Diseño e implementación de un robot serpiente controlado de forma remota

Raúl Estuardo Aguilar Recinos

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de un robot serpiente controlado de forma remota

Trabajo de graduación presentado por Raúl Estuardo Aguilar Recinos
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de un robot serpiente controlado de forma remota

Trabajo de graduación presentado por Raúl Estuardo Aguilar Recinos
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

Vo.Bo.:

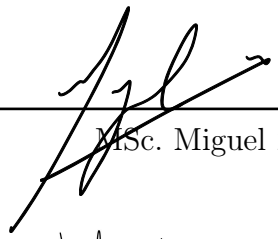
(f) 

Ing. Kurt Kellner

Tribunal Examinador:

(f) 

Ing. Kurt Kellner

(f) 

MSc. Miguel Zea

(f) 

Ing. Pablo Daniel Mazariegos

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

Prefacio

La disciplina de la Ingeniería Mecatrónica está ligada a grandes avances tecnológicos. En el campo de la robótica ha sido de gran utilidad para el desarrollo de productos innovadores que contribuyan a la resolución de problemas a todo nivel. Es por ello, por lo que, el presente trabajo de graduación se centra en la investigación, el diseño y el desarrollo de un robot que imita la anatomía de una serpiente.

Este se mueve de forma flexible para explorar rincones ocultos con gran facilidad. En el caso de los reptiles, esto les ha permitido sobrevivir en condiciones y terrenos en los que al humano le sería imposible hacerlo. Bajo este principio evolutivo, los robots serpiente son utilizados en áreas estrechas, peligrosas, de riesgo y poco accesibles, por ello son ideales para labores de rescate y en una variedad cada vez más amplia de sectores, como el industrial.

El robot serpiente, sujeto de esta investigación, es controlado de forma remota dentro del ecosistema Robotat y posee la capacidad de brindar las poses de los sistemas, de los que actualmente carece. Esto me motivó, junto a un grupo de estudiantes de la carrera Ingeniería Mecatrónica, a elaborar agentes que adquirieran la capacidad de operar en el ecosistema por medio de comandos a distancia, con el fin de establecer los cimientos de las próximas investigaciones de robótica, dentro del Robotat.

Esta experiencia académica no hubiera sido posible sin el apoyo de mis padres, Raul y Carol, a quienes les agradezco por ser mi guía en todo momento y por brindarme la valiosa oportunidad de estudiar en esta prestigiosa universidad. Su ejemplo como profesionales me ha motivado a alcanzar las metas académicas y personales que me proponga. Además, su cariño y sus consejos me han dado seguridad para desarrollarme de forma integral.

Agradezco también a mi hermano, Fabián, por su apoyo y su compañía. Mi gratitud también a mis abuelitos: Papa lorenzo, Mamamenchis y Elvia, por su cariño, apoyo y por motivarme para cumplir esta meta profesional.

Índice

Lista de figuras

Lista de cuadros

Resumen

El presente trabajo de graduación se centró en el diseño e implementación de un robot serpiente. Para facilitar la actuación y la locomoción de una serpiente biológica e imitar el movimiento de ondulación, se emplearon cinco servomotores.

Esto fue posible debido al microcontrolador ESP32, en el que se implementó una librería de control y se comandó el robot de forma remota, por medio del protocolo de comunicación MQTT.

Esta investigación apoyó el desarrollo de sistemas robóticos educativos para el ecosistema Robotat. Este diseño se convirtió en el primero en su tipo, al fusionar la naturaleza con la tecnología y al abrir las puertas a futuros estudios con el sistema de captura de movimiento.

Por lo que, la idea principal de la investigación fue incursionar en la captura de movimiento de los sistemas robóticos en tiempo real y hacer uso de las librerías para controlar al robot a través de protocolos de comunicación dentro del ecosistema Robotat.

Abstract

The present graduation work focused on the design and implementation of a snake robot. To facilitate the actuation and locomotion of a biological snake and to mimic the waving motion, five servomotors were used.

This was possible due to the ESP32 microcontroller, in which a control library was implemented and the robot was commanded remotely, via the MQTT communication protocol.

This research supported the development of educational robotic systems for the Robotat ecosystem. This design became the first of its kind, merging nature with technology and opening the door to future studies with the motion capture system.

Therefore, the main idea of the research was to venture into the motion capture of robotic systems in real time and make use of libraries to control the robot through communication protocols within the Robotat ecosystem.

CAPÍTULO 1

Introducción

La presente investigación surge de la necesidad de incorporar agentes robóticos en el sistema de captura de movimiento para su experimentación y análisis. Ya que, anteriormente no existía la infraestructura para realizar esta tarea.

Este forma parte del ecosistema robótico que está ubicado en la Universidad del Valle de Guatemala, en el salón Cit 106, el que cuenta con los recursos indispensables para estos fines, entre ellos una plataforma similar al Robotarium del Georgia Institute of Technology, en la que se pueden desarrollar actividades multirobot. De igual forma, cuenta con seis cámaras OptiTrack, las cuales tomaron las posiciones de los agentes robóticos y de una red de comunicación Wifi.

Por lo tanto, este documento profundiza en el diseño, el ensamblaje y la configuración de un robot serpiente, que pueda incorporarse al ecosistema robótico Robotat, por medio de una red WiFi. Es así, como el uso de un sistema de captura de movimiento permitirá realizar diferentes pruebas de comando a distancia al robot serpiente.

CAPÍTULO 2

Antecedentes

Los robots serpiente son dispositivos basados en el movimiento de los reptiles sin extremidades y con un esqueleto óseo muy flexible. La complexión de su cuerpo los hace perfectos para todo tipo de terrenos, en especial, para desplazarse en espacios reducidos. Por consiguiente, les han dado diferentes aplicaciones en múltiples campos. [1]

El pionero en la materia fue el Dr. Shigeo Hirose, nacido en Tokyo y profesor del Instituto de Tecnología de ese país asiático. El artículo Machine Design of Biologically Inspired Robots hace referencia al estudio y diseño de los robots serpiente que comenzó Hirose en 1971. Su interés por la simplicidad de la forma y la versatilidad de su estructura lo llevó a enfocarse en el análisis del movimiento estándar de las serpientes. En la investigación, clasificó los diseños mecánicos del robot serpiente en 5 categorías. [1]

La junta de flexión activa consiste de juntas conectadas serialmente y constituyen la forma más estándar de los robots serpiente. El primer modelo con movimiento de una serpiente real fue el ACM3 (Active Cord Mechanism 3) con 2m de largo y 28 Kg de peso. Se formó de 20 juntas conectadas entre sí, las que le proveían un movimiento oscilatorio, que era proporcionado por la transmisión de servomecanismos. El modelo también contaba con sensores táctiles en ambos lados del robot, que permitían el control de la forma del ACM3. Posteriormente, se desarrolló el ACM-R5 un modelo considerado anfibio. Este se caracterizó por la implementación de un cuerpo hermético, contra el agua y por contar con juntas universales accionadas con engranajes conectados a los motores. Así también, las partes externas estaban compuestas por tubos flexibles y aluminio. [1]

La junta de flexión activa y elongación destaca por su diseño, parecido a la elongación del movimiento de una oruga. Esto le da al robot la posibilidad de moverse de forma lineal, mientras su cuerpo se mantiene sin curvatura, y de desplazarse por una tubería recta, como el Slim Slime Robot. Este sobresale por estar compuesto de múltiples módulos de fuelles de acero, acoplados de forma paralela a los intervalos regulares, con una circunferencia idéntica. Los fuelles se flexionan al momento que el aire comprimido se suple y se comprimen cuando el aire se retira. [1]

La junta de flexión con oruga activa tiene como muestra a Souryu, que es conocido por ser un robot de rescate, utilizado para conducir operaciones de búsqueda en edificios colapsados en terremotos. Su cuerpo alargado consta de tres juntas. [1]

El artículo que refiere al Dr. Hirose contribuye a la introducción de los sistemas biomecánicos de las serpientes robot, los que más adelante fueron respaldados por las investigaciones de Hiroya Yamada, doctor en ingeniería mecánica aeroespacial por el Instituto de Tecnología de Tokyo. [1]

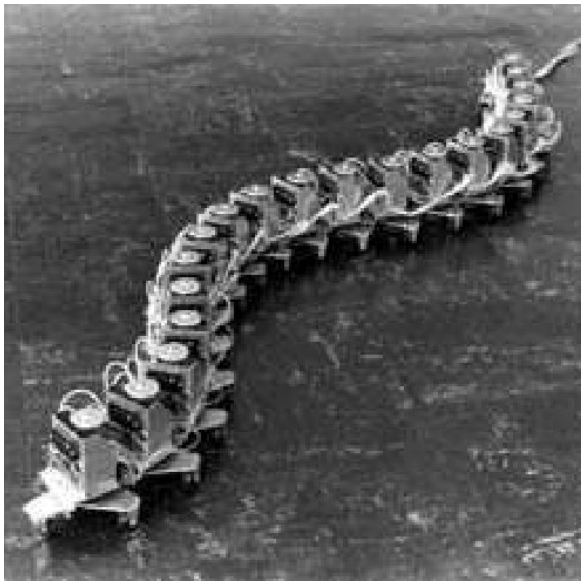


Figura 1: ACM3.

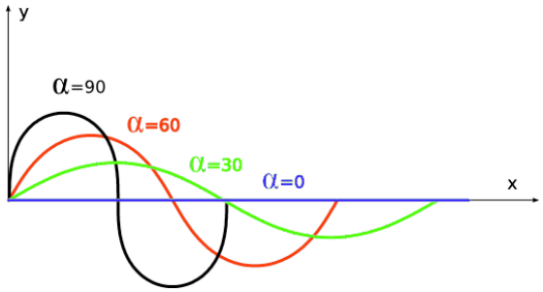


Figura 2: Cuerva de movimiento.

2.1 Sistema de captura de movimiento Optitrack

Para la mejor comprensión del sistema de captura de movimiento Optitrack, se tomó como referencia la tesis del ingeniero mecatrónico José David Pellecer, titulada "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"[2].

Su investigación explica que, por medio de este sistema se plantearon pruebas para demostrar el funcionamiento del manipulador serial dentro del ecosistema Robotat. En esta

se alcanzó el objetivo de verificar la información desplegada por el Optitrack sobre la posición del efector final. Esta se comparó, posteriormente, con las instrucciones mandadas desde el Robotat para controlar al manipulador. Para ello, se llevaron a cabo dos tipos de pruebas: con cinemática inversa y la ejecución de trayectorias.

La cinemática inversa es una prueba que consistió en mandar cinco posiciones meta desde el script hacia el módulo inalámbrico y comparar su valor teórico. Se demostró poca desviación de la posición final del efector final, error absoluto por debajo de los 30 mm y errores por debajo del 6 % por sistema de captura de movimiento. Se atribuyó el error a problemas en la calibración y reflejos del entorno que ingresaron al espacio de observación de mocap.

En cuanto a la **Ejecución de trayectorias**, se elaboró un análisis gráfico de la posición del efector final obtenida por el Optitrack para un recorrido preestablecido. Se definió por medio de seis posiciones, las que corresponden a los puntos teóricos por los cuales se supone que debe atravesar el efector final. Se demostró por medio de la gráfica que el recorrido hace sentido con los resultados obtenidos con anterioridad.

Con estas pruebas se validó la correcta integración del R17 con el ecosistema Robotat.

Así también, se consultó el trabajo de graduación del ingeniero mecatrónico, Camilo Perafán Montoya, titulado Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica", fundamental para comprender mejor el sistema Optitrack [3]. Su investigación relata cómo se logró ensamblar, calibrar y configurar este sistema de captura de movimiento, lo que dio lugar a la obtención de información de agentes dentro del rango de captura de las cámaras. Además, se desarrolló e implementó un programa capaz de enviar información por medio de la red Wifi y el protocolo MQTT. Asimismo, se diseñó una antena apta para conectarse al sistema de comunicación desarrollado, que permitiera la interacción de sistemas no robóticos dentro del ecosistema Robotat.

Para validar la correcta implementación del sistema, se procedió a comparar valores de orientación de la IMU y el sistema Optitrack. Se observó que, el error más grande dentro de las mediciones fue de 3.65 grados. Se le atribuyó que la métrica de evaluación de resultados definía que estos no debían variar de un valor de 8 grados, por lo que se aceptaron como válidos. Por lo que, se garantizó la correcta calibración del este.

2.2 Motor Inteligente

Según el trabajo del ingeniero informático de la Universidad de Madrid, Fernando Gallego Hernández, Diseño e Implementación de una Plataforma Autónoma para el Control de Sistemas Servomecánicos, [4], que tiene como objetivo diseñar e implementar una base digital que controle varios servos que utilicen firmware, se usa una placa Ebox 3310MX-AP para el control y la recepción de los comandos. Además de tener un robot ROBONOVA-1, compuesto por servos, que dispone de músculos formados por 16 servos: 5 en cada pierna y 3 en cada brazo, del tipo Hitec HSR-8498HR. La tesis estableció las razones de uso de este componente: su bajo costo, mayor facilidad de implementación al disponer de un menor número de instrucciones mejor detalladas y conexiones en paralelo, de manera independiente, con la posibilidad de hacer pruebas por módulos.

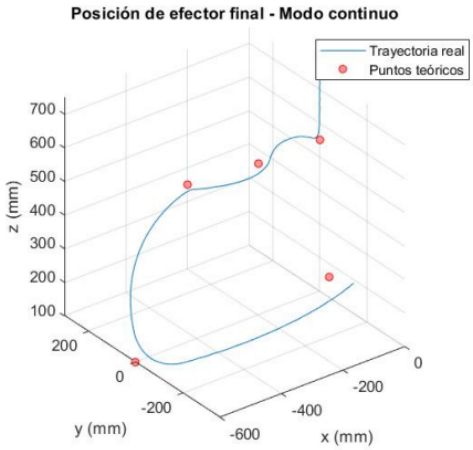


Figura 3: Trayectoria de prueba del efector final.

Corrida	Orientación	Orientación OptiTrack (°)	Orientación IMU (°)	Error (°)
1	pitch	5.718	-6.283	0.57
	yaw	-9.912	-9.805	0.11
	roll	-0.410	-2.598	2.19
2	pitch	-16.21	16.697	0.49
	yaw	-3.323	-2.748	0.58
	roll	1.933	-0.499	1.43
3	pitch	-5.328	-1.679	3.65
	yaw	-15.343	-17.360	2.02
	roll	-29.026	25.825	3.20
4	pitch	-25.663	22.270	3.39
	yaw	-7.834	-9.581	1.75
	roll	-20.851	21.044	0.19
5	pitch	-0.763	-1.402	0.64
	yaw	23.299	22.537	0.76
	roll	-20.747	18.355	2.39

Figura 4: Comparación de mediciones de orientación dadas por la IMU y mediciones del sistema Optitrack.

CAPÍTULO 3

Justificación

Actualmente, los sistemas robóticos son de suma importancia debido a sus múltiples aplicaciones, las que pueden ser industriales, médicas y educativas, por mencionar algunas. Estos sistemas ayudan a mejorar la eficiencia, la precisión de las tareas y la automatización. Por otro lado, el estudio de los movimientos de los animales y su replicación son una tarea enriquecedora que abre campo a la investigación para adaptar estos modelos a sistemas educativos y maniobras de rescate. Como se mencionó anteriormente, el robot Souryu fue pensado para apoyar a los socorristas en la búsqueda de personas atrapadas después de un terremoto, porque se adapta al suelo inestable y además porque le permite identificar de forma inmediata a las personas en lugares de peligro.

La parte fundamental este trabajo de graduación es el desarrollo de un robot serpiente, que permita la investigación del sistema de captura de movimiento en el ecosistema Robotat. Ya que, como se dijo, en la actualidad no existen sistemas robóticos que se muevan dentro de él. Por ello, este trabajo de graduación es de importancia para el análisis de las poses del robot serpiente, que son dirigidas de forma remota y por un sistema de control que recolecte la información dentro del ecosistema. Esto permite el acople de diferentes robots y su mejor entendimiento para un futuro desarrollo de nuevas tecnologías.

CAPÍTULO 4

Objetivos

4.1 Objetivo general

Diseñar e implementar un robot serpiente controlado de forma remota e integrarlo con un sistema de captura de movimiento.

4.2 Objetivos específicos

- Diseñar e implementar los sistemas mecánicos, de control y de potencia de un robot serpiente.
- Implementar el control remoto del robot vía Wifi utilizando el protocolo de comunicación MQTT.
- Integrar el robot serpiente al ecosistema Robotat.
- Utilizar servomotores para el movimiento del robot serpiente.

CAPÍTULO 5

Alcance

La investigación se enfoca en el desarrollo de dos bloques principales. El primero, consiste en el diseño y elaboración de un agente robótico que integre servomotores para su movimiento, y la incorporación de un microcontrolador capaz de establecer una conexión directa con la red Wifi del ecosistema y el control de cada uno de los motores que simulan los músculos principales de una serpiente.

El segundo aborda la familiarización del robot con el sistema de captura de movimiento OptiTrack para obtener información al momento de la integración del agente robótico al Robotat, por medio del sistema de captura de movimiento, con el fin de controlarlo a partir de la herramienta de software MATLAB.

CAPÍTULO 6

Marco teórico

6.1 Anatomía general de una serpiente

La forma en que evolucionaron las serpientes dio como resultado una asimetría visceral, con los órganos del lado derecho más craneales y de mayor tamaño que los del izquierdo.[5] El cuerpo de las serpientes se ha dividido en tres regiones como se puede evidenciar en la Figura(5).

Región craneal: En esta parte se encuentran el esófago, la tráquea, las glándulas paratiroides: timo, tiroides y corazón. [5]

Tercio medio: Aquí se localizan el pulmón, la continuación del esófago, el hígado, el estómago, el bazo, el páncreas, la vesícula biliar, el intestino delgado proximal y un pseudo saco de aire. La presencia de esta estructura les permite respirar, aun con el pulmón comprimido, después de la ingestión de alguna presa. [5]

Tercio caudal: Es el área en donde se encuentran localizados el intestino delgado caudal, las gónadas, las glándulas adrenales, los riñones, el ciego, el colon y la cloaca. [5]

La evidencia científica ha identificado la existencia de variaciones anatómicas entre distintas especies de serpientes arbóreas, terrestres y acuáticas, que reflejan diferentes estrategias evolutivas para adaptarse a diversos hábitats. Como la posición, la forma o el volumen de órganos internos. [5]

6.2 Locomoción de las serpientes

Los movimientos de una serpiente se clasifican en cuatro tipos: serpentino u ondulante, rectilíneo, concertina y movimiento acordeón. Cada uno de estos se caracteriza por la fluidez de su locomoción.

Ondulación: El cuerpo de la serpiente se flexiona hacia la derecha e izquierda alter-

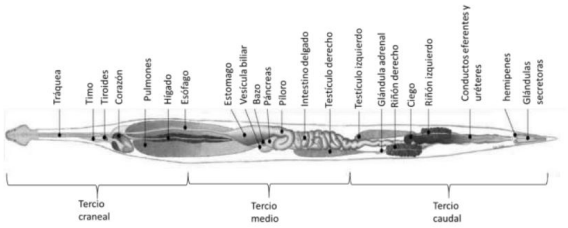


Figura 18. Regiones anatómicas de las serpientes.

Figura 5: Anatomía de una serpiente.

nadamente como se puede observar en la Figura (6), con movimientos en forma de ondas. La serpiente solo mantiene activos los músculos de un lado del cuerpo a la vez, los cuales se contraen desde la cabeza hasta la cola. Primero el lado izquierdo, luego el lado derecho, para permitir que la serpiente se desplace hacia adelante. [6]

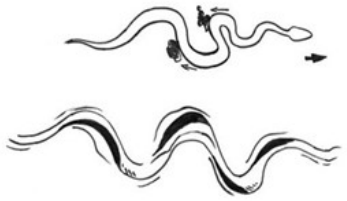


Figura 6: Movimiento ondulatorio de una serpiente.

Concertina: Como se puede evidenciar en la Figura(7) este movimiento consiste en anclar o presionar una parte del cuerpo, mientras tiran o empujan con otra parte. En otras palabras, presiona con la parte posterior, mientras estira la anterior, luego se invierte el movimiento lentamente. [6]

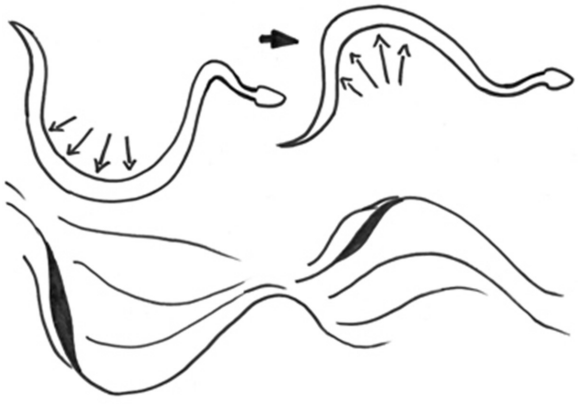


Figura 7: Movimiento concertino de una serpiente.

Rectilíneo: En este caso las serpientes levantan una parte del cuerpo, pero sin movimiento lateral, tal como se muestra en la Figura(8). Crean cierta ondulación a medida que van levantando un pequeño segmento de su cuerpo. [6]

Figura 8: Movimiento rectilíneo de una serpiente.

Acordeón: El cuerpo se estira y se recoge alternativamente mientras la serpiente se mueve desde un punto de anclaje hasta el siguiente. Se utiliza para cruzar superficies lisas o para trepar. Esto se evidencia en la Figura(9). [6]

Figura 9: Movimiento de acordeón de una serpiente.

En la Figura 10 se puede observar el modelo de las partes que conforman el cuerpo de una serpiente, formada por eslabones conectados serialmente con cierto largo. Estos hacen contacto con el suelo solo en los nodos (conexiones entre eslabones) con un torque Ti generado por los músculos en las conexiones Pi. El torque Ti aplica una fuerza fi al suelo en los nodos Pi. El resultado es una fuerza Fti que actúa a través del cuerpo tangencialmente. [1] A partir de ello se obtienen las siguientes ecuaciones:

$$Fti = (fi - fi_{-1}) + (fi_{+1} - fi)sin\frac{\theta i}{2} \cong \frac{Ti_{+1} - Ti_{-1}}{2\delta s}\theta i \tag{1}$$

$$Fni = (fi_{+1} - fi) - (fi - fi_{-1})cos\frac{\theta i}{2} \cong \frac{Ti_{+1} - Ti}{\delta s} - \frac{Ti - Ti_{-1}}{\delta s} \tag{2}$$

Al tener los huesos cortos, se asume que $\delta s = 0$. A partir de aquí, todas las relaciones se pueden expresar en forma de ecuación diferencial, asumiendo que θi para curvaturas $k(s) = \frac{\theta i}{\delta s}$ y Ti =T(s), fti=ft(s) δs y fni = fn(s)δs. De esta forma, las ecuaciones se pueden expresar como:

$$ft(s) = \frac{dT(s)}{ds} * K(s) \tag{3}$$

$$Fn(s) = Ft(s) = \frac{d^2T(s)}{ds^2} \tag{4}$$

En estas ecuaciones, el parámetro s indica la posición del cuerpo medido de la cabeza hasta la cola. Adicionalmente, T(s) indica torque, K(s) indica la curvatura, fn(s) indica la fuerza por unidad de medida que actúa del piso en dirección normal y ft(s) indica la fuerza por unidad de medida que se dirige del piso en dirección tangencial. La ecuación 3 y 4 expresan la relación entre el torque, la curvatura y la fuerzas que se aplican al piso, y muestran el principio básico del mecanismo de la locomoción de las serpientes.

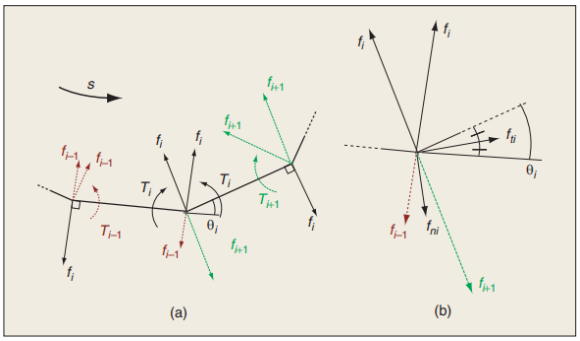


Figura 10: Fuerzas en el movimiento de una serpiente.

6.3 Sistemas de captura de movimiento

Los sistemas de captura de movimiento son un conjunto de técnicas que se usan para reproducir los movimientos de un elemento. Este movimiento se mapea a un modelo 3D para después registrarlo. Tiene múltiples usos, tales como, en el cine, los videojuegos, la investigación y los deportes. Existen dos tipos de sistemas de captura, los ópticos y los magnéticos de captura. [7]

Los sistemas de captura de movimiento ópticos emplean cámaras para rastrear el movimiento de los marcadores acoplados a las articulaciones del cuerpo. De esta forma se registran, interprentan y aplican el movimiento a un modelo digital. Por lo que, no es necesario el uso de un traje con sensores. A pesar de ello, los movimientos registrados se puede ver afectados por diversos factores como baja o alta iluminación, disminuyendo así la precisión del sistema. [7]

Los sistemas magnéticos de captura de movimiento consisten en el uso de un transmisor centralmente localizado y un set de receptores adheridos a partes del cuerpo del actor. Estos receptores son capaces de medir su relación espacial con el transmisor central. Cada recibidor es a su vez conectado a una interfaz que puede ser sincronizada. Este sistema tiene la ventaja de que no hay posibilidad de oclusión en la recepción de los cables, pero no soporta suficientes recibidores para capturar de una forma precisa el carácter del movimiento y limita el área.

Además, el sistema puede ser afectado por cualquier elemento metálico vecino en el área. [7].

6.4 Sistemas Optitrack

Los sistemas Optitrack incluyen el software de captura de movimiento y cámaras de gran velocidad que registran los datos enviados por el sistema de captura. Estos son usados en una variedad de tareas desde películas, juegos, deportes y biomecánica. Por estas características, estos sistemas serán utilizados para obtener los datos de los movimientos del robot serpiente y guardar la información. [8]

- Resolución : 2048x2048.
- Velocidad de fotogramas: 180Hz.
- Precisión 3D: +/- 0.10mm.
- Rango para Marcadores Pasivos: 30 m.
- Rango para Marcadores Activos: 45 m.
- Anillo de LEDs: 20 LEDs.
- Luces del tipo infrarrojo de 850 nm.
- Conexiones: Puerto de datos GigE.
- Sincronización de la cámara por medio de Ethernet.
- Tamaño: 12.6cm x 12.6 cm x 13.2 cm.
- Peso: 1.36 Kg.

6.5 Protocolo de comunicación MQTT

El protocolo MQTT (Message Queue Telemetry Transport) es un protocolo de mensajería asíncrona, usado en el ámbito de las comunicaciones *machine-to-machine*(M2M), en el campo de Internet de las cosas(IOT). Su funcionamiento se basa en el intercambio de mensajes mediante publicación y suscripción, es decir, cada vez que un mensaje sea publicado, este será recibido por el resto de los agentes adheridos al tópico correspondiente. Se adapta sobre cualquier protocolo de red que tenga soporte bi direccional y sin pérdida de información. Permite su implementación en redes con un ancho de banda limitado y con alta latencia. Por otra parte, su flexibilidad permite su aplicación en distintos dispositivos y servicios de IoT. [3]

Los clientes MQTT tienen comodidad de implementación, al ser un protocolo ligero por lo que el hardware no es un problema. Los suscriptores y los publicadores son cada uno de ellos un cliente MQTT, que puede ser tanto suscriptor como publicador, o ambos a la vez. Como se puede observar en la Figura(11).

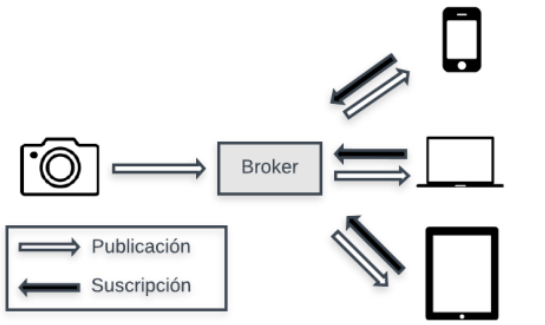


Figura 11: Protocolo MQTT.

6.6 Manipuladores seriales

Un manipulador serial es un robot de base fija que tiene como objetivo final manipular objetos. Estos se forman por cadenas cinemáticas abiertas, es decir, sus elementos se encuentran conectados uno a uno en serie y todas sus juntas son activas o actuadas. Los robots manipuladores seriales son herramientas muy útiles y eficaces para realizar tareas que demanden precisión y que son repetitivas. Por lo que, se busca conocer siempre la posición y orientación del efector final (pose). [2]

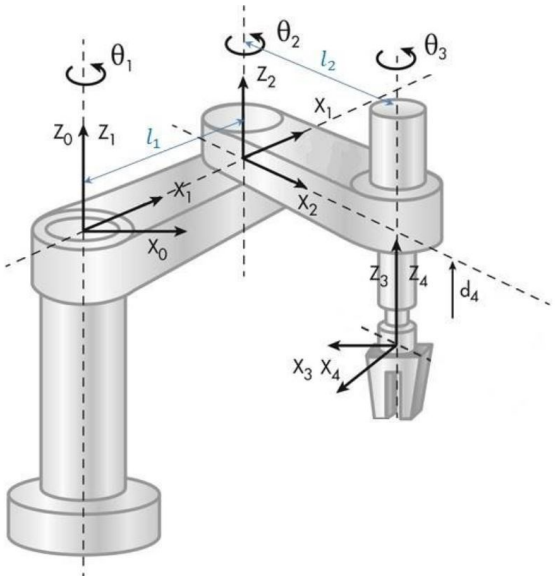


Figura 12: Diagrama cinemático de un manipulador serial.

6.7 Espacios de tarea y trabajo

Estos dos conceptos se relacionan con la configuración del efector final del robot.

- Espacio de tarea: es el espacio en donde la tarea del robot puede expresarse naturalmente. Por lo que, solo es necesario conocer la tarea a realizar, más no el robot como tal. [9]

- Espacio de trabajo: son especificaciones de las configuraciones que el efector final del robot puede alcanzar. Esta definición es manejada principalmente por la estructura del robot, independientemente de la tarea por realizar. [9]

  Ambas configuraciones son distintas al espacio de configuraciones del robot.

6.8 Cinemática directa de manipuladores

El problema de la cinemática directa implica encontrar la posición y orientación (pose) del efector final en un manipulador serial. En otras palabras, la cinemática directa es el mapeo de las coordenadas de las juntas o la configuración del robot a la pose del efector final. Es una forma clara e intuitiva de entender cómo se comporta el robot, por medio de un proceso que involucra colocar un marco de referencia en cada junta del manipulador, desde su base hasta el efector final, al describir cada elemento como una junta rotacional o una prismática. Por ello, se usa una secuencia de transformaciones homogéneas que permitan llegar de la base hasta el efector final. [9]

Para hacer este proceso más sistemático en la forma de describir la geometría de un manipulador serial, que posee juntas y prismáticas, se usa la convención de Denavit-Hartenberg.

6.9 La convención de Denavit-Hartenberg

Esta convención establece una secuencia estandarizada de juntas por medio de cuatro parámetros, que permite calcular la cinemática directa sin importar el manipulador serial.

Esta notación posee las siguientes restricciones:

- El manipulador posee N juntas y N+1 eslabones.
- Cada junta conecta únicamente 2 eslabones.
- Los eslabones son rígidos.
- Las únicas juntas aceptadas son juntas revolutas o prismáticas.

Por lo tanto, para un manipulador con N juntas y N+1 eslabones, numeradas del 0 a N. La junta j conecta al eslabón j-1 con el eslabón j+1. El movimiento relativo del marco de referencia en la junta j+1 a la j. El eslabón 0 es la base del robot, típicamente conectado al eslabón N, el último eslabón se integra al efector final o a una herramienta.[10] La transformación que describe este movimiento entre juntas se llama:

$$^{j-1}Tj = Rotz(\theta j)Translz(dj)Translx(aj)Rotx(\alpha_j) \tag{5}$$

Esta se define por las transformaciones elementales de rotación y traslación con parámetros de DH. Los cuales son:

- θ_j: ángulo de rotación entre x_{j-1} y x_j con respecto al eje z.

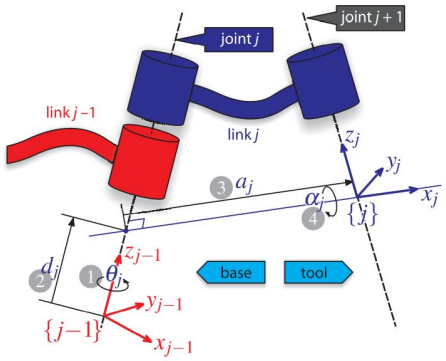


Figura 13: Definición estándar de juntas de Denavit Hartenberg.

- d_j: distancia sobre z hasta x_{j-1} y x_j que sean colineales.
- a_j: distancia entre Z_{j-1} y Z_j a través del eje x.
- α_j : ángulo Z_{j-1} hasta Z_j con respecto al eje x.

Los parámetros a_j y α_j son constantes. θ_j y d_j son variables, dependiendo el tipo de junta.

6.10 ¿Qué es un JSON?

JSON es una abreviación en inglés de JavaScript Object Notation o Notación de Objetos de JavaScript. Se puede describir como un formato de intercambio de datos, independiente del lenguaje. Utiliza convenciones que son ampliamente conocidas en la familia de lenguajes en C, C++, Java, JavaScript, Perl, Python y muchos otros. JSON se basa en un subconjunto de lenguaje de programación JavaScript, Estándar ECMA-262 3a Edición - Diciembre de 1999. El *String* resultante se le llama *JSON document* y puede ser enviado por una red o guardado en un archivo. [11] Estas características hacen de JSON un lenguaje de intercambio de datos ideal.

```
{"sensor":"gps","time":1351824120,"data":[
↪  48.756080,2.302038]}
```

Figura 14: Estructura Json.

JSON está compuesto por:

- Una colección de pares de nombre/valor o mejor conocido como *objeto*, registro, estructutra, diccionario, tabla hash o un arreglo asociado.
- *Array* de datos del tipo flotante o *float*, que contiene valores decimales.

JSON se usa como protocolo de comunicación entre cliente y servidor, debido a esto es empleado por la mayoría de los servicios web que tienen una API (Application Programming

Interface) basado en JSON, una forma de interactuar con los servicios web desde un programa de computadora. Algunas de las empresas que trabajan con JSON-based API son:

- Weather forecast.
  1. AccuWeather (accuweather.com)
  2. Dark Sky (darsky.net)
  3. OpenWeatherMap (openweather)
- Internet of Things (IoT).
  1. Adafruit (io.adafrui.com)
  2. Google Cloud Iot (cloud.google.com/iot)
  3. Thing Speak (thingspeak.com)
  4. Temboo (temboo.com)
- Sevicios de música y radios online
  1. Deezer(developers.deezer.com)
  2. Last.fm(last.fm)
  3. MusicBrainz(musicbrainz.org)
  4. Radio Browser(radio-browser.info)
  5. Spotify(developer.spotify.com)

6.11 Stack, heap y globales

Los datos dentro la memoria de corto plazo (RAM) se almacenan en una de las tres áreas de memoria que el programador tiene a su disposición. Se puede imaginar un enorme array que incluye todos los bytes de la memoria, desde el primer byte 0 de la RAM hasta 2048. Ya que, en el ATmega328 posee una capacidad de 2KB. El copilador y la rutina de la librería parten el array en tres áreas que usan diferentes tipos de data: Globales, heap y stack. También existe una zona libre de memoria sin usar. Como se muestra en la siguiente imagen, en la que se puede observar el manejo de la memoria de un microcontrolador Atmet ATmega328:

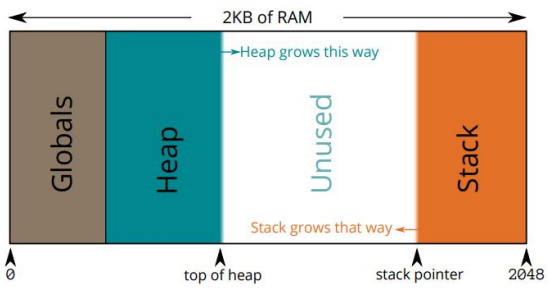


Figura 15: Memoria ATmega 328.

- Memoria Stack. Se utiliza para almacenar las variables denominadas automáticas, existen durante la ejecución de la función que las referencia. Los argumentos y variables locales son asignados y desasignados en forma dinámica, durante la ejecución de las funciones. Pero, de forma automática por el código generado por el copilador, debido a que, al terminar la ejecución libera el espacio. Se debe tomar en consideración que los fragmentos grandes de memoria, como arrays de gran tamaño, no deberían ser almacenados en el Stack, para prevenir desbordamientos. [12]

- Memoria Heap. El Heap a diferencia de la Stack no posee ninguna estructutra de asignación de espacios y su tamaño se ve limitado únicamente por el tamaño de la memoria virtual (RAM y espacio SWAP). Varía durante la ejecución, porque es usada comúnmente para variables de tamaño desconocido al momento de copilar. Es asignada durante la ejecución del programa mediante Malloc y Calloc (funciones de asignación de espacio de almacenamiento). La manipulación del heap (asignación, lectura, escritura) es más lenta que la del Stack. Además, esta memoria se mantiene en uso hasta que se libere explícitamente por el programa o sea liberada por el SO al termina la ejecución de este[12].

En resumen, las variables temporales deben ser almacenadas en el Stack, por su comodidad, rapidez y la facilidad de lectura, puesto que no hay que preocuparse por la asignación de la memoria. Para variables de gran tamaño, variables cuyo tamaño es dinámico o variables globales, es conveniente usar el Heap. Pero, hay que tener presente que es indispensable liberar su contenido una vez termina de usarse.

6.12 Serialización y deserialización

En ciencias de la computación se le conoce como serialización al proceso de convertir una estructura de datos en una serie de bytes, la cual puede ser almacenada o transmitida. Consecuentemente, la deserialización es el proceso de convertir una serie de bytes recibidos a una estructura de datos. En contexto de JSON, la serialización es la creación de un documento JSON a partir de un objeto en la memoria y la deserealización la operación contraria [11]. Como se puede evidenciar en la siguiente imagen:

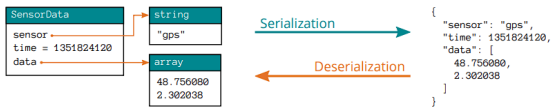


Figura 16: Proceso de serialización y deserialización de datos.

6.13 Multi-threading

Permite al CPU procesar varias tareas simultáneamente. Siendo más preciso procesar múltiples hilos al mismo tiempo. Cada hilo es una hembra de un proceso. Los programas pueden dividirse en procesos y estos, a su vez, en hilos individuales. Cada proceso consta al menos de un hilo. [13]

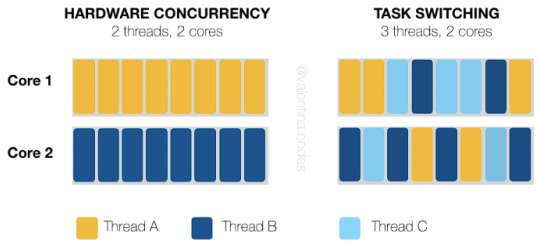


Figura 17: C++ Multithreading.

Mientras cada proceso solo puede cumplir una función al instante, el mecanismo de thread pools nos permite reciclar y reusar procesos y da al programa la ilusión de multitarea.

El objetivo final del multithreading es aumentar la velocidad de cálculo de un ordenador y, por lo tanto, su rendimiento. Para ello, se intenta optimizar la utilización de la CPU. En lugar de detenerse el sistema en un proceso durante mucho tiempo, aunque siga esperando datos, con el multithreading el sistema pasa rápidamente a la siguiente tarea. De este modo, apenas hay períodos de espera. [13]

Al mismo tiempo, el sistema reacciona más rápidamente a los cambios de prioridades. Cuando los usuarios o las aplicaciones necesitan repentinamente y sin planificación otra tarea, con la ayuda de los niveles de prioridad y los hilos cortos, el procesador también puede cambiar y dedicarse rápidamente a otra. [13]

CAPÍTULO 7

Diseño, ensamblaje, configuración y control del robot serpiente

7.1. Diseño y ensamblaje del robot

7.1.1. Primera iteración

El diseño del robot serpiente se desarrolla a partir de la técnica de lluvia de ideas, que toma como referencia las tesis que se muestran en la sección de antecedentes. En la primera fase, se procedió a elaborar bocetos del robot, en los que se definió su forma y las dimensiones aproximadas. Luego, se empleó el programa Autocad *Inventor* para plasmar las ideas preliminares del sistema. Así surgió el diseño introductorio de la articulación, con dos grados de libertad que se fueron modificando para brindarle las dimensiones con base a los servomotores Hitec D950TW. Este fue un proceso interactivo, ya que, a medida que se elaboraban los diseños en la cortadora laser se iban modificando hasta obtener un modelo funcional, tal como se muestra en la Figura 18.

Posteriormente, se trabajó en el ensamblaje del robot. Para el acople de cada articulación se usaron tornillos M4. Además, se instalaron los motores Hitec y se fijaron a la pieza con tornillos perforadores. Seguidamente, se procedió con el cableado y la alimentación. En este punto se optó por usar dos baterías de litio NCR18650B de 3.7 voltios y 3400mAh [14] , conectadas en serie. También se instaló un regulador de voltaje, porque al cargar las baterías de litio en su totalidad brindaron un voltaje máximo de 4.2 voltios, es decir, medio voltio más que lo calculado. Al ser superior al voltaje permisible de los motores, esto puede llegar a dañarlos, por ello se procedió a regular el voltaje de salida. Finalmente, se instaló al frente el microcontrolador ESP32 [15] el que se fijó con tornillos. El ensamblaje se muestra en la Figura 20.

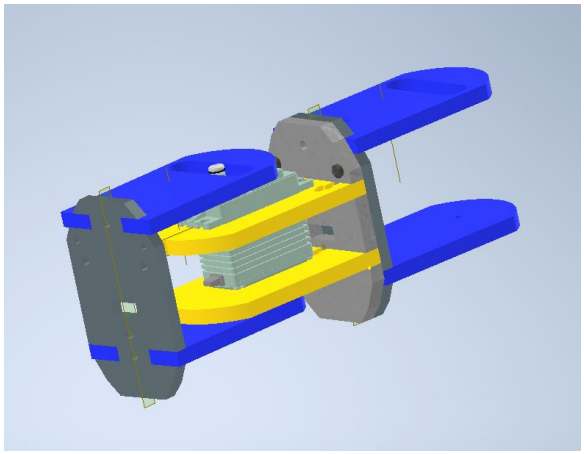


Figura 18: Diseño de articulación.

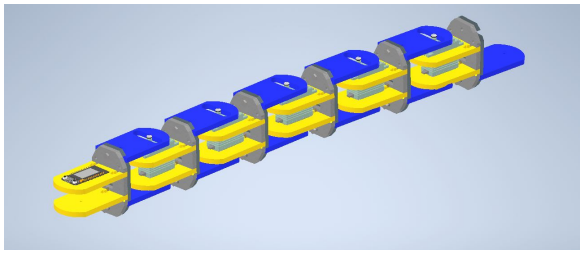


Figura 19: Diseño primera iteración.

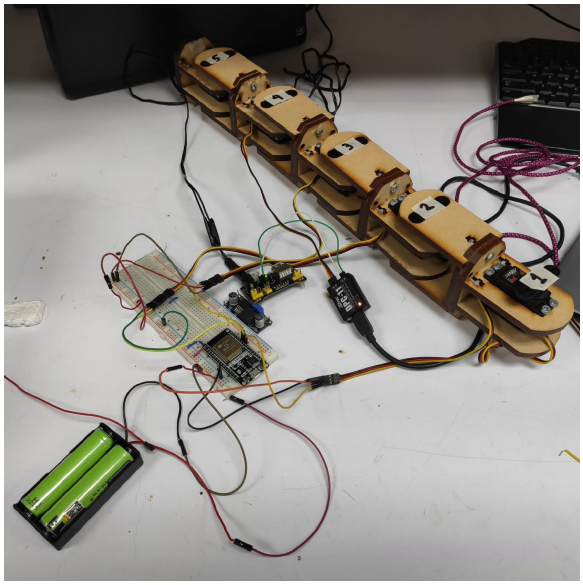


Figura 20: Robot físico.

7.1.2. Segunda iteración

Al tener el robot armado y funcionando se observaron problemas en el diseño. El primero fue el ensamblaje entre el motor y la pieza superior de la articulación. Al estar acoplado por un tornillo, se observó cierto desgaste en el orificio durante el movimiento, lo que ocasionó holgura entre el motor y la pieza superior, que impidió el movimiento en la articulación del robot. Entonces, se procedió a proveer a los motores de un acople que se pudiera introducir dentro de la pieza y transmitir torque al acople superior, sin provocar desgaste en la articulación.

El segundo se relaciona con el desplazamiento del robot. El exceso de fricción en los puntos de apoyo impidió el desplazamiento hacia delante. Por esta razón, se recurrió a la alternativa de dotar al robot de un sistema de ruedas pasivas que permitiera un movimiento ondulante similar al de una serpiente. Para ello, se procedió a diseñar una pieza capaz de introducir un eje en su interior con dos ruedas en sus extremos. Además, con la capacidad de atornillarse a los puntos de contacto con el suelo del robot. Como se puede observar en la Figura 21. Luego, se eligieron los rodamientos y se forraron de termoencogible, esto garantizó el desplazamiento frontal del robot. El resultado final se muestra a continuación:

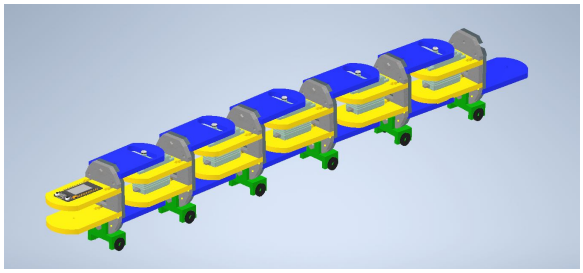


Figura 21: Diseño final en inventor.

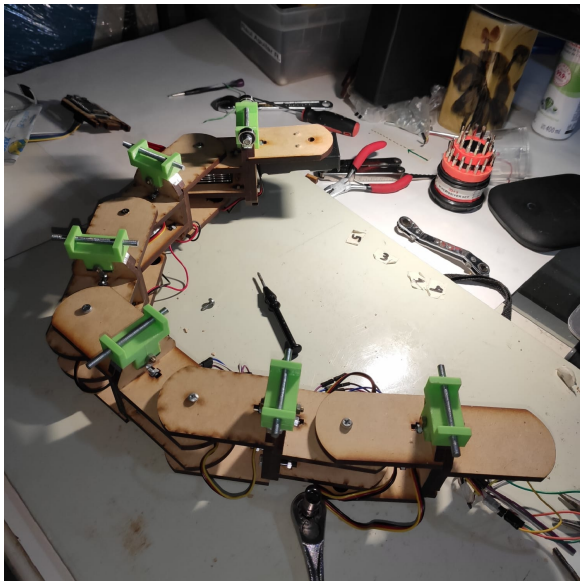


Figura 22: Instalación de la pieza.

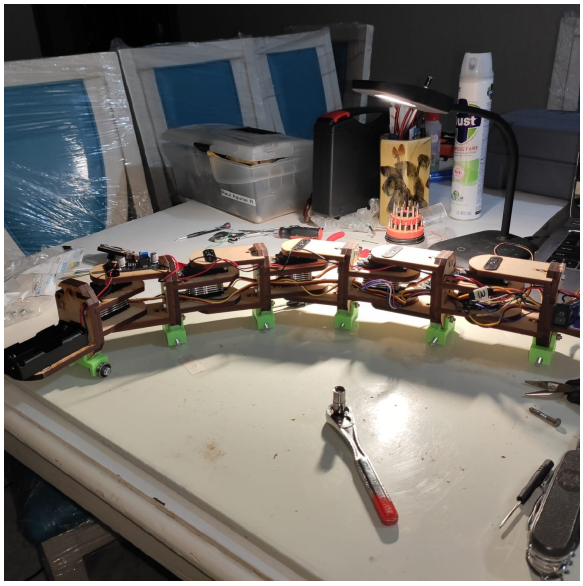


Figura 23: Ensamblaje final.

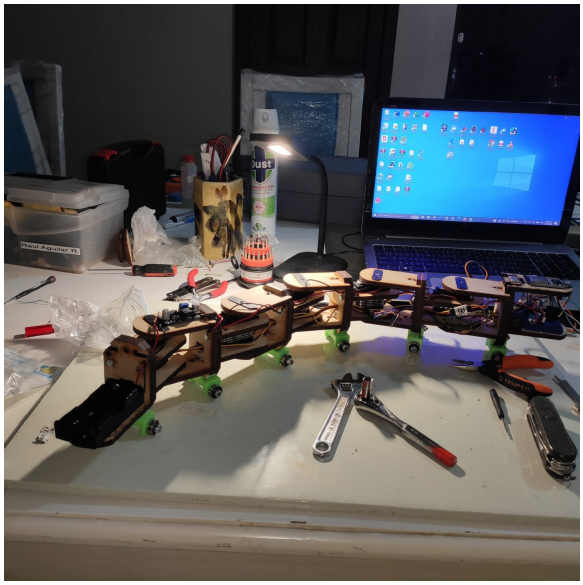


Figura 24: Resultado.

7.1.3. Diseño de placa

Después de probar todo el sistema con una Protoboard y su funcionamiento, se procedío a diseñar una placa de cobre para posicionar cada uno de los elementos. Con ello se buscó mejorar la estética del cableado de este prototipo, facilitar la manipulación de los componentes y obtener mayor orden.

La fabricación de esta placa se realizó por medio de una fresadora, propiedad de la universidad, la que posee un tamaño de 70x50 mm. Luego, se ensambló en la parte frontal del robot serpiente y se conectó a cada uno de los servomotores.

7.1.4. Esquemático

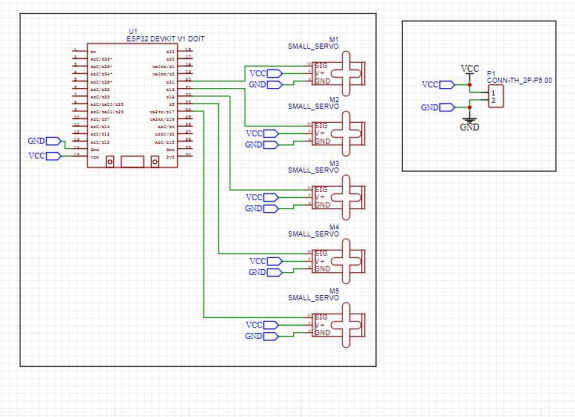


Figura 25: Esquemático.

La figura anterior muestra las conexiones de los componentes dentro de la placa. En el lado izquierdo de la imagen se observa el ESP32 conectado a sus seis voltios y a los servomotores que componen las articulaciones del robot. En el recuadro derecho se visualizan las conexiones de alimentación y tierra de la bornera al regulador de voltaje, la cual energiza el circuito completo.

7.1.5. Wire Routing

Una fase primordial durante el desarrollo del robot consistió en realizar el cálculo del tamaño de las pistas y la separación entre ellas. Para ello se tomaron en cuenta los siguientes parámetros:

- Paso de corriente de 2251.8 mA.
- Aumento máximo de temperatura de 30°C.
- Un espesor de cobre 1 oz/ft^2 .
- Temperatura ambiente 25°C.

- Una longitud promedio de conductor de 1 pulgada.
- Pico de voltaje de 6 voltios.

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)	2251.8	mA	Required Trace Width	1.26	mm	19.07	mil
Temperature Rise (max. 100°C)	30	°C	Cross-section Area	66.68	mil²	25.63	mil²
Cu thickness	1	oz/ft²	Resistance	0.01	Ω Ohms	0.03	Ω Ohms
Ambient Temperature	25	°C	Voltage Drop	0.03	Volts	0.07	Volts
Conductor Length	1	inches	Loss	0.06	Watts	0.15	Watts
Peak Voltage	6	Volts	Required Track Clearance	0.61	mm		

Figura 26: Cálculo de pistas.

Como se muestra en la imagen anterior, el ancho mínimo de pistas es de **1.26 mm** mientras que la separación fue de **0.61 mm.** Se actualizaron estas reglas en el programa de diseño y se procedió rutear la PCB, lo que dio como resultado la placa de una sola cara que se observa a continuación:

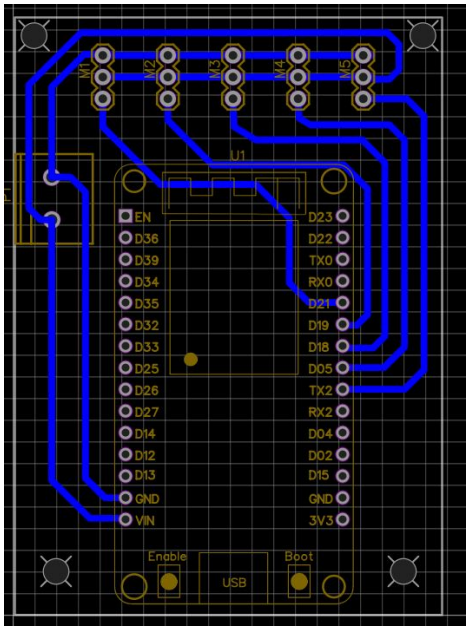
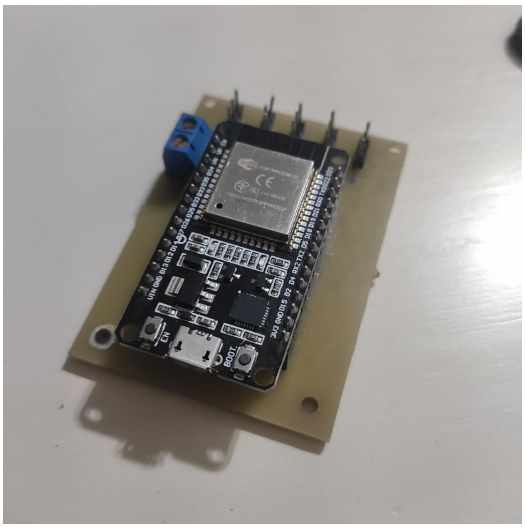
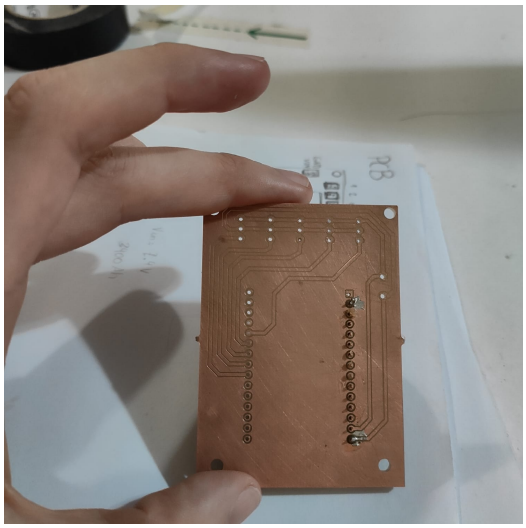


Figura 27: Diseño de placa.

Los orificios que se efectuaron en las esquinas de la PCB, permitieron posicionarla en la pieza frontal del robot, a través de cuatro tornillos M4. Los que se taladraron y se ajustaron con ocho tuercas métricas. Posteriormente, se conectó la alimentación a la bornera y se unieron los servomotores a los pines. Por último, se instaló el microcontrolador a la placa. El resultado puede observarse en la Figura 29.



(a) Placa de cobre parte superior.



(b) Placa de cobre parte inferior.

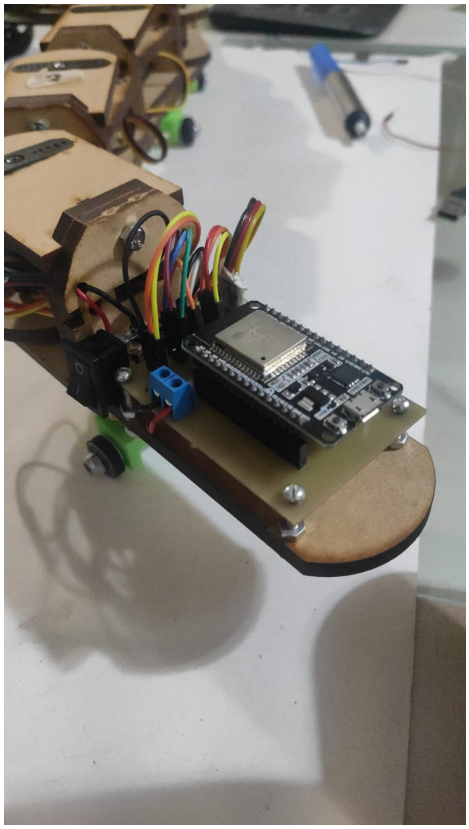


Figura 29: Resultado final.

7.2. Configuración y control de los servos

7.2.1. Fijación de parámetros y testeo

Parte de la configuración de cada motor por medio del programador DPC-11. Este se conectó primero a la computadora, después al servomotor y luego a una batería de 5 voltios. Se procedió a abrir el programa Hitec, se conectó al motor y se seleccionó la opción *Setting*, se ajustó el valor del centro y después se programaron los extremos derecho e izquierdo. Finalmente, se guardó la configuración, tal como se muestra en la Figura 30.

```
Servo1
CENTER:2474
LEFT:1762
RIGHT:3588

Servo2
CENTER: 2421
LEFT: 1731
RIGHT:2876

Servo3
CENTER:2574
LEFT:1483
RIGHT:3526

Servo4
CENTER:3464
LEFT:2102
RIGHT:4548

Servo5
CENTER:2876
LEFT:1390
RIGHT:3464
```

Figura 30: Posiciones.

Seguidamente, se usó un programa de prueba de arduino para probar la configuración de los motores, *Sweep*, en donde estos se definen al utilizar el objeto *Servo* y se les asigna un pin de salida del microcontrolador ESP32, por medio de la instrucción *myservo.attach*. Se asignaron los pines 17,5,18,19 y 21 a cada motor.

Después se le designó una frecuencia estándar de 50Hz a cada servo. Por último, se definió una condición *for* que aumenta el ángulo del servomotor de 0 a 180, con un incremento de 1, para luego cargar este valor al motor por medio de la instrucción *myservo.write* y hacer lo mismo, pero de forma contraria. El resultado es un movimiento coordinado en cada articulación, en donde el servo llega al ángulo máximo de 180° y regresa a su mínimo. Así lo muestra la Figura 32

Figura 31: Interfase.

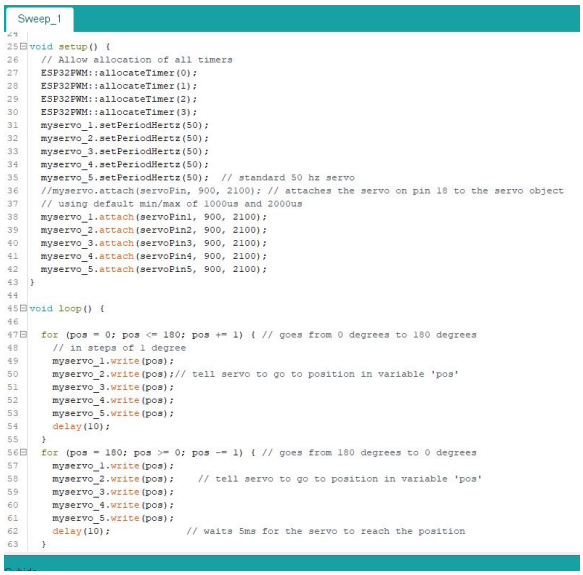


Figura 32: Programa Sweep.

7.2.2. Implementación de rutinas básicas

Se elaboraron dos rutinas básicas. La primera fue una posición neutral sin movimiento, en la que el robot estuviera totalmente vertical y que permitiera corregir cualquier imperfeción en su postura y dar lugar a la segunda rutina, el movimiento ondulatorio con desplazamiento frontal.

Se logró al mandar una onda coseoidal a través del cuerpo del robot, lo que dio como resultado un desplazamiento propulsado en dirección frontal. Los motores reciben el comando del ángulo, por lo que si este es de 90 grados el robot serpiente se modifica en línea recta; pero, si su ángulo es menor o mayor a 90 grados se escribirá a los servos y la articulación se curvará a la derecha o a la izquierda. Esto se debe al comando que se envía a cada servo:

$$sn.mwrite(90 + amplitud * cos(frecuencia * counter * 3.14159/180 - n * lag)) \quad (6)$$

En donde: n es el número de segmentos por cada servomotor, en este caso cinco. La amplitud determina que tan amplia es la onda, la frecuencia establece la velocidad de la serpiente y el counter permite el conteo de la rutina de ondulación y *lag* es la constante angular que da un retardo entre los segmentos. Por último, se convierte el valor de degrés a radianes por medio de 3.14159/180.

Se crea una condición *for* donde el counter aumente su valor desde 0 hasta 359. Esto le brinda un movimiento ondulatorio en dirección frontal.

CAPÍTULO 8

Modelo cinemático del robot serpiente

Fue necesario plantear el modelo cinemático del robot serpiente para simularlo y animarlo dentro del espacio de tarea. Esto, con el fin de comprender de mejor manera las posiciones y movimientos que sería capaz de realizar, a partir de simularlo como un manipulador serial con 5 actuadores. De igual forma, se validaron los comandos e instrucciones que se le enviaron para garantizar su funcionamiento. Esto fue posible por medio del software de Matlab.

A través de la convención de Denavit Hartenberg se describió al robot de forma general, a partir de una secuencia articulada de juntas con cuatro parámetros: θ_j, d_j, a_j, α_j que usan el Robotics system toolbox de Matlab. Como resultado se obtuvo la siguiente matriz:

ϕ(rad)	d (m)	a (m)	α(rad)
q1 + $\pi/2$	0	100	$\pi/2$
q2 - $\pi/2$	0	100	0
q3 + $\pi/2$	0	100	0
q4 - $\pi/2$	0	100	0
q5 + $\pi/2$	0	100	0

Cuadro 1: Matriz de parámetros de robot serpiente.

Para comprobar los valores anteriores, se generó una simulación gráfica del modelo como se muestra en la Figura 33. En la que se pueden observar que las juntas se representan con pequeños cilindros rojizos y los eslabones con líneas azules. A un costado de la representación gráfica se posicionan los controles de los ángulos θ_j, que permiten experimentar con diferentes valores para obtener las distintas orientaciones del robot.

Para el modelo planteado con la convención de DH se tomó el origen del robot (el primer actuador) y el ultimó actuador como el efector final. De acuerdo con este estudio, podemos deducir que el robot serpiente puede realizar una marcha serpenteante a partir de los valores

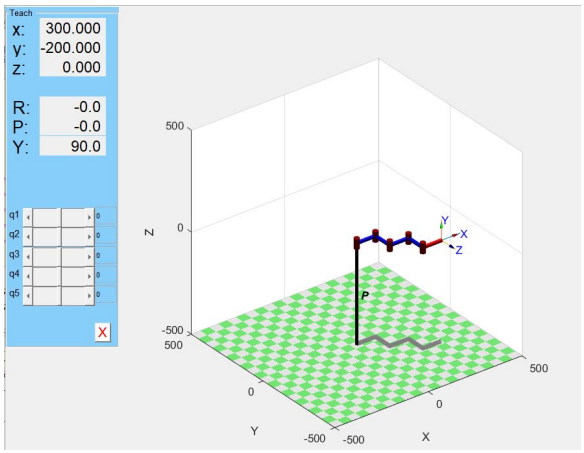


Figura 33: Simulación DH.

calculados. Por lo tanto, los valores calculados son correctos y pueden ser utilizados a través del código para la ejecución del robot.

CAPÍTULO 9

Implementación del control remoto del robot vía Wifi por medio del protocolo MQTT

9.0.1. Introducción al ecosistema

Uno de los objetivos principales de este trabajo de graduación, es implementar una conexión inalámbrica que permita la integración del robot serpiente con el ecosistema Robotat, para controlarlo de forma remota. El objetivo se alcanzó por medio de tres tareas: la conexión cliente-servidor entre el ESP32 y el sistema de captura de movimiento Optitrack, del que se recibieron los datos de posición del marcador y se guardaron en el microcontrolador. La siguiente tarea fue la conexión cliente-servidor entre el ESP32 y Matlab, que envió datos de rutinas al robot. Por último, se enviaron los datos del ESP32 a los motores.

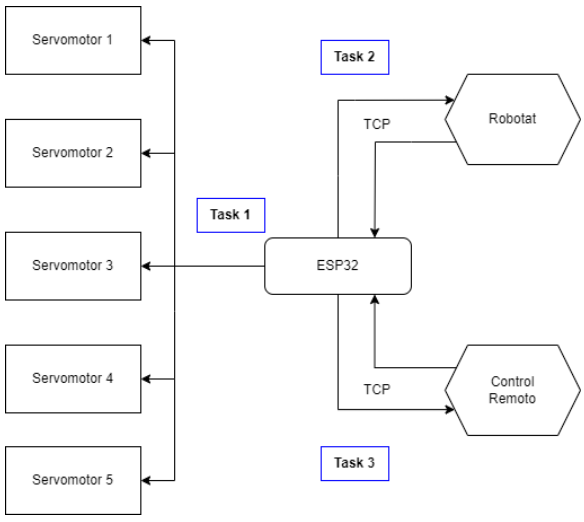


Figura 34: Tareas para la implementación

Para poder cumplir con las tareas se obtuvo una *MAC ADDRESS*, luego se le asignó una dirección IP estática, que permitió una conexión directa con la red del ecosistema.

9.0.2. Datos de Robotat

Primero se estableció la conexión con Optitrack y el ESP32. Para ello, fue necesario utilizar las librerías WiFi.h [16] y ArduinoJson.h[17] para habilitar la conexión WiFi y la transmisión de Json. Se configuró el ESP32 como cliente, con el fin de conectarlo a la red local del Robotat y recibir los datos de la posición del marcador en formato JSON. La comunición fue mediante el protocolo TCP.

Para ello, se estableció la IP: 192.168.50.200. Posteriormente, el robotat envió un mensaje de bienvenida. Esto confirmó la conexión servidor-cliente. Luego, se detalló el marcador en uso por medio de la función *cliente.write*, la cual le escribió al servidor del cliente al que se estaba conectando. En nuestro caso, al trabajar con el marcador cinco, se configuró como ciento cinco, por sintaxis del robotat, como se muestra en la Figura 35.

```
void loop()
{
  if (!MsgSent) {
    client.write("105");
    MsgSent = true;

  }

  if ( client.available() ) {

    c =  client.read();
    msg = msg + c ;


    if ( c == '}') {
      Serial.println(msg);
      Json();
      msg = "";
      delay(1000);

      while ( client.available()) {
        client.read();
      }
      MsgSent = false;
    }

  }

}
```

Figura 35: Parsing

Por medio de las funciones *cliente.read* y *cliente.print* fue posible recibir la información del robot y guardarla en un buffer. Luego, se deserializó con el documento de JSON y fueron guardadas en sus correspondientes variables(x,y,z,n,ex,ey,ez). Al final, se mostraron estos datos en el monitor serial del arduino. El resutado develo las posiciones dentro del Robotat del marcador cinco, como se puede observar a continuación.

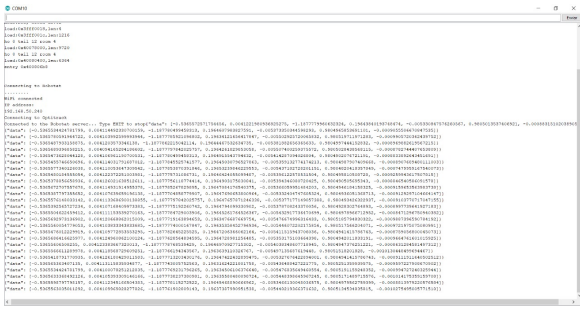


Figura 36: Datos recibidos por el ESP32

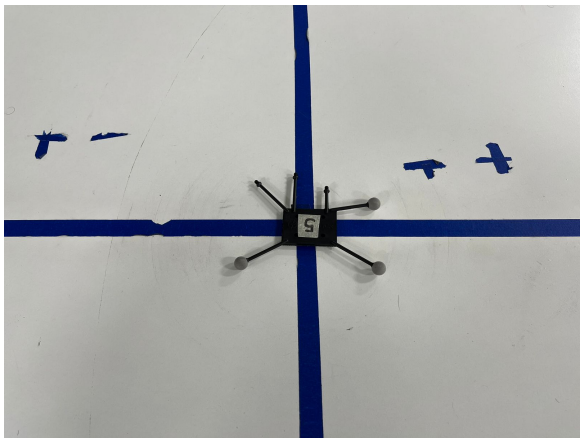


Figura 37: Marcador

9.0.3. Control remoto desde Matlab al ESP32

La segunda tarea consistió en establecer una conexión entre Matlab y el ESP32. Se usó la característica del ESP, de establecer un web server o servidor web con el fin de almacenar los datos de las rutinas ya calculadas en Matlab y enviarlas a cada servomotor.

En Matlab, fue necesario realizar la conexión por medio de un objeto de tipo TCP o protocolo de control de transmisión de datos, el que se concatenó a un Try-catch, ya adentro se creó *tcpclient*, en el que se describió la IP: 192.168.50.43 y un puerto: 8888. Con el fin de establecer la conexión con el servidor, para que, si existía algún problema, mostrarlo y terminar el programa. Luego, se definieron los valores que se deseaban enviar y se serializaron en JSON de cinco valores para los servos. Se envió el JSON por medio de la función *writeline* al servidor del ESP.

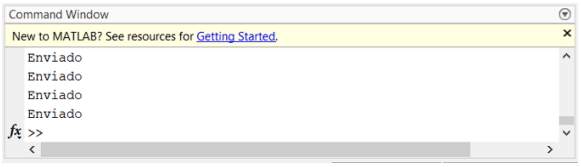


Figura 38: Datos enviados por Matlab

Se recibió el JSON en el ESP32 como se muestra en la Figura 39, al usar las librerías detallas con anterioridad. Se almacenó el mensaje en un buffer para luego deserializarlo y colocar cada valor en una variable independiente en el ESP32. Luego se escribió a cada motor con la variable correspondiente, se ejecutó y se varió el ángulo en las articulaciones del robot serpiente.

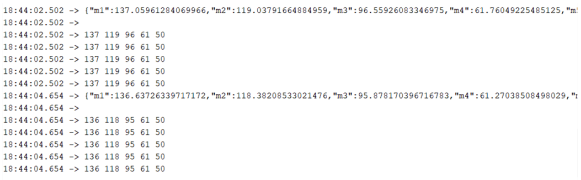


Figura 39: Datos recibidos por el servidor

CAPÍTULO 10

Conclusiones

- Es posible la elaboración del diseño del sistema mecánico de un robot serpiente con piezas de MDF de 6 mm de grosor en las articulaciones. Así también, el ensamblaje de cada una con tornillos M4 de cabeza hexagonal y la configuración de 5 motores Hitec, y que fluya su desplazamiento serpentoide por el ecosistema.
- La electrónica del robot serpiente a través de un microcontrolador ESP32, que utiliza dos baterías de litio, un regulador de voltaje, una PCB y jumpers para cada una de las conexiones, es idónea para la alimentación de los motores y el microcontrolador.
- Por medio del control remoto Wifi, que utilice Matlab como cliente y ESP32 como servidor, es posible tanto el envío de rutinas básicas -en formato JSON- al robot serpiente como su ejecución.

CAPÍTULO 11

Recomendaciones

- Se sugiere dotar al robot serpiente de más servos para que ejecuten una cantidad mayor de rutinas.
- Se recomienda integrar un módulo de carga en el sistema del robot serpiente para energizarlo de forma adecuada. Puesto que, anteriormente para cargar las baterías se debían extraer del robot, lo que lo hacía poco eficiente.
- Se debe agregar mayor peso al sistema, ya que de esta manera habrá más puntos de contacto que incrementen los coeficientes de fricción y que den como resultado un desplazamiento con mayor velocidad.

CAPÍTULO 12

Bibliografía

[1] H. Y. Shigeo Hirose, "*Machine Design of Biologically Inspired Robots*," *IEEE Robotics Automation Magazine*, págs. 1-11, 2009.

[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," pág. 119, 2022.

[3] C. P. Montoya, "Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica," pág. 74, 2022.

[4] F. G. Hernández, "Diseño e Implementación de una plataforma autónoma para el control de sistemas servomecánicos.," pág. 107, 2013.

[5] A. H. Cordero, "Serpientes: atención y cuidados en cautiverio," 2019.

[6] S. H. H. Sánchez Amezcua Hernández Hernández Carlos, "Diseño y construcción de un robot móvil tipo Snake," pág. 12, 2015.

[7] Y. C. Estrada, "Sistema de Captura de Movimiento para ambientes tridimensionales," pág. 23, 2006.

[8] "*About Optitrack*," 2022.

[9] K. L. y F.C. Park, "*Modern Robotics*," 2017.

[10] P. Corke, "*Robotics, Vision and Control*," 2017.

[11] B. Blanchon, "*Mastering Arduino Json6*," 2020.

[12] T. A. Vidal, "Memoria, Stack, Strings," 2010.

[13] D. G. IONOS, *Multithreading*, 2021. dirección: https://www.ionos.es/digitalguide/servidores/know-how/explicacion-del-multithreading/.

[14] E. C. of Panasonic Group, *NCR18650B Standard data*, 2012. dirección: https://www.shoptronica.com/files/Panasonic-NCR18650.pdf.

[15] Epressif, *ESP32*, 2022. dirección: https://www.espressif.com/en/products/socs/esp32.

[16] Arduino, *Wifi*, 2022. dirección: https://www.arduino.cc/reference/en/libraries/wifi/.

[17] A. Json, *JSON library for embedded C++*, 2022. dirección: https://arduinojson.org/v6/doc/installation/.

CAPÍTULO 13

Anexos

13.1. Planos de construcción

https://drive.google.com/drive/folders/1uHmpbodoXOkLTr6sbYmtHhjAjbzD33JR?usp=sharing

13.2. Código

https://drive.google.com/drive/folders/1VYSAruGEJF5UWn6vVgYBjjkbEwCeZd49?usp=sharing

13.3. Documentación

https://drive.google.com/drive/folders/1_hSfJ5dIcj9GLMx9rn8OYWpr0CZN0wm4?usp=sharing