Diseño de un robot hexapod con servos inteligentes para su implementación dentro del ecosistema Robotat

Luis Roberto Salazar Maldonado

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un robot hexapod con servos inteligentes para su implementación dentro del ecosistema Robotat

Trabajo de graduación presentado por Luis Roberto Salazar Maldonado
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

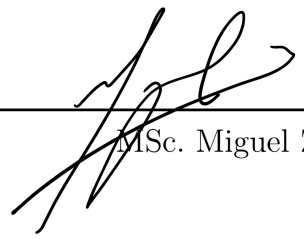
Diseño de un robot hexapod con servos inteligentes para su implementación dentro del ecosistema Robotat

Trabajo de graduación presentado por Luis Roberto Salazar Maldonado
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

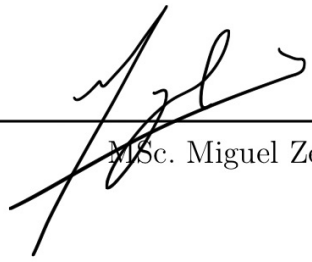
2023

Vo.Bo.:

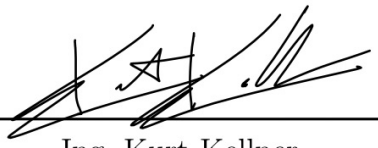
(f) 

MSc. Miguel Zea

Tribunal Examinador:

(f) 

MSc. Miguel Zea

(f) 

Ing. Kurt Kellner

(f) 

MAEB. Pablo Mazariegos

Fecha de aprobación: Guatemala, 5 de enero de 2023.

Índice

Lista de figuras

Lista de cuadros

Resumen

En este trabajo de graduación se desarrolló una plataforma robótica con la estructura de un robot hexapod de 6 extremidades y su implementación en el ecosistema Robotat de la Universidad del Valle de Guatemala. Esto se logró en 3 etapas, el diseño mecánico, el diseño de la electrónica interna y la conexión con Robotat.

Para la parte del diseño mecánico se utilizaron los servos AX-12A y XL-320 de la marca Dynamixel y a partir de esto se diseñaron las partes del cuerpo del robot con el uso del programa Inventor para luego ser manufacturadas usando corte láser e impresión 3D. En el torso se utilizó acrílico de 2.5 mm de grosor para crear 4 piezas y fueron unidas con tornillos a los servos AX-12A y Rivets para la cubierta. Para las extremidades se utilizaron 12 servos AX-12A y 6 XL-320, junto con piezas impresas en 3D y marcos compatibles con el AX-12A.

Para la electrónica interna se utilizaron los microcontroladores ESP32 y OpenCm 9.04, una placa electrónica personalizada para el ESP32, la expansión OpenCm 485 EXP y una expansión de conectores 3P para el AX-12A. Se utilizó el programa ArduinoIDE para la programación de ambos microcontroladores, gracias a su compatibilidad y librerías. También se utilizó el programa Altium Designer para el diseño de la PCB del ESP32, la cual se diseño para tener una comunicación UART con el OpenCm 9.04 y una entrada de voltaje para toda la plataforma.

En la implementación del ecosistema Robotat se realizó la conexión entre el ESP32, Optitrack y Matlab con una comunicación cliente-servidor en formato JSON. El objetivo de estas conexiones era, recibir los datos de posición de un marcador de Optitrack y almacenar los datos en el ESP32, establecer las secuencias de los servos en Matlab para luego enviarlos al ESP32 y al OpenCm 9.04 para su ejecución. Por otro lado, se calcularon secuencias de movimiento de prueba usando cinemática directa e inversa.

Abstract

In this graduation work, a robotic platform was developed with the structure of a hexapod robot with 6 extremities and its implementation in the Robotat ecosystem of the Universidad del Valle de Guatemala. This was achieved in 3 stages, the mechanical design, the design of the internal electronics and the connection with Robotat.

For the mechanical design, Dynamixel AX-12A and XL-320 servos were used and from this, the body parts of the robot were designed using the Inventor program and then manufactured using laser cutting and 3D printing. For the torso, 2.5 mm thick acrylic was used to create 4 pieces and they were attached with screws to the AX-12A servos and Rivets for the cover. For the limbs, 12 AX-12A and 6 XL-320 servos were used, along with 3D printed parts and AX-12A compatible frames.

ESP32 and OpenCm 9.04 microcontrollers, a custom electronics board for the ESP32, OpenCm 485 EXP expansion and a 3P connector expansion for the AX-12A were used for the internal electronics. The ArduinoIDE program was used for programming both microcontrollers, thanks to its compatibility and libraries. The Altium Designer program was also used for the design of the ESP32 PCB, which was designed to have a UART communication with the OpenCm 9.04 and a voltage input for the entire platform.

In the implementation of the Robotat ecosystem, the connection between the ESP32, Optitrack and Matlab was made with a client-server communication in JSON format. The objective of these connections was, to receive position data from an Optitrack marker and store the data in the ESP32, establish the sequences of the servos in Matlab to then send them to the ESP32 and OpenCm 9.04 for execution. On the other hand, test motion sequences were calculated using direct and inverse kinematics.

CAPÍTULO 1

Introducción

El objetivo principal de este proyecto de graduación es diseñar y ensamblar un robot hexapod construido a partir de servos inteligentes, en específico, dos tipos de servos Dynamixel que son: 6 servos AX-12A y 12 servos XL-320. Esta plataforma robótica se desarrolla con el fin de ser compatible con el ecosistema Robotat de la Universidad del Valle de Guatemala. Como se mencionó anteriormente, al ser un desarrollo, esta plataforma debe ser construida desde cero, por lo que se deben desarrollar tres aspectos importantes: el diseño mecánico de la plataforma, el diseño de la electrónica interna y por último su implementación al ecosistema Robotat.

Para que sea una plataforma funcional el robot debe realizar 4 movimientos básicos, caminar hacia adelante y hacia atrás, y girar a los lados. Cada una de estas secuencias de movimiento se pueden combinar para lograr la traslación del robot a cualquier parte del ecosistema, esto se logra gracias a los microcontroladores que posee, ya que uno de ellos, el OpenCm 9.04, controla las secuencias del movimiento de los motores mientras que el otro, el ESP32, controla que secuencias debe hacer. En palabras simples, el microcontrolador OpenCm 9.04 contiene el algoritmo que dice como hacer el movimiento y el ESP32 contiene el algoritmo de cuando hacer las secuencias.

En el mundo existen varios proyectos de robots hexapods, pero ninguno de ellos es autónomo, debido a que en su mayoría se controlan manualmente ya sea con un control remoto, una aplicación de celular, etc. Con este proyecto se busca la implementación del robot con un sistema de captura de movimiento, con el fin de implementar rutinas simples de movimiento y logrando así un movimiento autónomo. El proyecto está desarrollado para que en futuras investigaciones se implementen rutinas complejas.

A continuación se detalla el proceso del desarrollo de la plataforma el cual incluye diseño mecánico, diseño electrónico, programación de microcontroladores, análisis de movimiento, implementación del sistema de captura de movimiento y automatización de rutinas.

CAPÍTULO 2

Antecedentes

Para lograr comprender de forma correcta el funcionamiento de la plataforma robótica a proponer en el presente trabajo se deben tener en consideración tres partes importantes, las cuales se muestran a continuación en forma de ejemplos. El primer antecedente muestra un proyecto de un hexápodo en donde se implementaron servo motores como fuente de movimiento, diseño mecánico e impresión 3D y electrónica interna. En este proyecto se muestra el control por medio de un mando Bluetooth. El segundo antecedente es de una tesis en donde se presenta el desarrollo de un paquete de herramientas de software en donde se ejecutan trayectorias del manipulador R17 observado por un sistema de captura de movimiento que es Optitrack. En el último antecedente se detalla la forma en que se implementa la red de comunicación dentro de un ambiente de Robotat y se realizan simulaciones en donde se evalúan las condiciones del proyecto.

2.1. Hexapod PhantomX MK-IV por Interbotix

En el año 2020 la empresa Interbotix desarrolló un hexapod llamado PhantomX MK-IV, el cual es construido en su totalidad por partes impresas en 3D y servos inteligentes de la serie Dynamixel XL de Robotis. Viene pre-ensamblado y probado y es totalmente compatible con Ubuntu 20.04c/ROS Noetic. La plataforma ofrece opciones de WiFi, conexión anclada e inalámbrica a través del Bluetooth de un control de Playstation 4. Por otra parte tiene una capacidad de carga útil de aproximadamente 1 lb, una velocidad de marcha de aproximadamente 129 mm/s (marcha predeterminada) y un tiempo de funcionamiento estimado con batería de 45 minutos. [1].

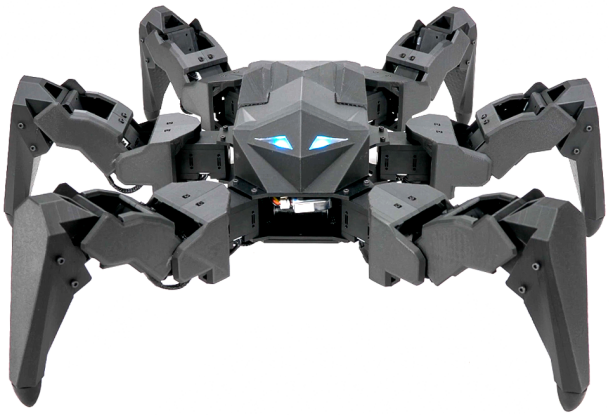


Figura 1: Hexapod PhantomX MK-IV.[1]

Como se mencionó anteriormente el sistema utiliza un control de Playstation 4 programado en Raspberrypi utilizando Ubuntu Mate y Ross Melodic. Los desarrolladores utilizaron estas herramientas ya que la Raspberrypi ofrece mucho mas poder de procesamiento y RAM que cualquier otro controlador. Por otra parte ofrece conexión Wi-Fi para controlar al hexapodo de forma remota. [1].

2.2. Diseño e implementación de un paquete de herramientas de software para controlar inalambricamente un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento

En este trabajo se presenta el desarrollo e implementación de un paquete de herramientas de software que permite la ejecución de trayectorias del manipulador R17 en un espacio de tarea observado por un sistema de captura de movimiento. Se abarcó desde el levantamiento del sistema hasta el desarrollo y pruebas de software del manipulador. El sistema de captura de movimiento utilizado fue el OptiTrack, con el cual se contaba con 6 cámaras Primex 41. [2].

El proceso para su levantamiento abarca el posicionamiento de las cámaras, sus conexiones, el proceso de calibración y la obtención de datos (poses de cuerpos rígidos). Además se trabajó con los SDK's disponibles, exportando la información obtenida a Python. Con respecto al manipulador R17, se estableció el modelo cinemático usando la convención de Denavit-Hartenberg, y se hizo pruebas de cinemática directa, inversa y ejecución de trayectorias. [2].

Figura 2: Manipulador R17 sobre ecosistema Robotat[2]

2.3. Diseño e implementación de una red de comunicación Wi-Fi e interfaz gráfica para una mesa de pruebas de robótica de enjambre.

En este documento se plantea el desarrollo del software de una mesa de pruebas de robótica de enjambre de bajo costo, por medio de la cual se pueden evaluar los algoritmos de robótica de enjambre para ser validados con agentes físicos en entornos reales. Se describe el desarrollo de la aplicación basada en multithreading desarrollada en python para la interacción entre el usuario y la plataforma, al igual que el despliegue de la información en tiempo real. [3].

También se detalla el software implementado en los agentes y cómo es que este interactúa con el software de la plataforma para poder realizar las simulaciones y cargar el algoritmo de control de una forma eficiente, permitiéndole al usuario obtener la información necesaria para evaluar una gran variedad de algoritmos swarm. También se detalla la forma en que se implementa la red de comunicación dentro del sistema, cómo esta la diseña el usuario y es creada por los agentes al momento de iniciar la simulación. Explicando la flexibilidad que presenta esta red para poder ser implementada en diferentes topologías de comunicación, las cuales pueden ser diseñadas por el usuario desde la aplicación. [3].

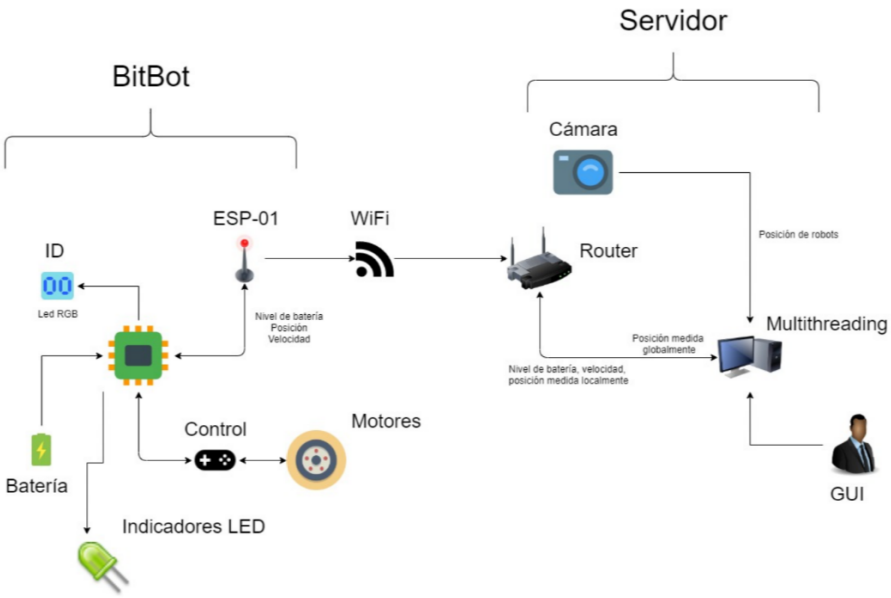


Figura 3: Esquema de trabajo de Robotat[3]

CAPÍTULO 3

Justificación

Los robots autónomos son una herramienta muy importante en la tecnología, los cuales pueden llegar a realizar desde tareas sencillas hasta complicadas para las personas, sin necesidad de controlarlos remotamente, sin arriesgar a los trabajadores y sin tener el factor de error que podría cometer un humano. Como ejemplo claro tenemos a Boston Dynamics, el cual ha diseñado robots bípedos y cuadrúpedos inteligentes capaces de realizar tareas que se han aplicado en industrias, incrementando su eficiencia. Esto lo logran los robots a través de su capacidad para recorrer las instalaciones de las empresas, realizando determinadas tareas, ya sea la búsqueda de fallos en la maquinaria por medio de lectura de mediciones utilizando de sensores y cámaras o simplemente realizar el mapeo de las instalaciones. Esto se logra a través de años de investigación y desarrollo de los robots, lo cual en países avanzados tecnológicamente no es algo reciente, por lo que se está viendo su implementación cada vez más en muchas industrias y empresas. Lamentablemente, en nuestro país no se han implementado robots en procesos industriales debido a la falta de desarrollo nacional y desinterés en su inversión, pero con forme el tiempo pasa esto se ha vuelto una necesidad y Guatemala debe desarrollarse en ese ámbito, ya que ese paso es una gran oportunidad para salir adelante como país y ser un ejemplo internacional en temas de tecnología y eficiencia industrial.

Como se mencionó anteriormente, los robots autónomos pueden llegar a ser una ventaja para las industrias, pero pueden llegar a ser importantes para el entretenimiento, transporte o su aplicación en medicina, ya que es algo que cada vez va creciendo y puede llegar a aplicarse en cualquier ámbito de la vida. Es por eso que este proyecto de graduación se enfoca en el desarrollo de un robot hexapod autónomo el cual se implementará en un ecosistema Robotat y así tener una plataforma para realizar experimentos, lo cual significa que se utilizará un sistema de captura de movimiento para obtener la posición del robot en tiempo real y en base a eso realizar distintas tareas de movimiento sin necesidad de controlarlo de manera remota utilizando algoritmos en su programación y usando una red de comunicación Wi-Fi. El desarrollo de esta plataforma es un gran paso para las necesidades del país y en la Universidad del Valle de Guatemala tenemos la responsabilidad de acercarnos cada vez más a estar en la vanguardia de la tecnología abriendo el camino para nuevos sistemas e implementándolo en aplicaciones prácticas y útiles para los guatemaltecos.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar y ensamblar un robot hexapod construido a partir de servos inteligentes compatible con el ecosistema Robotat.

4.2. Objetivos específicos

- Realizar el diseño mecánico de la plataforma para su movimiento óptimo utilizando herramientas CAD e implementarlo con técnicas de manufactura rápida.
- Diseñar el sistema electrónico interno del robot, agregando los microcontroladores OpenCm 9.04 y ESP32, junto con un dimensionamiento adecuado del sistema de potencia.
- Implementar y configurar el sistema de captura de movimiento de Optitrack y utilizarlo como medio de control del hexapod a través del ESP32 por medio de WiFi.

CAPÍTULO 5

Alcance

La meta del presente trabajo es principalmente realizar una plataforma robótica e implementar rutinas simples de movimiento por medio del ecosistema robotat. Para ser más específicos se desarrolló un robot hexápodo en donde se utilizan partes impresas en 3D y piezas de acrílico en conjunto con los servos Dynamixel para la parte mecánica de la plataforma, y se utiliza el microcontrolador OpenCM 9.04 junto con su expansión OpenCM 485 EXP y un PCB personalizado para el microcontrolador ESP32 y lograr así una conexión con el OpenCM 9.04 para la parte electrónica. Posteriormente, se realiza la implementación al sistema Robotat, del cual se ejecutan secuencias simples de movimiento utilizando cinemática directa o inversa.

El enfoque principal es el desarrollo de la plataforma robótica y que sea compatible con el ecosistema robotat, lo que significa que la plataforma debe ser capaz de recibir datos del sistema de captura de movimiento, recibir las secuencias de movimiento a través de MATLAB y enviar esos datos a los servos. Por último, se aplican rutinas de prueba simples, con el objetivo de que la plataforma robótica sea optimizada para que en trabajos posteriores se puedan aplicar rutinas complejas.

Por otro lado, se tuvieron retrasos en el desarrollo del proyecto ya que se tenían que obtener los servos Dynamixel AX-12A y XL-320, mismos que no venden en el país y los cuales se obtuvieron gracias a la Universidad, además, como son modelos diferentes se tuvo que fabricar seis adaptadores que conectan los dos tipos servos. Además, por motivos de funcionamiento se reemplazaron 6 XL-320 por otros 6 AX-12A, ya que estos no poseen el torque suficiente para sostener al robot en una secuencia de movimiento. Teniendo un total de 6 XL-320 y 12 AX-12A.

CAPÍTULO 6

Marco teórico

6.1. Movimiento de hexápodo

La habilidad de un robot hexápodo para eludir obstáculos, caminar en distintos tipos de terreno y su eficiencia energética depende en gran medida de la relación entre el largo de sus patas y su peso. [4].

El caminado de un hexápodo de acuerdo al número de patas que se encuentran en apoyo, se puede clasificar en secuencias de caminado, que pueden ser de tipo onda, tetrápodo, transición y trípode. Para crear estas secuencias de caminado se puede utilizar un generador central de patrones, redes neuronales, algoritmos genéticos o un generador aleatorio basado en aprendizaje. En el caminado cada pata puede tener dos estados: avance y recuperación.[4].

En el movimiento y desplazamiento de un robot hexápodo se tienen que controlar y coordinar al mismo tiempo sus seis patas, cada una de las cuales se representa normalmente utilizando la estructura de una cadena cinemática de tres eslabones con tres grados de libertad, por lo que el control de un robot hexápodo es un problema en el que se tienen que controlar 18 uniones al mismo tiempo. [4].

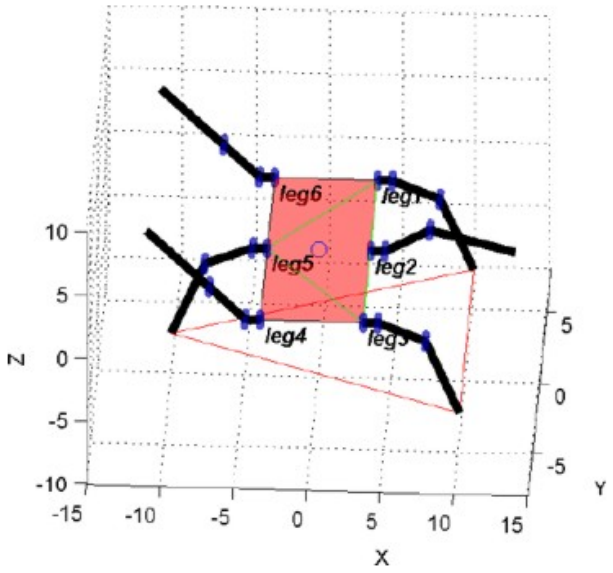


Figura 4: Movimiento de hexápodo gráfico[4]

Un robot hexápodo al considerarse como un sistema aislado sus patas se pueden mover libremente sin embargo al poner las patas sobre la superficie estas ya no se pueden mover libremente sino que estarán restringidas por la fricción entre el piso y las patas, sus movimientos se limitan de manera similar a lo que sucede cuando una mano robótica con tres dedos sujeta un objeto uniforme . [4].

Durante la etapa de avance en el caminado trípode un robot hexápodo se asemeja a un robot paralelo tipo Delta, la plataforma móvil es el cuerpo del robot y la plataforma fija es el triángulo de apoyo que forman las tres patas que se encuentran en contacto con el suelo.

El caminado trípode del robot hexápodo se divide en seis estados a partir de la posición inicial en la cual las patas 1, 3 y 5 están en apoyo y las patas 2, 4 y 6 están levantadas. El proceso de caminado se muestra en la siguiente figura.[4].

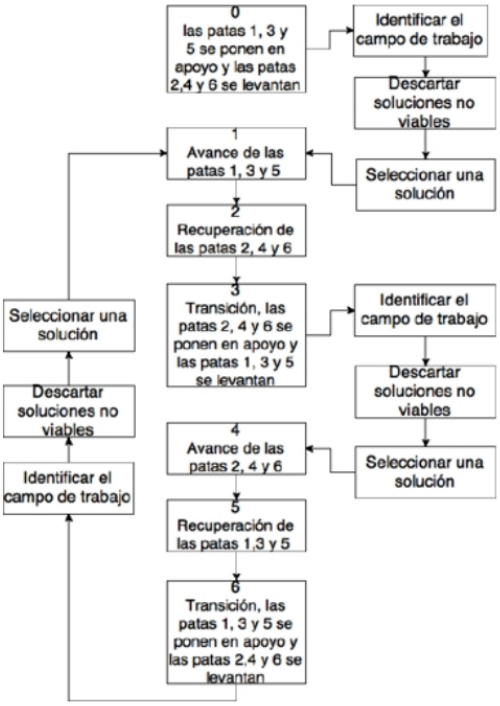


Figura 5: Proceso de movimiento de hexápodo[4]

En el primer estado las dos patas del lado izquierdo 1 y 3, y la pata central 5 del lado derecho se encuentran en apoyo, estas patas se desplazan hacia atrás cambiando el valor de los ángulos del actuador, esto provocará que el robot avance en línea recta hacia enfrente. [4].

En el segundo estado (recuperación), las dos patas del lado derecho 4 y 6, y la pata central 2 del lado izquierdo se ponen en la posición inicial, y los ángulos del actuador se les asigna el valor cero.[4].

En el tercer estado (transición), las patas 4, 6 y 2 se ponen en apoyo, posteriormente las patas 1, 3 y 5 se levantan.

En el estado cuatro las patas 4, 6 y 2 realizan el avance.

En el estado cinco las patas 1, 3 y 5 se recuperan a su posición inicial.

En el estado seis se realiza la transición y se vuelve a repetir el ciclo.

El primer y el cuarto estado son los que controlan el desplazamiento del hexápodo. En el avance en línea recta el ángulo de las patas del lado derecho y del lado izquierdo en apoyo deben tener la misma magnitud pero con sentido diferente. [4].

6.2. Técnicas de modelado en Autodesk Inventor

Las diversas técnicas y enfoques disponibles para modelar piezas y crear ensamblajes pueden afectar al rendimiento. El enfoque de modelado escogido determina el número de incidencias, la complejidad de la geometría, los métodos de restricción y la creación de ensamblajes. Normalmente se mezclan varias técnicas para ajustarse al propósito de los productos y del diseño. Puede utilizar el modelado descendente para diseñar y crear una estructura o un modelado ascendente para insertar y restringir componentes de una biblioteca.[5].

El modelado ascendente es el método tradicional para crear ensamblajes. Primero se definen las piezas individuales. A continuación, se incluyen en subensamblajes mediante las restricciones de ensamblaje. Después los subensamblajes se insertan en los ensamblajes de nivel superior hasta llegar al ensamblaje del nivel superior. Se trabaja de manera ascendente. Este método de ensamblaje da lugar a ensamblajes con un gran número de relaciones entre piezas y ensamblajes. [5].

El modelado descendente es un método que permite empezar definiendo el resultado y generar todos los criterios de diseño conocidos. El resultado se convierte en la base para los subensamblajes y piezas subyacentes. Hay un único archivo conceptual que contiene toda la información del diseño con una ubicación única para incorporar los cambios del diseño. Este método puede proporcionar actualizaciones más rápidas, más recursos disponibles para la gestión de grandes conjuntos de datos y un modo más fácil de trabajar en un entorno de colaboración. En general, es una forma más adecuada para los trabajos de diseño. [5].

6.3. Servos

Normalmente los motores habituales transforman la energía eléctrica (o química) en un giro continuo que se puede utilizar para desarrollar trabajo mecánico. Sin embargo, los servos son también motores de corriente continua, pero en lugar de diseñarse para obtener un giro continuo, se diseñan para que se muevan un ángulo fijo en respuesta a una señal de control y se mantengan fijos en esa posición. [6].

Estos servos o servomotores son muy frecuentes en Aeromodelismo y en robótica, por la capacidad que presentan para moverse a un ángulo concreto y mantenerse allí. Se suelen diseñar para que giren un ángulo proporcional a una señal PWM, de forma que su control es muy preciso. [6].

6.4. Información general de servos Dynamixel AX-12A y XL-320

Figura 6: AX-12A[7]

Baud Rate	7,843 [bps] ∼1 [Mbps]
Weight	AX-12 (53.5 [g]), AX-12+ (53.5 [g]), AX-12A (54.6 [g])
Dimensions (W x H x D)	32 X 50 X 40 [mm] 1.26 X 1.97 X 1.57 [inch]
Resolution	0.29 [°]
Running Degree	0 ∼300 [°] Endless Turn
Motor	Cored
Gear Ratio	254 : 1
Stall Torque	1.5 [N.m] (at 12 [V], 1.5 [A])
No Load Speed	59 [rev/min] (at 12V)
Operating Temperature	-5 ∼+70 [°C]
Input Voltage	9.0 ∼12.0 [V] (Recommended : 11.1V)
Command Signal	Digital Packet
Physical Connection	TTL Level Multi Drop Bus Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
ID	254 ID (0∼253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Gear Material	Engineering Plastic(Full)
Case Material	Engineering Plastic(Front, Middle, Back)

Cuadro 1: Especificaciones de servomotor[7]

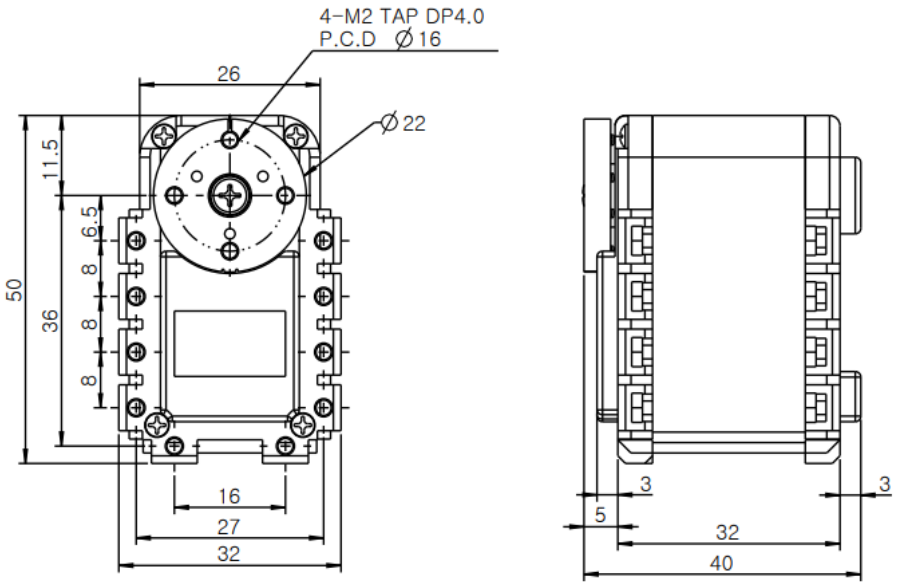


Figura 7: Dimensiones del AX-12A[7]

Para el modo de operacion Joint Mode el rango de posicion es de 0 hasta 1023 (0X3FF) y el valor por unidad es de 0.29°. En la siguiente figura se muestra de forma ilustrativa el funcionamiento del Goal Position para ambos servos.

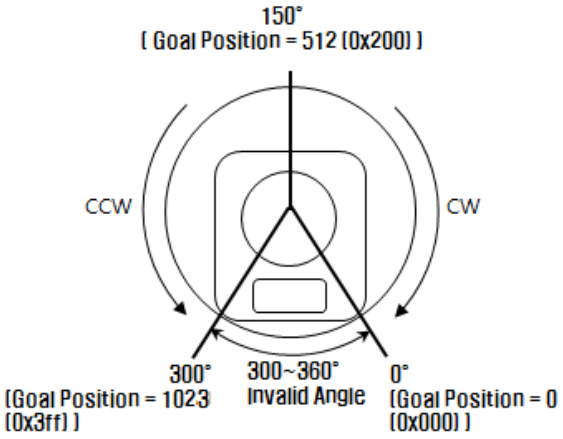


Figura 8: Limites de Goal Position del AX-12A[7]

Figura 9: XL-320[8]

Motor	Cored
Baud Rate	7343 [bps] ∼1 [Mbps]
Resolution	0.29 [°]
Operating Modes	Joint Mode (0 ∼300 [°]) Wheel Mode (Endless Turn)
Weight	16.7 [g]
Dimensions (W x H x D)	24 x 36 x 27 [mm]
Gear Ratio	238 : 1
Stall Torque	0.39 [N.m] (at 7.4 [V], 1.1 [A])
No Load Speed	114 [rev/min] (at 7.4 [V], 0.18 [A])
Operating Temperature	-5 ∼+65 [°C]
Input Voltage	6 ∼8.4 [V] (Recommended : 7.4 [V])
Command Signal	Digital Packet
Protocol Type	TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity
Physical Connection	TTL Multidrop Bus
ID	253 ID (0 ∼252)
Feedback	Position, Temperature, Load, Input Voltage, etc
Case Material	Engineering Plastic
Gear Material	Engineering Plastic

Cuadro 2: Especificaciones de servomotor[8]

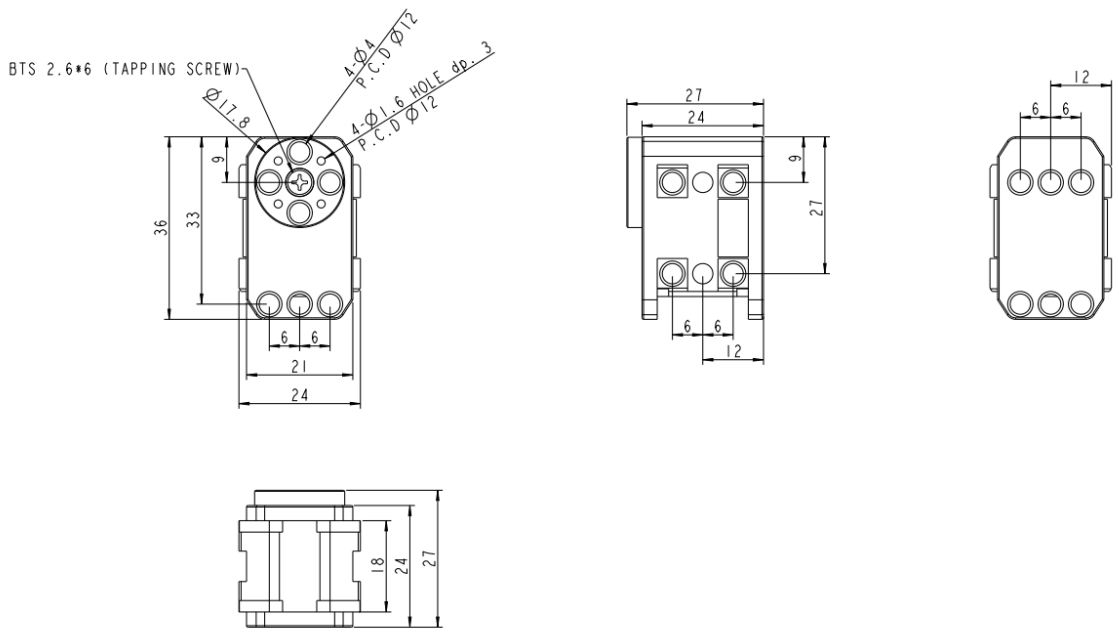


Figura 10: Dimensiones del XL-320[8]

La manera en que el servo XL-320 es conectado es a través de uniones que se denominan Rivets, en concreto Rivets RS-10 de 6 mm los cuales tienen las dimensiones exactas para la unión de la base y el eje de rotación del motor.

Figura 11: Rivets[8]

6.5. OpenCm 9.04 y 485 EXP

Figura 12: OpenCm 9.04 Type C[9]

CPU	STM32F103CB (ARM Cortex-M3)
Operation Voltage	5V ∼16V
I/O	GPIO x 26
Timer	4 (16bit)
Analog Input(ADC)	10 (12bit)
Flash	128Kb
SRAM	20Kb
Clock	72Mhz
USB	1 (2.0 Full Speed) Micro B Type
USART	3
SPI	2
I2C(TWI)	2
Debug	JTAG & SWD
DYNAMIXEL TTL BUS	4 (Max 1Mbps)
Dimensions	27mm x 66.5mm

Cuadro 3: Especificaciones de OpenCm 9.04[9]

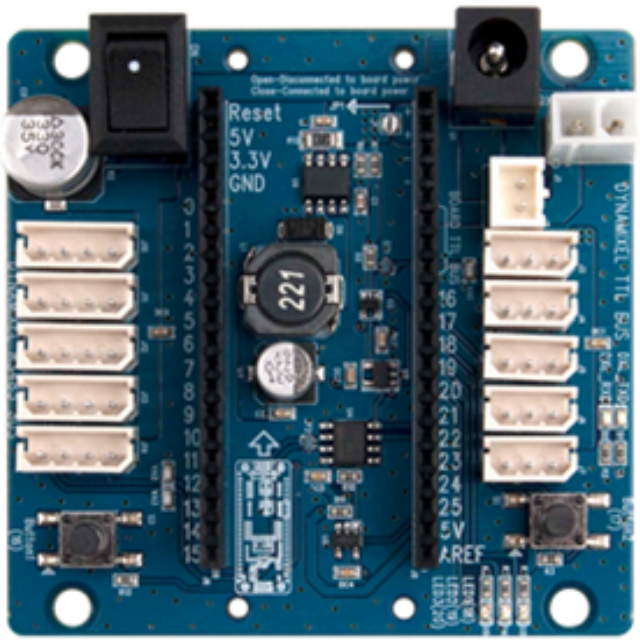


Figura 13: OpenCm 485 EXP[10]

Input voltage	5 ∼30V
Power	SMPS, LiPo, DXL PRO 24V
Power Switch	1
DYNAMIXEL Port	4Pin x 5, 3Pin x 5
Button	2
LED	5
Size	68 mm X 66.5 mm
Weight	32g
Serial3 TX	Header Pin #24
Serial3 RX	Header Pin #25
Direction Control	Header Pin #22

Cuadro 4: Especificaciones de OpenCm 485 EXP [10]

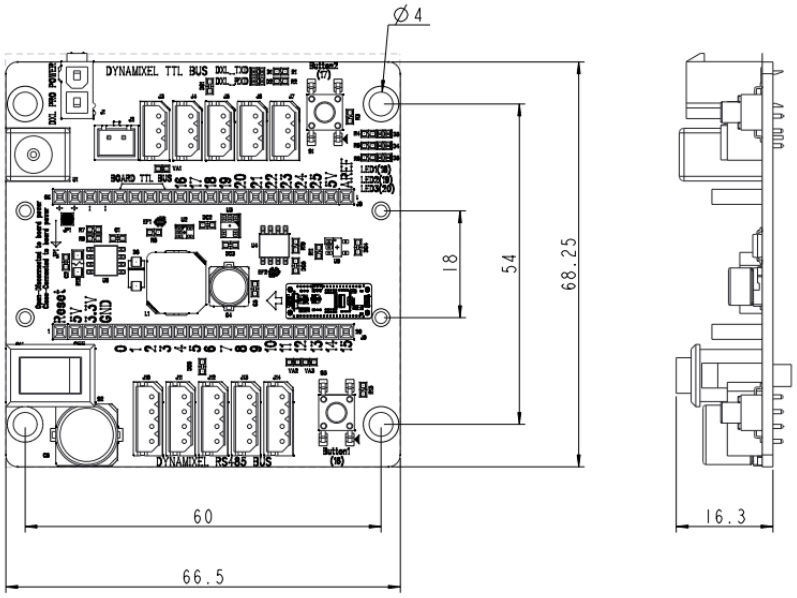


Figura 14: Dimensiones de OpenCm 485 EXP[10]

6.6. WiFi

Wi-Fi es una tecnología de red inalámbrica que permite que dispositivos tales como computadoras, dispositivos móviles y otros equipos interactúen con Internet. Permite que estos dispositivos, y muchos más, intercambien información entre sí, creando una red.[11].

La conectividad a Internet se produce a través de un enrutador inalámbrico. Cuando accede a Wi-Fi, se está conectando a un enrutador inalámbrico que permite que sus dispositivos compatibles con Wi-Fi interactúen con Internet.[11].

Este se basa en ondas de radio, las cuales operan en las frecuencias de 2.4 GHz en el estándar 802.11 n y 5 GHz en el estándar 802.11 ac. El estándar IEEE 802.11 define los protocolos que permiten la comunicación con dispositivos inalámbricos, entre los cuales se incluyen routers y puntos de acceso inalámbrico.[11].

6.7. ESP32

ESP32 es un sistema de bajo costo y bajo consumo en una serie de chips (SoC) con Wi-Fi y capacidades Bluetooth de modo dual. Cuenta con un microprocesador Tensilica Xtensa LX6 de doble núcleo o de un solo núcleo con una velocidad de reloj de hasta 240 MHz. ESP32 está altamente integrado con interruptores de antena incorporados, balun de RF, amplificador de potencia, amplificador de recepción de bajo ruido, filtros y módulos de administración de energía.[12].

Diseñado para dispositivos móviles, dispositivos electrónicos portátiles y aplicaciones IoT, ESP32 logra un consumo de energía ultra bajo a través de características de ahorro de energía que incluyen activación de reloj de resolución fina, múltiples modos de energía y escalado dinámico de energía.[12].

Este microcontrolador puede ser programado en distintos lenguajes de programación como C++ de Arduino, Micropython o C. Estos entornos de programación proveen una gran variedad de librerías que permiten el fácil acceso de todos los recursos provistos por el microcontrolador.[12].

Características:

- Procesador principal: microprocesador Tensilica Xtensa LX6 de 32 bits Núcleos: 2 o 1 (según variación) Frecuencia de reloj: hasta 240 MHz Rendimiento: hasta 600 DMIPS
- Coprocesador de potencia ultrabaja: le permite realizar conversiones de ADC, cálculos y umbrales de nivel mientras está en suspensión.
- Wi-Fi: 802.11 b/g/n /e/i (802.11n a 2,4 GHz hasta 150 Mbit/s)
- Bluetooth: v4.2 BR/EDR y Bluetooth de bajo consumo (BLE)
- ROM: 448 KiB
- SRAM: 520 KiB
- Entrada/salida de periféricos: periféricos con DMA que incluye toque capacitivo, ADC (convertidor de analógico a digital), DAC (convertidor de digital a analógico), I^2C (circuito inter integrado), UART (receptor/transmisor asíncrono universal)), CAN 2.0 (Red de área del controlador), SPI (Interfaz periférica en serie), I^2S (Sonido inter-IC integrado), RMII (Interfaz independiente de medios reducidos), PWM (modulación de ancho de pulso).
- Seguridad: Todas las funciones de seguridad estándar IEEE 802.11 son compatibles, incluidas WFA, WPA/WPA2 y WAPI Arranque seguro Cifrado flash OTP de 1024 bits, hasta 768 bits para clientes Aceleración de hardware criptográfico: AES, SHA-2, RSA, criptografía de curva elíptica (ECC), generador de números aleatorios (RNG).

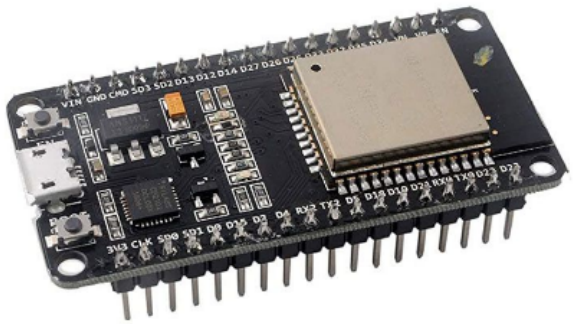


Figura 15: Microcontrolador ESP32[12]

6.8. Sistemas de captura de movimiento

Los sistemas de captura de movimiento o motion capture en inglés son un conjunto de técnicas que se usan para reproducir los movimientos de un sujeto real (o actor) en un personaje virtual. Este movimiento se traslada a un modelo digital 3D para animarlo con los movimientos registrados. [13].

Los sistemas de captura de movimiento tienen múltiples usos, pero destaca su uso en el cine y en los videojuegos. También se suele usar en el mundo del deporte y en investigación médica. Existen dos tipos de sistemas principales, los ópticos y los no ópticos, que emplean distintas tecnologías para capturar el movimiento. Cada sistema tiene sus ventajas y sus desventajas.[13].

Los sistemas de captura de movimiento ópticos emplean un conjunto de cámaras que captan el movimiento del sujeto y un software que lo interpreta. De esta forma el sujeto o actor se sitúa frente a las cámaras y estas registran, interpretan y aplican el movimiento a un modelo digital.[13].

La ventaja principal de estos sistemas es su comodidad. Al no requerir de un exoesqueleto el actor no debe "cargar" con un traje lleno de sensores que limitaría los movimientos que se puede realizar. Además, estos sistemas son notablemente más baratos que los no ópticos.[13].

Entre las desventajas que tienen los sistemas de captura ópticos destaca su baja precisión. Al funcionar mediante cámaras la precisión de los movimientos registrados se puede ver afectada por diversos factores como poca o mucha luz en el ambiente o una falta de contraste entre la vestimenta del sujeto y el fondo.[13].

Los sistemas de captura de movimiento no ópticos destacan por el uso de un traje o exoesqueleto. Se dividen en dos grupos principales los mecánicos y los electromagnéticos.En los sistemas de captura de movimiento mecánicos el sujeto cubre su cuerpo con un traje que incorpora sensores (acelerómetros y giroscopios) que registran el movimiento. En los electromagnéticos se emplea un emisor que genera un campo electromagnético y un conjunto de sensores que miden las alteraciones que realiza el sujeto en ese campo para registrar el movimiento. [13].

La principal ventaja de los sistemas de captura de movimiento no ópticos es su enorme precisión, sobre todo comparada con los ópticos, ya que están calibrados para obtener la máxima precisión. Su desventaja principal es el precio ya que son más caros que los ópticos. [13].

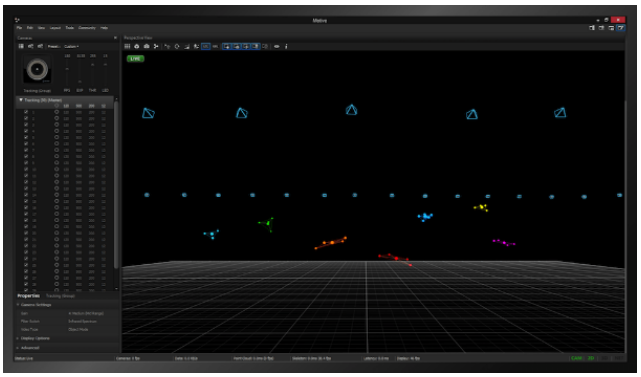


Figura 16: Software Motive aplicando el sistema de captura de movimiento[13]

6.9. Sistema OptiTrack

OptiTrack es el mayor proveedor de captura de movimiento del mundo y ofrece seguimiento óptico de alto rendimiento a los precios más asequibles de la industria. La línea de productos OptiTrack incluye software de captura de movimiento y cámaras de seguimiento de alta velocidad.[14].

Los sistemas de seguimiento en tiempo real OptiTrack son la elección mundial para el seguimiento 6DoF (6 grados de libertad) de precisión y baja latencia para robótica terrestre y aérea (UAV). Producen los sistemas de posicionamiento interior y exterior más precisos y fáciles de usar para la entrada en todos los principales sistemas de control.[14].

Los sistemas de seguimiento de drones y robots terrestres de OptiTrack producen un error de posición inferior a 0,3 mm y un error de rotación inferior a 0,05°. Varía levemente en el entorno de captura, pero a menudo es incluso mejor que esas cifras.[14].

Las cámaras de la serie Prime pueden rastrear marcadores reflectantes en el exterior, a plena luz del Sol, ideal para rastrear quadrotors en entornos brillantes, sin agregar el peso y la complejidad de los marcadores activos. Todas las cámaras OptiTrack pueden rastrear un marcador del mismo tamaño en exteriores a aproximadamente la mitad de su distancia en interiores.[14].

6.10. Cámaras Prime 41

Las cámaras Prime 41 de OptiTrack ofrecen luz estroboscópica infrarroja, amplio campo de visión y gran alcance de la cámara capaz de rastrear un marcador de captura de movimiento de 16 mm desde 100 pies de distancia. En la Figura 12 se muestran sus características.[15].

- Peso: 3,2 libras (1,45 kg)
- Lente estándar: 12 mm F1.8 (banda ancha con revestimiento AR)
- Campo de visión horizontal: 51°
- Campo de visión vertical: 51°
- Foco ajustable y f-stop
- Filtro de paso de banda de 850 nm
- Resolución: 2048 × 2048
- Tamaño de píxel: 5,5 µm × 5,5 µm
- Velocidad de fotogramas: 30–180 FPS (ajustable)
- Latencia: 5,5ms
- Tipo de obturador: global
- Velocidad de obturación: Predeterminado: 500 µs (0,5 ms) Mínimo: 10 µs (0,01 ms) Máximo: 8100 µs (8,1 ms) a 120 FPS 5300 µs (5,3 ms) a 180 FPS
- Datos: GigE (1000BASE-T)
- Sincronización de cámara: Ethernet
- Alimentación: PoE o PoE+ 1

Figura 17: Camara OptiTrack Prime 41[15]

6.11. Comunicación TCP

El Protocolo TCP/IP o Transfer Control Protocol consiste en un acuerdo estandarizado sobre el que se realiza la transmisión de datos entre los participantes de una red informática. [16].

Los programas que forman redes de datos en una red de ordenadores emplean el protocolo TCP para crear conexiones entre sí, de forma que se pueda garantizar el flujo de datos entre las partes. A través de este protocolo se asegura que los datos lleguen a su destino en el mismo orden que se transfirieron y sin errores. [16].

La inmensa mayoría de comunicaciones que se realizan en internet utilizan el protocolo TCP IP, como es el caso de navegadores, programas de intercambio de ficheros, servicios FTP y un largo etcétera. [16].

Las principales características del TCP Protocol son:

- Es un protocolo que funciona mediante la conexión mutua entre cliente y servidor.
- Ordena los segmentos provenientes del protocolo IP.
- Monitorea el flujo de los datos y permite evitar la saturación de la red.
- Entrega los datos al protocolo IP en forma de segmentos de longitud variable.
- Permite circular de forma simultánea a la información proveniente de diferentes fuentes.

Para saber cómo funciona el protocolo TCP es necesario tener en cuenta que se trata de un estándar que se basa en la transmisión bidireccional de la información. La comunicación se realizaría en base a paquetes o segmentos, que funcionarían como las unidades básicas de transmisión. Tanto la conexión entre terminales, como la transmisión de los datos, es llevada a cabo por un software TCP, el cual se ejecuta dentro del conjunto de protocolos de red del sistema operativo. [16].

Las aplicaciones de red, como los servidores o los navegadores de internet, cuentan con interfaces que permiten activar este software TCP. Para establecer la conexión, es necesario que se reconozcan dos puntos de acceso y destino (cliente y servidor). Sin embargo, no importa qué parte hace de cliente y cuál hace de servidor, Lo único que necesita el software TCP para establecer la conexión es que cada una de las partes cuente con una dirección IP y un puerto asignado. [16].

6.12. JSON

JavaScript Object Notation (JSON) es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript. Es comúnmente utilizado para transmitir datos en aplicaciones web (por ejemplo: enviar algunos datos desde el servidor al cliente, así estos datos pueden ser mostrados en páginas web, o vice versa). [17].

Es posible incluir los mismos tipos de datos básicos dentro de un JSON que en un objeto estándar de JavaScript - cadenas, números, arreglos, booleanos, y otros literales de objeto. [17].

En algunas ocasiones, se recibirá una cadena JSON sin procesar, y será necesario convertirla en un objeto. Y cuando sea necesario enviar un objeto Javascript a través de la red, será necesario convertirlo a un JSON (una cadena) antes de ser enviado. Afortunadamente, estos dos problemas son muy comunes en el desarrollo web por lo que un objeto JSON integrado está disponible en los navegadores, que contiene los siguientes dos métodos:

parse(): Acepta una cadena JSON como parámetro, y devuelve el objeto JavaScript correspondiente.

stringify(): Acepta un objeto como parámetro, y devuelve la forma de cadena JSON equivalente. [17].

```
{
  "name": "Molecule Man",
  "age": 29,
  "secretIdentity": "Dan Jukes",
  "powers": [
    "Radiation resistance",
    "Turning tiny",
    "Radiation blast"
  ]
},
```

Figura 18: Ejemplo JSON[17]

6.13. Librerías de Arduino

El entorno Arduino se puede ampliar mediante el uso de bibliotecas, al igual que la mayoría de las plataformas de programación. Las bibliotecas brindan funcionalidad adicional para usar en bocetos, por ejemplo, trabajar con hardware o manipular datos. A continuación se muestran las librerías a utilizar. [18].

- Arduino WiFi [19].
- Arduino JSON [20].
- DynamixelWorkbench [21].

6.14. Multi-threading

Para aumentar la velocidad del núcleo del procesador sin tener que cambiar la frecuencia del reloj, el *multithreading* permite a la CPU procesar varias tareas simultáneamente. Siendo más precisos: se procesan varios hilos al mismo tiempo. Un hilo puede entenderse como una hebra de un proceso. Los programas pueden dividirse en procesos y éstos, a su vez, en hilos individuales. Cada proceso consta de al menos un hilo. [22].

Los procesos suelen procesarse de forma secuencial, es decir, un proceso tras otro. Esto no es óptimo, porque de esta manera las tareas tediosas bloquean el hardware. Si se necesita otro proceso de forma espontánea, este tiene que esperar su turno en la cola. Con el multithreading se procesan varios hilos de forma simultánea. Sin embargo, esta afirmación es solo parcialmente correcta: la simultaneidad de verdad solo puede garantizarse en raras ocasiones, aunque puede conseguirse en algunos casos. [22].

De igual forma, la llamada pseudosimultaneidad también proporciona un aumento del rendimiento: el sistema organiza y calcula los hilos de forma tan inteligente que el usuario tiene la impresión de estar procesando simultáneamente. Esta forma de simultaneidad no debe confundirse con las capacidades de las CPU multinúcleo, en el que, si el sistema tiene varios microprocesadores, también se procesan varios procesos de forma simultánea. [22].

Para que el multithreading se utilice eficazmente, el software debe estar preparado para ello. Si los desarrolladores no dividen (o no son capaces de dividir) sus programas en varios hilos, la tecnología no funcionará. [22].

6.15. Comunicación Serial Asíncrono - UART

La electrónica integrada se conforma con circuitos interconectados (procesadores u otros circuitos integrados) para crear un sistema en el que están repartidas las funciones. Para que esos circuitos individuales intercambien su información, deben compartir un protocolo de comunicación común. Se han definido muchos protocolos de comunicación para lograr este intercambio de datos y, esencialmente, cada uno puede ubicarse en una de dos categorías: Paralelo o Serie. [23].

La comunicación en serie transmite sus datos un bit a la vez. Estas interfaces pueden operar con tan solo un cable, por lo general nunca más de cuatro. El protocolo serie asincrónico tiene una serie de reglas integradas: mecanismos que ayudan a garantizar transferencias de datos sólidas y sin errores. Estos mecanismos, que obtenemos para evitar la señal del reloj externo, son: Bits de datos, Bits de sincronización, Bits de paridad, Velocidad en baudios. [23].

El protocolo es altamente configurable. La parte crítica es asegurarse de que ambos dispositivos en una línea serie estén configurados para usar exactamente los mismos protocolos. Las velocidades en baudios pueden ser casi cualquier valor dentro de lo que permite el hardware. El único requisito es que ambos dispositivos funcionen a la misma velocidad. Una de las velocidades en baudios más comunes, especialmente para cosas simples donde la velocidad no es crítica, es de 9600 bps. Otras velocidades en baudios «estándar» son 1200, 2400, 4800, 19200, 38400, 57600 y 115200. Un bus serie consta de solo dos cables, uno para enviar datos y otro para recibir. Entonces, los dispositivos serie deben tener dos pines serie: el receptor: RX y el transmisor: TX. [23].

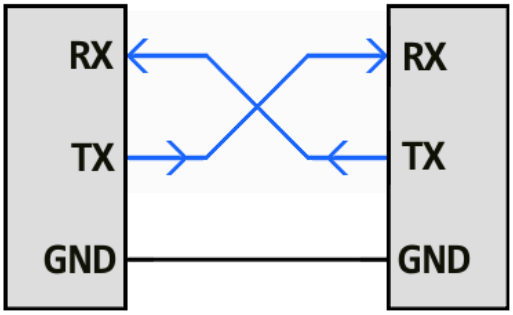


Figura 19: Conexión UART[23]

Los UART más avanzados pueden enviar los datos que reciben a un archivo de memoria de respaldo, llamado búfer, donde pueden permanecer hasta que el microcontrolador vaya a buscarlos. Los UART generalmente publicarán sus datos almacenados en un búfer con un sistema de “el primero que entra es el primero que sale” (First In First Out = FIFO). Los búfer pueden tener apenas unos pocos bits, o pueden ser de gran tamaño, como miles de bytes. [23].

CAPÍTULO 7

Diseño mecánico y manufactura del hexápodo

Para el diseño de la plataforma primero se deben establecer las partes que tendrá el robot, en este caso al ser un robot hexápodo tendrá únicamente torso y 6 extremidades, las cuales se deben diseñar con en fin de que se acoplen de manera perfecta a los motores Dynamixel y a la electrónica interna. A continuación se presentan las dos secciones previamente mencionadas en donde se sigue un procedimiento de diseño utilizando Autodesk Inventor [5] hasta su manufactura y ensamble.

7.1. Torso

En el primer paso para diseñar el torso del robot se tenía que definir la geometría óptima, misma que no debía interferir en el movimiento de las extremidades pero lo suficientemente grande para alojar las PCB: de la electrónica, cables y las baterías. Por lo tanto, se tomó la decisión de realizar un diseño de rectángulo de doble nivel, lo que significa que dentro del torso estarían las PCB: y los cables, y encima de la estructura se colocarían las baterías para fácil acceso a ellas. Por otro lado, se colocaron en la estructura del torso 6 motores AX-12A que afortunadamente poseen un montaje de doble nivel y gracias a eso fue posible mantener unida la estructura. En la Figura 20 se muestra el diseño del primer nivel, en donde se debe tomar en cuenta las dimensiones de los 6 motores mencionados, la placa OpenCM 485 EXP, la PCB del ESP32 y una extensión para el cableado de los motores.

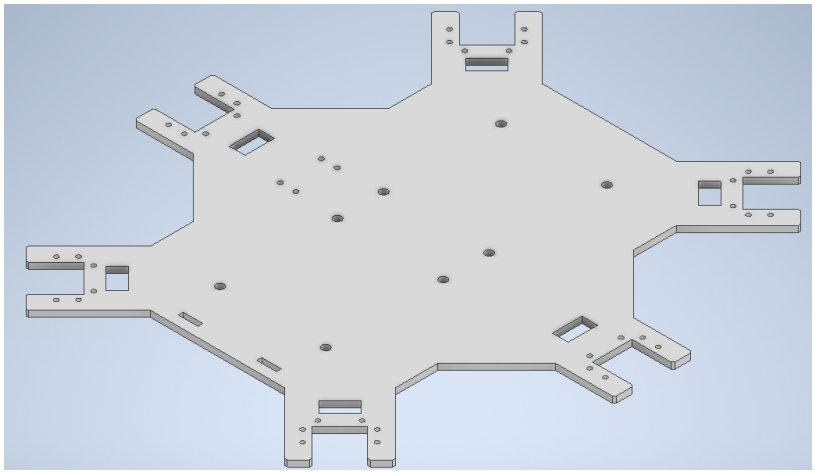


Figura 20: Parte baja del torso.

Como se puede observar en la Figura 20, se diseñó una estructura con 6 montajes para los motores, los cuales tienen la forma para colocar 6 tornillos por motor y una abertura para el cable. Además, se colocaron varios agujeros en la parte central con las medidas de la placa OpenCM 485 EXP y una placa de extensión que se explicará de mejor manera en el capítulo de electrónica interna. En la Figura 21 se observan las dimensiones que se utilizaron para esta parte y se define el tamaño del torso de 125mm x 200mm.

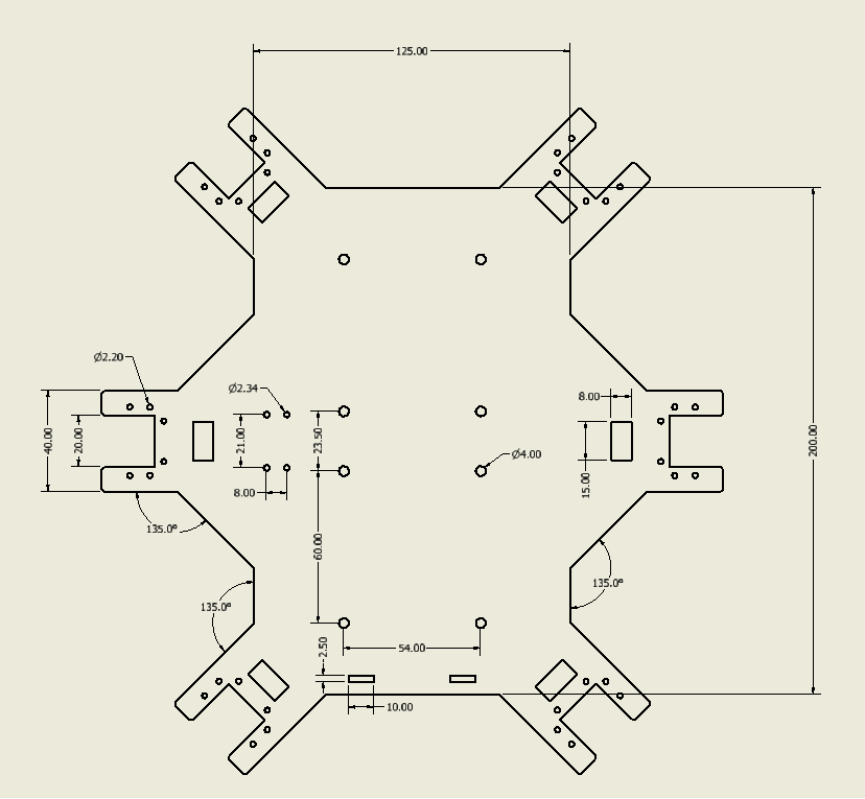


Figura 21: Dimensiones generales de la parte baja del torso.

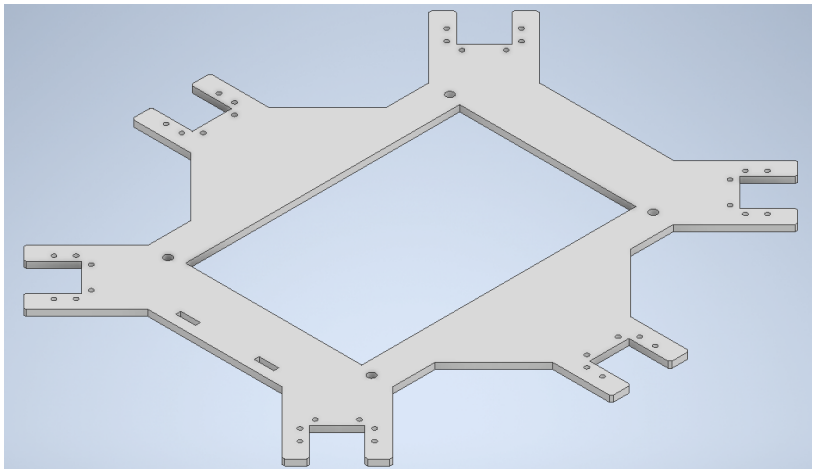


Figura 22: Parte superior del torso.

Para la estructura superior se deben colocar las mismas dimensiones pero en la parte central se colocó un agujero cuadrado con el fin de poder manipular la electrónica con facilidad. Como se mencionó anteriormente, se tenía planificado que las baterías fueran colocadas encima de esta parte, por lo que se agregaron 4 agujeros para una cubierta removible.

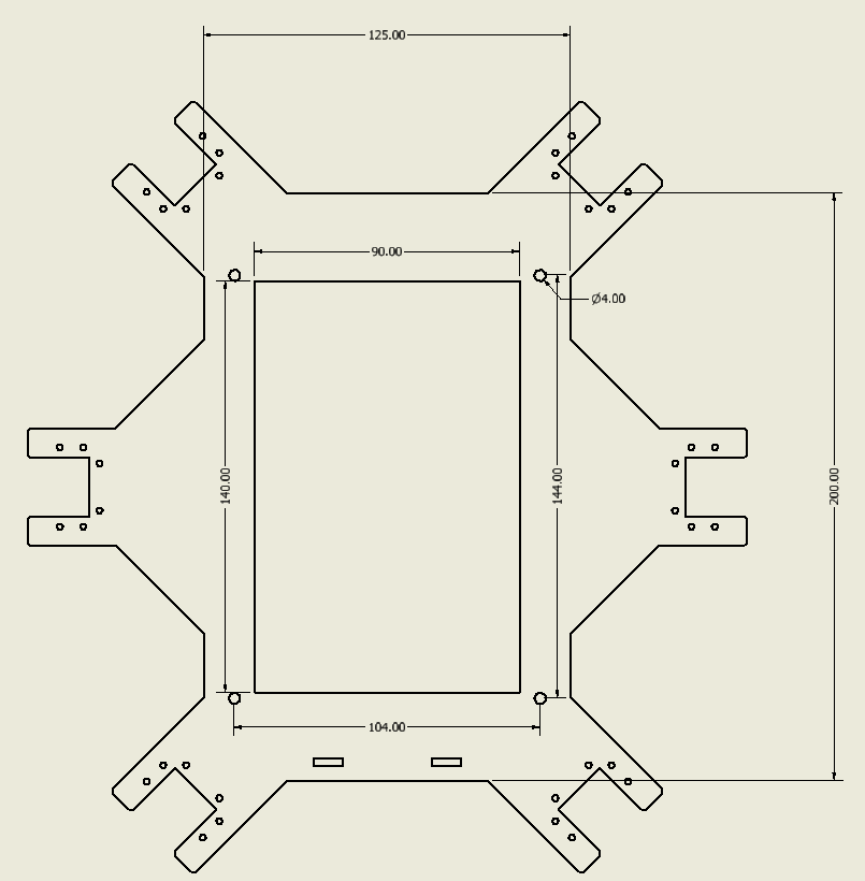


Figura 23: Dimensiones generales de la parte superior del torso.

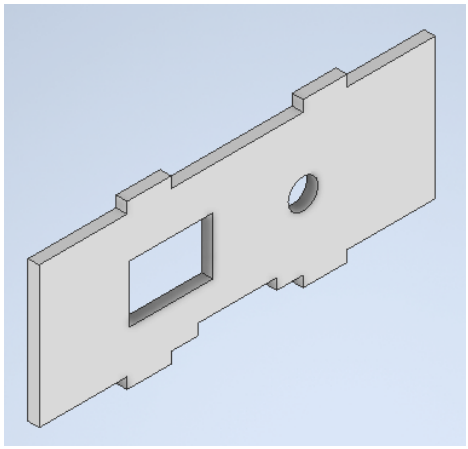


Figura 24: Soporte para interruptor de encendido y botón de comando.

En la Figura 24 se observa una parte extra que se colocó en el torso y el objetivo de esta fue agregarle un interruptor de encendido y un botón para algún comando extra que se necesite, para que no fuera necesario abrir la cubierta o conectar las baterías para encender o apagar al robot.

Por último, se tuvo la decisión de que estas partes fueran de acrílico, ya que las partes deben ser delgadas pero lo suficientemente fuertes para soportar todo el peso. El acrílico seleccionado es de 2.5mm de grosor, el cual es perfecto para utilizar con los tornillos de fábrica que tienen los motores AX-12A debido a que estos no son de largo suficiente para cualquier material y se realizó la manufactura a través de corte láser con equipo de la Universidad. En la Figura 25 se muestra el torso ensamblado junto con los servos. La plataforma presentó una altura interna de 32 mm, lo que da el suficiente espacio para colocar la electrónica.

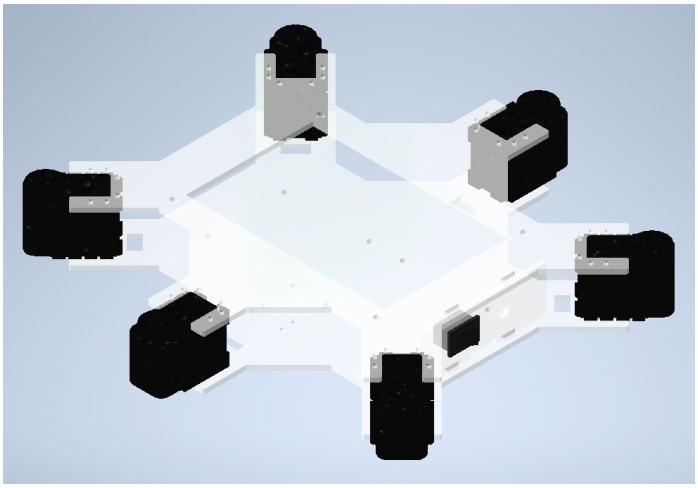


Figura 25: Torso completo.

7.2. Extremidades

Para comenzar con las extremidades se debía realizar una parte que uniera los servos AX-12A con los XL-320, por lo que se diseñó un marco para el XL-320 similar al marco FP04-F2 de Robotis [24] compatible con el AX-12A como se observa en la Figura 26. Este posee agujeros de 4mm de diámetro para ensamblarlos con ayuda de Rivets: [25] específicos para el servo.

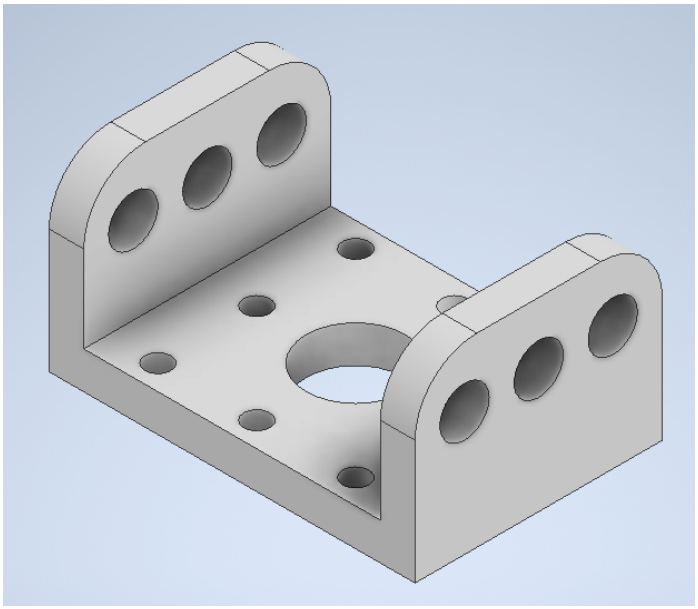


Figura 26: Marco de XL-320.

Luego de esto fue el momento de unir dos XL-320 y simplemente se diseñó una pieza plana compatible con la estructura del servo y su eje de rotación. De esta parte se utilizaron dos piezas por extremidad ya que irán colocadas a los lados de los servos, junto con un pin de seguridad mostrado en la Figura 27 el cual sirve para evitar movimientos no deseados y mantener fijada la estructura.

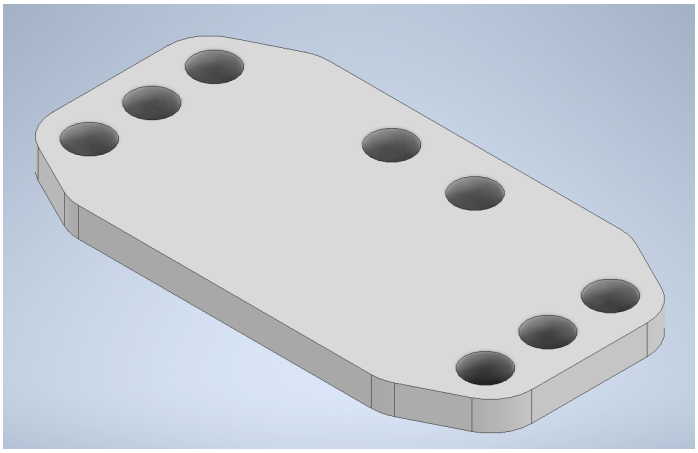


Figura 27: Soporte para motores XL-320.

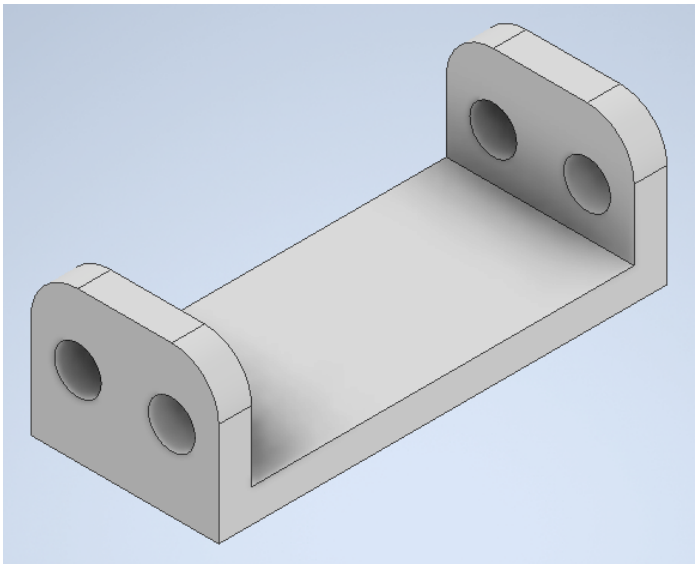


Figura 28: Unión de rigidez para soportes.

La siguiente parte fue la más complicada, ya que se debían tomar varias consideraciones. Primero debía ser lo suficientemente grande para mantener elevado al robot con una posición de 90 grados, debía tener las dimensiones para colocar el eje de movimiento del servo y ser eficiente en el uso del material en su manufactura. Tomando en cuenta esto se realizó el diseño que se puede observar en la Figura 29, el cual posee una estructura rígida y al igual que las partes anteriores se conecta el servo XL-320 utilizando los Rivets: especiales de 6 mm.

Para la manufactura de todas las partes de las extremidades se utilizó el método de impresión 3D, el cual entrega piezas resistentes y precisas en dimensiones. Todos los diseños fueron pensados para este tipo de manufactura ya que no tienen formas tan complicadas, sino simples pero funcionales.

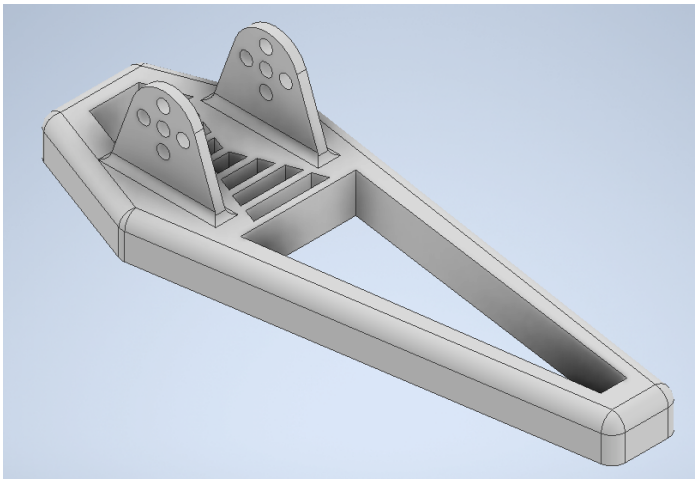


Figura 29: Soporte de la extremidad

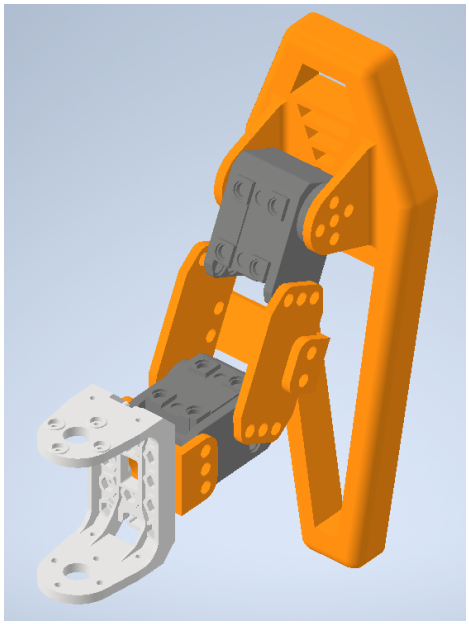


Figura 30: Extremidad completa.

En la Figura 30 se muestra el ensamble de todas las partes formando así la extremidad completa. Se puede observar las partes impresas en 3D en naranja junto con los servos y el marco del AX-12A. En la Figura 31 se tiene la extremidad física ya ensamblada en donde se utilizaron 18 Rivets: por extremidad para su ensamble y se colocó un material anti-fricción al extremo de las patas para mayor firmeza en el movimiento.

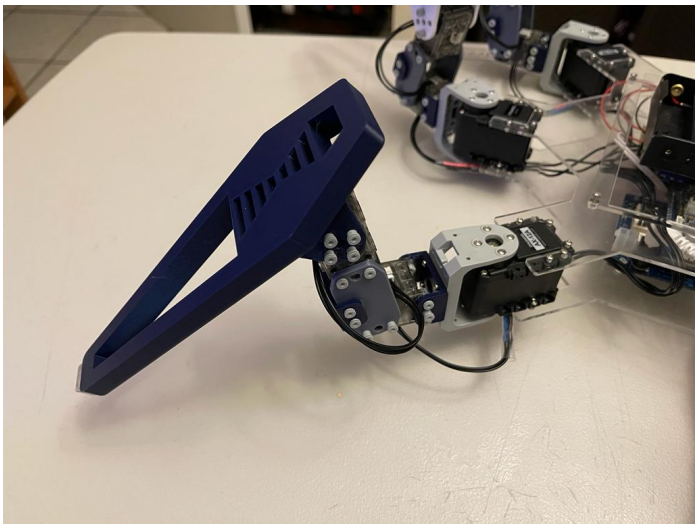


Figura 31: Extremidad física

.

Figura 32: Diseño mecánico completo.

Finalmente, se obtuvo el diseño completo de la plataforma mostrado en la Figura 32, la cual posee 6 servos AX-12A, 12 servos XL-320, 5 piezas impresas en 3D por extremidad y 4 partes de acrílico incluyendo la cubierta que se muestra en la Figura 33, la cual tiene aberturas para los cables de las baterías.

Cabe mencionar que cada extremidad del robot posee 3 revolutas (2 horizontales y 1 vertical) tal como cualquier otro hexápodo, por lo que la plataforma cuenta con 18 grados de libertad para utilizarlos en su movimiento.

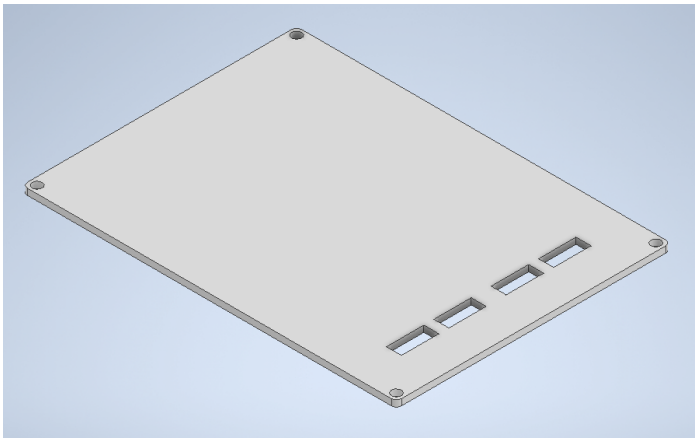


Figura 33: Cubierta de la parte superior del torso

En la Figura 34 se muestra el hexápodo ya ensamblado físicamente. Este diseño cuenta con un armado y desarmado fácil, ya que se utilizó en un su mayoría tornillos y Rivets: para el ensamble y no algún tipo de adhesivo o pegamento que pueda interferir a la hora del desarmado.

Figura 34: Diseño final de hexápodo.

CAPÍTULO 8

Electrónica interna

8.1. OpenCM 9.04 y 485 EXP

En el diseño mecánico se estableció que para cada extremidad se tendrá primero 1 AX-12A y luego 2 XL-320, y para que el cableado de los servos fuera eficiente debía ser prácticamente un solo cable. Afortunadamente, ambos tipos de servos funcionan con una conexión Daisy Chain:, el problema fue que los extremos de los cables para ambos servos era diferente, por lo que se realizó un adaptador para conectar el AX-12A con el primer XL-320. A partir de eso se obtuvo otro problema, el OpenCM 9.04 no posee el conector para el primer servo, y la solución fue adquirir su expansión 485 que sí la tiene junto con una expansión de conectores 3P ya que esta placa solo tiene 5 salidas y se necesitan 6.

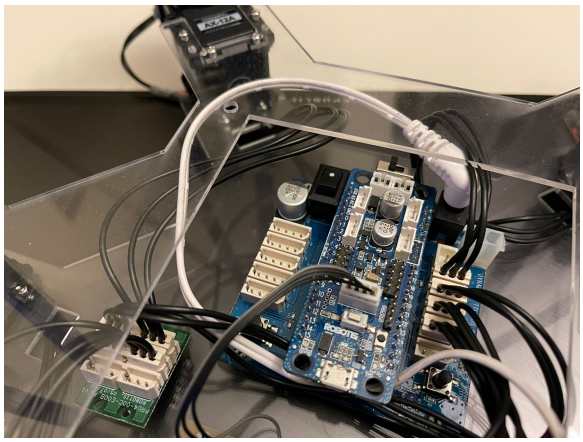


Figura 35: OpenCM 9.04 montado en la EXP 485 junto a la expansión de conectores.

Con esas conexiones fue posible mover cada servo por separado, utilizando como fuente de voltaje dos Baterías 18650: de 3.7V , teniendo en total un voltaje de operación de 7.4V y utilizando el programa Dynamixel Wizard 2.0, con el cual fue posible cambiar los identificadores de cada servo y hacer pruebas visuales de movimiento. Por otro lado, se realizó una actualización de firmware para todos los servos para su correcto funcionamiento.

Debido a que el robot debía recibir comandos de movimiento a través de una conexión Wi-Fi era necesario la conexión hacia otra placa que tuviera el microcontrolador ESP32. Esto se realizó por medio del conector serial 2 del OpenCM 9.04, como se muestra en la Figura 36. A partir de esta conexión se conectaron a los pines dl puerto serial 2 correspondientes del ESP32 y se obtuvo la fuente de voltaje del ESP32 directamente del OpenCM 9.04, en otras palabras, el ESP32 trabaja con un voltaje de 3.3V.

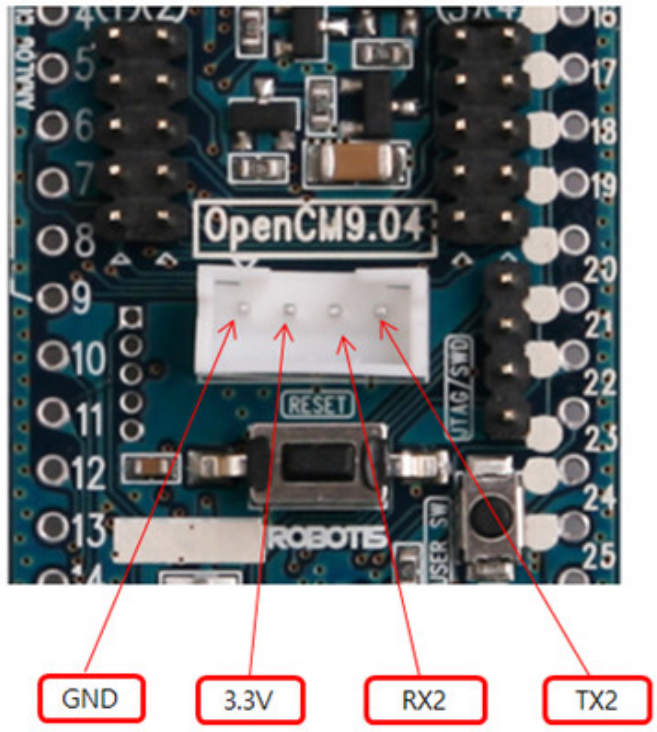


Figura 36: Conector de Serial 2 para OpenCM 9.04.

8.2. PCB de ESP32

La PCB: del ESP32 debía tener entradas y salidas externas a la placa, es por eso que se conectó por medio de conectores del tipo bornera en caso de ser necesario remover las conexiones. Como segunda restricción, el tamaño de la placa debía ser idéntico a de la placa de OpenCM 485 EXP para colocarla en el diseño mecánico. El modo de conexión para el ESP32 se hizo por medio de pines *headers* hembra y, dado que es un ESP32 Devkit V1[12], cuenta con 19 pines de ambos lados.

Con todo lo anterior establecido se procedió a hacer el esquemático del PCB:, que se observa en la Figura 37. Este posee las conexiones desde los *headers* hasta las borneras correspondientes a la conexión serial 2 y un botón de comando conectado al pin digital dos del ESP32.

Adicionalmente, la placa tiene conexiones ajenas al ESP32, las cuales fueron: dos conexiones de fuente de voltaje de 7.4V (conexiones en paralelo) en caso fuera necesario agregar 2 baterías extra para aumentar la corriente, salidas a un interruptor de encendido y una salida de voltaje que va conectada directamente al OpenCM 485 EXP. Como se ve en la Figura 37, lo relacionado con el ESP32 está justo en el centro, mientras que las conexiones para el OpenCM son lo que rodea el esquemático.

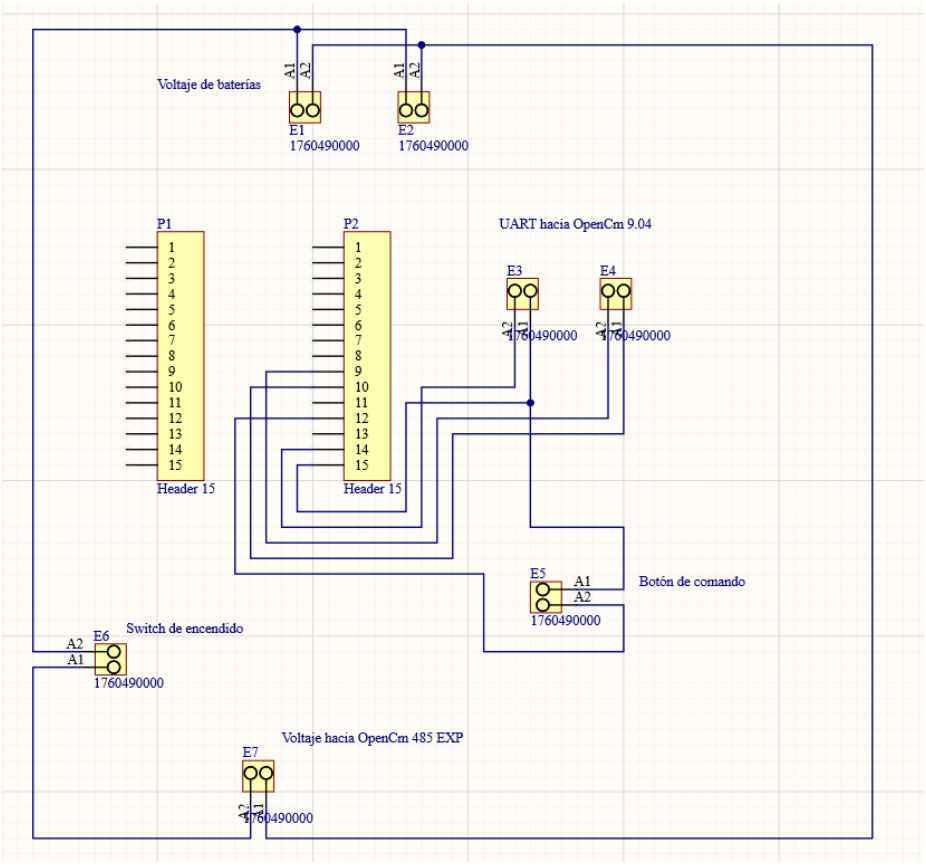


Figura 37: Esquemático de PCB.

Para obtener una PCB funcional se utilizó una calculadora de ancho de tracks de tipo ANSI, en donde se ingresaron valores deseados de temperatura, corriente, voltaje, grosor de cobre, largo del track para la placa. Como se observa en la Figura 38 se obtuvo un ancho de track de 74 mils y un espacio entre tracks de 26 mils.

ANSI PCB TRACE WIDTH CALCULATOR

Input Data			Results Data				
				Internal Traces		External Traces	
Field	Value	Units	Trace Data	Value	Units	Value	Units
Current (max. 35A)	3	Amps	Required Trace Width	73.7	mil	28.33	mil
Temperature Rise (max. 100°C)	30	°C	Cross-section Area	99.04	mil²	38.07	mil²
Cu thickness	1	oz/ft²	Resistance	0.02	Ω Ohms	0.06	Ω Ohms
Ambient Temperature	25	°C	Voltage Drop	0.07	Volts	0.18	Volts
Conductor Length	3	inches	Loss	0.21	Watts	0.55	Watts
Peak Voltage	12	Volts	Required Track Clearance	25.4	mil		

Figura 38: ANSI PCB Trace Width Calculator

Para posicionar las borneras en el diseño de la PCB: se tomó en cuenta que la conexión serial iba por el lado superior (GND y 3.3V - E3; TX y RX - E4), la conexión del interruptor y el botón iban del lado inferior (interruptor - E6; Botón - E5), entrada de voltaje de baterías (E1 y E2, siendo positivo A1) y por último la salida de voltaje hacia el OpenCM 485 EXP (E7, siendo A1 como GND). Para el montaje sobre el diseño mecánico se utilizaron Rivets: para sujetar tanto la PCB: del ESP32 como el OpenCM 485 EXP. Como se muestra a continuación en la Figura 39, se ve la vista general de la electrónica completa y montada. Las conexiones de las extremidades del lado derecho del robot están conectadas directamente en la placa, sin embargo, se utilizó la expansión de conectores 3P para la conexión de las extremidades del lado izquierdo del robot.

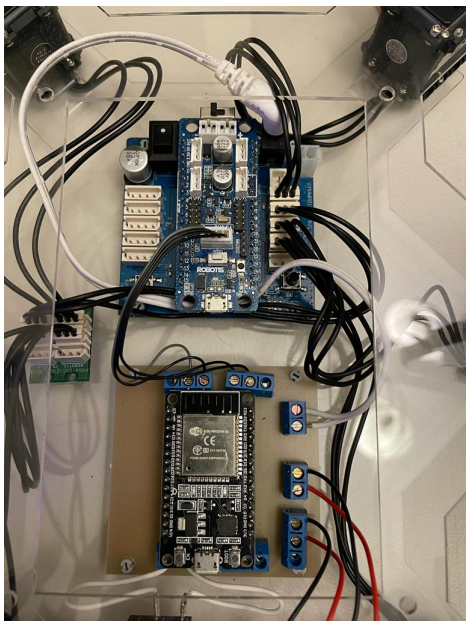


Figura 39: Vista general de electrónica completa.

CAPÍTULO 9

Derivación cinemática

Luego de obtener la plataforma robótica armada, el siguiente paso es hacer una derivación cinemática, la cual se utilizara para realizar un modelo digital del robot.

El primer paso para modelar la plataforma es establecer un torso rectangular en donde se colocan 4 Revolutas: en las esquinas para simplificar el modelo. En las Figuras 40 y 41 se muestran las medidas de ancho y largo para el modelo.

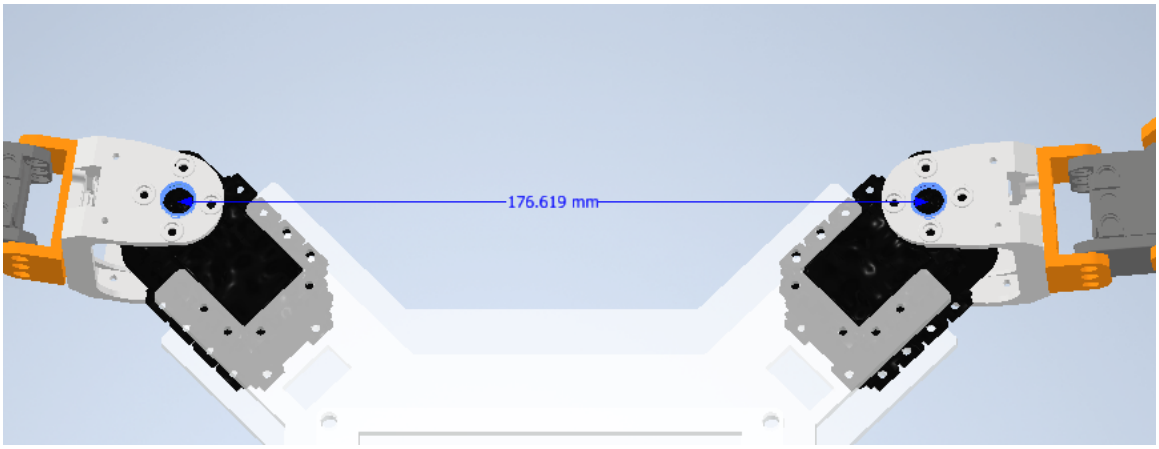


Figura 40: Medida de ancho para el torso modelado (W = 176.619 mm)

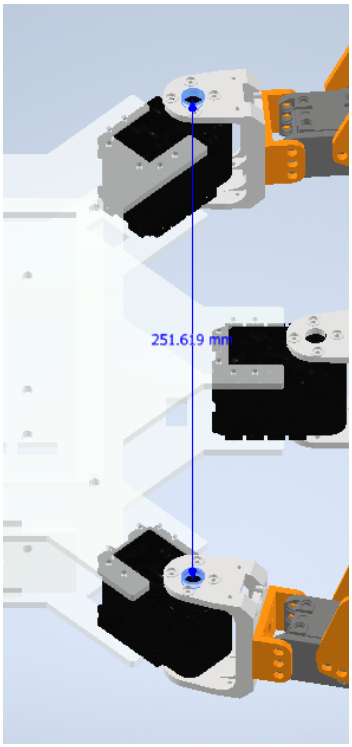


Figura 41: Medida de largo para el torso modelado (L = 251.619 mm)

Posteriormente se define el marco de referencia base que es el centro del cuerpo del robot y se realizan las mediciones desde la referencia hasta las Revolutas: de los servos AX-12A de las extremidades del medio. En la Figura 42 se observa la medida previamente mencionada, la cual es igual en ambos lados del robot.

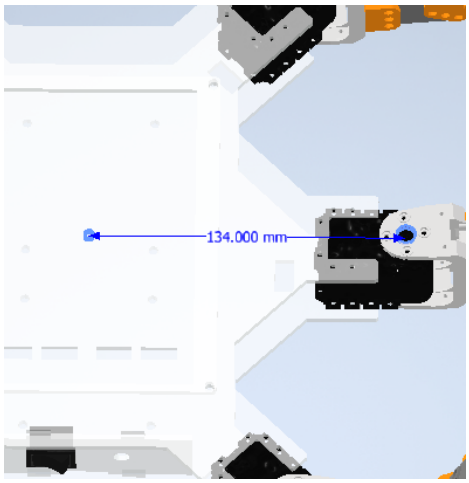


Figura 42: Medida de referencia a primera revoluta de extremidad media (R = 134 mm)

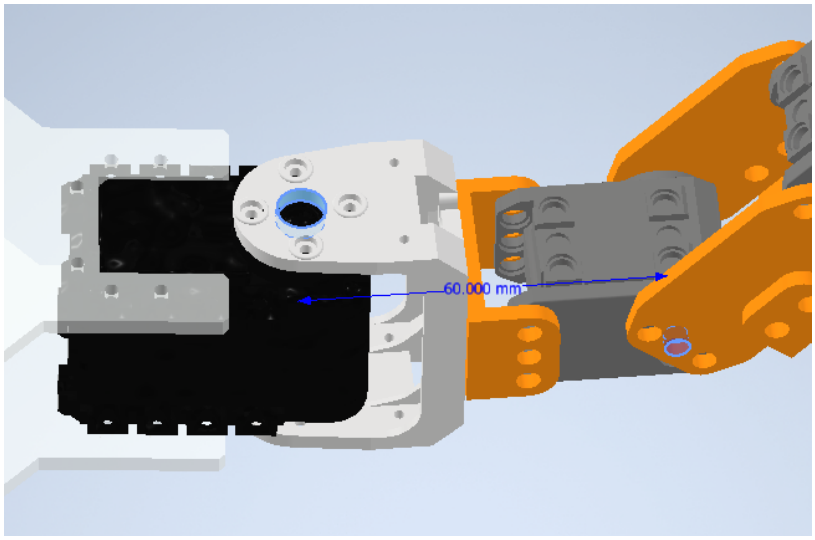


Figura 43: Medida de primera a segunda revoluta (L1 = 60 mm)

Las medidas de las Figuras 43, 44 y 45 son las distancias entre Revolutas: (eslabones) para todas las extremidades, ya que cada una de estas son idénticas. Gracias a esto es posible hacer un modelo individual para cada extremidad y solo replicarla en su debida posición y orientación.

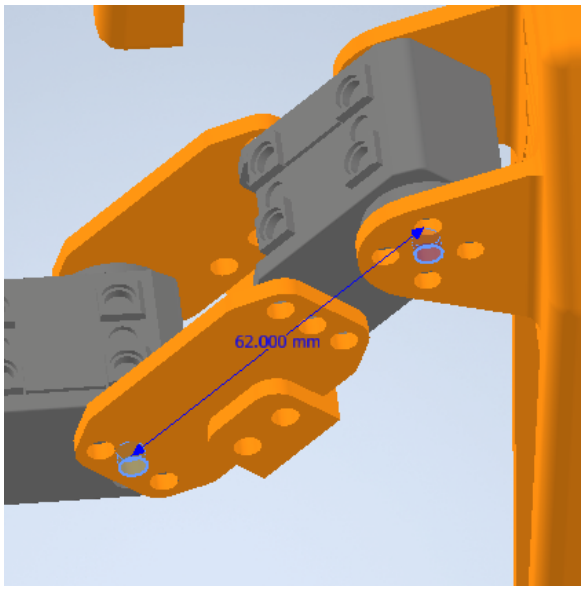


Figura 44: Medida de segunda a tercera revoluta (L2 = 62 mm)

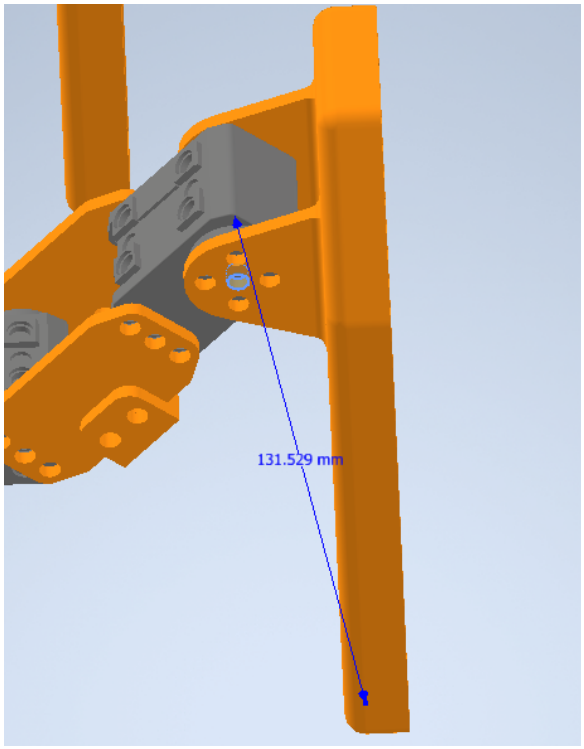


Figura 45: Medida de tercera revoluta al extremo de la pata (L3 = 131.529 mm)

Con las medidas establecidas se procede a realizar una derivación cinemática del robot siguiendo como modelo el ejemplo de la sección 7.5.2 del libro Robotics, Vision and Control de Peter Corke [26]. El cual indica que se debe determinar los Parámetros de Denavit-Hartenberg: y establecer los ejes de referencia. Se define el eje X positivo hacia al frente del robot, el eje Y positivo hacia la derecha del robot y por ultimo el eje Z positivo hacia arriba del robot.

Las secuencias de movimiento según el modelo físico del robot serian las siguientes:

- Rz(q1): La primera revoluta de la extremidad, eje de AX-12A con rotación en el eje Z
- Ty(L1): Traslación en eje Y desde la primera revoluta hasta el eje del primer XL-320
- Rx(q2): Rotación en eje X de la revoluta del primer XL-320
- Ty(L2): Otra traslación en eje Y desde la revoluta del primer al segundo XL-320
- Rx(q3): Rotación en eje X de la revoluta del segundo XL-320
- Tz(L3): Traslación en eje Z desde la última revoluta hasta el final de la extremidad

La secuencia general de cada extremidad se establece como:

- Rz(q1)Ty(L1)Rx(q2)Ty(L2)Rx(q3)Tz(L3)

Esta secuencia puede ser aplicada en el Robotics Toolbox de Peter Corke [26] del programa Matlab:, el cual permite obtener los Parámetros de Denavit-Hartenberg:. Al tener la extremidad ya modelada se puede aplicar una cinemática directa o en otras palabras, hacer un cambio en la orientación de las revolutas para simular un movimiento en los ejes de los servos y observar un cambio de posición del efector final. En la Figura 46 se observa el modelo de una extremidad con posiciones de revolutas en valor cero.

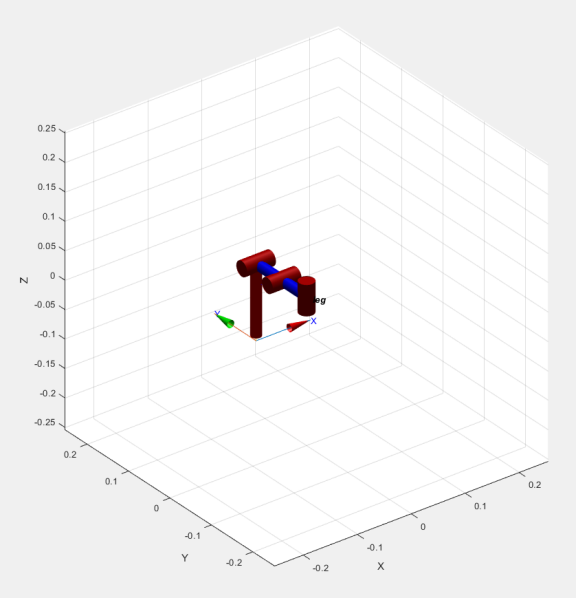


Figura 46: Simulación de extremidad individual en Matlab

Como se mencionó anteriormente, para simular la plataforma robótica era necesario modelar una extremidad con sus respectivos movimientos, replicarla y colocar cada extremidad en su posición, tomando en cuenta las medidas tomadas para el torso establecido. En la Figura 47 se muestra la simulación de la plataforma completa en donde se observa que se agregaron 6 extremidades y se colocaron según su posición real, 4 en los extremos del torso y 2 más colocados al costado según la distancia con la referencia base. Cabe mencionar que cada revoluta está modelada en posición 0 y posteriormente se cambiarán las posiciones utilizando cinemática directa e inversa utilizando el Toolbox previamente mencionado.

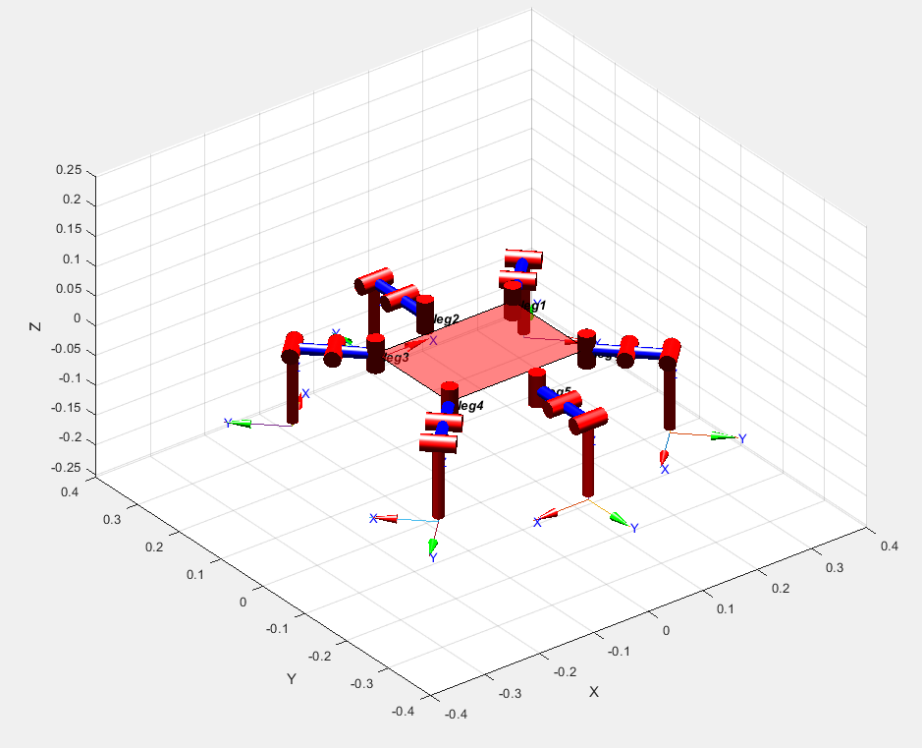


Figura 47: Simulación de plataforma robótica en Matlab

CAPÍTULO 10

Prototipo V2

Ya que se tenía la plataforma robótica armada se realizaron pruebas de movimiento con valores arbitrarios en donde se tenían secuencias para avanzar, retroceder y girar hacia los lados. Esto fue posible gracias a una conexión entre el ESP32 y Adafuit IO:, junto con una interfaz de usuario. Sin embargo, debido al peso del robot, los servos XL-320 no contaban con el suficiente torque para un movimiento en donde hay 3 patas como soporte.

Por lo que se tuvo la decisión de cambiar 6 servos XL-320 por otros 6 servos AX-12A, de los cuales poseen un mayor torque. Al realizar este cambio se tuvo que remover del diseño el soporte para los servos XL-320, figura 27 y la unión de rigidez para los soportes, figura 28. Por otro lado se agrego el frame FP04-F3 [27] del AX-12A.

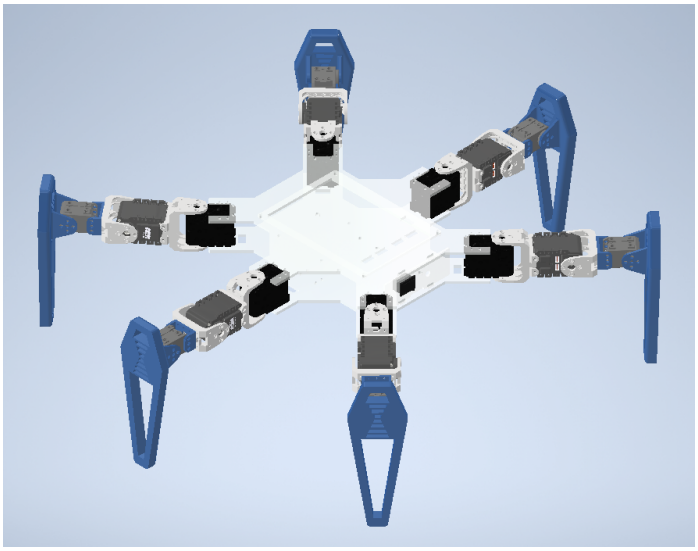


Figura 48: Segundo prototipo del hexápodo

En la Figura 48 se observa la segunda versión de la plataforma robótica, en la cual se colocaron los AX-12A en la misma orientación que los servos reemplazados y con los mismos identificadores para que no afecte la programación, además al realizar el cambio se modificaron algunos valores de la derivación cinemática que se mencionan a continuación.

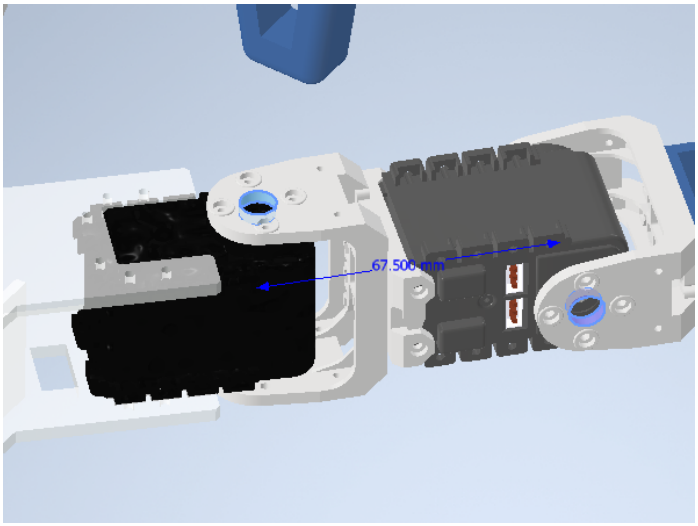


Figura 49: Medida de primera a segunda revoluta - V2

Las medidas de ancho y largo del torso (W y L), de referencia a primera revoluta (R) y de tercera revoluta al extremo de la pata (L3) se mantienen iguales aunque las medidas de primera a segunda revoluta (L1) y segunda a tercera revoluta (L2) cambian. Como se observa en la Figura 49, L1 es de 67.5 mm y en la Figura 50, L2 es de 60.0 mm.

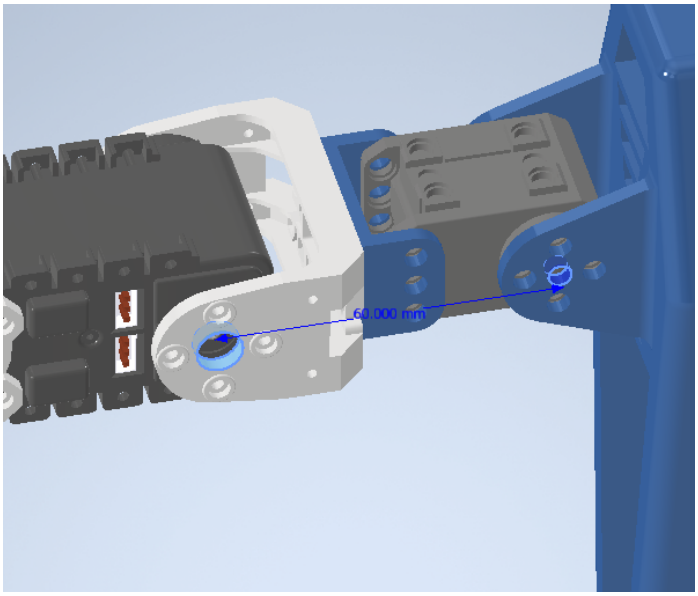


Figura 50: Medida de segunda a tercera revoluta - V2

CAPÍTULO 11

Implementación de Robotat

Al tener la versión final del robot se realizó la implementación del ecosistema Robotat, el cual constaba de tres pasos: La conexión cliente-servidor entre el ESP32 y el sistema Optitrack, del cual se debían recibir los datos de posición de un marcador, con el objetivo de guardar esos datos en el ESP32. El segundo paso era la conexión cliente-servidor entre el ESP32 y Matlab, del cual se debía mandar las secuencias de movimiento. El tercer paso era la comunicación entre el OpenCm 9.04 y el ESP32, en donde fue necesario el uso de librerías JSON y una conexión UART.

Al finalizar estas tareas fue necesario unir los tres códigos en uno solo y obtener una conexión bi-direccional entre el Optitrack, Matlab, ESP32 y una conexión simple para el OpenCm 9.04. Más adelante se explica a detalle la realización de estos pasos. A continuación, en la Figura 51 se muestran gráficamente las conexiones de la implementación.

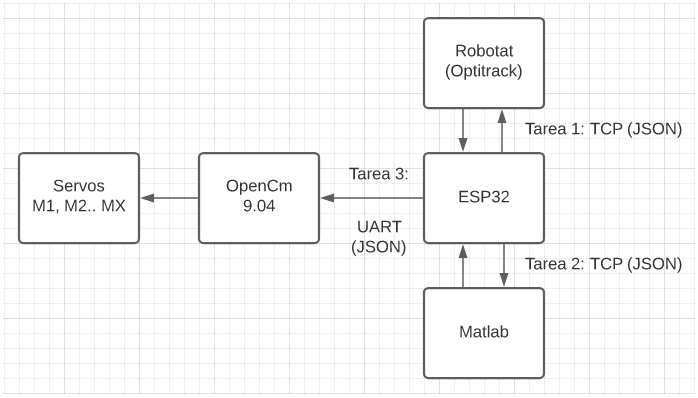


Figura 51: Metodología de implementación de Robotat

11.1. Optitrack

Para la parte de la conexión con Optitrack se estableció el ESP32 como cliente ya que el sistema Robotat posee una red local y un servidor que envía los datos del Optitrack en formato JSON y se comunica utilizando el protocolo TCP. Para comunicarse con el sistema se debe conectar por medio de la IP: 192.168.50.200, que al realizar la comunicación envía un mensaje de bienvenida y espera el mensaje del cliente para luego enviar la información del marcador. El mensaje enviado debe ser algún identificador de los marcadores, que en este caso se utilizo el "105". La estructura de los datos recibidos en formato JSON se muestra en la Figura 52.

```
{
  data: [
    x,
    y,
    z,
    n,
    ex,
    ey,
    ez
  ]
}
```

Figura 52: Estructura de los datos en formato JSON

En el lado del ESP32 se utilizaron las librerías WiFi.h [19] y ArduinoJson.h [20] con las cuales fue posible utilizar el ESP32 como cliente y por medio de funciones como client.read y client.print se logró establecer la comunicación. Los datos recibidos por el servidor se almacenan en un buffer para luego deserializarlos y colocar cada dato en una variable que se almacenan en la memoria del ESP32. En la Figura 53 se muestran los datos recibidos del marcador 105 en formato JSON y en variables individuales. Para verificar los valores enviados se colocó el marcador al centro del sistema y como se observa en la Figura 53 los valores de 'x' y 'y' se acercan a 0.

```
Connecting to Robotat
.....
WiFi connected
IP address:
192.168.50.42
Connecting to Optitrack
Connected
Connected to the Robotat server... Type EXIT to stopſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſſ
deserializeJson() returned InvalidInput
{"data": [-0.010888566263020039, 0.00513529684394598, 0.03668757155537605, -0.592383086681366, 0.004928352311253548, -0.8056354522705078, 0.003077418077737093]}
x = -0.01, y = 0.01, z = 0.04, n = -0.59, ex = 0.00, ey = -0.81, ez = 0.00
{"data": [-0.010886762291193008, 0.005137189757078886, 0.03669309988617897, -0.5923930406570435, 0.0048926761373877525, -0.805628418922424З, 0.0030544993933290243]}
x = -0.01, y = 0.01, z = 0.04, n = -0.59, ex = 0.00, ey = -0.81, ez = 0.00
{"data": [-0.010886762291193008, 0.005137189757078886, 0.03669309988617897, -0.5923930406570435, 0.0048926761373877525, -0.8056284189224243, 0.0030544993933290243]}
x = -0.01, y = 0.01, z = 0.04, n = -0.59, ex = 0.00, ey = -0.81, ez = 0.00
{"data": [-0.010889699682593346, 0.005134425591677427, 0.03668644651770592, -0.5923485159873962, 0.004852012265473604, -0.8056613206863403, 0.0031036012805998325]}
x = -0.01, y = 0.01, z = 0.04, n = -0.59, ex = 0.00, ey = -0.81, ez = 0.00
{"data": [-0.010885154828429222, 0.005135294981300831, 0.036681775003671646, -0.5923884510993958, 0.004931892734020948, -0.8056317567825317, 0.0030508856289088726]}
x = -0.01, y = 0.01, z = 0.04, n = -0.59, ex = 0.00, ey = -0.81, ez = 0.00
{"data": [-0.010888699442148209, 0.005131909158080816, 0.03668525442481041, -0.5923911929130554, 0.004903765860944986, -0.805629670619964б, 0.00310487556271255]}
x = -0.01, y = 0.01, z = 0.04, n = -0.59, ex = 0.00, ey = -0.81, ez = 0.00
{"data": [-0.010891013778746128, 0.00513340812176466, 0.036685097962617874, -0.5923898220062256, 0.004871467128396034, -0.805630862712860l, 0.003093620063737035]}
x = -0.01, y = 0.01, z = 0.04, n = -0.59, ex = 0.00, ey = -0.81, ez = 0.00
```

Figura 53: Valores recibidos en ESP32 desde Robotat

11.2. Matlab

Luego de la conexión con el sistema Robotat, la siguiente tarea era la conexión con Matlab. Para esto fue necesario establecer al ESP32 como servidor y Matlab como cliente, con el objetivo de utilizar a Matlab como medio de cálculo de trayectorias para las extremidades. En la red Robotat se configura una IP fija para el ESP32 y se establece como 192.168.50.42 utilizando la dirección MAC del ESP32.

Para realizar la conexión se crea un objeto de tipo TCP en Matlab, incluyendo la IP y el puerto deseado. Luego de esto se crea una variable que contiene la estructura serializada en JSON de 18 valores para los servos, que en este caso se utiliza el valor de 512 para todos. Como se puede observar en la Figura 54, al correr el programa se muestra el servidor conectado (ESP32) y este envía dos mensajes, el primero al comienzo del proceso de envió de datos y el segundo cuando termina el mensaje.

```
>> Client
Server connected.
Enviando...
Enviado
```

Figura 54: Mensaje de conexión en Matlab

Al igual que en la sección anterior, el ESP32 utiliza las mismas librerías y funciones para su conexión y almacena el mensaje enviado en un buffer para luego deserializarlo y colocar los valores de los servos en variables independientes almacenados en la memoria del ESP32. A diferencia de la sección anterior este mensaje se envía una sola vez en comparación a los datos del Optitrack que se reciben constantemente. En la Figura 55 se muestran los mensajes correspondientes al ESP32 de conexión con el cliente y los datos recibidos en formato JSON.

```
.....Connected
IP Address:192.168.50.42
[Client connected]
Received: {m1: 512, m2: 512, m3: 512, m4: 512, m5: 512
[Client disconnected]
```

Figura 55: Mensaje de conexión en ESP32

11.3. Comunicación entre OpenCm 9.04 y ESP32

La tercera tarea era la comunicación entre el OpenCm 9.04 y el ESP32 utilizando el formato JSON para el envío de datos. Como se mencionó en el capitulo de Electrónica Interna, la conexión entre microcontroladores es tipo UART, lo que significa que ambos deben estar en la misma frecuencia o baudrate y se estableció como 115200. Esta frecuencia permite una lectura rápida y correcta de los datos, lo cual es necesario si posteriormente se envía una secuencia de movimiento.

En el ESP32 se serializan 18 variables para los servos en formato JSON para luego enviarlos por medio de un buffer a través de la conexión UART. La estructura del JSON se muestra en la Figura 56 con un valor de 512 para todas las variables.

```
{
  m1: 512,
  m2: 512,
  m3: 512,
  m4: 512,
  m5: 512,
  m6: 512,
  m7: 512,
  m8: 512,
  m9: 512,
  m10: 512,
  m11: 512,
  m12: 512,
  m13: 512,
  m14: 512,
  m15: 512,
  m16: 512,
  m17: 512,
  m18: 512
}
```

Figura 56: Estructura de los datos en formato JSON

Luego de esto, el OpenCm 9.04 realiza la lectura de los datos que son almacenados en un buffer para luego ser deserializados y puestos en variables, tal como en las secciones anteriores. Al tener todos los datos en la memoria del OpenCm, se realiza la ejecución de cada uno de los valores para su respectivo servo, esto se logra gracias a la función goalPosition de la librería DynamixelWorkbench.h [21], en donde se le asigna el identificador del servo y un valor entre 0 y 1023 como se muestra en la Figura 8.

A continuación, en la Figura 57 se muestra un mapa del hexapod con cada servo numerado en relación a la asignación de variables, de los cuales posteriormente, en el Cuadro 5 se muestra cuales variables se le asignan a los identificadores de los servos.

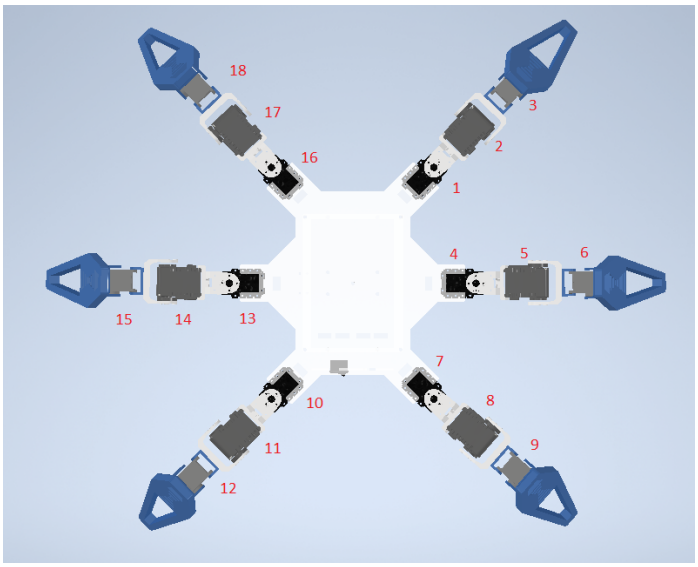


Figura 57: Mapa de asignación de variables para servos

ID Servo	Variable
12	M1
1	M2
18	M3
13	M4
2	M5
3	M6
14	M7
4	M8
5	M9
15	M10
6	M11
7	M12
16	M13
8	M14
9	M15
17	M16
10	M17
11	M18

Cuadro 5: Identificadores de servos y variables asignadas para goalPosition

CAPÍTULO 12

Ejecución de trayectorias

Para la ejecución de trayectorias se utilizaron dos conceptos importantes, cinemática directa e inversa. Estas son técnicas usadas en gráficos 3D por computadora para calcular la posición de partes de una estructura articulada a partir de sus componentes fijos y las transformaciones inducidas por las articulaciones de la estructura.

Sin embargo, el objetivo de esta parte era la ejecución de secuencias simples por lo que se establecieron 3 secuencias para la parte de cinemática directa y 4 secuencias para cinemática inversa. A continuación se explica más a detalle como se obtuvieron dichos parámetros.

12.1. Cinemática directa

La cinemática directa utiliza ecuaciones cinemáticas para calcular la posición de su actuador final a partir de valores específicos denominados parámetros. Para simplificar el cálculo de los parámetros, tanto de cinemática directa como de cinemática inversa, se utilizó el Toolbox de Peter Corke [26] mencionado en el capitulo de derivación cinemática, en el cual se utilizan las funciones fkine y ikine respectivamente.

Como se observa en la Figura 58 se establecen 3 vectores con valores en radianes, que son las 3 secuencias mencionadas previamente. Estos valores corresponden a la posición para cada extremidad y luego se calcula la posición del actuador final usando la función fkine.

```
%% Cinematica Directa
q0 = [0, -pi/4, -pi/4];
q1 = [0, -pi/2, -pi/2];
q2 = [0, -pi/2, pi/4];
T0 = leg.fkine(q0);
T1 = leg.fkine(q1);
T2 = leg.fkine(q2);
```

Figura 58: Código de cinemática directa en Matlab

Luego de obtener las tres secuencias para cada extremidad es necesario establecer una matriz que contenga los 18 valores de los servos (filas) y las secuencias a enviar (columnas) como se observa en la Figura 59, además se realiza una conversión de radianes a los valores aceptados por los servos (0 - 1023), siendo 512 el valor de 0 radianes.

```
sec =

   512   512   512
   357   202   667
   357   202   202
   512   512   512
   357   202   667
   357   202   202
   512   512   512
   357   202   667
   357   202   202
   512   512   512
   357   202   667
   357   202   202
   512   512   512
   357   202   667
   357   202   202
   512   512   512
   357   202   667
   357   202   202
```

Figura 59: Matriz de 3 secuencias de cinemática directa

Al establecer la matriz de secuencias se hace un ciclo de conexión con el ESP32, en donde se envía cada columna de la matriz en formato JSON y se le agrega un tiempo de espera para que los servos lleguen a la posición recibida. En las Figuras 60, 61, 62 se observa la simulación de las 3 poses del robot de su respectiva secuencia, agregando el vector de secuencia a la función plot de la derivación cinemática.

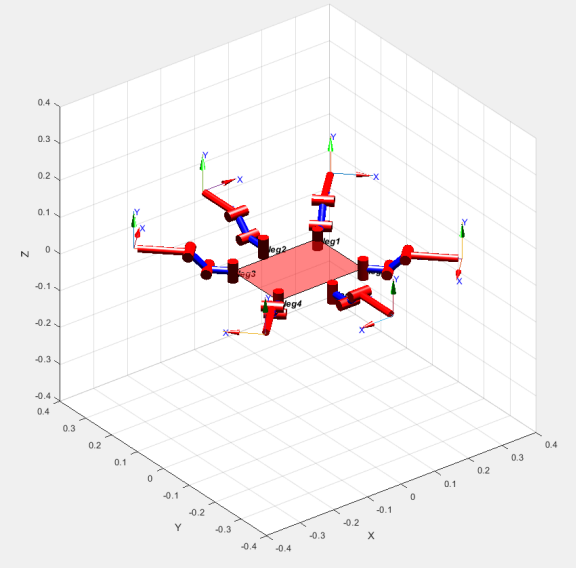


Figura 60: Pose 1 de cinemática directa

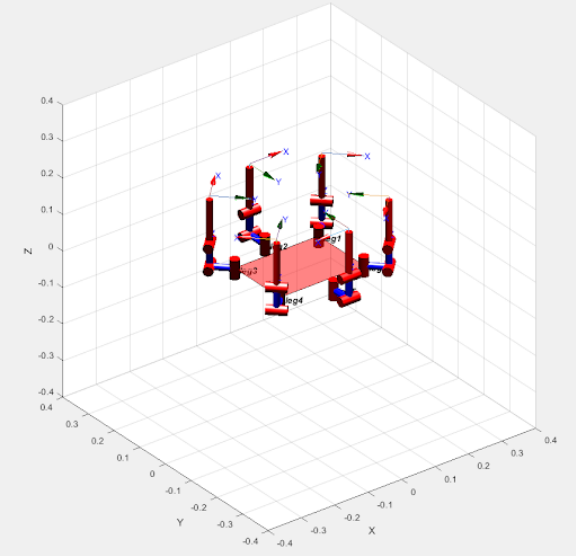


Figura 61: Pose 2 de cinemática directa

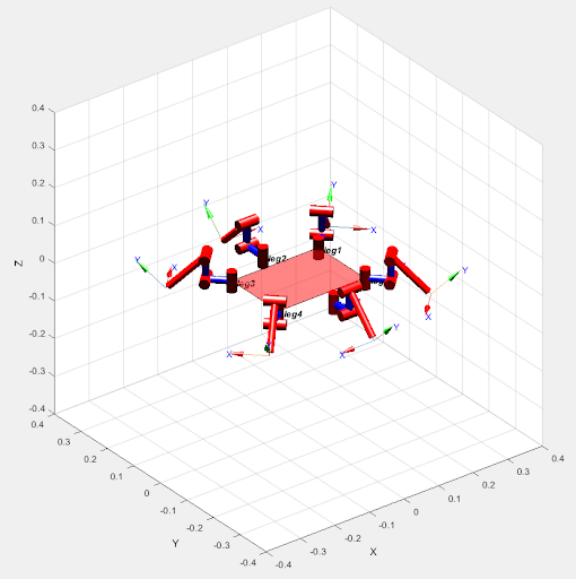


Figura 62: Pose 3 de cinemática directa

12.2. Cinemática inversa

Al igual que la cinemática directa, la cinemática inversa utiliza ecuaciones cinemáticas pero calcula los parámetros en función a la posición del actuador final. En este caso se utiliza la función ikine para la simplificación de los cálculos. En la Figura 63 se muestra que se estableció una trayectoria del punto M1 a M2 con 4 iteraciones o secuencias, para luego realizar el cálculo de los parámetros con base en esta trayectoria.

```
%% Cinematica Inversa
M1 = transl(0.1, 0.1, 0); % define the start point
M2 = transl(0.1, 0.1, -0.2);    % and destination
Mi = ctraj(M1, M2, 4);
q = leg.ikine(Mi, 'q0', [0, 0, 0], 'mask',[1, 1, 0, 0, 0, 0]);
qg1 = q(1,:); %Visualizacion
qg2 = q(2,:);
qg3 = q(3,:);
qg4 = q(4,:);
ql = 512 + round(rad2deg( q )/0.29);
ql1 = ql(1,:);
ql2 = ql(2,:);
ql3 = ql(3,:);
ql4 = ql(4,:);
```

Figura 63: Código de cinemática inversa en Matlab

Posteriormente se realiza la matriz de secuencias, Figura 64, mencionada en la sección anterior que posee 18 valores para los servos y 4 columnas correspondientes a las 4 secuencias calculadas.

```
sec =

   357   357   357   357
   587   544   525   522
   592   544   524   521
   357   357   357   357
   587   544   525   522
   592   544   524   521
   357   357   357   357
   587   544   525   522
   592   544   524   521
   357   357   357   357
   587   544   525   522
   592   544   524   521
   357   357   357   357
   587   544   525   522
   592   544   524   521
   357   357   357   357
   587   544   525   522
   592   544   524   521
```

Figura 64: Matriz de 4 secuencias de cinemática inversa

Finalmente, en las Figuras 65 y 66 se muestra la simulación de la pose inicial y final de la trayectoria. Además se puede observar que la trayectoria es ascendente, tal como se estableció en los puntos de inicio y final.

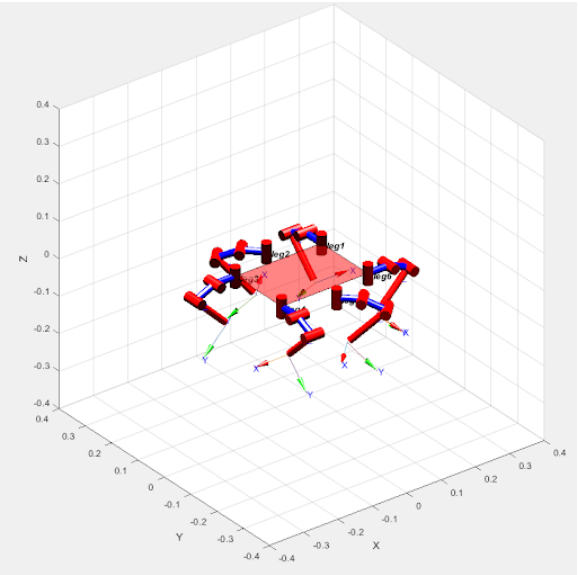


Figura 65: Pose 1 de cinemática inversa

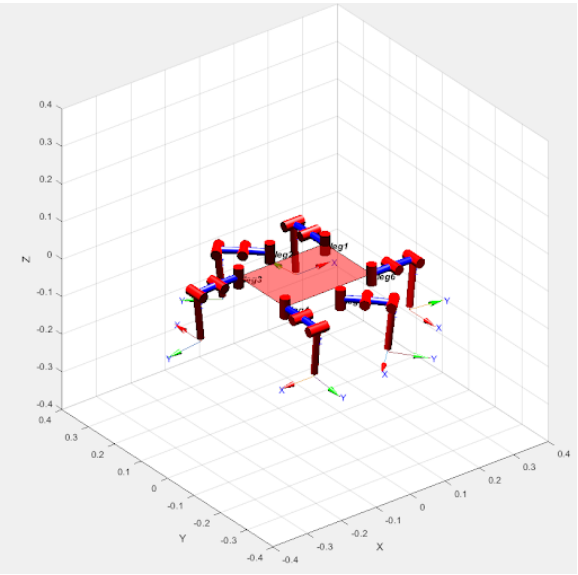


Figura 66: Pose 2 de cinemática inversa

CAPÍTULO 13

Conclusiones

- Se ejecutó el diseño mecánico de la plataforma con éxito, para esto se utilizaron piezas de acrílico de 2.5mm de espesor para la parte del torso y piezas manufacturadas con impresión 3D para las extremidades las cuales se diseñaron con unas dimensiones apropiadas para acoplarse satisfactoriamente a los 18 servos, por otro lado el diseño completo obtuvo un funcionamiento correcto para el movimiento de la plataforma con un manejo eficiente.
- Se logró realizar el diseño de la electrónica utilizando 2 tipos de microcontroladores, el OpenCm 9.04 y el ESP32. La implementación de estos controladores hicieron posible el funcionamiento correcto y se logró controlar 12 servos XL-320 y 6 servos AX-12A junto con su compatibilidad con un sistema conectado por Wi-Fi. Ambos controladores se conectaron entre si por una conexión UART, con el fin de que el ESP32 reciba los comandos y se envíen al OpenCm 9.04 para ejecutarlos.
- Se realizó la implementación al ecosistema robotat, el cual incluye la lectura de datos de un marcador en Optitrack a través de una comunicación cliente-servidor entre el Robotat y el ESP32, el envío de secuencias desde Matlab como medio de calculo de trayectorias y a través de una comunicación cliente-servidor con el ESP32 y la comunicación entre microcontroladores junto a la ejecución de dichas secuencias.

CAPÍTULO 14

Recomendaciones

- Para el desarrollo de un robot de plataforma móvil como lo es el hexapod se recomienda utilizar un solo tipo de servos para que su diseño y conexión sea lo más eficiente posible, o como alternativa colocar servos con el torque suficiente que soporten por sí solos el peso del robot utilizando 6 servos.
- Para este tipo de proyectos se recomienda tener elementos de sujeción extra para los servos, ya que los servos Dynamixel cuentan con una cantidad justa de tornillos, los cuales si por algún motivo se pierden son muy difíciles de reemplazar. Lo mismo sucede para el caso de los Rivets.
- Al utilizar una librería para controlar servos tan complejos se recomienda investigar cómo funciona y qué funciones posee, debido a que puede presentarse el caso de necesitar una función que es vital para su correcto funcionamiento.
- Se recomienda la implementación de un módulo de baterías con más capacidad y con la opción de una carga directa a la plataforma robótica.

CAPÍTULO 15

Bibliografía

[1] Interbotix, *PhantomX MK-IV Hexapod*, https://www.trossenrobotics.com/phantomx-hexapod-mk4.aspx#documentation, Visitado: 10-04-2022.

[2] 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," Tesis de licenciatura, Universidad del Valle de Guatemala, 2022.

[3] J. A. C. Forno, "Diseño e implementación de una red de comunicación wifi e interfaz gráfica para una mesa de pruebas de robótica de enjambre," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[4] CITEDI, *Movimiento de hexapodo*, http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462017000200305#B18, Visitado: 4-05-2022.

[5] Autodesk Inventor, *Tecnicas de modelado en Inventor*, https://knowledge.autodesk.com/es/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2019/ESP/Inventor-Help/files/GUID-C68767BC-9398-4D2A-8DF5-B87184E396B3-htm.html, Visitado: 4-05-2022.

[6] Prometec, *Que es un servo*, https://www.prometec.net/servos/, Visitado: 4-05-2022.

[7] Robotis, *AX-12A Especificaciones*, https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/#control-table-of-eeprom-area, Visitado: 4-05-2022.

[8] ——, *XL-320 Especificaciones*, https://emanual.robotis.com/docs/en/dxl/x/xl320/#control-table-of-eeprom-area, Visitado: 4-05-2022.

[9] ——, *OpenCm 9.04 Especificaciones*, https://emanual.robotis.com/docs/en/parts/controller/opencm904/, Visitado: 15-09-2022.

[10] ——, *OpenCm 485 EXP Especificaciones*, https://emanual.robotis.com/docs/en/parts/controller/opencm485exp/, Visitado: 15-09-2022.

[11] CISCO, *What Is Wi-Fi?* https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html, Visitado: 4-05-2022.

[12] Internet of things, *ESP32 Datasheet*, http://esp32.net/#Development, Visitado: 4-05-2022.

[13] Teseo, *¿Qué es y cómo funciona la captura de movimiento?* https://teseo.es/noticias/que-es-y-como-funciona-la-captura-de-movimiento/, Visitado: 4-05-2022.

[14] OptiTrack, *OptiTrack for Robotics*, https://optitrack.com/applications/robotics/#open-architecture, Visitado: 4-05-2022.

[15] NaturalPoint, Inc., *OptiTrack Primer 41*, https://optitrack.com/cameras/prime-41/, Visitado: 4-05-2022.

[16] Ayudaley, *Protocolo TCP*, https://ayudaleyprotecciondatos.es/2021/07/29/protocolo-tcp/, Visitado: 25-11-2022.

[17] MDN, *Trabajando con JSON*, https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON, Visitado: 25-11-2022.

[18] Arduino, *Librerias Arduino*, https://www.arduino.cc/reference/en/libraries/, Visitado: 25-11-2022.

[19] ——, *Libreria WiFi*, https://www.arduino.cc/reference/en/libraries/wifi/, Visitado: 25-11-2022.

[20] ——, *Libreria JSON*, https://www.arduino.cc/reference/en/libraries/arduinojson/, Visitado: 25-11-2022.

[21] Robotis, *Libreria DynamixelWorkbench*, https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_workbench/, Visitado: 25-11-2022.

[22] IONOS, *Multi Threading*, https://www.ionos.es/digitalguide/servidores/know-how/explicacion-del-multithreading/, Visitado: 25-11-2022.

[23] RD, *Comunicación Serial*, https://robots-argentina.com.ar/didactica/que-es-la-comunicacion-serie/, Visitado: 25-11-2022.

[24] Robotis, *Frame FP04-F2 (AX-12A)*, https://www.robotis.us/fp04-f2-10pcs/, Visitado: 28-09-2022.

[25] ——, *Rivets (XL-320)*, https://www.robotis.us/rivet-set-rs-10/, Visitado: 28-09-2022.

[26] P. Corke, *Robotics, Vision and Control*, ép. Springer Tracts in Advanced Robotics. Springer, 2017, ISBN: 978-3-319-54412-0.

[27] Robotis, *Frame FP04-F3 (AX-12A)*, https://www.robotis.us/fp04-f3-10pcs/, Visitado: 27-11-2022.

CAPÍTULO 16

Anexos

16.1. Programación

16.1.1. Movimiento de prueba - Adafruit y ESP32

https://github.com/luisro21/Hexapod

16.1.2. Simulación y modelado digital de la plataforma robótica

https://github.com/luisro21/Hexapod-Simulacion

16.1.3. Programación final - ESP32, OpenCm 9.04 y Matlab

https://github.com/luisro21/Robotat

16.2. Diseño mecánico y electrónico

https://drive.google.com/drive/folders/1atJzlQob9QyXcdflpaKthgL57iXE9UU3?usp=share$_l ink$

16.3. Diseño de PCBs

16.3.1. Vista de cara inferior

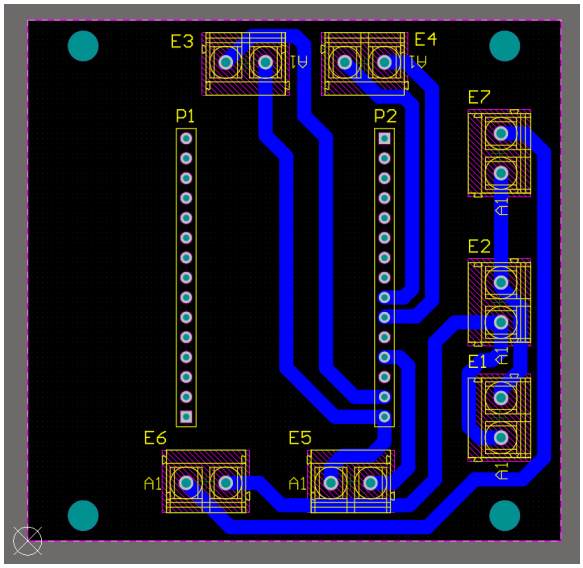


Figura 67: Diseño de PCB.

16.3.2. Diseño 3D

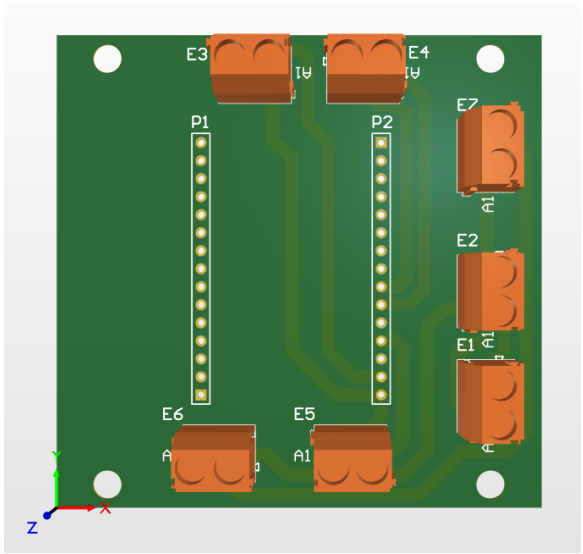


Figura 68: Diseño de PCB en 3D.

16.4. Interfaz de adafruit - Movimiento de prueba

Figura 69: Interfaz gráfica para envió de secuencias de movimiento

CAPÍTULO 17

Glosario

Adafuit IO: Es un servicio de mensajería que emplea el protocolo MQTT con un esquema de publicador/suscriptor y un Broker que acepta los mensajes publicados por el cliente y los difunde entre los clientes suscritos. 48

Baterías 18650: Batería recargable modelo 18650 de Li-ion con voltaje de 3.7V. 39

Daisy Chain: Daisy Chain es un esquema de cableado en el que varios dispositivos se conectan juntos en secuencia o serie en donde se comparte la alimentación y la señal de control es un bus de datos. 38

Matlab: Es una plataforma de computación numérica y programación utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos. 46

Parámetros de Denavit-Hartenberg: Los parámetros de Denavit-Hartenberg se utilizan para describir cómo se vincula una articulación con la siguiente. Estos parámetros describen básicamente rotaciones y translaciones. 45, 46

PCB: Una placa de circuito impreso, o placa de PC, o PCB, es un material no conductor con líneas conductoras impresas o grabadas. Los componentes electrónicos se montan en la placa y las trazas conectan los componentes para formar un circuito o conjunto de trabajo. 29, 40, 41

Revolutas: La conexión entre dos eslabones que permiten movimiento, en este caso rotación de un grado de libertad. 42–44

Rivets: Remaches de 6 mm para ensamblar marcos ROBOTIS. 33–35, 37, 41