Diseño e implementación de un robot cuadrúpedo controlado de forma remota mediante un módulo WiFi

Fernando Andrés Figueroa Hernández

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de un robot cuadrúpedo controlado de forma remota mediante un módulo WiFi

Trabajo de graduación presentado por Fernando Andrés Figueroa Hernández para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

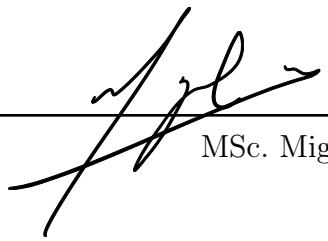
Diseño e implementación de un robot cuadrúpedo controlado de forma remota mediante un módulo WiFi

Trabajo de graduación presentado por Fernando Andrés Figueroa Hernández para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

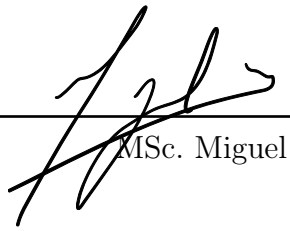
2023

Vo.Bo.:

(f) 

MSc. Miguel Zea

Tribunal Examinador:

(f) 

MSc. Miguel Zea

(f) 

Dr. Luis Alberto Rivera Estrada

(f) 

MAEB. Pablo Mazariegos

Fecha de aprobación: Guatemala, 3 de enero de 2023.

Prefacio

Quiero agradecer a mi familia, quienes estuvieron apoyándome durante la carrera y han estado pendientes de mi durante toda mi carrera, en especial de los avances en esta última etapa. Quiero darles las gracias a mis padres, Fernando y Lilian, quienes siempre me han ayudado a mejorar y no rendirme ante las adversidades que se fueron presentando. A mi hermano, por su incondicionalidad a pesar de la distancia, a mis abuelos por siempre estar pendientes del proceso y quiero agradecerle a mi novia Yoshira, quien siempre me apoyo en lo que pudo durante todo lo que tuve que realizar para llegar hasta acá.

De la misma manera quiero agradecer a mi asesor, Miguel Zea el cual siempre me exhortó a realizar de la mejor manera todas las pruebas y fases que abarcaron este trabajo de graduación.

Finalmente quiero agradecer a todos mis amigos tanto dentro como fuera de la carrera que fueron viendo cómo iba avanzando en la carrera universitaria, y les deseo lo mejor a los que tengan también su trabajo de graduación pendiente de aprobación.

Índice

Lista de figuras

Lista de cuadros

Resumen

En este trabajo de graduación se diseñó, manufacturó y programó una plataforma robótica, que debía de funcionar y poder ser controlada mediante el ecosistema robótico denominado como Robotat.

Primeramente se realizaron pruebas muy básicas con los motores que se iban a utilizar con el robot, esto para familiarizarse con el funcionamiento que estos tenían.

Posteriormente con las pruebas realizadas y las medidas de los motores, se procedió a comenzar con un bosquejo para el primer diseño, con esto también se empezó con el diseño en inventor. Al terminar todo el diseño se comenzó a manufacturar cada parte utilizando como material PLA en impresoras 3D. Para unir los motores con cada eslabón se utilizaron tornillos y pines especiales de la misma marca de los motores.

Con las piezas ya manufacturadas se ensambló todo el robot y se comenzaron a realizar las rutinas para el movimiento de la plataforma robótica utilizando la placa OpenCM9.04-A. Para esto se tenía que inicializar el protocolo pertinente para cada modelo. Como estos actuadores funcionan con un número de identificación individual se tuvo que enumerar cada uno y programar ese mismo número para poder darle una instrucción distinta a cada uno, todo fue programado en Arduino. Debido al tipo de conexiones que la placa tenia, se manufacturaron adaptadores para conectar los motores AX-12 hacia el OpenCM.

Finalmente con las rutinas listas se realizó la conexión de la plataforma robótica con el ecosistema robótico utilizando la placa ESP32, la cual recibe y procesa toda la información de todas las placas utilizadas así como del ecosistema Robotat. Por último con toda la información que procesa la ESP32 esta logra recibir y enviar las posiciones de los motores hasta la placa OpenCM9.04 lo que logra el movimiento remoto del robot.

Abstract

In this graduation work, a robotic platform was designed, manufactured and programmed, which had to work and be controlled by the robotic ecosystem called Robotat.

First, very basic tests were carried out with the motors that were going to be used with the robot, in order to become familiar with their operation.

Subsequently, with the tests performed and the measurements of the motors, we proceeded to start with a sketch for the first design, with this we also started with the design in inventor. Once the design was finished, we started to manufacture each part using PLA as material in 3D printers. To join the motors with each link we used special screws and pins of the same brand of the motors.

With the parts already manufactured, the whole robot was assembled and the routines for the movement of the robotic platform were started using the OpenCM9.04-A board. For this, the relevant protocol had to be initialized for each model. As these actuators work with an individual identification number, we had to enumerate each one and program that same number in order to give a different instruction to each one, everything was programmed in Arduino. Due to the type of connections that the board had, adapters were manufactured to connect the AX-12 motors to the OpenCM.

Finally, with the routines ready, the robotic platform was connected to the robotic ecosystem using the ESP32 board, which receives and processes all the information from all the boards used as well as from the Robotat ecosystem. Finally, with all the information processed by the ESP32, it manages to receive and send the positions of the motors to the OpenCM9.04 board, which achieves the remote movement of the robot.

CAPÍTULO 1

Introducción

En la Universidad del Valle de Guatemala se han realizado avances significativos en el campo de la robótica. Se ha trabajado en años pasados con el robot R17, como lo fue en el caso del trabajo de graduación *Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento*[1], en el que se desarrolló un software capaz de controlar el manipulador serial R17 dentro de un ecosistema que utiliza el Optitrack. Además el último año también se trabajó con el ecosistema robótico Robotat en conjunto con el sistema de cámaras Optitrack, como se puede ver en el trabajo de graduación *Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica*[2], en el cual se desarrolló el propio ecosistema Robotat para trabajos futuros. Este trabajo de graduación presenta un proyecto el cual toma aspectos de estas dos ideas mediante el desarrollo de una plataforma robótica la cual debe de funcionar dentro del ecosistema Robotat.

Con este proyecto se presenta todo el proceso de manufactura de una plataforma robótica, tanto de la parte mecánica como la parte electrónica que conlleva la fabricación de un robot, así como del desarrollo e implementación de las rutinas para que este pueda movilizarse y ser controlado de forma remota. Bajo esta idea, este trabajo de graduación se puede dividir en: la manufactura y diseño, conexiones eléctricas entre los motores y la placa utilizada y todo el aspecto informático que conlleva la conexión de la plataforma robótica al sistema de cámaras Optitrack así como las rutinas que logran movilizar el robot.

CAPÍTULO 2

Antecedentes

En el campo de la robótica podemos encontrar robots utilizados dentro de distintos ámbitos. Aprovechando la velocidad y precisión que una maquina puede tener sobre un humano, muchas industrias han logrado automatizar procesos para producciones en masa. Todo esto se logra gracias a las formas en las cuales los robots logran interactuar con el mundo exterior, ya que estos pueden utilizar sensores, actuadores o cámaras las cuales son las herramientas empleadas para que el robot pueda obtener los datos pertinentes y así poder ejecutar alguna acción específica. Se ha pasado de robots que apenas podían desplazarse o mantener el equilibrio, a robots industriales capaces de inspeccionar terrenos peligrosos. Un ejemplo de esto se puede encontrar dentro de la empresa automotriz Ford [3], la cual utilizó robots cuadrúpedos con sensores y cámaras integradas para poder escanear una de sus fábricas y así poder rediseñarla. Gracias a estos robots utilizados se logró escanear áreas las cuales eran de muy difícil acceso para una persona [3]. Incluso, se han llegado a utilizar robots en hospitales para comunicación entre el personal o pacientes para poder disminuir la exposición al COVID-19 [4].

En la actualidad el diseño de sistemas de locomoción para robots móviles es derivado del estudio de sistemas biológicos [6], como mamíferos comunes o algunos insectos. Estos sistemas de movimiento utilizan eslabones y juntas, las cuales pueden denominarse como

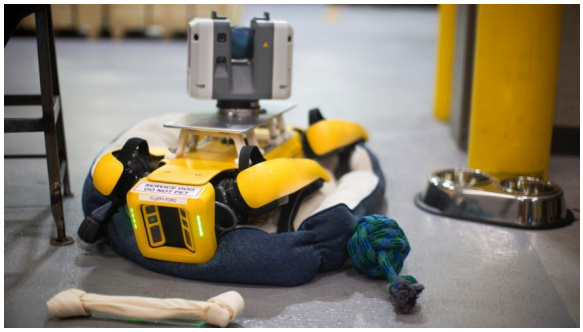


Figura 1: Robot cuadrúpedo utilizado en Ford [3].

Figura 2: Robot interactivo Petoi [5].

"patas". Las maquinas móviles de la actualidad se aprovechan de estos sistemas de locomoción biológicos mediante una cantidad distinta de puntos de apoyo, ya sea solo uno o más de uno (bípedos, hexápodos, bípedos, etc.) 2, por esto los robots actuales se han convertido en la mejor opción para recorrer terrenos peligrosos o de difícil acceso. Los robots que utilizan estos sistemas son más complejos que los que se movilizan utilizando ruedas, debido a que requieren de un nivel más alto de sensorización además los sistemas de control y mecánicos son un poco más complejos [7].

Motores Dynamixel en robots cuadrúpedos

Existen motores tipo servo de la marca dynamixel los cuales son los actuadores más avanzados a nivel de robótica personal. Estos cuentan con altas prestaciones para robots que son totalmente programables y proporcionan una alta cantidad de información de retroalimentación [9]. Estos servos pueden emplearse en cualquier tipo de robot, como ejemplo de esto se encuentra el robot cuadrúpedo llamado Enik 3. Este robot es un prototipo el cual podría ser útil si se quisiera diseñar un sistema de locomoción con 4 patas un poco más grande de lo normal. Por el tamaño que tiene Enik, se utilizan servos Dynamixel AX 12-A [8]. Este fue diseñado con el propósito de aportar a estudiantes e investigadores una forma de comprender de una forma más práctica, como se debe de implementar la codificación para un robot de este tipo. Este usa una fuente de energía de 12 voltios, ya que no se puede utilizar ninguna batería tipo LiPo debido al tipo de enchufe que tiene la placa OpenCM 485

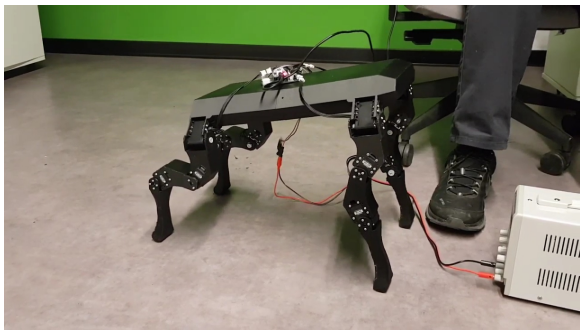


Figura 3: Robot cuadrúpedo Enik [8].

que se utiliza en Enik. Para la comunicación se emplea un cable USB conectado directamente a la computadora pero el microcontrolador tiene un conector de comunicación RX TX, por lo que cualquier modulo inalámbrico se puede usar para la comunicación, como el modulo Bluetooth HC-06 [8].

Ecosistema Robotat

En la Universidad del Valle de Guatemala se encuentra un ecosistema robótico, que utiliza el sistema de captura de movimiento Optitrack. El ecosistema fue denominado como Robotat y se desarrolló con el fin de tener una red de comunicación WiFi para distintos agentes y no sólo para un campo de investigación específico. Para poder implementar la red WiFi se utilizó el microcontrolador ESP32 y se creó una librería en el lenguaje de programación C, esto para poder conectarse con el sistema de captura de movimiento y así recibir los datos pertinentes que enviaba el Optitrack, también se utilizó Python, con este último se creó un programa el cual tomaba los valores de las poses de los agentes y los publicaba en un broker utilizando el protocolo MQTT[2].

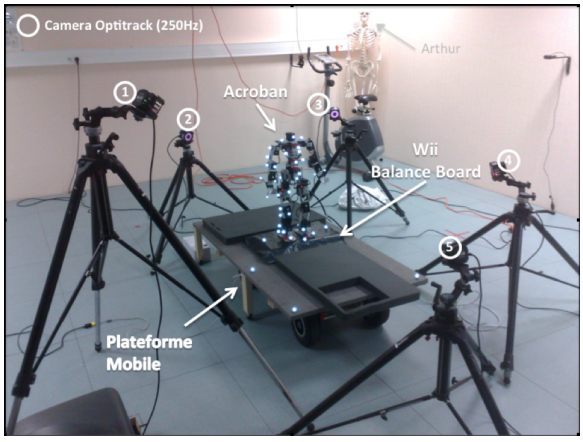


Figura 4: Ejemplo sistema Optitrack [10].

CAPÍTULO 3

Justificación

Actualmente el desarrollo de plataformas robóticas ha tomado una gran relevancia en distintos ámbitos. Estas proporcionan una gran eficacia al realizar las tareas que tienen asignadas, ayudando así al ser humano a cumplir con varios procesos a la vez, sin la necesidad de tener que sacrificar calidad o precisión. Además de esto, el diseño, la fabricación y programación de una plataforma robótica proporciona una alta gama de oportunidades para el público a quienes se encuentra dirigida, además de esto los científicos también se encuentran ante una experiencia enriquecedora debido a los retos que se presentan con un proyecto de este tipo.

A pesar que existen distintos tipos de robots, las investigaciones se ven centradas en su mayoría en bípedos y hexápodos. Los robots cuadrúpedos destacan sobre los bípedos en la capacidad de carga que pueden soportar así como el consumo energético, debido a que los actuadores consumen mucho menos energía para poder mantener estable el robot. En comparación contra los hexápodos, utilizar un robot cuadrúpedo puede reducir los costos debido a la cantidad de actuadores que estos normalmente utilizan en su sistema de locomoción.

Estamos en una etapa en la que la tecnología ha avanzado de tal manera que ya tenemos la posibilidad de poder utilizar robots autónomos en diversas tareas y aplicaciones, en su mayoría debido al tema de la industrialización, sin embargo debido a que el campo de la robótica ha avanzado tanto, ya no nos encontramos limitados a que una entidad robótica pueda cumplir únicamente con una sola tarea a la ves creando así entornos más dinámicos. El uso de este tipo de plataformas en Guatemala aún es muy limitado. No existe ninguna empresa o quizás muy pocas las cuales las utilicen para poder facilitar o agilizar sus procesos, cuando en otras partes del mundo este ámbito ya ha progresado de manera significativa. Es por esto que es de gran importancia comenzar a formar parte de este avance tecnológico no solo comprendiendo de una mejor forma cómo funcionan estas plataformas robóticas sino que además adentrarnos en lo que sería el diseño, fabricación e implementación de una plataforma de este tipo.

Por otra parte para lograr el desarrollo de una plataforma robótica son necesarias varias herramientas las cuales serán claves para el éxito de la implementación de una entidad

robótica. Es por esto que el ecosistema robótico que se encuentra dentro de la Universidad del Valle de Guatemala es una parte fundamental para el desarrollo del ámbito de la robótica en Guatemala. Este ecosistema nos permite no solo realizar una interacción en tiempo real entre varias plataformas robóticas a la vez sino que también tener una interacción humano-maquina ayudándonos así a comprender mejor el funcionamiento de estas.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar e implementar de un robot cuadrúpedo que funcione dentro del ecosistema Robotat de la Universidad del Valle de Guatemala

4.2. Objetivos específicos

1. Diseño y ensamblaje de todos los componentes de un robot cuadrúpedo.
2. Implementación de los servos Dynamixel para poder controlar los movimientos del de un robot cuadrúpedo.
3. Selección y dimensionamiento de la batería que se utilizara para el robot.
4. Utilizar el microcontrolador OpenCM9.04-C para poder implementar la parte autónoma de un robot cuadrúpedo.
5. Utilizar un módulo WiFi SP32 para poder controlar el robot de forma remota.
6. Utilizar el sistema de cámaras Optitrack que se encuentra dentro de la Universidad del Valle de Guatemala para poder monitorear el movimiento del robot.

CAPÍTULO 5

Alcance

El enfoque principal de este trabajo de graduación es el desarrollo de plataformas robóticas que utilicen una red de comunicaciones inalámbricas junto con un sistema de captura de movimiento, sin embargo las rutinas desarrolladas no tienen un nivel de complejidad muy elevado ya que el fin es poder implementar rutinas más complejas en trabajos posteriores lo cual podría abrir la posibilidad de montar sensores o cámaras en los robots para poder escanear áreas de difícil acceso o incluso superar obstáculos, como ya lo logran hacer otras plataformas robóticas de este tipo. Las rutinas creadas tienen únicamente la funcionalidad de realizar el movimiento del robot, a diferencia de otros robots de este tipo los cuales si tienen la capacidad de subir gradas, saltar, entre otros.

Las rutinas de movimiento fueron desarrolladas en la placa OpenCM9.04 la cual utiliza el lenguaje de programación Arduino. Para la conexión WiFi se utilizó el módulo ESP32 el cual también utiliza el lenguaje de Arduino. Gracias a este último fue posible realizar la conexión con el sistema de captura de movimiento, con el cual se podrá controlar los movimientos del robot de forma remota.

Fueron empleados 12 servomotores de la marca Dynamixel, de los cuales 8 fueron el modelo XL-320 mientras que los otros 4 fueron de modelo AX-12. A pesar que existen servomotores mucho más complejos que podrían brindar una mejor estabilidad y movilidad, se utilizaron los dos modelos ya mencionados debido a que estos eran los que tenía el departamento de Electrónica por lo que el uso de estos modelos se convirtió en parte de los requerimientos que tenía este proyecto de graduación.

Cabe mencionar que los servos que se usaron para la plataforma robótica por ser de distintos modelos tenían conectores distintos, mientras que el modelo que se tenía de la placa OpenCM solo tenía entrada para uno de los modelos de los motores siendo estos los XL-320. Esto afectó en el desarrollo y en las pruebas ya que se tuvieron que fabricar cables adaptadores, para poder conectar los motores AX-12 hacia el OpenCM. El tiempo de fabricación de estos adaptadores fue lo que retrasó un poco las pruebas de la plataforma robótica.

CAPÍTULO 6

Marco teórico

6.1. Servomotores Dynamixel

Actualmente existe la marca Dynamixel, quienes producen los actuadores más avanzados a nivel de robótica personal. Estos cuentan con altas prestaciones para robots que son totalmente programables y proporcionan una alta cantidad de información de retroalimentación. Estos servos cuentan con la topología daisy-chain, la cual permite conectar varios motores al mismo tiempo. Estos pueden proporcionar una gran cantidad de información de lo que le está ocurriendo al motor y como se está desempeñando como por ejemplo:

- Posición actual del motor[9].
- Velocidad [9].
- Temperatura interna [9].
- Torque[9].
- Tensión de alimentación [9].

Cuando alguna de esta información sobrepasa los márgenes óptimos de funcionamiento también existe una función de alarma la cual desconecta el actuador además de encender un LED que actúa como aviso, todo esto para no dañar el mecanismo del servo [4]. Gracias a la alta carga que pueden soportar estos servos se puede lograr que un robot se desplace de forma horizontal pero además pueda desplazarse de forma vertical, siempre y cuando tenga forma de poder sostenerse en la superficie que este quiere escalar.

Estos servomotores utilizan el protocolo de comunicación Half-Dúplex. Un dispositivo que posee este protocolo de comunicación solo pueden transmitir en una sola dirección a la vez. Los datos sí pueden desplazarse en dos direcciones diferentes pero no se podrá

realizar al mismo tiempo, la conexión si es bidireccional pero no simultánea. Las redes Half-Dúplex requieren un mecanismo que evite la colisión de datos. Se debe verificar si existe algo que aún se encuentre transmitiendo antes de enviar cualquier otro tipo de dato evitando así una colisión. Este tipo de comunicación se utiliza normalmente cuando se puede tener riesgo de colisión, esto generalmente puede ocurrir cuando más de un dispositivo o usuario intenta conectarse al mismo tiempo [11]. El controlador principal que controla los actuadores Dynamixel establece la dirección de comunicación que deberá tener el modo de entrada y cuando se desea transmitir un paquete de instrucciones este se convertirá la dirección a modo de salida. En UART Half-Dúplex, el tiempo en el que finaliza la transmisión de datos es importante para poder cambiar al modo de recepción. Existen 3 definiciones de bits que indican el status de la UART, estos serían los siguientes:

- TXD BUFFER READY BIT: indica que los datos de transmisión ya se pueden cargar en el buffer[11].
- TXD SHIFT REGISTER EMPTY BIT: se establece cuando ya terminó el envió de datos y han salido del CPU[11].
- TXD BUFFER READY BIT: se utiliza cuando se va a transmitir un BYTE a través del canal de comunicación en serie[11].

En la Figura 5 se puede observar los distintos tipos de instrucción y el campo que define cada uno.

6.2. Cinemática directa y cinemática inversa

La cinemática de un robot es el estudio de los movimientos que este puede realizar. En un análisis de cinemática, la velocidad, aceleración y posición son calculadas sin tomar en cuenta las fuerzas que causan los movimientos. Si se menciona el estudio de la cinemática de manipuladores, se hace énfasis en todas las propiedades geométricas y basadas en el tiempo del movimiento. Los dos problemas dentro del estudio de la manipulación mecánica reciben el nombre de cinemática directa y cinemática inversa[12].

El problema de la cinemática directa se plantea en términos de encontrar una matriz de transformación la cual es utilizada para relacionar un sistema de coordenadas que se encuentra ligado al cuerpo en movimiento, respecto a un segundo sistema de coordenadas el cual se usa de referencia. Para poder llegar a esta representación de forma exitosa se utilizan matrices de 4×4 dentro de las cuales se incluyen operaciones de orientación y traslación.

Valor	Instrucciones	Descripción
0x01	Silbido	Instrucción que verifica si el Paquete ha llegado a un dispositivo con la misma ID que la ID del Paquete
0x02	Leer	Instrucciones para leer datos del dispositivo
0x03	Escribe	Instrucciones para escribir datos en el dispositivo
0x04	Registro de escritura	Instrucción que registra el Paquete de Instrucción en un estado de espera; El paquete se ejecuta más tarde a través de la instrucción de acción
0x05	Acción	Instrucción que ejecuta el Paquete que se registró previamente usando Reg Write
0x06	Restablecimiento de fábrica	Instrucción que restablece la mesa de control a su configuración inicial predeterminada de fábrica
0x08	Reiniciar	Instrucción que reinicia DYNAMIXEL (Ver productos soportados en la descripción)
0x83	Escritura sincronizada	Para múltiples dispositivos, instrucción para escribir datos en la misma dirección con la misma longitud a la vez
0x92	Lectura masiva	Para múltiples dispositivos, instrucciones para escribir datos en diferentes direcciones con diferentes longitudes a la vez (ver productos compatibles en la descripción)

Figura 5: Definición de los distintos tipos de instrucciones para los motores Dynamixel [11]

$$T = \begin{bmatrix} rotacin & posicin \\ f_{1x3} & escalado \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Figura 6: Representación de matriz homogénea [12].

La matriz de transformación homogénea transforma un vector el cual se encuentra expresado en coordenadas homogéneas en un sistema, hacia coordenadas expresadas en un sistema de coordenadas distinto. En la Figura 6 se puede observar la estructura que tiene la matriz homogénea, donde los vector $\vec{a}$, $\vec{s}$, $\vec{n}$ son vectores ortogonales unitarios y $\vec{p}$ sera un vector que describe la posición $\vec{x}$, $\vec{y}$, $\vec{z}$ del sistema actual respecto del sistema que se tiene como referencia [12].

El problema de la cinemática inversa consiste en hallar las coordenadas articulares de todo el robot, esto teniendo conocimiento previo de la posición y orientación que tendrá el efector final del robot. La solución más habitual suele ser mediante el uso de métodos geométricos porque lo más común es que un robot tenga cadenas cinemáticas sencillas. Cada vez que se especifica una posición y orientación de destino, se debe calcular la cinemática inversa, con esto se lograr despejar los ángulos requeridos de las articulaciones. Este tipo de análisis permite por ejemplo, realizar los cálculos para los parámetros de la unión de un brazo para que este logre levantar un objeto. A pesar de todo esto la cinemática inversa cuenta con algunos problemas como:

- Obtener los valores de todas las variables articulares para poder llegar a una posición determinada puede convertirse en una tarea complicada [12].
- Se deben resolver ecuaciones no lineales de forma simultánea [12].
- Pueden existir múltiples soluciones, ninguna solución o incluso singularidades [12].

6.3. Representación Denavit-Hartenberg

Este tipo de notación es un método que se utiliza para describir la relación cinemática entre un par de eslabones adyacentes que forman parte de una cadena cinemática abierta. Este método representa la posición y orientación de un cuerpo rígido a través de una matriz 4×4, en el cual se utiliza un número mínimo de parámetros que describen completamente la relación cinemática [13].

Para este método se le debe asignar a cada articulación un sistema de referencia local con un origen en un punto Q_i y ejes ortonormales $\{X_i, Y_i, Z_i\}$ comenzando con un primer eje fijo e inmóvil dado por los ejes $\{X_0, Y_0, Z_0\}$ anclado a un punto fijo Q_0 de la base [13].

La manera más sistemática para lograr este proceso requiere colocar un marco de referencia en cada uno de las juntas del manipulador, desde la base hasta el efector final.

Seguido de esto la idea es tener una secuencia de transformaciones que nos permitan trazar, por decirlo así, un camino hasta el efector final.

La idea detrás de esta representación es derivar la cinemática directa del manipulador mediante rotaciones y desplazamientos entre los eslabones adyacentes. La representación de Denavit-Hartenberg cuenta con las siguientes restricciones:

- Cada junta conecta únicamente dos eslabones.
- Los eslabones son rígidos.
- Las únicas juntas aceptadas son prismáticas o revolutas.
- El manipulador tiene N juntas y N+1 eslabones.

Entonces un manipulador con N juntas y N+1 eslabones puede ser descrito completamente utilizándolos parámetros de Denavit-Hartenberg o parámetros DH. Las transformaciones que se realizan entre juntas están dadas por las transformaciones elementales de rotación y traslación. Los parámetros de las rotaciones y traslaciones elementales son los siguientes:

- α_j : ángulo de rotación de z_{j-1} hasta z_j con respecto al eje x.
- θ_j : ángulo de rotación de x_{j-1} hasta x_j con respecto al eje z.
- a_j : distancia sobre el eje x que hace coincidir los orígenes de los marcos de referencia de los eslabones z_{j-1} y j.
- d_j : distancia sobre z hasta que x_{j-1} y x_j sean colineales.

De los parámetros anteriormente mencionados a_j y α_j serán constantes, θ_j y d_j podrían ser variables esto en dependencia del tipo de junta que se está analizando. Cuando se tienen juntas revolutas θ_j será variable y d_j será constante, mientras que en una junta prismática θ_j es constante y d_j es variable. Entre estos parámetros los variables serán correspondientes con la configuración del robot los cuales son representados mediante un vector q.

La siguiente fórmula 2 representa un ejemplo para los argumentos de rotaciones y traslaciones elementales. Esta transformación está representada por ${}^{j-1}T_j$ la cual representa el movimiento que se realiza entre juntas y está dada por las transformaciones elementales de rotación y traslación.

$$ {}^{j-1}T_j = Rot_z(\theta_j)Transl_z(d_j)Transl_x(a_j)Rot_x(\alpha_j) \tag{2} $$

6.4. Sistema de captura de movimiento OptiTrack

Los sistemas OptiTrack ofrecen la mejor combinación de precisión de medición y flujo de trabajos simples lo que brinda a ingenieros y diseñadores los datos de seguimiento 3D

necesarios para sus estudios. Este sistema es de los más precisos que existen en la actualidad. Este normalmente suele producir un error de medición menor a 0.2mm en áreas de seguimiento de hasta 10000 pies cuadrados o más. Cuando se tienen áreas más pequeñas el error de medición puede bajar hasta un 0.1mm más o menos. Gracias a que los sistemas OptiTrack tienen un sistema de auto calibración, estos pueden funcionar durante todo un día sin perder precisión en las mediciones. Dentro de la robótica se utilizan estos sistemas para analizar los movimientos que tiene un robot, así como la posición del mismo en el espacio. Esto de tal forma que se pueda realizar cualquier cambio cuando sea necesario [10]. Las cámaras Primex 41 utilizadas en el sistema de captura de movimiento que se encuentra dentro de La Universidad del Valle de Guatemala cuentan con las siguientes especificaciones[2]:

- Resolución: 2048x2048
- Velocidad de fotogramas: 180 Hz.
- Latencia: 5.5 ms.
- Precisión 3D: +/- 0.10 mm.
- Rango para marcadores pasivos: 30 m.
- Rango para marcadores activos: 45 m.
- 20 LEDs.
- Luces del tipo infrarrojo de 850 nm.
- Puerto de datos GigE.
- Sincronización de la cámara por medio de Ethernet.
- Alimentación por medio de PoE o PoE+.
- Tamaño: 12.6cm x 12.6 cm x 13.2 cm.
- Peso: 1.36 Kg.

Estos sistemas de captura de movimiento permiten capturar movimientos modelándolos en un ecosistema virtual, esto permite aplicar los datos dentro del campo de la robótica. Existen técnicas diferentes que se pueden emplear en los sistemas de captura de movimiento, estas pueden ser:

- Técnicas sin marcadores.
- Técnicas inerciales.
- Técnicas ópticas pasivas.
- Técnicas ópticas activas.

En las técnicas ópticas pasivas se utilizan marcadores retroreflectivos y son posicionados en el objeto el cual se desea capturar. Las cámaras del sistema generan una luz infrarroja la cual es reflejada por los marcadores, cuando esto es detectado se calcula la posición en un espacio tridimensional. Las técnicas activas funcionan de la misma manera con la única diferencia en que la luz no es reflejada, sino que emitida por los marcadores. Esto causa que los marcadores utilizados en este último procedimiento sí necesiten una fuente de poder para poder funcionar. Luego se tienen las técnicas sin marcadores. Esta ya no utiliza sensores, sino que se basan en algoritmos los cuales son capaces de seguir y grabar los objetos a analizar utilizando cámaras con sensibilidad a la profundidad. El inconveniente con este tipo de técnicas es que por la falta de sensores se puede perder precisión en las mediciones. Por último se encuentran las técnicas inerciales, solo utilizan dispositivos como acelerómetros, giroscopios, entre otros. Estos dispositivos que permiten determinar la posición del cuerpo que se desea medir. Esta técnica ya no utiliza ningún tipo de cámaras a diferencias de las tres anteriores [2].

6.5. Robots cuadrúpedos

La gran mayoría de robots cuadrúpedos están destinados a la exploración de terrenos peligrosos o irregulares los cuales serían de difícil acceso para un ser humano. Estas máquinas tienen muchos propósitos distintos como los que se mencionaron en la sección de antecedentes. Cabe mencionar que además de este tipo de tareas, debido a las características que poseen este tipo de robots, estos pueden convertirse en mecanismos utilizados no solo en superficies horizontales sino que también en superficies verticales, toman de nombre robots escaladores [14].

Al tener un mayor número de patas se obtiene la ventaja de alcanzar mayores velocidades así como una mejor estabilidad, a pesar de esto también se incrementa la complejidad mecánica y de control. Un robot cuadrúpedo disminuye esta complejidad si se compara contra robots que poseen una mayor cantidad de patas, sin embargo logran mantener las características de una maquina caminante, las cuales serían:

- Capacidad de adaptación al terreno.
- Control de fuerza en cada apunto de apoyo.
- Posee una gran capacidad omnidireccional.

La estructura básica que poseen estos robots normalmente cuenta con cuatro patas las cuales tendrán 3 articulaciones cada una, teniendo en total 12 grados de libertad. Para la parte del chasis normalmente se opta por lograr una figura cuadrada o rectangular dando espacio suficiente para todos los componentes electrónicos que fuera a tener el robot. Las patas del robot son ubicadas en cada esquina de forma simétrica para poder ubicar el centro de gravedad en el eje central y así poder distribuir los esfuerzos realizados en cada pata, de forma simétrica[14].

Para realizar un control de este tipo de robots de forma cinemática, se pueden encontrar un vector de valores articulares que posicionan y orientan los extremos de los efectores del robot en un lugar deseado o punto objetivo, logrando así controlar la postura del mismo. Si se

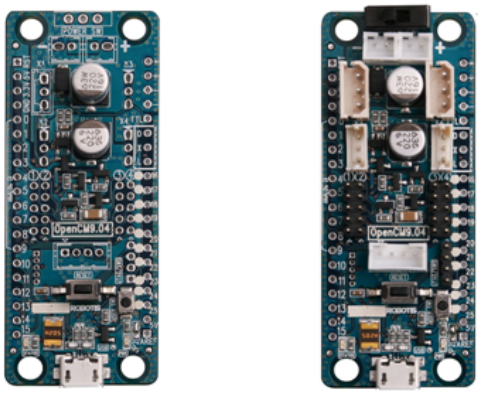


Figura 7: Placa OpenCM y sus distintos modelos[15]

habla, más específicamente, de la cinemática inversa, se puede utilizar un método geométrico y analítico para determinar los valores articulares para una posición deseada, todo esto a partir de funciones trigonométricas. De esta manera puede realizarse la orientación del robot cuadrúpedo así como el movimiento del mismo.

6.6. OpenCM 9.04

La placa OpenCM 9.04 se utiliza para servomotores Dynamixel, además de poseer hardware y software de código abierto basado en lenguajes ROBOTIS OpenCM, tiene un IDE similar a Arduino que permite programar en C o C++. Esta es una placa integrada la cual se basa en la CPU ARM Cortex-M3 de 32 bits. Viene en 2 versiones distintas, estas serían OpenCM9.04-A y OpenCM9.04-B[15].

OpenCM9.04-A es simplemente la placa impresa con componentes SMD (sin interruptores o encabezados pre soldados) mientras que la OpenCM9.04-B viene con todos los conectores de 3 pines pre soldados así como interruptores listos para usar. La versión B de la placa también le da la opción al usuario de poder soldar los encabezados para poder acceder a los pines GPIO, como puede observarse en la Figura 7.

6.7. Formato de texto tipo JSON

El JavaScript Object Notation (JSON) es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript. Normalmente este es utilizado para la transmisión de datos que, en la mayoría de casos pueden ser aplicaciones web, como por ejemplo envió de datos entre un servidor y un cliente, de esta manera los datos podrán ser mostrados en una página web o viceversa).

Los JSON en realidad son cadenas. Este debe de ser convertido a un objeto nativo de JavaScript cuando se desea acceder a sus datos. Esto se logra de manera sencilla ya

Figura 8: Formato de un JSON[16]

que JavaScript posee un objeto global JSON que tiene los métodos disponibles para lograr transformarlo. Un objeto de tipo JSON pueden ser almacenados en su propio archivo, el cual sería básicamente un archivo con una extensión tipo .json. El formato de un JSON es casi idéntico al de los objetos de JavaScript. Es posible incluir tipos de datos básicos en un JSON de la misma manera que en un objeto estándar de JavaScript.

Los JSON están constituidos por dos estructuras:

1. Una colección de pares de nombre/valor. En varios lenguajes esto puede ser conocido un objeto, registro, estructura o un arreglo asociativo.
2. Una lista de valores, esto puede ser implementado en varios lenguajes como arreglos, vectores, listas o secuencias.

Estos son estructuras universales por lo que virtualmente todos los lenguajes de programación los soportan de una manera o de otra. Es razonable y muy común que un formato de intercambio de datos que sea independiente del lenguaje de programación que se utiliza sea basado en este tipo de estructuras.

6.8. ESP32

Esp32 es la denominación de una familia de chips SoC de bajo costo y consumo de energía. Estos poseen tecnología WiFi además de Bluetooth. El ESP32 utiliza un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo, este también posee interruptores de antena, amplificador de potencia, amplificador receptor de bajo ruido, filtros y módulos de administración de energía. Este fue creado por Espressif Systems y actúa como el sucesor del ESP8266. El ESP32 posee una mayor capacidad en cuanto al proceso de datos lo que ayuda a que se reduzca la carga sobre el OpenCM. Este tipo de chip en su versión de dos núcleos se podría realizar un intercambio de información a la nube y además en simultáneo se podrían administrar datos de un sensor de forma precisa sin tener mucha dificultad.

Figura 9: Chip ESP32[17]

6.9. Multihilo

La técnica conocida como multihilo permite a las unidades centrales de procesamiento (CPU) aumentar su rendimiento sin necesidad de aumentar la frecuencia. Utilizando esta técnica se pueden ejecutar varias tareas de forma simultánea. Hablando de manera más precisa, se puede decir que se ejecutan varios hilos al mismo tiempo los cuales están conectados a un proceso diferente cada uno.

Normalmente un proceso se ejecuta de manera secuencial, es decir, un proceso después de otro. Esto causa que el rendimiento disminuya lo que lo convierte en algo poco óptimo porque las tareas bloquean el hardware debido a que si se desea ejecutar alguna otra tarea se debe esperar a que la anterior termine. Usando el multihilo reduce este problema porque se pueden procesar varios hilos de forma simultánea. Para que el multihilo funcione el software debe de estar preparado para ello, ya que si este no se encuentra optimizado al intentar utilizarse puede tener un efecto perjudicial.

El objetivo del multihilo es aumentar la velocidad de cálculo y por lo tanto el rendimiento. El sistema en lugar de detenerse mucho tiempo en un proceso, este pasara de forma rápida a la siguiente tarea, lo que ayuda a reducir esos tiempos de espera. Además esto también ayuda a que el sistema reaccione de manera más rápida a los cambios de prioridades. Cuando se necesita un cambio de tarea de forma repentina con la ayuda de la prioridad y los hilos cortos el procesador puede cambiar y dedicarse rápidamente a otra tarea.

6.10. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para varios sistemas operativos. Este incluye depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código y refactorización de código además este también se puede

Figura 10: PlatformIO[18]

personalizar, lo que permite cambiar el tema del editor[18].

Este es combatible con muchos lenguajes de programación y muchas de las características de este editor no se encuentran directamente en la interfaz de usuario sino que se debe acceder a ellas usando línea de comando o utilizando archivos con una extensión de tipo .json.

Utilizando Visual Studio podemos acceder a PlatformIO se puede observar en la Figura 10 . Esta se instala como extensión dentro de Visual Studio Code. PlatformIO es un IDE abierta de programación en C y C++. El objetivo de PlatformIO es facilitar el desarrollo de varios sistemas integrados. Normalmente cada plataforma incluye su propio IDE estas se incluyen en PlatformIO [18].

CAPÍTULO 7

Diseño y manufactura del robot

La parte fundamental de este trabajo fue el diseño, la manufactura y la creación de la plataforma robótica. Sin este proceso de diseño todas las demás fases del no tendrían sentido alguno. Por lo que el objetivo de este capítulo es resaltar el proceso que se llevó a cabo para la fabricación, manufactura y programación del robot.

7.1. Metodología

En esta sección se describen los procesos que se llevaron a cabo para el funcionamiento de la plataforma robótica (aspecto mecánico, electrónico y de la manufactura). Cabe resaltar que el tipo de robot que se iba a construir ya había sido seleccionado antes de comenzar con este trabajo por lo que el diseño se basó en algunos de los robots ya mencionados en los antecedentes.

7.1.1. Pruebas iniciales con los servomotores Dynamixel

Fue necesario realizar ciertas pruebas básicas con los servomotores Dynamixel además tomar medidas de los mismos. Esta primera parte fue necesaria para además de saber cómo funcionaban estos motores poder realizar un primer bosquejo del diseño del robot utilizando las medidas tomadas. Debido a que la placa que controla los motores trabaja con Arduino, para realizar las primeras pruebas se utilizaron dos librerías [19] [20] las cuales ya contenían las instrucciones para poder mover los motores. Estas primeras pruebas consistieron no sólo en lograr mover los servomotores, sino que también en poder utilizar el protocolo de comunicación daisy-chain el cual es una sucesión de enlaces tal que un dispositivo A se conecta a un dispositivo B, ese mismo B se conecta a un dispositivo C y así sucesivamente. Con

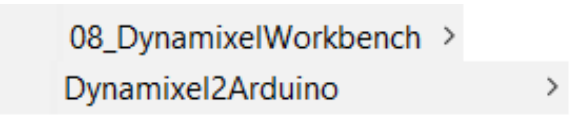


Figura 11: Librerías utilizadas [20].

este protocolo se logró conectar varios motores al mismo tiempo pero cada uno realizando acciones distintas, por ejemplo, moverse a distintas posiciones y/o velocidades.

7.1.2. Diseño realizado en Autodesk Inventor

Para el proceso de diseño se tomó como base el libro de Robert L. Norton de diseño de maquinaria [21], en este se detallan ciertos pasos los cuales se deben seguir para cualquier diseño mecánico. Cabe resaltar que varios de estos pasos ya habían sido cubiertos con la redacción inicial de este proyecto como lo sería la investigación preliminar, planteamiento de objetivos, entre otros. Se prosiguió directamente con las iteraciones iniciales del robot.

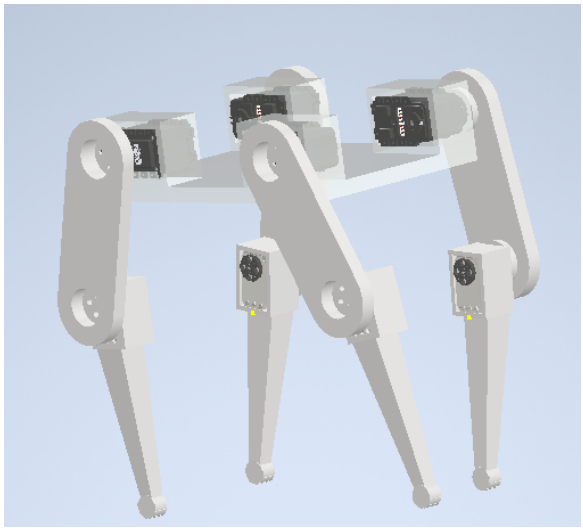


Figura 12: Primera iteración del robot.

Como referencia principal para el diseño se tomó el robot Petoi [5], el cual es un pequeño robot cuadrúpedo interactivo [5]. Con el primer bosquejo se realizó también un diseño en Autodesk Inventor, aunque el problema que se dio con esta primera iteración, la cual puede verse en la Figura 12, fue que son 12 servos en total los que se debían usar pero hacía falta un eslabón para poder utilizarlos todos a la vez, esto causó que esta primera iteración fuera descartada.

Como segunda iteración, la cual puede verse en la Figura 13, se conservó como base el mismo diseño de la primera iteración para el cuerpo pero se agregó un eslabón extra para las patas del robot junto con un cambio en la pieza que se encuentra adjunta a las mismas, esto para realizar la conexión entre estos dos eslabones. Este cambio se agregó para cumplir con la cantidad de servos requerida, pero además para lograr una mejor estabilización cuando este se movilizara. Junto con el bosquejo de la segunda iteración también se modificó el

diseño en Inventor.

Cuando se tenía completa la segunda iteración se comenzaron las rutinas para que el robot lograra desplazarse, pero a pesar que este si se lograba desplazar se tuvieron que realizar algunos cambios en los dos últimos eslabones para que este lograra aún una mejor estabilidad por lo que con estos cambios, el nuevo diseño se podría tomar como la tercera iteración la cual puede apreciarse en la Figura 14.

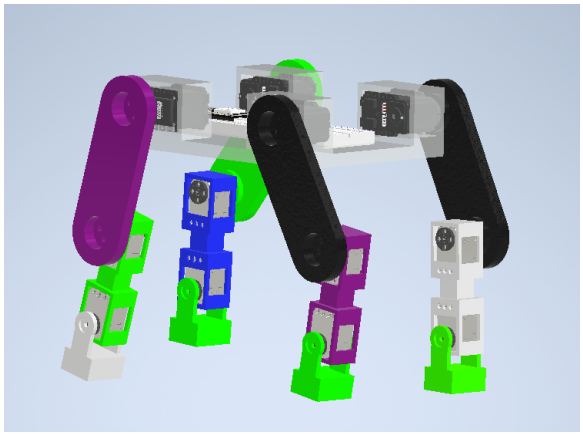


Figura 13: Segunda iteración del robot.

Para lograr el montaje de los servomotores con las piezas, se utilizaron pines especiales los cuales eran propios de los motores Dynamixel, pero estos podían ser utilizados únicamente para los motores con el modelo XL-320, como se muestra en la Figura 15. Para lograr montar los motores AX-12, Figura 16, se utilizaron los mismos tornillos que estos motores traían. En los eslabones fueron diseñadas cajas con las medidas exactas de los motores utilizados (de los dos modelos). Estas cajas fueron utilizadas para poder montar los servomotores y que no existiera ningún tipo de movimiento de los mismos cuando se encontraran en funcionamiento.

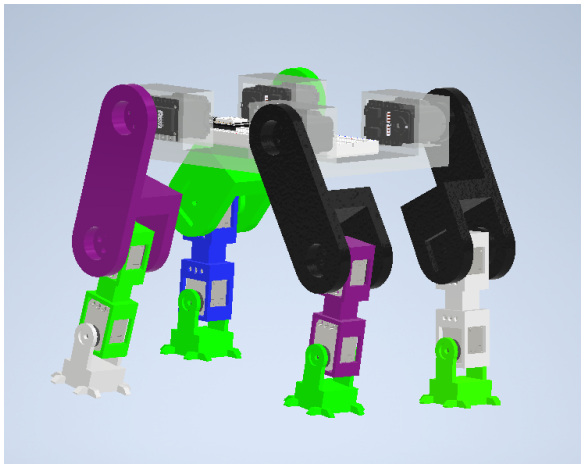


Figura 14: Tercera iteración del robot.

7.1.3. Programación de los motores

Para poder lograr la programación de los motores, se utilizó la placa OpenCM [15]. Al principio se dieron algunos problemas debido a que los motores no respondían cuando estos eran conectados en conjunto. Estos servos funcionan con un número de identificación (ID) individual lo que les permite saber que acción debe realizar cada uno de forma individual. Se utilizó un ejemplo en una de las librerías (DynamixelWrokbench) [20] para poder escanear el número de identificación de los servos y así saber cuál numero tenia cada uno. Se descubrió que todos los servos estaban utilizando como número identificación el número 1, lo que causaba conflictos con las rutinas que usaban un número distinto. Con ayuda de otro de los ejemplos de la misma librería mencionada anteriormente se cambió el ID de cada uno de los motores y se enumeró cada servo de 1 a 12, siendo los primeros 8 los XL-320 mientras que los últimos 4 serían los AX-12.

```
bool jointMode(uint8_t id, int32_t velocity = 0, int32_t acceleration = 0, const char **log = NULL);
bool wheelMode(uint8_t id, int32_t acceleration = 0, const char **log = NULL);
bool currentBasedPositionMode(uint8_t id, int32_t current = 0, const char **log = NULL);

bool goalPosition(uint8_t id, int32_t value, const char **log = NULL);
bool goalPosition(uint8_t id, float radian, const char **log = NULL);
```

Figura 15: Instrucciones utilizadas.

Con los IDs correctamente establecidos se encontró otro problema, este consistía en que los dos modelos de los motores que se estaban usando (XL-320 y AX-12), funcionaban con un protocolo de comunicación distinto, causando que al conectar los dos modelos juntos sólo funcionara uno. Para corregir esto lo que se hizo fue inicializar cada servo de forma individual cada vez que se fuera a usar, gracias a esto siempre se inicializa el protocolo correcto si se desea estar cambiando el movimiento de un modelo de motor diferente.

Figura 16: Motores modelo XL-320[22].

Con estos problemas corregidos se logró trabajar directamente en lo que fue la locomoción de la plataforma robótica. Se utilizó principalmente la librería DynamixelWorkbench que se había utilizado con anterioridad. De forma más específica se utilizaron las instrucciones de "jointmodez "goalPosition-15. Con estas dos instrucciones se logró establecer la posición a la cual se desea que el servo llegue, lo que en conjunto causó el movimiento completo de la plataforma robótica. En la Figura 18 se puede observar cómo funcionan los valores para los

ángulos en los dos modelos que se están utilizando.

Figura 17: Motores modelo AX-12[23].

7.1.4. Manufactura del robot

Con la segunda iteración del robot se comenzó a manufacturar las piezas. Todas fueron realizadas con impresora 3D utilizando PLA. No se utilizó MDF debido a la forma que tenían las piezas así como el cuerpo del robot eran muy complejas por lo que fue mucho más fácil lograr alcanzar la geometría deseada mediante impresión 3D que intentando unir distintas piezas de MDF. A las primeras impresiones se les tuvo que agrandar los agujeros con un taladro luego se ser impresas ya que se encontraban un poco pequeños, para las siguientes impresiones esto fue corregido para evitar la modificación extra de las piezas y que esto pudiera causar desperfectos cuando todo fuera ensamblado.

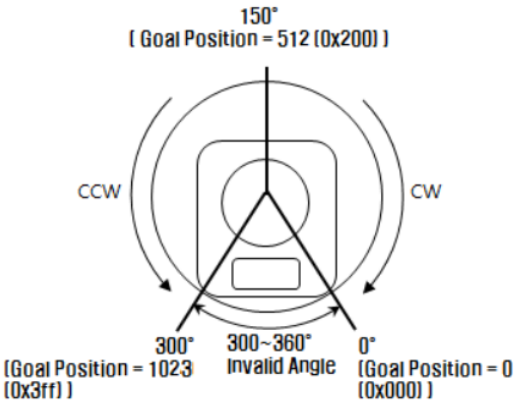


Figura 18: Ángulos que determinan la posición de los motores[23].

Debido a que la placa que se estaba utilizando tenía solo conexiones para los motores XL-320 se fabricaron unos adaptadores especiales para poder conectar los motores AX-12

con los conectores de la placa. Para fabricar esto se tuvo que cortar cables tanto de los motores AX-12 y XL-320 para luego soldarlos entre ellos, lo que resultaba en un cable con dos conexiones en cada extremo, siendo una para un modelo y la otra para el otro.

CAPÍTULO 8

Modelo cinemático de la plataforma robótica

Para poder implementar las rutinas de movimiento de la plataforma robótica fue necesario realizar una simulación y animación del mismo. Esto para poder comprender las posiciones que este debía tener cuando fuera a movilizarse, además de que agilizaba la implementación las rutinas de movimiento, porque se corroboraba con la simulación que los movimientos si fueran los correctos, esto evito la implementación de los movimientos fuera algo basado en prueba y error, ya que al tenerlo de forma correcta en la simulación era mucho más sencillo realizar pequeños ajustes de forma física, sin tener que estar replanteando todos los ángulos de movimiento al cual debían de llegar los servos.

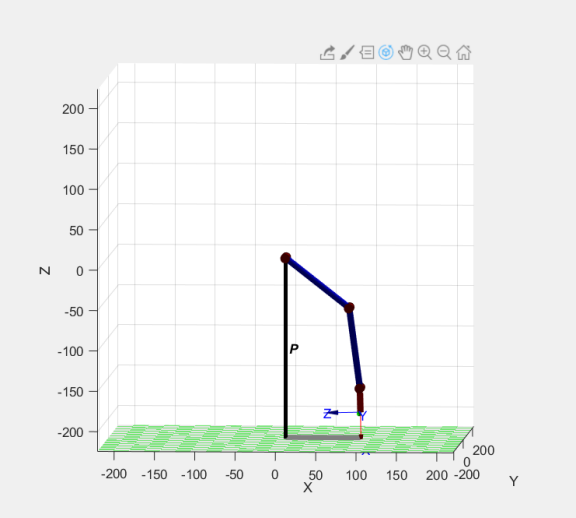


Figura 19: Simulación de la la plataforma robótica.

8.1. Simulación de las patas de la plataforma robótica

La meta de este trabajo de graduación era no solo lograr manufacturar una plataforma robótica sino que también lograr que esta fuese móvil. Para cumplir con este último objetivo fue necesario realizar una simulación en Matlab de las patas del robot. Las patas pueden ser tomadas como un brazo robótico de 3 eslabones, gracias a esto es posible representarlas como un manipulador serial utilizando Denavit Hartemberg [13]. Se sabe de antemano que todas las juntas del robot serán revolutas además de tener las longitudes efectivas de los eslabones.

Para entender cómo funcionan los eslabones de esta plataforma robótica de mejor manera de hacerlo es mediante una simulación en Matlab para poder animar la plataforma robótica y ver si los movimientos se realizan de la manera correcta.

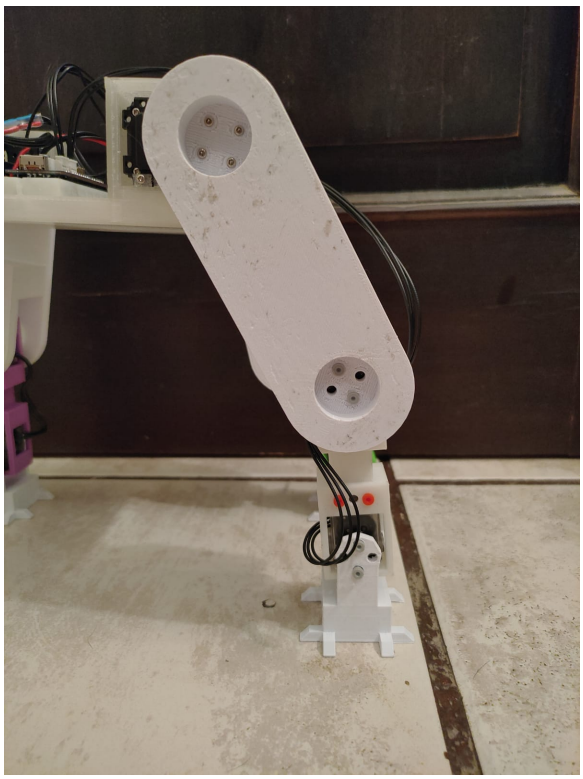


Figura 20: Foto real de la plataforma robótica

Como se mencionó anteriormente conociendo el tipo de juntas junto con las longitudes se aplicó Denavit-Hartemberg. Primero se encontraron los parámetros DH Cuadro 1. Con estos parámetros establecidos se utilizó Matlab para corroborar que estos si estuvieran correctos. Como se puede observar en la Figura 19 se realizó la simulación únicamente de una de las patas, pero se puede ver que esta tiene todos los eslabones en las direcciones correctas, lo que nos indica que la tabla de parámetros está bien planteada. En la Figura 20 podemos observar como el robot se encuentra con una de sus patas en la misma posición que la simulación.

Luego de haber corroborado que la tabla de parámetros de Denavit-Hatemberg estaba

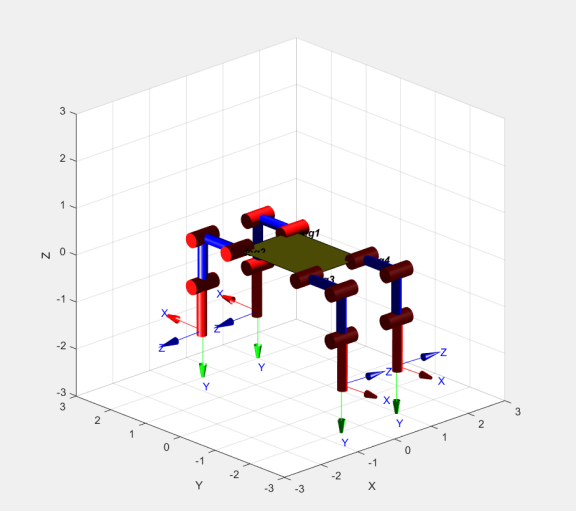


Figura 21: Simulación de toda la plataforma robótica

correctamente planteada utilizando esa primera simulación como base se planteó una segunda simulación pero esta ya fue de toda la plataforma robótica completa, las 4 patas junto con el cuerpo del robot, esto puede observarse en la Figura 21. Como podemos observar, la Figura 14 y la Figura 21 representan la plataforma robótica tanto de manera física como en una simulación en Matlab, teniendo las mismas juntas y eslabones en posiciones idénticas.

ϕ	d	a	α
q1 + $\pi/2$	0	30	$-\pi/2$
q2 - $\pi/2$	0	100	0
0	0	100	0

Cuadro 1: Matriz de parámetros de una de las patas de la plataforma robótica.

CAPÍTULO 9

Intercambio de datos utilizando formato tipo JSON

Para realizar los intercambios de datos entre el Robotat, Matlab y las dos placas utilizadas se utilizó el formato tipo JSON. Se decidió utilizar este tipo de datos para todas las conexiones debido a que es universal y es soportado por todas las plataformas que se utilizaron para las rutinas del robot. Esto nos benefició ya que no fue necesario estar convirtiendo los datos enviados a otro tipo cada vez que se debían leer en una de las múltiples plataformas utilizadas.

Utilizar los JSON también nos ayudara a reducir la carga de datos a la que se somete del OpenCM ya que este ya no se encarga de enviar, crear y además de procesar los datos, sino que ahora solo leerá los valores de los ángulos que son mandados como un JSON desde Matlab para poder inicializar los motores.

En la Figura 22 se puede observar un diagrama con todas las conexiones de la plataforma robótica.

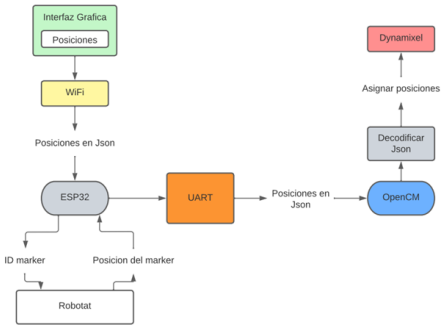


Figura 22: Diagrama de las conexiones de la plataforma robótica

9.1. Conexión entre Matlab y la Esp32

El envió de datos comienza utilizando Matlab. Este se encarga se enviar las posiciones y ángulos se los motores hacia la ESP32 para que este último pueda procesarlos. Matlab se conecta a la ESP32 como cliente mediante la instrucción de *"tcpclient"*, la cual se conecta utilizando un puerto y una dirección IP. Junto con esto se crea el JSON con los ángulos y las posiciones. La creación del JSON se realiza en una interfaz gráfica igualmente en Matlab 23. Todos los botones realizan la misma conexión, la única diferencia entre cada uno son los valores y la cantidad que se envían, ya que cada botón de la interfaz representa un movimiento distinto para el robot.

La ESP32 utiliza nuevamente la función *"deserializeJSON"* para poder revisar el JSON enviado desde Matlab. Debemos conectar la ESP32 a cualquier conexión WiFi esto nos dará una dirección IP la cual es la que usamos en Matlab. Cuando Matlab envía los datos mediante de la interfaz, la ESP32 lee los datos que se están enviando del cliente. Es en este momento que se utiliza *"deserializeJSON"* pero debemos especificarle que no debe leer los datos de una conexión serial sino de un cliente que se encuentre conectado. Esto nos permite revisar el JSON y poder comenzar a utilizar los datos que vienen en el para asignarlo a las variables necesarias que luego serán mandadas al OpenCM.

Además de las posiciones para los motores, también se está enviando desde Matlab una variable llamada "bandera". Esta variable se manda para cada movimiento diferente, en otras palabras se está enviando la misma variable sin importar cual botón de la interfaz se presione. Lo que cambia es que se le asigna un valor distinto, para cada movimiento, a esa variable "bandera". Esto ayuda a que, cuando el JSON es revisado por la ESP32, gracias al valor asignado de la variable "bandera", este no procesa los datos de todos los movimientos al mismo tiempo sino que solo revisara y enviara los que se encuentran asignados al número que la variable "bandera"tiene actualmente. Este filtrado utilizando la variable bandera ayuda al OpenCM para que no deba abrir un JSON con una carga de datos tan grande, reduciendo la cantidad de procesamiento de datos que esta debe hacer. En las rutinas del OpenCM también se encuentra un filtrado utilizando la variable bandera, esto se realiza para que el OpenCM no se mantenga esperando a recibir la información de todos los movimientos, esta solo ejecutara el movimiento que se encuentre asignado al valor de la variable "bandera".

9.2. Conexión entre Robotat y la Esp32

Se realizó una conexión entre el Robotat y la ESP32. Esta se hizo con la finalidad de poder leer y procesar la información que el Robotat tendrá acerca del robot, como lo sería la posición del robot dentro del rango del ecosistema robótico. No se está usando esta información con ningún otro fin, ya que el poder utilizar la información de las posiciones del robot será implementado en su totalidad en los siguientes años en distintos trabajos de graduación.

Para realizar la conexión entre el Robotat y la ESP32 se debe conectar la ESP32 a la red el Robotat usando WiFi, esto se realiza al igual que con Matlab con una dirección IP además del nombre y la contraseña dela red. Luego la ESP32 debe conectarse a la dirección

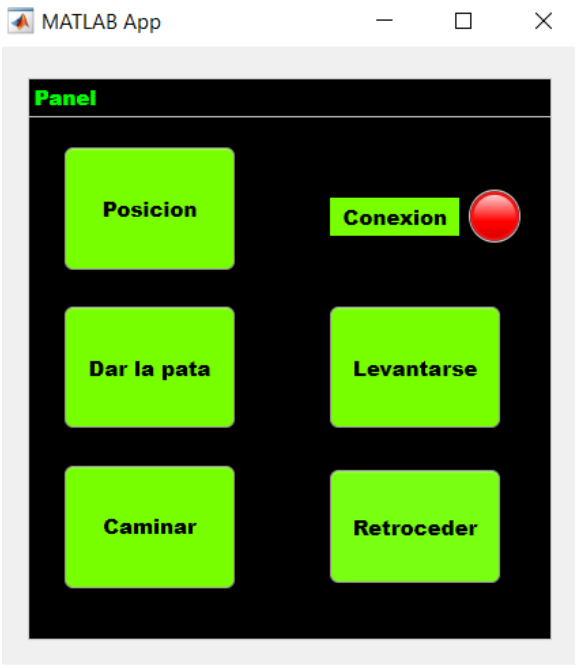


Figura 23: Interfaz gráfica en Matlab

IP del servidor de las cámaras del sistema Optitrack, esto nos permitirá leer los datos de las posiciones que nos envía el Optitrack y los desplegara en el monitor serial como se puede ver en la Figura 24.

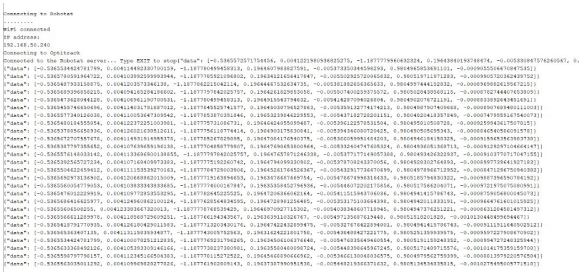


Figura 24: Datos del ecosistema Robotat

9.3. Conexión entre OpenCM y la Esp32

Para realizar él envió de datos entre el OpenCM y la ESP32 se utilizaron los puertos serial de cada chip, siendo el serial2 para la ESP32 y el serial 3 para el OpenCM. Además de esto mientras que el OpenCM está siendo alimentado por dos baterías de 3.7 cada una, la ESP32 está siendo alimentada por el propio OpenCM utilizando el pin que provee 5V de salida.

Se comenzó realizando la conexión entre la ESP32 y el OpenCM utilizando únicamente dos jumpers, pero se encontró con el inconveniente que esta no se estaba conectando a una red WiFi cuando era alimentada de esta manera. Esta se conectaba a una red WiFi únicamente

cuando se alimentaba usando la conexión micro USB que tiene el chip. Se descubrió que este modelo de ESP32 trae conectado, de fábrica, el módulo WiFi con la entrada micro USB mientras que el pin Vin no se encontraba conectado con el módulo WiFi, lo que causaba que la placa si se energizara mientras que el módulo WiFi no. Para resolver este problema se creó un adaptador conformado por dos jumpers y un cable micro USB 27. Simplemente se cortó el cable en dos y se le soldaron dos jumpers uno para tierra y otro para voltaje, logrando conectar los pines Vin y tierra del OpenCM con la entrada micro USB de la ESP32. Esto resolvió el problema completamente.

Figura 25: Adaptador usado para la ESP32

Entre la ESP32 y el OpenCM se está enviando un JSON con los ángulos y posiciones necesarias para que los motores puedan moverse. La ESP32 se encarga de realizar la conexión WiFi para poder recibir datos de Matlab además de asignarle el nombre a las variables que representaran los ángulos de los motores, seguido de esto envía por conexión serial estos datos de los ángulos al OpenCM, este último solo se encargara de establecer las velocidades de los motores, sus posiciones así como de inicializarlos. Para poder analizar el contenido del JSON dentro del ESP32 se utiliza la función *"deserializeJson"*, mientras que para poder crear un JSON con todos los valores de los motores se utiliza la función *"serializeJson"*. Esta última no solo crea el JSON sino que también la envía por conexión serial hacia el OpenCM. Debido a que estamos usando el lenguaje de Arduino en los dos chips, se están utilizando las mismas funciones para el OpenCM como para la ESP32. Con la función *"deserializeJson"* se puede revisar el JSON y así asignarle los valores de los ángulos a cada motor para luego ser inicializados.

Gracias a esta distribución entre estos dos chips el OpenCM tiene una menor carga de procesamiento de datos, mientras que la ESP32 tiene una mayor carga pero esto no nos afecta ya que esta es mucho más superior al OpenCM en cuanto a procesamiento de datos.

9.4. Multihilo con la plataforma robótica

Para poder optimizar los procesos que deben de realizar los microcontroladores utilizados en la plataforma robótica. Se necesitaba implementar programación de multihilo. Se optó por utilizar *"Visual Studio Code"*, como se ve en la Figura 26 ya que este cuenta con más funciones que el IDE de arduino. Utilizando la ESP32 se implementó la programación de multihilo y *"Visual Studio Code"* fue de ayuda ya que la velocidad de compilación es mucho más eficiente que la de arduino lo que agilizo las pruebas.

No fue posible utilizar *"Visual Studio Code"* con el OpenCM ya que, a pesar que *"Visual Studio Code"* si cuenta con la posibilidad de poder programar en el lenguaje de arduino las librerías de Dynamixel usadas con el OpenCM no están implementadas en *"Visual Studio Code"*. Se intentó agregar y modificar las librerías de dynamixel para poder ser usadas en *"Visual Studio Code"* pero esto no fue posible.

La solución para esto fue seguir utilizando arduino únicamente para el OpenCM mientras que para la ESP32 si se está utilizando *"Visual Studio Code"* ya que todas las librerías de la ESP32 que se estaban usando si eran combatibles con esta plataforma. Entonces el multihilo fue implementado únicamente en la ESP32, pero esto fue beneficioso ya que el OpenCM únicamente está encargado de inicializar los servos con los datos que le manda la ESP32, mientras que la ESP32 si está encargada de revisar que todos los datos se envíen de manera correcta además de crear el JSON con los datos recibidos de Matlab y decodificar los valores de las posiciones que nos está enviando el Robotat, por lo que la técnica de multihilo se usa para optimizar estos procesos.

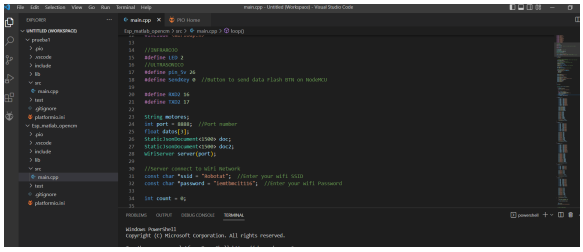


Figura 26: Plataforma Visual Studio Code

Figura 27: Adaptador usado para la ESP32

CAPÍTULO 10

Conclusiones

- Se logró diseñar, manufacturar y programar una plataforma robótica móvil.
- Se desarrollaron e implementaron rutinas poco complejas que lograron el movimiento de la plataforma robótica.
- Se logró realizar una conexión exitosa entre dos modelos distintos de motores Dynamixel utilizando las rutinas que inicializan cada modelo de motor.
- Se logró realizar una conexión física exitosa entre diferentes modelos de motores Dynamixel
- Se consiguieron las baterías correctas para poder tener un buen tiempo de uso de la plataforma robótica
- Luego de varias iteraciones se logró encontrar una con buena estabilidad mientras el robot se encuentra en funcionamiento

CAPÍTULO 11

Recomendaciones

- Ampliar la funcionalidad de la plataforma robótica mediante el uso de sensores o cámaras además de la modificación de las rutinas de movimiento para que estas sean más complejas, esto para dar paso a usos más complejos del robot
- Utilizar el mismo modelo para todos los actuadores, para así evitar el problema de tener que usar protocolos distintos.
- Utilizar la expansión de la placa OpenCM9.04 para evitar la fabricación de los adaptadores con doble conexión.
- A pesar que las baterías se recargaban cuando el robot comenzaba a tener movimientos torpes, lo que ayudaba a que estas no tuvieran una descarga completa, es necesario el implementar un circuito de protección de descarga para no dañar las baterías en un descuido, debido a que la vida de estas disminuye cada vez que estas se descargan completamente. De cada celda utilizada debe salir una conexión hacia el circuito de protección que se debería de implementar.

CAPÍTULO 12

Bibliografía

[1] J. D. P. Orellana, "Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento," Trabajo de graduación de Licenciatura en Ingeniería Mecatrónica, Universidad del Valle de Guatemala, 2021.

[2] C. P. Montoya, "Robotat:Un ecosistema robótico de captura de movimiento y comunicación inalámbrica," Trabajo de graduación de Licenciatura en Ingeniería Mecatrónica, Universidad del Valle de Guatemala, 2021.

[3] M. DEARBORN. "Robots Cuadrúpedos se convierten en los nuevos compañeros de ingenieros de Ford." (), dirección: `https://media.ford.com/content/fordmedia/fna/mx/es/news/2020/07/27/no-bones-about-it-ford-experiments-with-four-legged-robots.html`. (accessed: 04.17.2022).

[4] "La robótica en la asistencia sanitaria." (), dirección: `https://www.intel.es/content/www/es/es/healthcare-it/robotics-in-healthcare.html`. (accessed: 04.27.2022).

[5] "Programable open source robot." (), dirección: `https://www.petoi.com/`. (accessed: 05.08.2022).

[6] H. Kimura, "Adaptive Motion of Animals and Machine," en Springer Tokyo, 2006.

[7] G. de Santos, "" SILO6: Desing and configuration of a legged robot for humanitarian demining," 2002, págs. 21-26.

[8] E. Tezel. "Robots De Muchas Patas." (), dirección: `https://www.wevolver.com/wevolver.staff/enik.robot`. (accessed: 04.27.2022).

[9] "Actuadores de muy altas prestaciones en red para robots." (), dirección: `https://www.ro-botica.com/tienda/ROBOTIS-DYNAMIXEL`. (accessed: 04.17.2022).

[10] "OptiTrack for Robotics." (), dirección: `https://optitrack.com/applications/robotics/`. (accessed: 04.18.2022).

[11] "Protocolo Dynamixel 1.0." (), dirección: `https://emanual.robotis.com/docs/en/dxl/protocol1/`. (accessed: 05.27.2022).

[12] "Cinemática de Robots," págs. 1-30.

[13] J. C. Parejo. "La representación Denavit-Hartenberg." (), dirección: https://docplayer.es/70397649-La-representacion-denavit-hartenberg.html.

[14] J. Z. E. Calle I. Ávila, "Diseño e Implementación de un Robot Móvil Cuadrúpedo," 2007, págs. 65-72.

[15] "Tarjeta de control OpenCM9.04-A para servomotores Dynamixel." (), dirección: https://www.generationrobots.com/en/401596-opencm904-a-control-board-for-dynamixel-servomotors.html. (accessed: 05.08.2022).

[16] "Trabajando con JSON." (), dirección: https://www.json.org/json-es.html. (accessed: 09.29.2022).

[17] "ESP32 Wi-Fi." (), dirección: https://www.espressif.com/en/products/socs/esp32. (accessed: 09.29.2022).

[18] "PlatformIO." (), dirección: https://pypi.org/project/platformio/. (accessed: 11.30.2022).

[19] "Dynamixel2Arduino." (), dirección: https://www.arduino.cc/reference/en/libraries/dynamixel2arduino/. (accessed: 09.29.2022).

[20] "Dynamixel Workbench." (), dirección: https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_workbench/. (accessed: 09.14.2022).

[21] R. L. Norton. "Diseño de Maquinaria." (), dirección: https://lsbunefm.files.wordpress.com/2018/10/disec3b1o-de-maquinaria-robert-l-norton-4.pdf. (accessed: 09.14.2022).

[22] "XL-320." (), dirección: https://emanual.robotis.com/docs/en/dxl/x/xl320/. (accessed: 09.14.2022).

[23] "Diseño de Maquinaria." (), dirección: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/. (accessed: 09.14.2022).

CAPÍTULO 13

Anexos

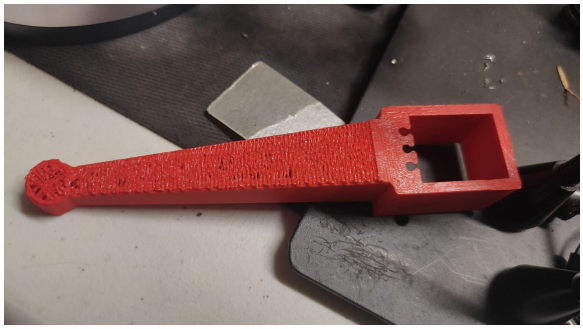


Figura 28: Pieza fallida iteración1

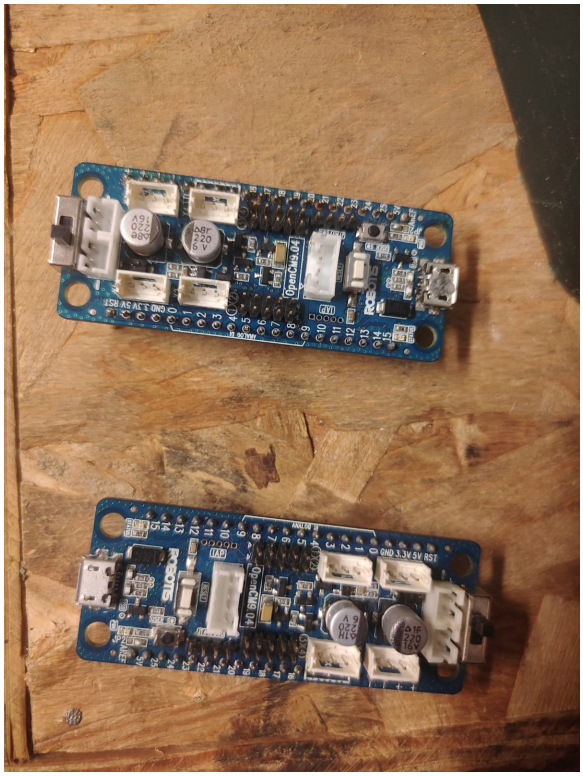


Figura 29: Placas que se quemaron durante el proyecto

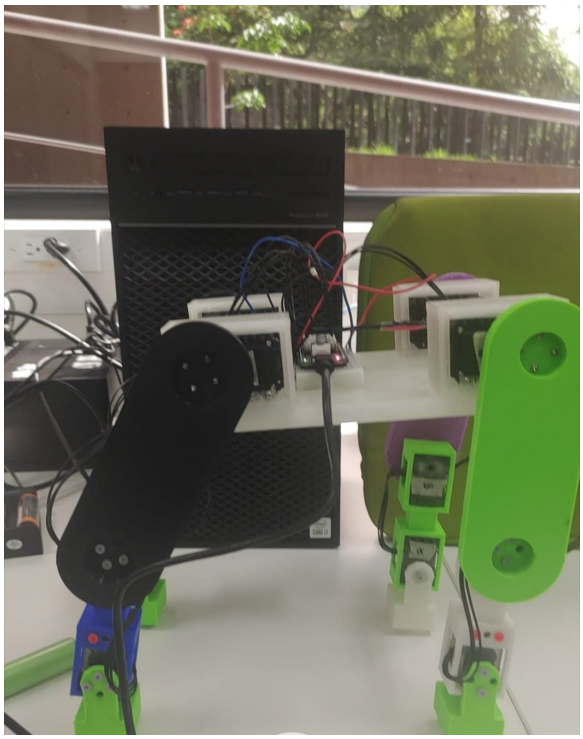


Figura 30: Pruebas realizadas con la iteración 2

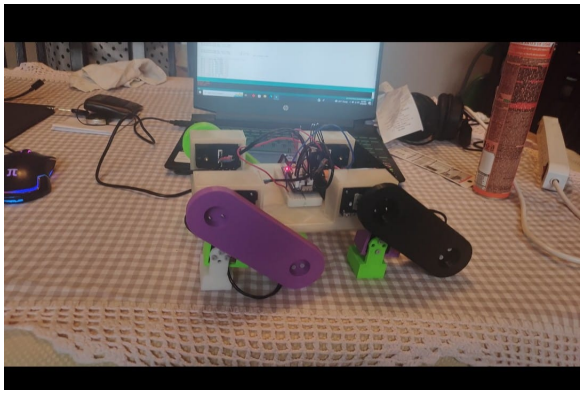


Figura 31: Primeras pruebas realizadas de las rutinas de movimiento con la iteración 2

Figura 32: Pieza de la iteración 2 que fue modificada

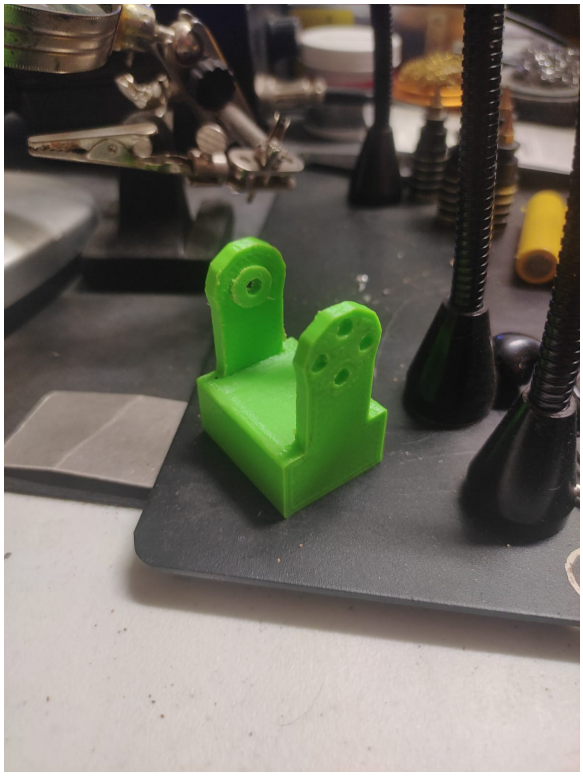


Figura 33: Segunda pieza de la iteración 2 que fue modificada