}{\cos\left(\textbf{Encoder Pos}\right)} \right) \frac{180^\circ}{\pi}. \tag{18}$$

Al final en la ecuación (18), se puede observar que el valor resultante se convierte de radianes a grados. El valor en grados es útil ya que puede servir como referencia en la consola serial y es más fácil de interpretar. El resultado final es la variable `EncoderPosGo` (encoder position goal) que se utiliza como sensor en la retroalimentación negativa del lazo de control del programa.

7.4. Librería de MPU6050

Para la utilización de la IMU (MPU 6050), se utilizó la librería provista por Tiva Ware [13]. Esta librería aprovecha distintas funciones del modulo MPU 6050, la principal siendo la obtención de datos del giroscopio en radianes sobre segundos y la del acelerómetro en gravedades. Esta librería es parte del Sensor Lib de Tiva Ware [13] y aprovecha el protocolo de comunicación I2C para establecer una comunicación con la IMU.

El protocolo I2C utiliza dos pines, uno es el SCL y el otro es el SDA. Uno se encarga del reloj mientras en el otro se transmite la información del proceso. En un ciclo de instrucción hay 19 bits en total, una secuencia que inicia con un bit de inicio luego un bit reconocimiento y luego empieza un byte donde se identifica al dispositivo al cual se quiere comunicar. Después sigue otro bit de reconocimiento y hay otro byte para la dirección de un registro. De ultimo hay otro bit de reconocimiento junto a un byte donde se coloca la data o información del registro seleccionado. Al final se termina con un bit de finalización de la secuencia, similar al bit con el que se empezó. En un bus de dispositivos conectados por I2C se pueden comunicar y hablar distintos dispositivos y se comunican siempre con el byte del dispositivo.

Es importante mencionar la inicialización del I2C, ya que también hay una interrupción con la cual se toman las medidas del MPU 6050. Al principio de la inicialización, se debe habilitar el módulo, configurar los pines del I2C0 que usa la Tiva C, luego ajustar los bits por segundo, después se limpia el bus de datos y por último se activa el módulo para que funcione con cierto reloj. De la misma manera que se

modificó el handler en el temporizador, también se modifica en el I2C. De esta manera se crea una función que sirve como función de interrupción.

El MPU 6050 también requiere de una función de inicio. En este caso se define una variable que funciona como bandera de reconocimiento y con esta se puede ir ejecutando las instrucciones de configuración paso a paso a medida que leen y reciben los comandos en el bus del I2C. Primero, se ejecuta la inicialización del MPU6050, después se ajusta las gravedades del acelerómetro, que en el código se ajustaron para que fueran de $+/-2g$. Luego se ajusta los grados por segundo del giroscopio, en este caso se ajustó un valor de $+/-250dps$. De ultimo se corren unas instrucciones para desactivar los magnetómetros, ya que estos no serán de utilidad en la aplicación.

Al final, en el ciclo principal del programa se realiza una instrucción para leer los datos de la IMU. Como la librería se encarga de hacer la conversión de los datos en bruto, solo se solicitan los valores del acelerómetro y giroscopio en las dimensionales que utiliza. Con los valores del acelerómetro en los ejes x y z, se pudo calcular la posición angular del eje Y, la cual se describe en la siguiente ecuación:

$$\theta_y = \arctan 2\left(\frac{\alpha_x}{\alpha_z}\right). \tag{19}$$

7.5. UART y lectura de señales

El módulo del UART fue simple en su implementación, y no fue del todo necesario en el programa, pero se utiliza para la adquisición y visualización de datos.Específicamente se utilizó para mostrar el valor de los datos de la IMU y del encoder magnético.

Para inicializar el UART, primero se habilita el módulo, luego se habilitan los GPIO que se utilizan como los pines TX y RX. De último se coloca el baudrate con el que se transmitirán los datos, para esto se utilizó un valor de 115200. En este caso, se especifican los pines. pero de igual manera no fue necesario utilizarlos ya que se pudo usar el puerto USB de la Tiva C. Para la impresión de los datos en la consola se utilizó la función de "UARTprintf". Con la que se puede imprimir textos y variables.

7.6. Implementación de control PID

Para el controlador PID, se usó el mismo algoritmo descrito en la sección 6.2 del marco teórico. En este algoritmo se inician las variables de error. Está el error actual, con el cual se podría hacer todo el control de lazo cerrado, el error futuro y el acumulado. El error futuro se multiplica con la constante derivativa y el error acumulado con la constante integral. Para el error actual (ecuación (20)) se hizo la diferencia del valor obtenido por la IMU menos el valor obtenido por el encoder

magnético. La variable w_k es la referencia, obteenida como el dato procesado con los filtros complementarios de la IMU.

$$e_k = w_k - \textbf{EncoderPosGo}. \tag{20}$$

El error actual y acumulado tienen una parte pasada, la cual es el valor previo del ciclo. Al principio del ciclo del programa estas variables se les asigna un valor de cero. El valor final que se obtiene del PID es la señal que se coloca en la instrucción de velocidad del motor y con esto se realiza la función principal del programa la cual es el movimiento del eje del motor con respecto al eje de roll de la IMU. El índice con la letra k, hace referencia al número iteraciones en el código. en la ecuación 9 se muestra el resultado.

Las constantes integral, diferencial y proporcional que se utilizaron no afectaron el comportamiento a simple vista. En el prototipo de auto balance, se seleccionó únicamente las constantes con un valor de uno. Se observó claramente que el efecto que realizan a simple vista no producen pequeñas vibraciones ni retraso en el tiempo en el cual se mueve el eje del motor. La constante integral y derivativa se operó con el tiempo del ciclo del temporizador, en este caso con 0.001 segundos. La constante integral se multiplicó con este valor y en la derivativa se dividió por este valor.

7.7. Diseño de filtros complementarios

Se desea poder expresar la posición angular del ángulo de roll, ya que con este se puede manejar la posición del eje del motor con respecto a la posición de la plataforma que esté conectada al mecanismo de auto corrección. El filtrado complementario es importante ya que con este se puede realizar un mejoramiento significativo en la lectura de datos de la IMU. Con este precisamente se obtienen los valores del giroscopio y del acelerómetro para poderlos unir y de esta manera obtener una lectura limpia de la señal.

El filtrado complementario consiste en la suma de dos filtros digitales. En un filtro se procesa una señal de posición angular obtenida por el giroscopio y en el otro se procesa la señal de posición angular obtenida con con acelerómetros, descrita en la ecuación (19). El filtro que se utiliza para la parte de los acelerómetros es un filtro pasa bajas y el que se utiliza para la parte del giroscopio es un pasa altas.

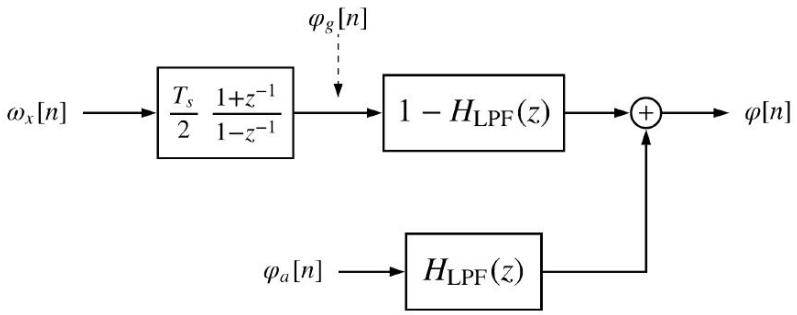


Figura 21: Filtrado complementario demostrado en un diagrama de bloques [12].

En este diagrama (21) se muestran cuatro variables importantes. La variable $\omega_x[n]$ es la entrada del giroscopio, $\varphi_g[n]$ es la posición angular del ángulo roll pero obtenido por medio del giroscopio, $\varphi_a[n]$ es la posición angular obtenida con los acelerómetros (ecuación (19)) y $\varphi[n]$ es el resultado final de la posición angular. En la Figura 21 también se muestra una ecuación de diferencias. Esta se utilizó para poder derivar la velocidad angular y de esta manera poder obtener una posición angular.

$$\frac{\varphi_g(z)}{\omega_x(z)} = \frac{\Delta t}{2}\frac{1+z^{-1}}{1-z^{-1}}, \tag{21}$$

$$\varphi_g\left[k\right] = \tfrac{\Delta t}{2}\left(\omega_x\left[k\right] + \omega_x\left[k-1\right]\right) + \varphi_g\left[k-1\right], \tag{22}$$

$$\varphi_g\left[k\right] = \Delta t \omega_x\left[k\right] + \varphi_g\left[k-1\right]. \tag{23}$$

La ecuación (21) necesita transformarse en el dominio de tiempo para que esta pueda ser implementada en un programa de manera iterativa, esto se expresa en la ecuación (22). La variable $k-1$ hace referencia a las variables pasadas dentro del programa similar a como se trabajaron los errores pasados del controlador PID. En la ecuación 23 se simplificó la expresión ya que el valor pasado del giroscopio se toma como el valor presente. El dos que divide a Δt se acumula como una constante. Las siguientes ecuaciones describen el desarrollo para implementar el filtro pasa bajas y pasa altas que se utilizaron. Las variables $X(z)$ y $Y(z)$ de las ecuaciones (26) y (24) respectivamente, corresponden a la salidas del filtro pasa altas y pasa bajas.

$$\frac{Y(z)}{\Phi_a(z)} = \frac{1-\lambda}{1-\lambda z^{-1}}, \tag{24}$$

$$y\left[k\right] = (1-\lambda)\varphi_a\left[k\right] + \lambda y\left[k-1\right]. \tag{25}$$

Las ecuaciones (24) y (25) corresponden al desarrollo del filtro pasa baja que se utilizará para procesar la posición angular de los acelerómetros. Las siguientes ecua-

ciones ((26),(27) y (28)) corresponden al desarrollo del filtro pasa altas que procesa el valor obtenido con la ecuación (23). La constante lambda en el código utilizó un valor de 0.9.

$$\frac{X(z)}{\Omega_x(z)} = 1 - \frac{1-\lambda}{1-\lambda z^{-1}}, \tag{26}$$

$$\frac{X(z)}{\Omega_x(z)} = \frac{\lambda\left(1-z^{-1}\right)}{1-\lambda z^{-1}}, \tag{27}$$

$$x\left[k\right] = \lambda\left(\omega_x\left[k\right] - \omega_x\left[k-1\right]\right) + \lambda * x\left[k-1\right]. \tag{28}$$

Al final se realiza la suma para obtener el resultado final de la posición angular.

$$\varphi\left[k\right] = x[k] + y[k] \tag{29}$$

En el código que se muestra en la sección de Anexos 12.1 El resultado final tiene una variable con diferente nombre.

CAPÍTULO 8

Implementación de hardware y diseño de robot

8.1. Actualización de piezas

Es importante resaltar que uno de los objetivos específicos fue la verificación de lo realizado en la fase anterior del proyecto [1]. En este documento se propone un diseño inicial para la plataforma del robot saltarín, y también se deja como tal un diseño CAD ya realizado.

Dentro de la fabricación real de los componentes se necesitaba de la impresión en 3D de dichos componentes. Los componentes fueron impresos y se lograron obtener buenas piezas de estos, aun así no se han utilizado todavía. Un detalle que no se pudo captar al principio de la impresión fue que el tamaño del robot es relativamente pequeño y carece de ciertos detalles con las cualidades necesarias. El robot fue diseñado de tal manera que los componentes electrónicos tienen que estar compactos en un espacio y el microcontrolador como tal tenía que ser únicamente un circuito integrado o microchip.

Este tipo de microchips son complejos de evaluar y programar sin el equipo especializado correcto. Necesitan de mayor recursos ya que deben estar montados superficialmente sobre un PCB y no tienen la flexibilidad de ser programados varias veces. Los microcontroladores que pueden ser programados por usuarios finales la mayoría de veces se venden en placas o kits de desarrollo como en el caso de la Tiva C y Arduino. Seleccionando la Tiva C se observa que las piezas diseñadas por Rivera [1] no cumplen con el requerimiento del tamaño.

Analizando este problema del tamaño pequeño de las piezas se tiene como alternativa necesaria el aumento de estas, y con esto se tendrá en mente tener piezas

que cumplan con el tamaño, que se puedan adaptar y ensamblar fácilmente con el controlador que se utilizó.

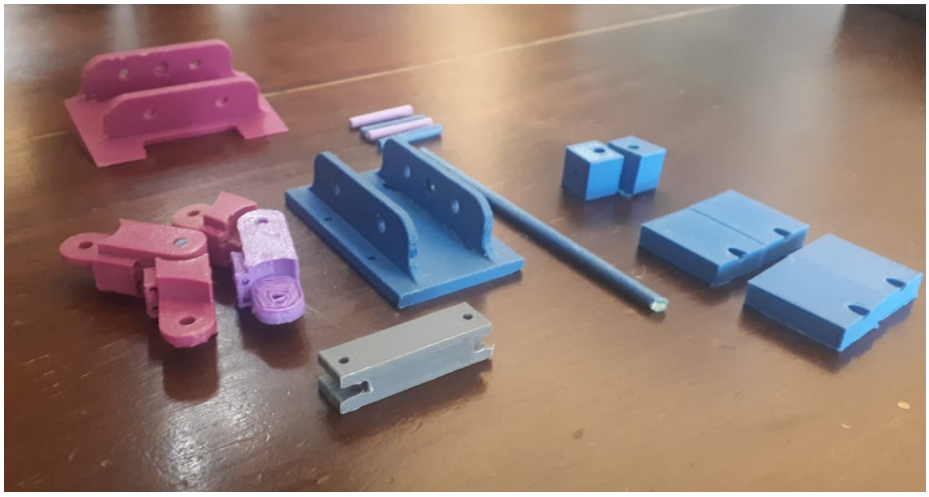


Figura 22: Resultado de las piezas impresas.

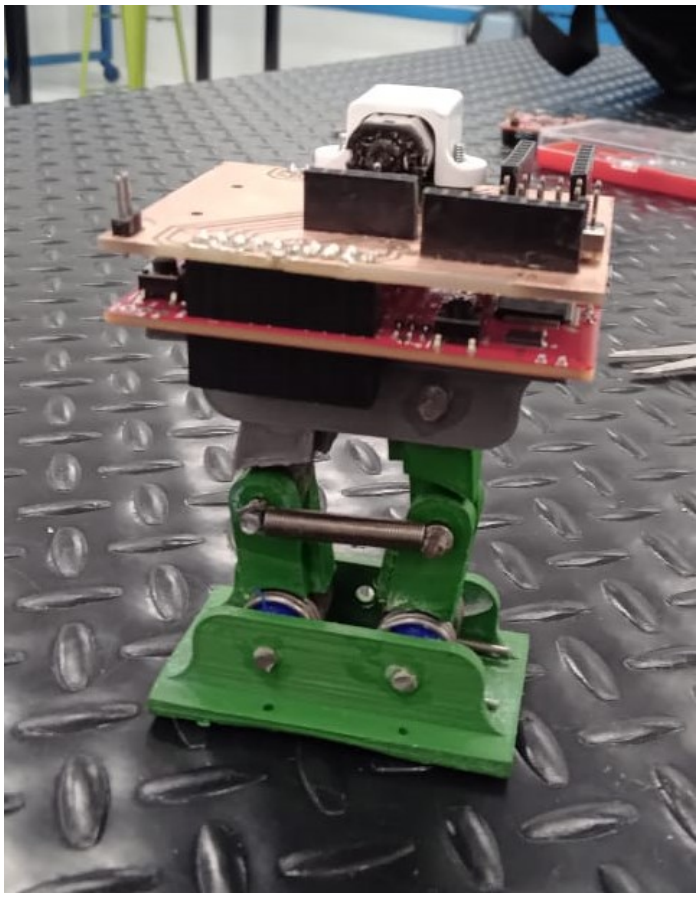


Figura 23: Visualización de robot armado.

En las Figuras 22 y 23 se puede apreciar todas las piezas que fueron impresas para el ensamblaje de dicho robot. En la Figura 23 se muestra el robot armado con todos los pines y piezas correspondientes. Los pines que se utilizaron al final fueron de hierro al igual que se colocaron unas roldanas de Poliuretanos termoplásticos (TPU:) en las uniones donde se colocan los resortes junto a los eslabones que conectan la base inferior con la superior. Estas roldanas (Figura 24) son de utilidad ya que sirven para sujetar y mantener firme las uniones de los resortes. En total se adquirieron 8 resortes y el ensamblaje fue el mismo descrito por Rivera [1] (el cual se describe en la Figura 26. En el diseño original se tienen unos pequeños orificios en la placa inferior y superior, estos orificios se utilizan para insertar el alambre de los resortes y sujetarlos. Estos no se utilizaron ya que se comprobó que estos hacen que el robot tenga muy poca carrera a la hora de comprimirse, almacenando de esta manera poca energía potencial. Por no utilizar los agujeros, se utiliza las roldanas(24) que generan soporte para los resortes flojos y las piezas flojas en la base.

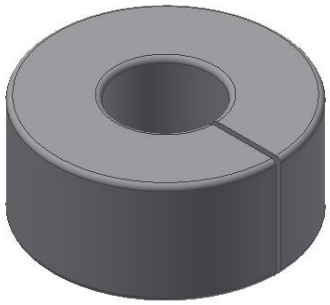
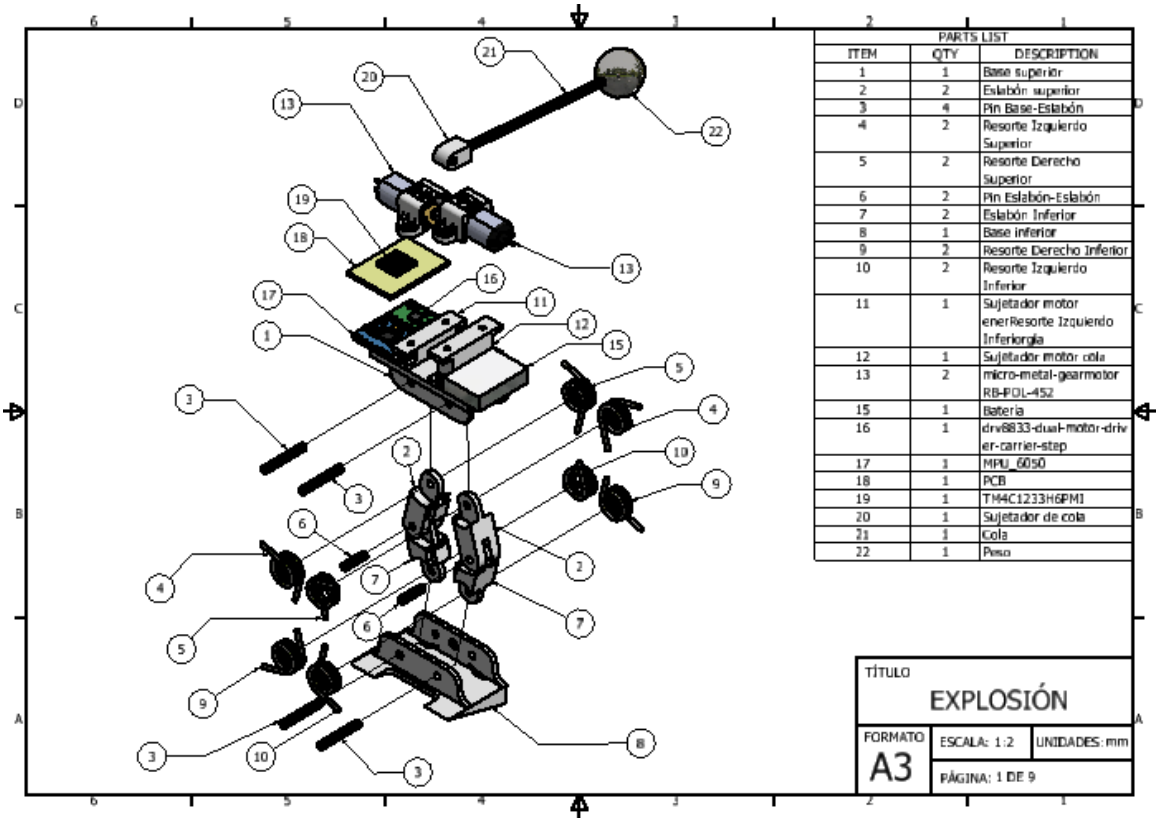


Figura 24: Roldana diseñada.

Figura 25: Resortes empleados.



PARTS LIST

ITEM	QTY	DESCRIPTION
1	1	Base superior
2	2	Eslabón superior
3	4	Pin Base-Eslabón
4	2	Resorte Izquierdo Superior
5	2	Resorte Derecho Superior
6	2	Pin Eslabón-Eslabón
7	2	Eslabón Inferior
8	1	Base inferior
9	2	Resorte Derecho Inferior
10	2	Resorte Izquierdo Inferior
11	1	Sujetador motor enerResorte Izquierdo Inferiorgia
12	1	Sujetador motor cola
13	2	micro-metal-gearmotor RB-PDL-452
15	1	Batería
16	1	drv8833-dual-motor-driver-carrier-step
17	1	MPU_6050
18	1	PCB
19	1	TM4C1233H6PM1
20	1	Sujetador de cola
21	1	Cola
22	1	Peso

TÍTULO: EXPLOSIÓN

FORMATO: A3 | ESCALA: 1:2 | UNIDADES: mm | PÁGINA: 1 DE 9

Figura 26: Plano de ensamblaje [1].

Los resortes se adquirieron por medio de un servicio dedicado a la fabricación de resortes, estos fueron diseñados con las mismas medidas que los resortes propuestos

por Rivera. A estos se les tuvo que modificar un poco, ya que se requirió cortar una parte del alambre. En la figura se muestran junto al diseño de la roldana empleada en el ensamblaje del robot.

Para la impresión de los componentes se utilizó el programa de Ultimaker Cura [14] siendo esto un entorno de "Manufactura asistida por computadora (CAM:)" para tecnologías de manufactura aditivas. Se especificaron varias opciones de la configuración de cada pieza. La altura de capa de impresión fue un ajuste importante, a este se le coloco un valor de 0.15 mm para que la impresión fuera mas rápida. Hubo piezas como los pines y la base de la plataforma que necesitaron de una estructura de soporte durante la impresión, esto para que la impresión fuera efectiva y el ácido poliláctico (PLA:) caliente no se derramara sobre la mesa. La base inferior se imprimió en un principio de una manera en la que la cara plana estuviera sobre la mesa de impresión, ya que era la posición más lógica y rápida de imprimir. La impresión resultó efectiva, pero la resistencia del material no fue la mejor. Ya que la posición de los filamentos era horizontal, se tuvo una fractura en el material, por lo cual se tuvo que imprimir la pieza de manera que la cara lateral estuviera encima del plano de impresión. En la Figura 27 se puede observar la comparación entre la misma pieza impresa en diferentes posiciones, cabe aclarar que la pieza en la derecha se tardó el doble de tiempo en imprimir. La pieza en la izquierda está fracturada y se puede ver como los filamentos son paralelos a la parte que se fracturó.

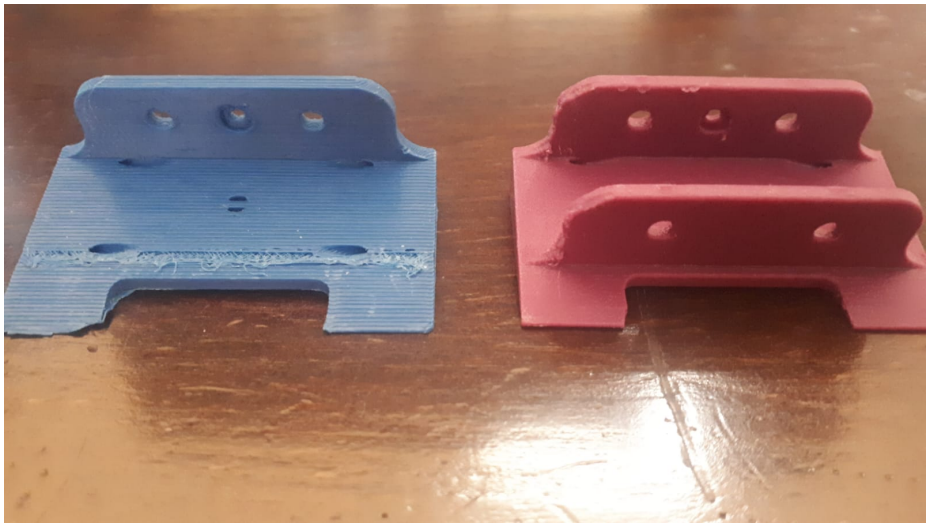


Figura 27: Comparación de pieza impresa en diferentes posiciones.

Para el aumento de las piezas, estas se fabricaron con un veinte por ciento adicional a su tamaño original. En la figura se muestra la configuración seleccionada para estas piezas.

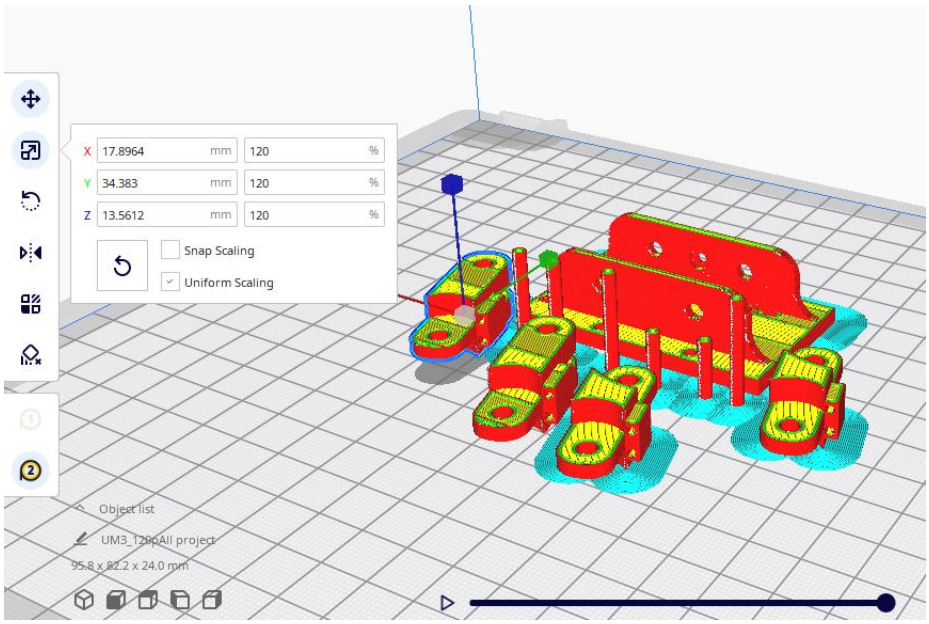


Figura 28: Configuración seleccionada para piezas aumentadas en tamaño.

Otro punto importante que no se resaltó en la fase previa del proyecto fue el mecanismo con el cual saltara el robot, necesario para poder impulsar los resortes y liberar la energía. Es necesario tener este tipo de mecanismo ya que la transmisión del motor necesita ser transformada para poder actuar sobre los resortes. Como propuesta se implementó un mecanismo sencillo de manivela corredera donde se tiene un motor en la parte superior del robot, este motor se conecta con un pequeño eslabón. Que también está conectado a un eslabón más grande. En un ciclo completo del motor se comprime y se extiende el robot, en la parte que se extiende es cuando el robot libera la energía potencial de la compresión. Se utilizo el programa "Linkage" para realizar una animación y una síntesis de posición, para encontrar las dimensiones correctas de los eslabones que se utilizaran en la Figura se muestra el diagrama empleado para encontrar las posiciones 29.

Figura 29: Diagrama de mecanismo manivela corredera .

Es importante resaltar que cuando se libera el salto, este no lo hará en su totalidad ya que siempre hay un torque en oposición cuando se mueve . Esta propuesta del mecanismo con un eslabón es similar a la que se utilizó en el proyecto del MSU jumper [2]. En las figuras 30 y 32 se muestra el mecanismo, la barra es el segundo eslabón y esta se conecta a la base inferior del robot. También se puede observar cómo el motor está en el borde de la base superior. Los motores estarán sujetados a la PCB por medio de abrazaderas de Polulu [3], la abrazadera que sujeta al motor requiere de una pequeña placa para ser levantada junto el motor. El motor esta en la mitad de la PCB y este interfiere con los pines soldados de los header que hay en la placa, por esto mismo se diseño una pequeña placa para levantar el motor, la cual se muestra en la Figura 33.

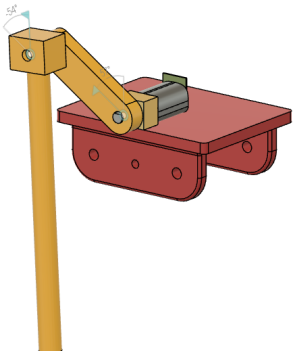


Figura 30: Disposición de mecanismo de salto.

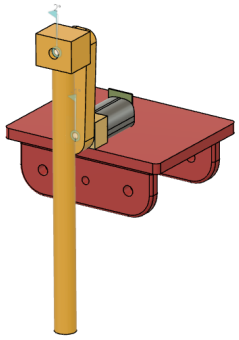


Figura 31: Mecanismo de salto.

Figura 32: Piezas para mecanismo de salto.

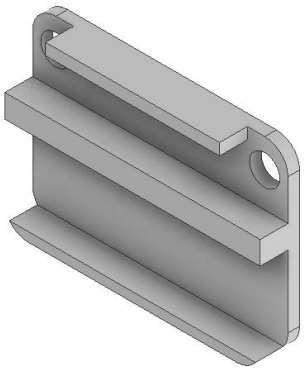


Figura 33: Placa diseñada para levantar abrazadera.

Una gran consideración que se tuvo durante la fabricación es que la parte superior del robot se diseñó de tal manera que los motores estén colocados en la PCB junto con los otros módulos y sensores. Abajo de la PCB esta la Tiva C y en medio se coloca la batería tipo LiPo. La Tiva C se coloca arriba del mecanismo de salto y se coloca con un buen adhesivo.

Una vez teniendo el mecanismo armado junto a los eslabones correspondientes se realizaron pruebas para poder probar sí el mecanismo en efecto podía comprimir la plataforma con los 8 resortes. En los videos que se muestra en la sección de Anexos se quitaron 4 resortes para 2 pruebas distintas que se realizaron sobre la plataforma. En la primera prueba se arma la plataforma con 4 resortes de manera cruzada donde los resortes no se colocan en los pequeños orificios de las placas. En esta prueba se tenía mayor carrera para la compresión, pero no suficiente rigidez en la plataforma. La segunda prueba se colocan los resortes también de manera cruzada en la plataforma solo que ahora se colocan con los agujeros de las placas. En esta disposición si se logra obtener mayor rigidez en la plataforma, pero no se tiene la misma carrera.

Ni en la primera y segunda prueba se logra comprimir la plataforma, ya que el pandeo es bastante que no se puede eliminar ni con las roldanas propuestas anteriormente. En la Figura 34 se muestra el robot ensamblado completamente con el segundo PCB diseñado y con el mecanismo de auto balance. Al final sí se logra validar las piezas de transmisión de potencia y se comprueba que no se llegó al comportamiento deseado.

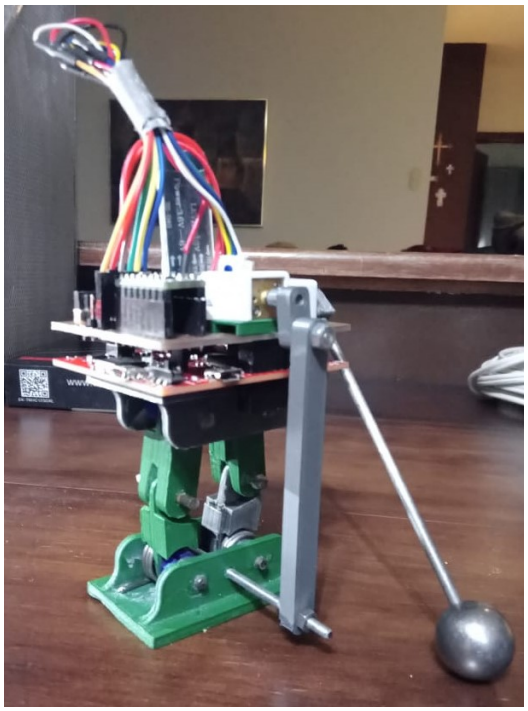


Figura 34: Modelo terminado del robot.

8.2. Diseño de esquemático y PCB

Para el diseño de la Placa de circuito impreso (PCB:) se utilizo el software "KiCad" [15]. En este se diseñó el esquemático principal de como se conforma el robot y con este mismo se diseñó una placa de circuitos impresos para poder colocar y controlar al robot de una manera efectiva. De diseñaron dos PCB, la primera se muestra a continuación como ejemplo y la segunda se muestra en la sección de Anexos. La segunda placa es la que se conforma para el diseño final, en esta la IMU esta en diferente orientación y al mismo tiempo hay agujeros para colocar las abrazaderas junto a los motores. La segunda placa tiene el mismo tamaño que una Tiva C y se diseño de esta manera para aprovechar la geometría y el diseño modular del robot, En esta también se incluyó un módulo ZS-040 para la conexión a bluetooth, donde este se usará para el control remoto por medio del modulo UART.

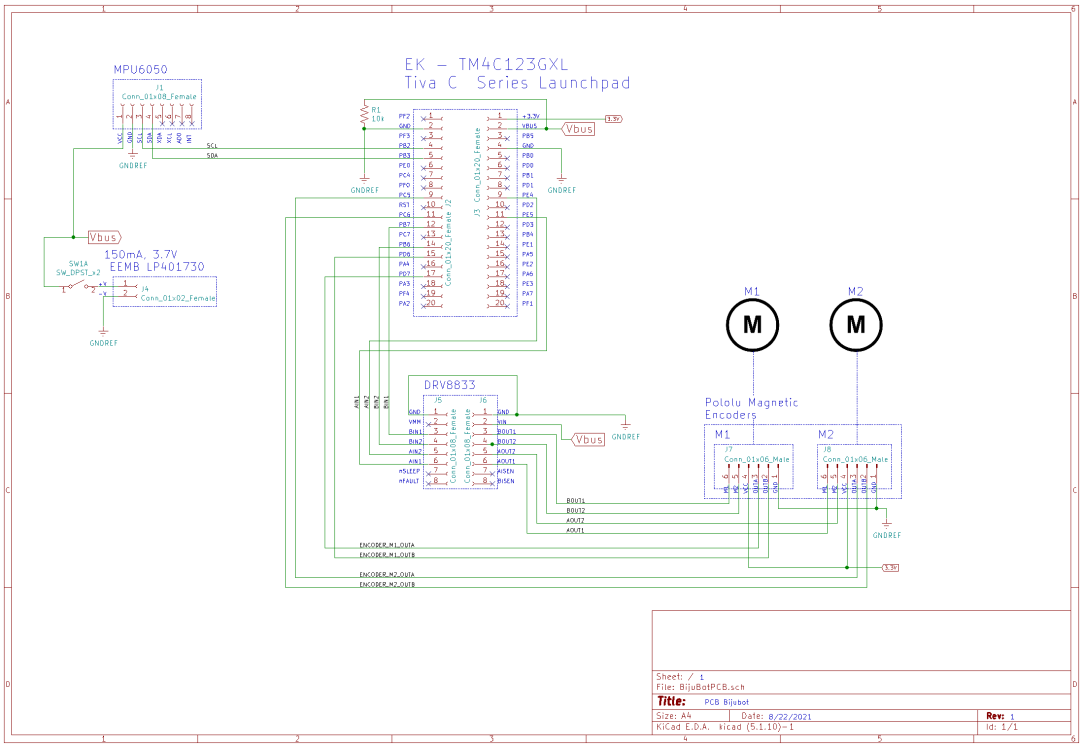


Figura 35: Esquemático general.

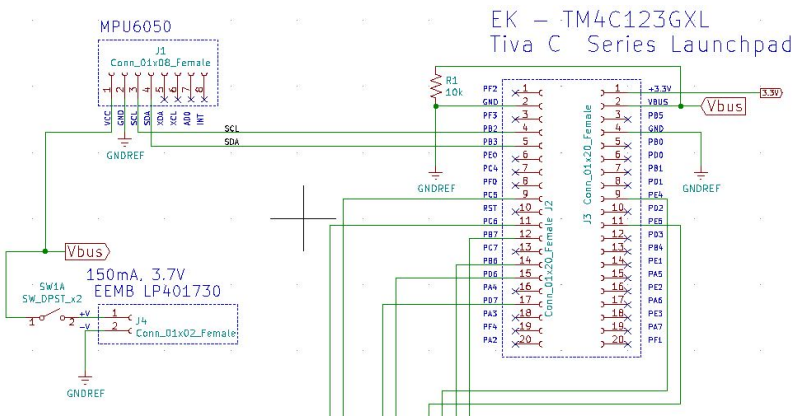


Figura 36: Conexión de MPU6050 y batería.

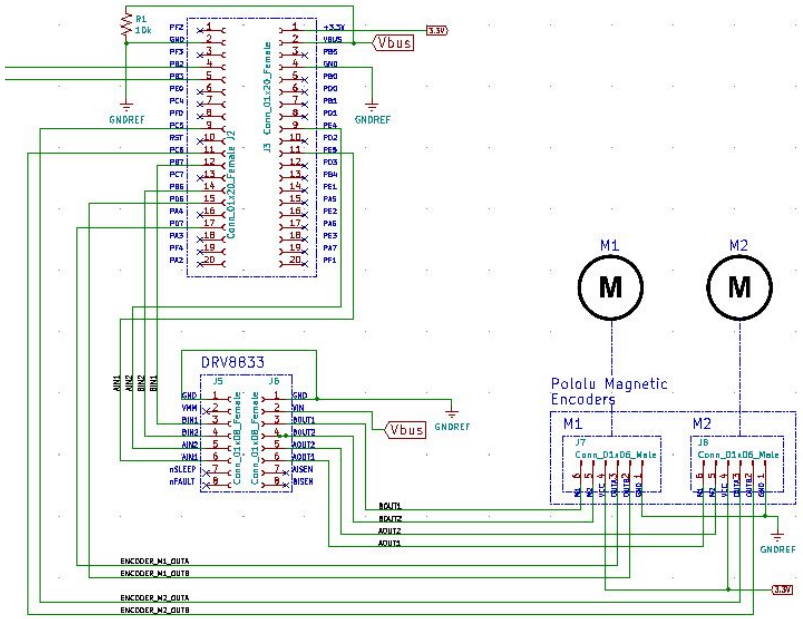


Figura 37: Conexión del Driver y encoders magnéticos.

En el esquemático se observa el orden y la distinta clasificación en la cual se tienen colocados los distintos componentes. Primero se observa la conexión del MPU 6050 a la Tiva C. Está conectado al módulo cero del I2C y los pines a los que se conectan son el PB2 y PB3. La batería Polímero de Litio (LiPo:) está conectada directamente a la Tiva C ya que el voltaje de 3.7 voltios es necesario para la alimentación de todo el robot. En la Figura 36 se muestra la conexión con el driver y los encoders magnéticos, la fuente que se usa para el driver y encoders es la misma batería LiPo.

Para el diseño de la PCB se tomó en consideración el tamaño de las pistas. Con una corriente de 2 A que se obtiene de la alimentación de salida del driver DRV8833. Con una densidad de capa de 2 oz/ft^2, se tiene un ancho de pista mínimo de 1 mm. Esta cantidad fue apropiada para fines prácticos y, por la capacidad de la fresadora

de PCBs, se utilizó un ancho de la misma cantidad. Las vías, para poder conectar las pistas de ambos lados de placa, se diseñaron con un agujero de 0.8 mm de diámetro estas se colocaron con pequeños remaches del mismo tamaño. En el esquemático de la Figura 35 se colocó una pequeña resistencia, esta se quita en el segundo diseño ya que no es necesaria y colocó como un error de diseño. En la fabricación de los PCBs, no se empleo y la conexión se queda abierta.

Los componentes que se seleccionaron para la placa fueron en su mayoría headers hembra. Se usaron para los pines de la IMU, del driver del motor y para las conexiones con la Tiva C. Los headers tipo macho se utilizaron para la conexión de los encoders magnéticos. La placa se acopla con los pines macho que tiene la Tiva C, ya que de esta manera esta puede estar sujeta en con estos pines y estar arriba de la Tiva C. En la sección de anexos se muestra el diseño de las placas.

8.3. Comportamiento y resultados de plataforma de auto balance.

Para empezar a probarlo se construyó el mecanismo en un protoboard de una galleta con el cual se demostraró el funcionamiento del sistema de manera simple y eficaz. Este contó con la base superior del robot donde se coloca la palanca de inercia. En la Figura 38 se muestra el prototipo desarrollado.

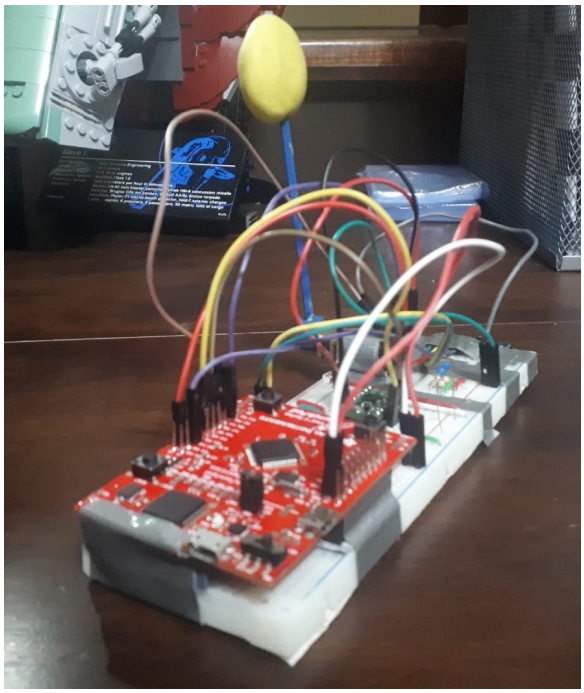


Figura 38: Prototipo de plataforma de auto balance.

Para poder realizar las pruebas y poder verificar el mecanismo de auto balance,

se lanzó el mecanismo y se observó cómo este se comportaba en el aire viendo si la posición era aceptable en el tiempo de vuelo. Para esto, se colocaron los componentes en la placa fabricada y se colocó el motor. Este sistema montado en la placa sigue teniendo aplicación como prototipo pero la placa y el orden presentan el diseño que se busca implementar. En la Figura 39 se muestra este.

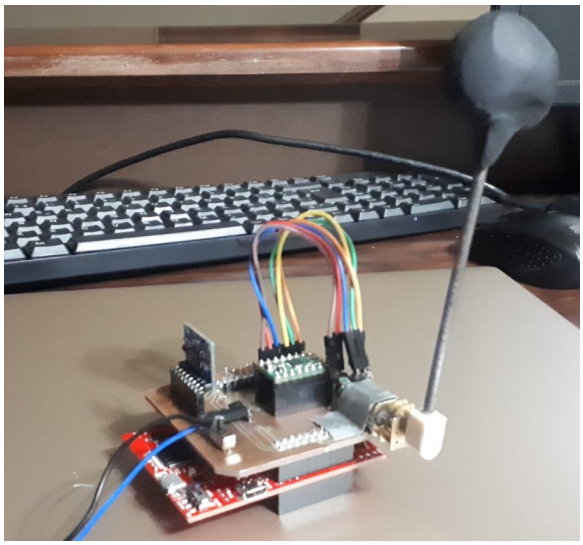


Figura 39: Sistema montado en placa conectado a la Tiva C.

Es importante especificar que la barra que se utilizó fue una de hierro, ya que las oscilaciones y las constantes caídas hacían que la barra original de PLA fuera inestable, ya que la flexibilidad de esta y la resistencia no fueron suficientes. Para el acople que se utilizó entre el eje del motor y la barra de inercia fue el mismo diseño realizado por Rivera [1], en la Figura 40 se muestra la pieza. Otra característica importante en la barra de inercia es el contrapeso. Para poder ir iterando en el proceso de pruebas se decidió utilizar plastilina en la punta de la barra, ya que con esta se puede ir removiendo material para poder ajustar el peso y de esta manera poder manufacturar un pequeño hierro que se ajuste a la barra inercial. En el último diseño se utilizó una bola de acera, obtenida de la esfera que se usa en los ratones antiguos, en la Figura 41 se muestra el resultado. Esta se tuvo que perforar con un barreno de banca para colocarla en la barra.

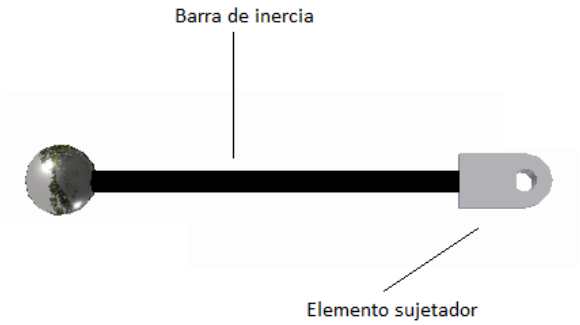


Figura 40: Acople de barra y barra inercial [1].

Figura 41: Esfera empleada para diseño final.

En general, la barra presentó más vibraciones cuando está en la placa que cuando está en el prototipo, ya que la IMU no está completamente fijada a los ejes, (hay un poco de holgura). El movimiento del motor y los pequeños movimiento generaron oscilaciones alrededor de la placa mecánica, estas oscilaciones afectaron a la sensibilidad de la IMU. En la sección de Anexos se muestra el comportamiento en la placa y en el prototipo.

Se realizaron dos escenarios de las pruebas de lanzamiento. En la primera prueba se coloco la barra de inercia de manera que girara en el mismo sentido del ángulo de roll de la IMU. En la segunda se se procuró que la barra girara en dirección contraria a la que gira la IMU en el ángulo de roll, esta es la manera adecuada ya que de esta manera se equilibra el sistema en el aire y la palanca maneja el peso de la placa compensando el movimiento contrario.

La palanca de inercia en el sistema de salto funciona como un péndulo que apunta a la dirección a la cual quiere saltar el robot. Este péndulo se comporta de manera tal que mantiene su trayectoria y dirección en el aire, de manera que si saltara el robot apuntando con la cola, la cola mantendría una posición igual en todo momento.

Experimentalmente no siempre fue el caso, ya que en los vídeos del primer escenario mostrados en la sección de Anexos 12.3 se puede observar el comportamiento de la palanca y de cómo se realiza cierta compensación en esta. Aun así, hay parámetros físicos que necesitan ser modificados, como es el caso del peso de la plataforma, el largo de la barra y el contrapeso de la barra. En el segundo escenario mostrado también en la sección de Anexos 12.3, se puede observar como la plataforma es arrojada y al mismo tiempo nunca cambia de dirección en el aire porque se compensa el movimiento con la barra inercial.

Para verificar el cambio de movimiento se utilizó el software de "Tracker" para capturar el movimiento del cambio de la palanca en el aire. En este programa se obtenían puntos de coordenadas que se definían manualmente, durante el proceso de la captura de movimiento. Los puntos que se capturaron fue el movimiento de la esfera y los puntos de la placa. Lo que se trató al final fue demostrar el ángulo que hay entre la barra inercial y la placa, con esto mostrar una tendencia en el cambio del movimiento en el aire, este ángulo se obtuvo con ciertas relaciones trigonométricas para calcularlo. En Matlab se realizó un pequeño script donde se emplean las ecuaciones de trigonometría desarrolladas y se calcula el ángulo correspondiente entre la barra de inercia y la placa. En las Figuras 42 y 43 se muestran los resultados obtenidos de la variación del ángulo de la palanca conforme el tiempo de vuelo. Se puede verificar efectivamente que esta cambia su posición y si se genera una compensación de movimiento en el aire, donde si puede afirmar que el sistema funciona como se esperaba y se logra cumplir con uno de los objetivos que era validar esto mismo. En la Figura 44 se muestra también la configuración con las trayectorias del primer escenario de los puntos medidos en el software de Tracker.

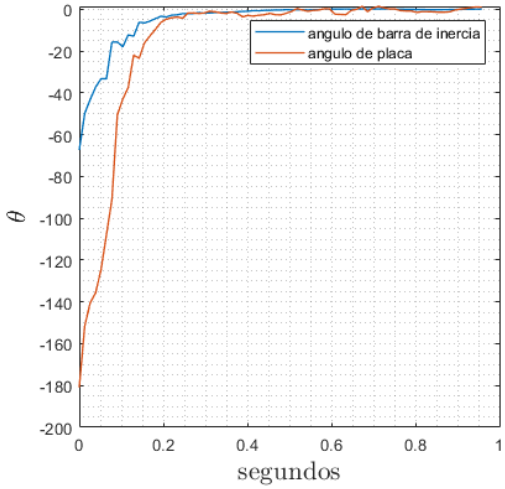


Figura 42: Gráfica generada para verificación de cambio de ángulo.

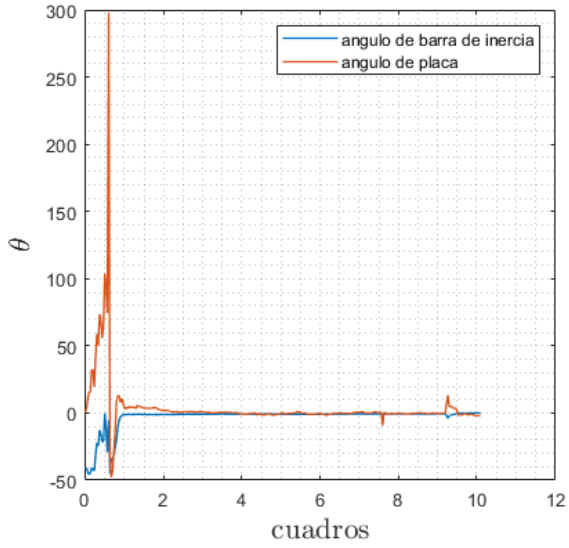


Figura 43: Gráfica generada para verificación de cambio de ángulo durante segundo escenario.

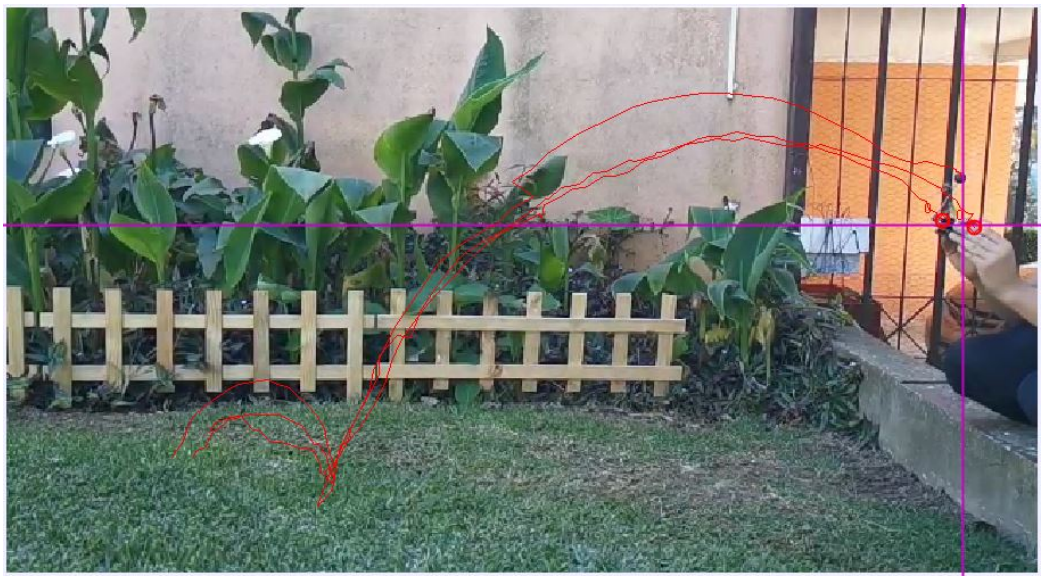


Figura 44: Trayectorias en el tiempo de vuelo medidas en programa de captura de movimiento.

Figura 45: Segundo escenario prueba de lanzamiento

También se realizaron pruebas con respecto al movimiento del eje del motor y la IMU a través del módulo serial con una gráfica en tiempo real que demuestra el comportamiento de cambio de ángulos en el tiempo. Estas pruebas se realizaron en el prototipo de la Figura 38 ya que en este prototipo se presentaba un poco más de fluidez de movimiento en la barra, ya que el espacio entre componentes no era tan justo y se presentaban menos vibraciones en general. En las siguientes figuras se muestran los distintos comportamientos obtenidos. Cabe resaltar que la función de color celeste representa a a la IMU y la de color negro es el encoder magnético.

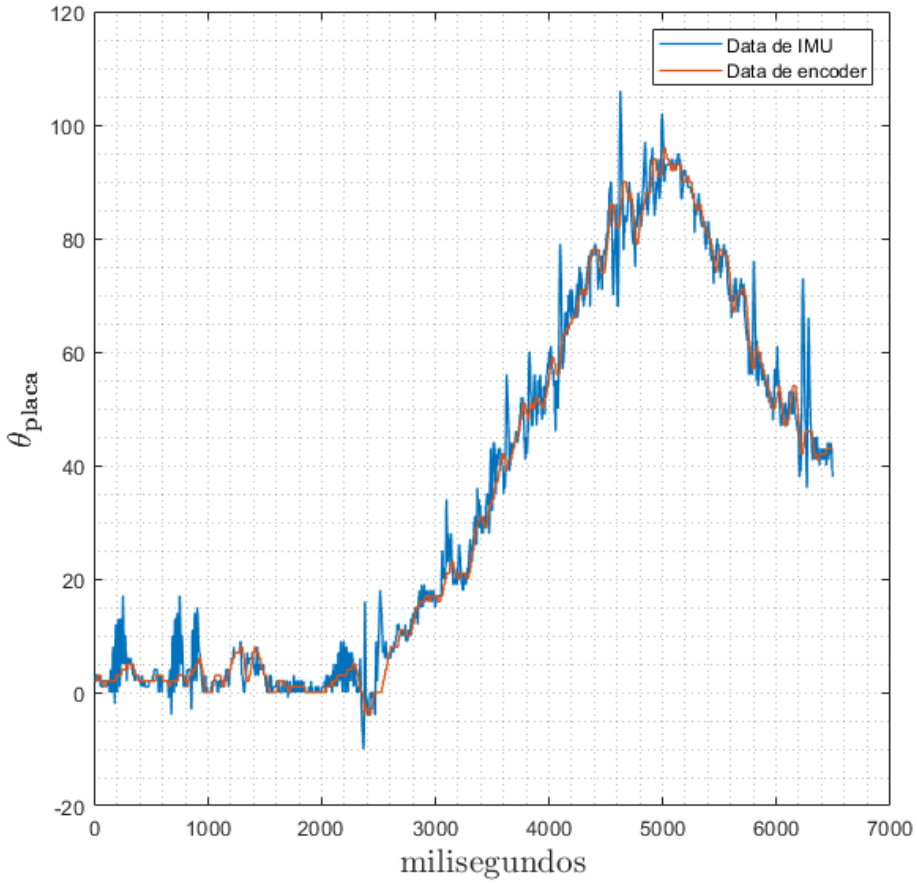


Figura 46: Comportamiento de IMU y encoder magnético.

En esta Figura 46 se muestra el rastreo realizado por el encoder magnético, claramente se puede observar como la señal se aproxima y casi se adhiere a la señal manejada por la IMU.

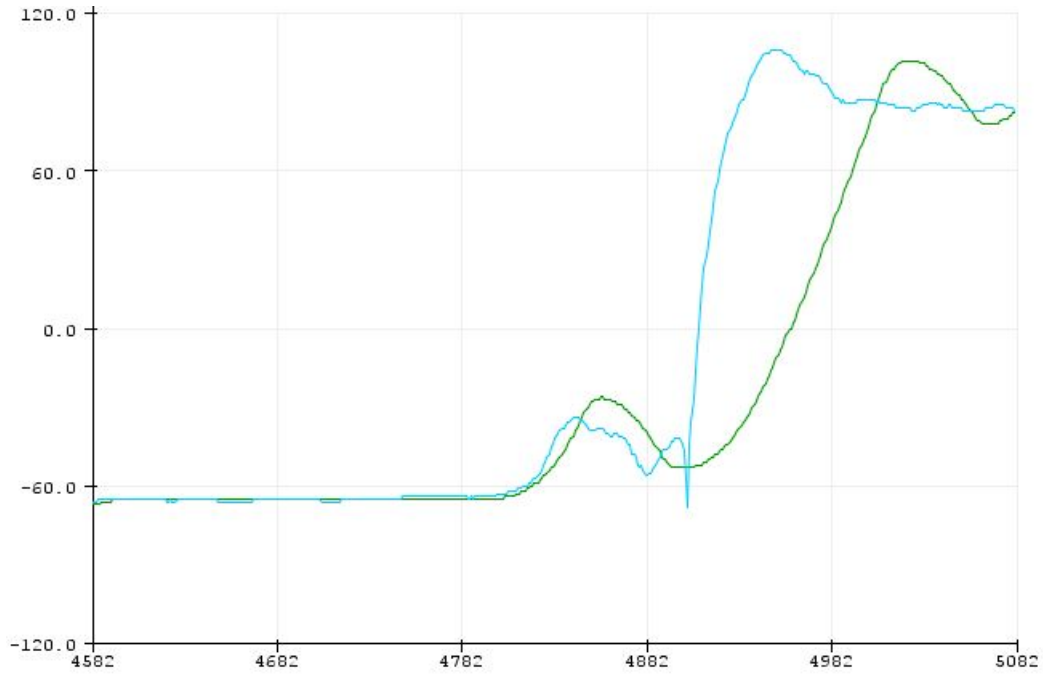


Figura 47: Respuesta rápida con delay en encoder

En la Figura 47 se puede observar cómo existe un delay de tiempo entre las dos funciones. El eje horizontal corresponde al tiempo, cada número colocado en la escala horizontal corresponde a 100 milisegundos, y el eje vertical son los ángulos representados. Este fue el comportamiento que presentó el sistema de auto balance cuando se mueve rápidamente de -90 a 90 grados respectivamente. Se observa como hay un cambio brusco y el encoder se separa cierto tiempo de la IMU. Estos datos fueron tomados cuando se tiene una retroalimentación negativa sin ningún lazo de control. Se vio que este control fue bueno pero no se pudo concluir que era el definitivo todavía ya que era necesario probar el controlador que se propuso en el trabajo de Rivera [1].

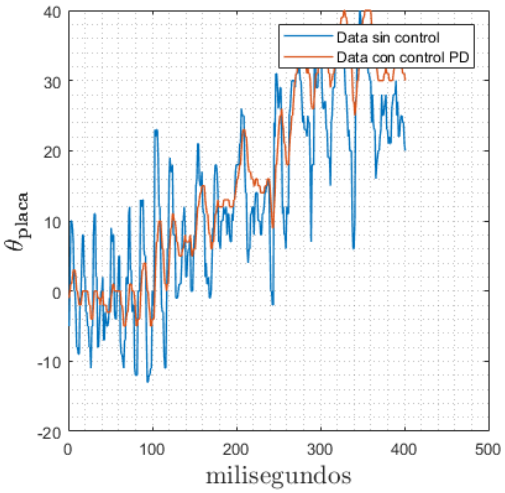


Figura 48: Señal con sistema de control aplicado.

En la Figura 48 se muestra el comportamiento de la señal con control, esta es la señal que se colocó en la función del motor después que se realiza el sistema de control con el procesamiento de la IMU y del encoder magnético. Esta señal es el resultado del controlador PID y con esta se pudo verificar que a movimientos más bruscos la gráfica se disparan. Cuando hay movimientos suaves la gráfica se mueve generalmente entre 1 y -1. En este caso se tomo la gráfica con un movimiento repentino del sistema de auto balance.

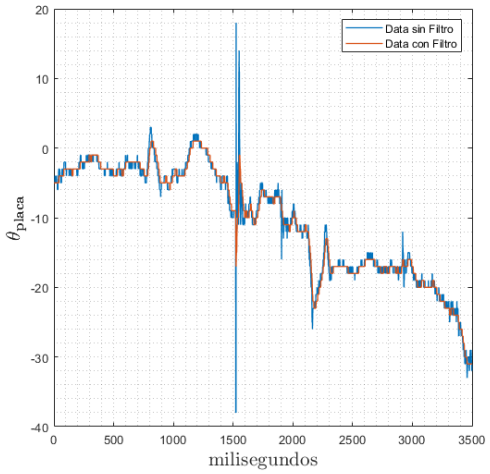


Figura 49: Ruido en la señal de la IMU.

La Figura 49 verifica el comportamiento de la IMU sin y con filtros complementarios. El filtro complementario también fue importante porque la lectura del ángulo con los acelerómetros sólo puede captar una rotación entorno a la gravedad mientras que cuando se implementan los filtros complementarios se puede captar un ángulo cuando

se esta girando en un desplazamiento horizontal como es en el caso del sistema de auto balance. También se puede observar en las figuras como el filtro complementario elimina una buena parte del ruido que se presentan en el ángulo de roll de los acelerómetros. Los pequeños pulsos que se muestran en la Figura 48 son despreciables, y son menos frecuentes que el ruido en la Figura 49.

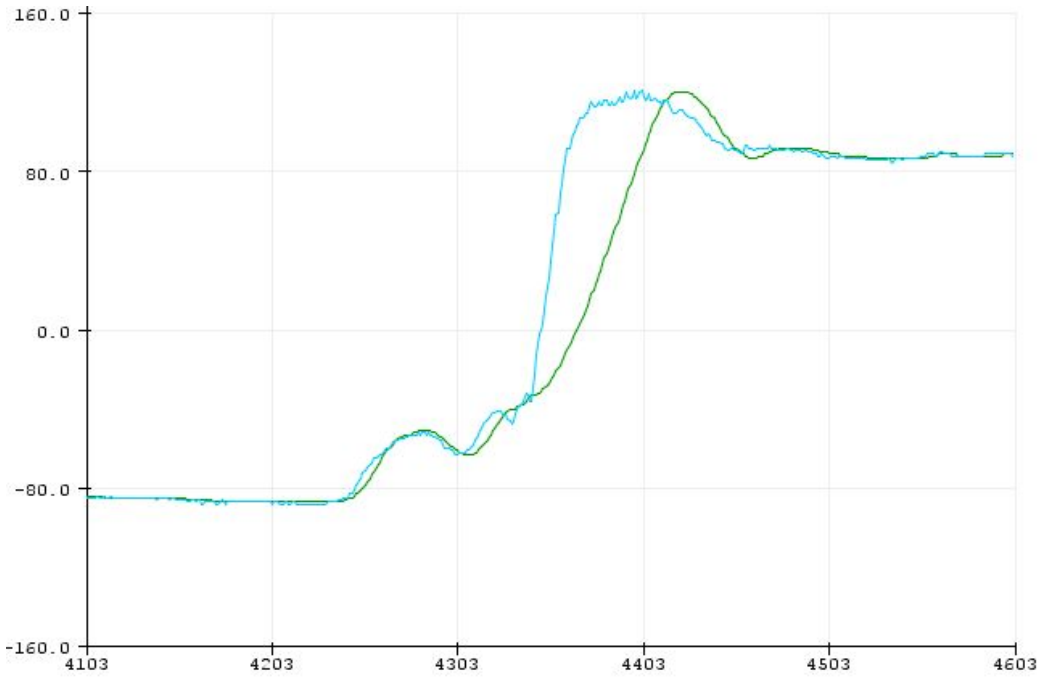


Figura 50: Respuesta utilizando control PD recomendado.

A la hora de utilizar el controlador PD con constante k_p de 0.86 y constante k_d de 0.055, se observa un comportamiento muy similar al realizado por la Figura 50. Lo que se puede evidenciar a simple vista es que el retraso y sobre amortiguamiento disminuye significativamente, siendo alrededor de 30 milisegundos de diferencia para el tiempo de asentamiento, por lo que se puede afirmar que el control utilizando con las variables presentadas es eficiente, ya que garantizó un tiempo de asentamiento adecuado.

8.4. Ajustes y observaciones

Como se comentó en la sección 8.1, el aumento del mecanismo de salto cambiara en su gran parte el diseño que se tenía establecido en un principio. El robot puede que no logre saltar los 88 cm que se tenía pensado pero va a lograr desplazarse saltando Si se cuenta el peso adicional que se estará utilizando sera alrededor de 39 gramos. Este peso cambia la dinámica de la palanca de inercia de cierta manera ya que la masa del robot es una variable a considerar.

En las recomendaciones realizadas por Rivera [1] se comentó que se necesitará

observar y realizar cálculos similares a [16] que consiste en el control aéreo de la palanca de inercia. Lo que se escribe en este paper, es cuáles serán los parámetros mecánicos óptimos para que el sistema de auto balance logre estabilizar al robot en el aire. El cálculo de estos parámetros en el paper fue realizado con el desarrollo de una ecuación lagrangiana de conservación de energía y sistemas dinámicos. Lo que se buscaba al final era poder optimizar los parámetros como es el caso del largo de la barra, la masa que se coloca en la barra y la relación del tren de engranajes reductor para el motor. Teniendo en cuenta estas variables se utilizaba una función objetivo y se resolvía empleando la toolbox de Matlab para optimización.

Este procedimiento puede realizarse dadas las circunstancias, pero el método que se seguirá utilizando es por medio de las iteraciones en las pruebas del mecanismo de auto balance donde se logrará corregir hasta obtener el mejor grupo de medidas para los parámetros buscados para la cola.

CAPÍTULO 9

Conclusiones

Para la parte del software se pudo llegar a desarrollar un programa que incluye las características necesarias para el funcionamiento completo de la palanca de inercia en el robot, siendo esta el medio principal con el cual el robot se auto ajustará en el aire. Por ende se puede afirmar que se implemento un algoritmo efectivo para el rastreo y posicionamiento del motor atreves de la unidad de medición inercial.

Teniendo en cuenta el controlador que se implementó en el algoritmo del código también se pudo afirmar que las constantes que se utilizaron en el control PD son adecuadas y generan un comportamiento deseable, donde se minimizan las vibraciones de la barra de inercia.

Para la parte de hardware, se diseñaron placas de circuitos impresos donde se pudo verificar el funcionamiento de la palanca de inercia, la cual fue diseñada adecuadamente pero se tendrá que modificar debido al rediseño mecánico que se tendrá que realizar para el robot. Teniendo en cuenta el rediseño también, se puede asegurar que el diseño del mecanismo presentado es el más sencillo y seguro de implementar ya que se basa en un mecanismo de manivela corredera y la liberación de la energía potencial se puede dar debido al rápido movimiento del eje del motor. En las pruebas finales del funcionamiento de este mecanismo no se logro comprimir los 8 resortes al mismo tiempo. Se colocaron 4 resortes y con estos sí se podía comprimir la placa. El mecanismo no logro comprimir el robot por sí solo ya que no había rigidez suficiente en los eslabones del robot. Con todo esto se logro manufacturar y evaluar la plataforma robótica junto a sus piezas de transmisión de potencia.

CAPÍTULO 10

Recomendaciones

Es importante destacar que el mecanismo de salto es indispensable para que se pueda realizar una compresión de los resortes de torsión en el robot. Para esto se necesitará de un mecanismo muy similar al planteado, solo que esta vez colocarlo en medio de la plataforma para que el movimiento sea completo y el mecanismo no genere pandeo como se muestra en los resultados de la placa. Puede ser un mecanismo similar al implementado por el MSU Jumper [17].

Es recomendable realizar cierta modularidad en el código a implementar debido a la facilidad de depuración y visualización de código a la hora de la búsqueda de variables, instrucciones y datos en esta. En el caso del código implementado para el mecanismo de auto balance, se realizo cierto orden y secciones para el código, pero no se evitó el diseño de ciertas librerías como es el caso del control PID y filtros complementarios.

Para la palanca de inercia es recomendable la utilización de un material sumamente rígido, debido a que la flexibilidad puede afectar en vibraciones y en las perturbaciones del sistema. En este proyecto se implementó en un principio con un material de PLA, luego se utilizó una barra de hierro y los resultados en cuanto al movimiento de la palanca mejoraron significativamente.

También es recomendable añadir pequeñas modificaciones al código, donde se realizan programas para el control remoto del robot, por medio del conexión por "bluetooth" y también un pequeño programa para poder mover la palanca de inercia.

Es importante resaltar que también es importante diseñar piezas con tolerancias dentro del software de Autodesk Inventor, ya que durante el ensamblaje es importante poder colocar y sujetar piezas con las medidas estipuladas. De esta manera no se deforman las piezas y no es necesario realizar un proceso de desgaste.

CAPÍTULO 11

Bibliografía

[1] J. Rivera, "Diseño e implementación de un robot bioinspirado con mecanismo de salto y control de orientación por medio de palanca de inercia," tesis de licenciatura en ingeniería mecatronica, Universidad del Valle de Guatemala, 2020.

[2] J. Zhao, J. Xu, B. Gao, N. Xi, F. J. Cintrón, M. W. Mutka y L. Xiao, "MSU Jumper: A Single-Motor-Actuated Miniature Steerable Jumping Robot," *IEEE Transactions on Robotics*, vol. 29, n.º 3, págs. 602-614, 2013.

[3] P. Corporation. (2020). "100:1 Micro Metal Gearmotor LP 6V with Extended Motor Shaft," dirección: `https://www.pololu.com/product/2204`.

[4] I. Kollár, R. Pintelon y J. Schoukens, "Frequency Domain System Identification Toolbox for MATLAB," *IFAC Proceedings Volumes*, vol. 24, n.º 3, págs. 1243-1247, 1991, 9th IFAC/IFORS Symposium on Identification and System Parameter Estimation 1991, Budapest, Hungary, 8-12 July 1991, ISSN: 1474-6670. DOI: `https://doi.org/10.1016/S1474-6670(17)52521-5`. dirección: `https://www.sciencedirect.com/science/article/pii/S1474667017525215`.

[5] E. Tutorials. (2021). "Hall Effect Sensor," dirección: `https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html`.

[6] P. Corporation. (2021). "Magnetic Encoder Pair Kit for Micro Metal Gearmotors, 12 CPR, 2.7-18V," dirección: `https://www.pololu.com/product/3081`.

[7] ——, (2021). "Dual MC33926 Motor Driver Carrier," dirección: `https://www.pololu.com/product/1213`.

[8] ——, (2021). "DRV8833 Dual Motor Driver Carrier," dirección: `https://www.pololu.com/product/2130`.

[9] M. Zea, "El Controlador PID," en *IE3006 - Sistemas de Control 1*, 2019.

[10] TDK$_{I}$*nvenSense*. (2021). "MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices," dirección: `https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/`.

[11] R. Z. y. M.-W. L. Li Jin Shi-Yang Qin. (2020). "High-sensitivity tunneling magnetoresistive micro-gyroscope with immunity to external magnetic interference," dirección: https://www.nature.com/articles/s41598-020-73369-6.

[12] M. Zea, "Fusión de Sensores para una IMU," en *IE3032 - Procesamiento de Señales*, 2019.

[13] T. I. Incorporated, *TivaWare™ Sensor Library*, Texas Instruments 108 Wild Basin, Suite 350 Austin, TX 78746, 2020.

[14] Ultimaker. (2021). "Ultimaker Cura," dirección: https://ultimaker.com/es/software/ultimaker-cura.

[15] KiCad. (2021). "KiCad EDA Design Tool," dirección: https://componentsearchengine.com/library/kicad?gclid=CjwKCAjw49qKBhAoEiwAHQVTo4tiWBgedsjgCg6g_I0WcIrszXHqlD-AezCtJg2VBkTwwz82mWRvXxoCs0cQAvD_BwE.

[16] J. Zhao, T. Zhao, N. Xi, F. J. Cintrón, M. W. Mutka y L. Xiao, "Controlling aerial maneuvering of a miniature jumping robot using its tail," en *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, págs. 3802-3807.

[17] J. Koh, S. Jung, M. Noh, S. Kim y K. Cho, "Flea inspired catapult mechanism with active energy storage and release for small scale jumping robot," en *2013 IEEE International Conference on Robotics and Automation*, 2013, págs. 26-31.

CAPÍTULO 12

Anexos

12.1. Repositorio de código de plataforma de auto balance

En este enlace se encuentra el repositorio del programa del sistema de auto balance. Se puede acceder a varios archivos con formato tipo C, como el caso de las librerías y el codigo principal:

`https://github.com/choriegojose/BijuBot`

12.2. Drive de archivos y documentos trabajados

En este enlace se encuentra el google drive donde se muestran los documentos trabajados y distintas versiones

`https://drive.google.com/drive/folders/1AjQ_TRFWhqBjpflNYIy-4wtCmK3iOLcs?usp=sharing`

12.3. Vídeos de funcionamiento, prototipo y placa

12.3.1. Vídeo demostrando funcionamiento de sistema de auto balance con el prototipo

`https://www.youtube.com/watch?v=LWmH-J5yoro`

12.3.2. Vídeo demostrando funcionamiento de sistema de auto balance con PCB construido

https://www.youtube.com/watch?v=Xl26l_FUdOM

12.4. Vídeos de prueba

12.4.1. Primera prueba de lanzamiento de sistema de auto balance de primer escenario

https://drive.google.com/file/d/1vowSoz5e4Yj-gM61hP2-t-zPvdeDk_uS/view?usp=sharing

12.4.2. Segunda prueba de lanzamiento de sistema de auto balance de primer escenario

https://drive.google.com/drive/u/1/folders/17RTrbzqRJdh4DeCO1CCHMZHDV5bTJWy8

12.4.3. Tercera prueba de lanzamiento de sistema de auto balance de primer escenario

https://drive.google.com/file/d/1Xp1ct3qPYTTQVxcNDWtRLQe_sgQmE9Nb/view?usp=sharing

12.4.4. Primera prueba de lanzamiento de sistema de auto balance de segundo escenario

https://drive.google.com/file/d/1e4EDc9MaSJbGa3uLSW8hBPJ7b3jTW60L/view?usp=sharing

12.4.5. Segunda prueba de lanzamiento de sistema de auto balance de segundo escenario

https://drive.google.com/file/d/1ZYbJ-LWOaCmxc6wN1fgptNZOBeB1ob_f/view?usp=sharing

12.4.6. Tercer prueba de lanzamiento de sistema de auto balance de segundo escenario

https://drive.google.com/file/d/1a0R4HU9xOg3xpUtIMB2EisGABqwi6kIf/view?usp=sharing

12.5. Pruebas de compresión de resortes

12.5.1. Prueba de mecanismo para comprimir resortes con la carrera larga

https://drive.google.com/file/d/1RMCFkJuGJZ5sLKop2YV6SBdHbfsKfIiD/view?usp=sharing

12.5.2. Prueba de mecanismo para comprimir resortes con la carrera corta

https://drive.google.com/file/d/1_vJdUhkEZ8vzf9IccuUEajsekuq8Enzo/view?usp=sharing

12.6. Esquemáticos y diseños de PCB

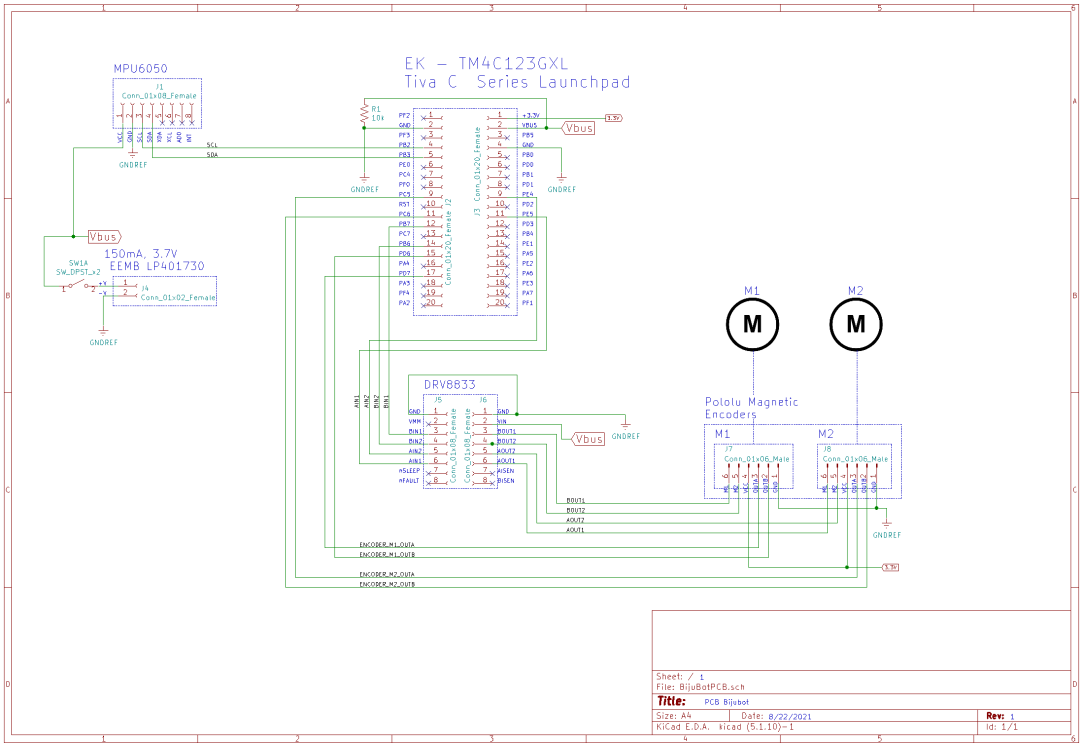


Figura 51: Esquemático general de primer placa.

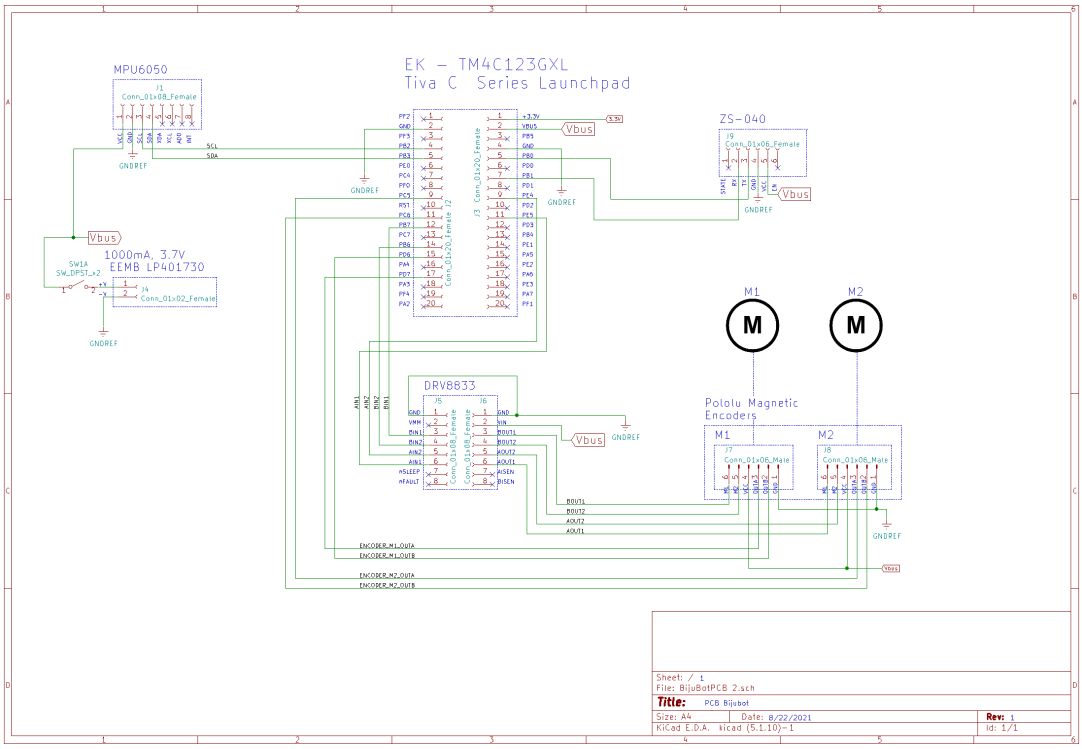


Figura 52: Esquemático general de segunda placa.

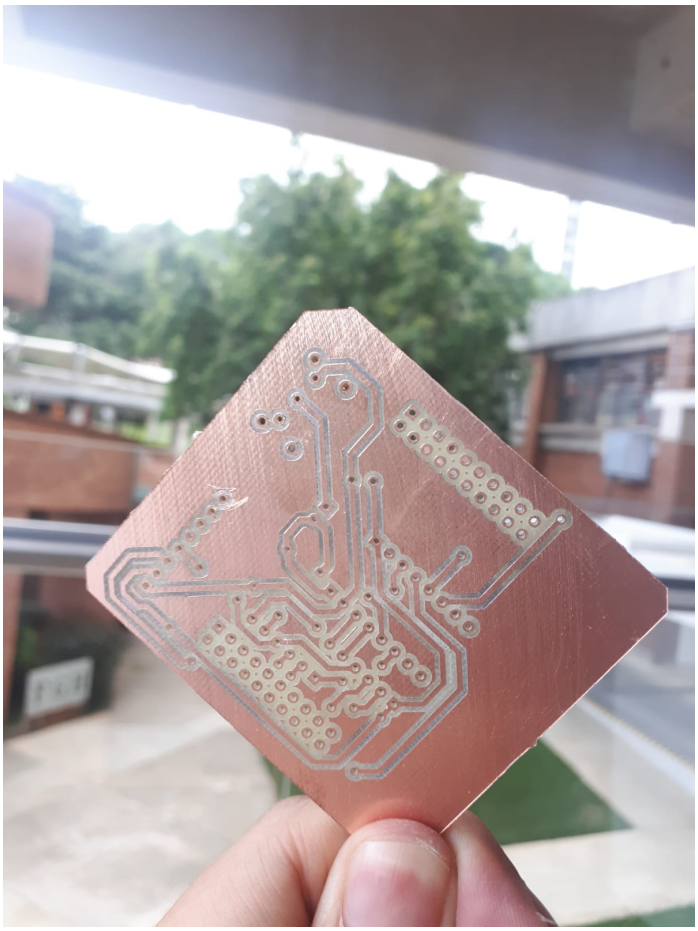


Figura 53: Placa sin soldar.

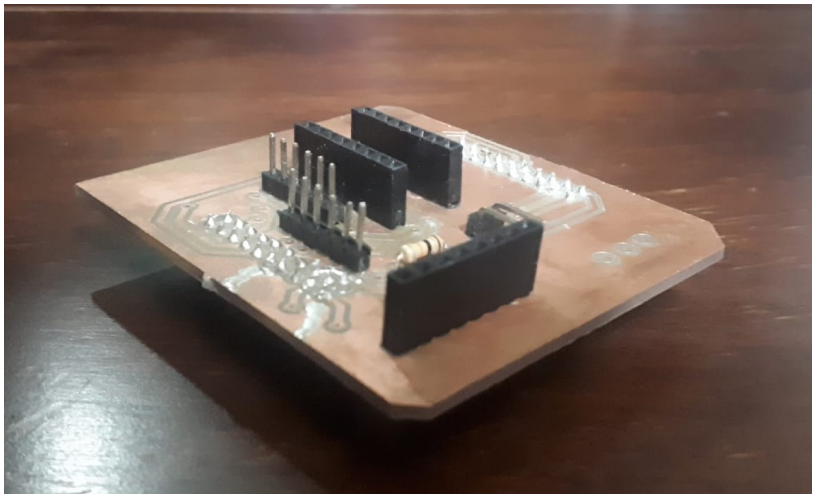


Figura 54: Resultado final de placa.

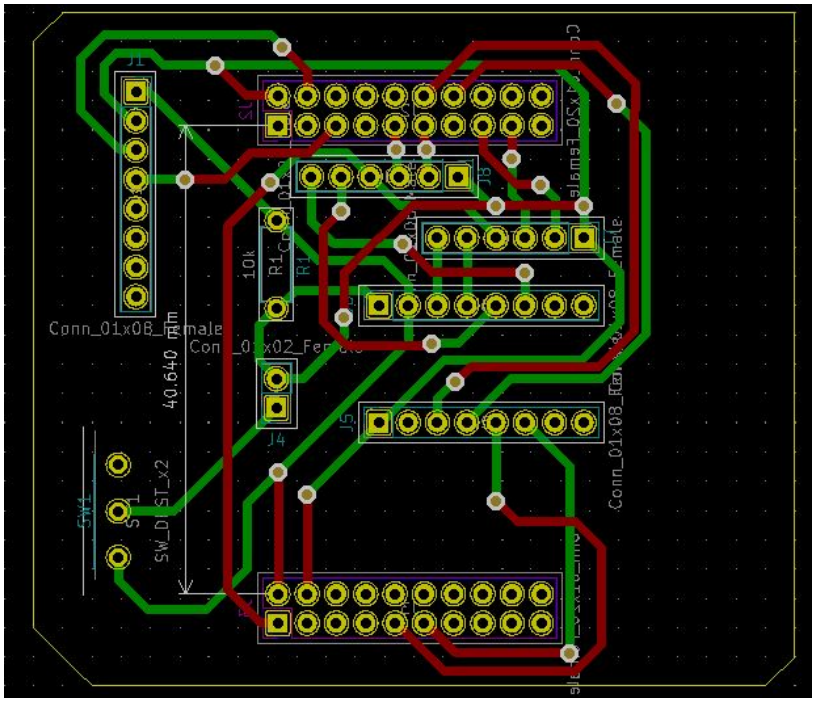


Figura 55: Diagrama del primer diseño de PCB.

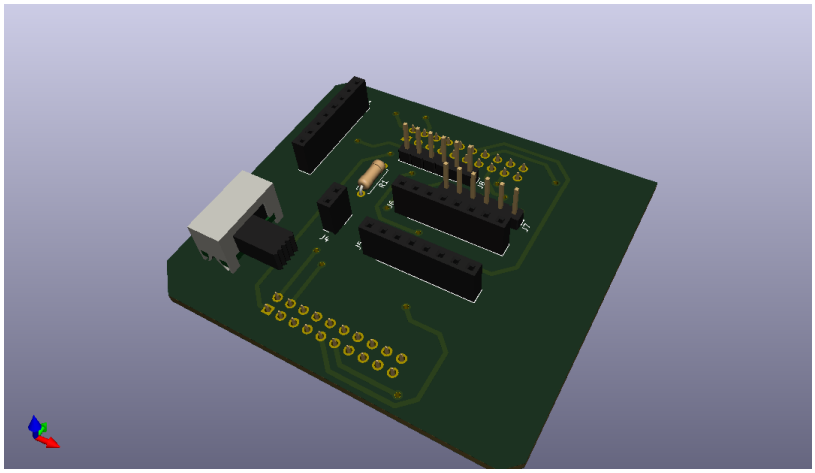


Figura 56: Render de placa real.

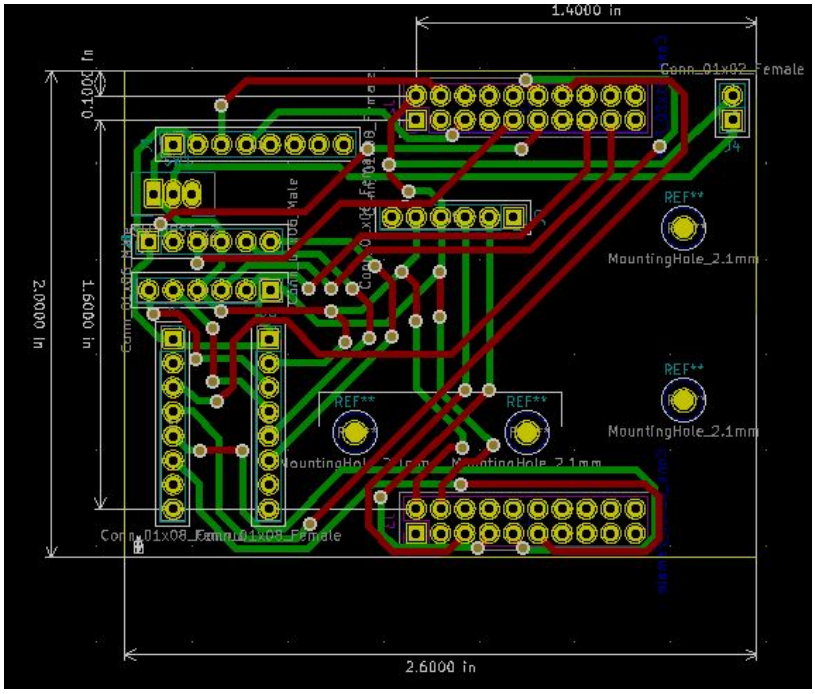


Figura 57: Diagrama del segundo diseño de PCB.

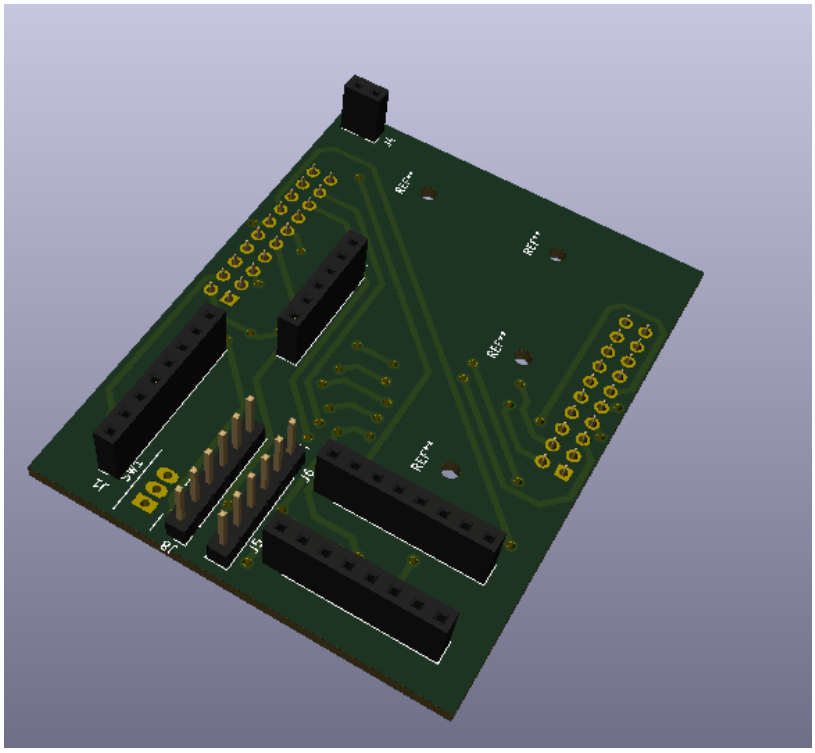


Figura 58: Render de placa real de segunda placa.

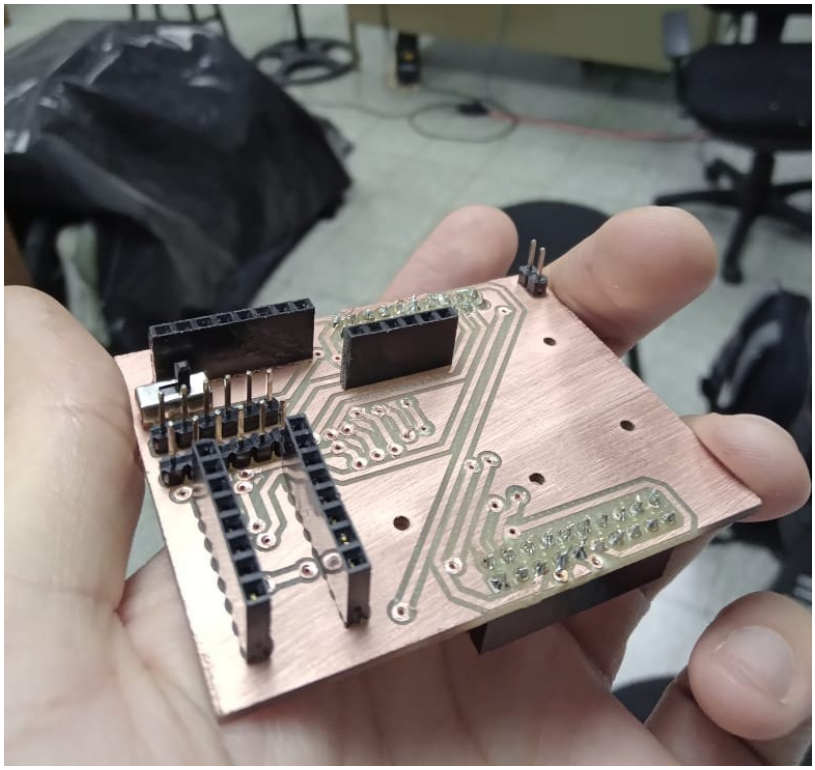


Figura 59: Segunda placa armada y soldada

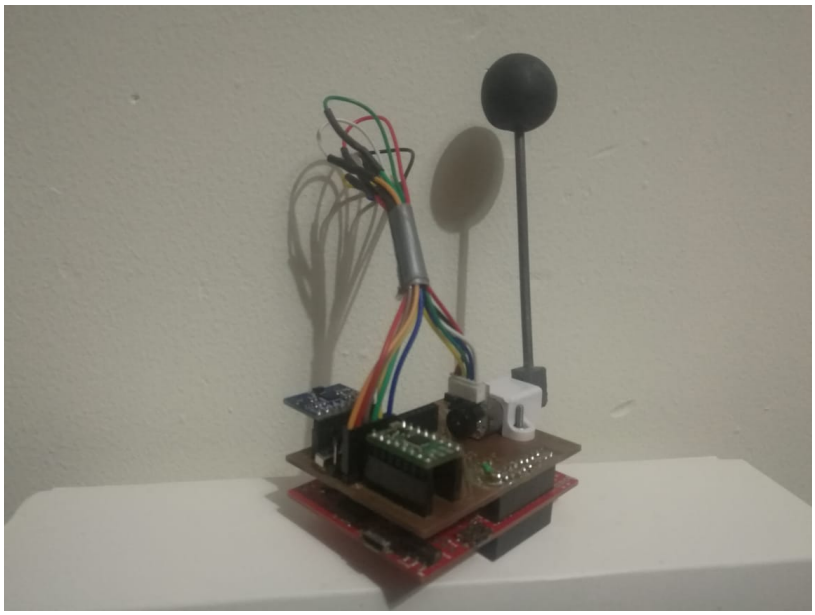


Figura 60: Prototipo armado con segunda placa

Figura 61: Prototipo final de mecanismo de salto

12.7. Planos de detalle

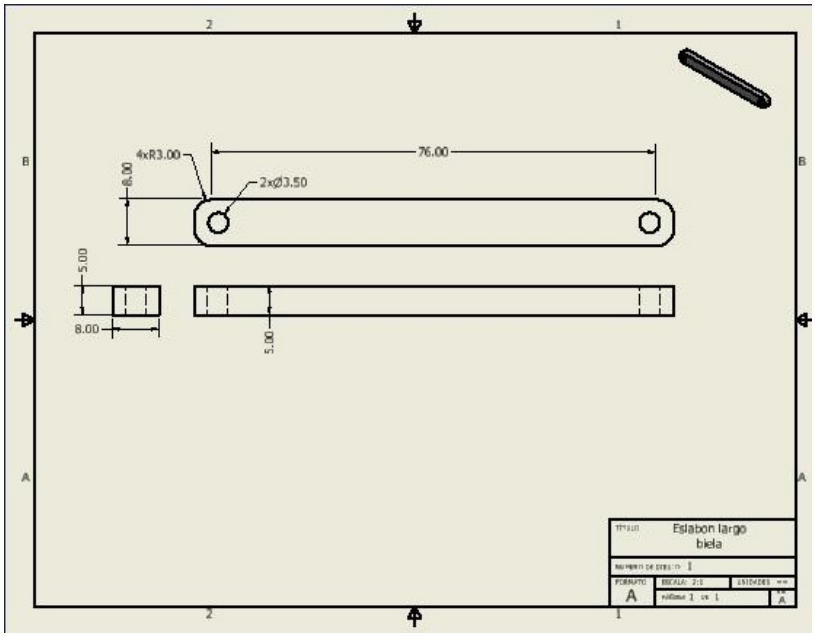


Figura 62: Plano de biela

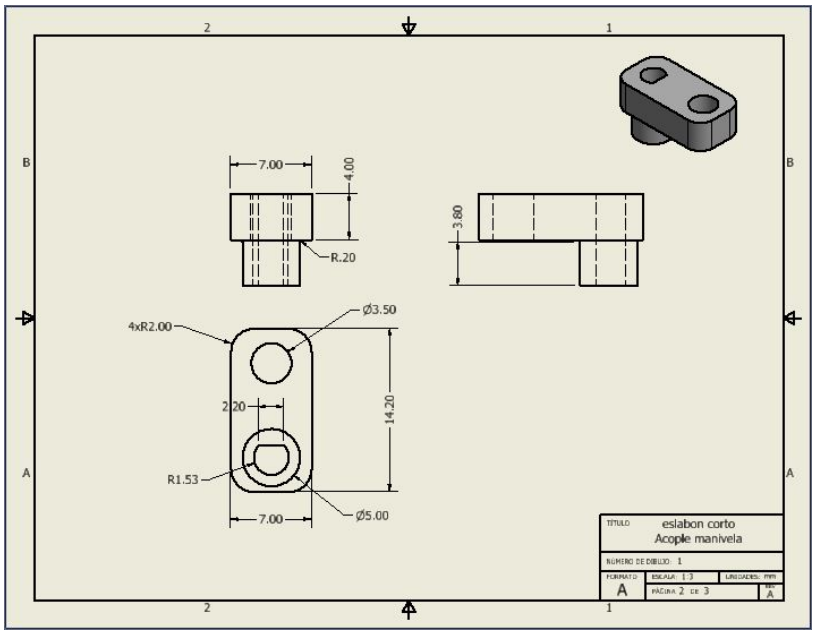


Figura 63: Plano de manivela

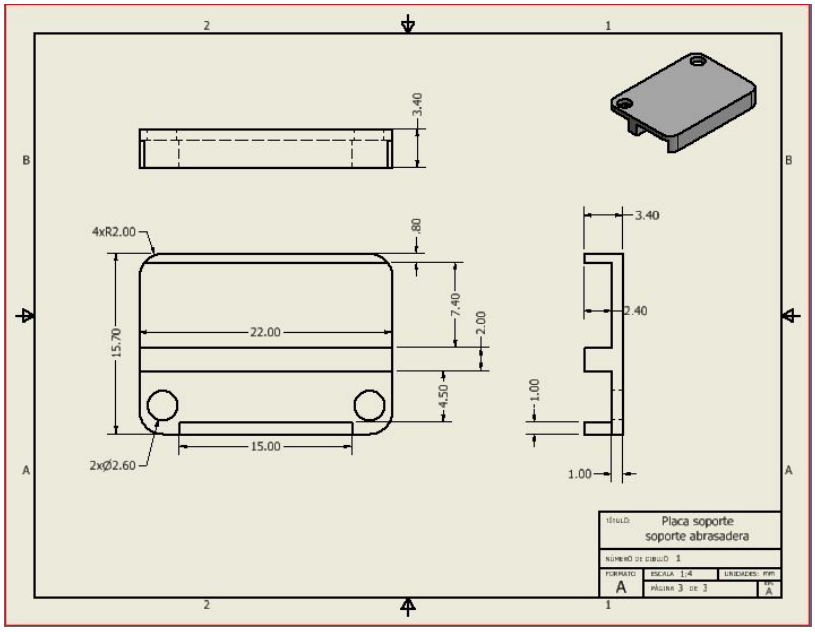


Figura 64: Plano de placa de abrazadera

CAPÍTULO 13

Glosario

CAM: Manufactura asistida por computadora. 37

IMU: Unidad de Medición Inercial. 6, 20

LiPo: Polímero de Litio. 43

PCB: Placa de circuito impreso. 42

PLA: ácido poliláctico. 37

PWM: Pulse Width Modulation. 16

Tiva C: EK TM4C123GXL, Tiva C Series Launchpad: Es una placa de desarrollo y evalución que contiene el microcontrolador de 16 bits TM4C123GH6PM.. 24

TPU: Poliuretanos termoplásticos. 35