



**INSTITUTO POLITÉCNICO NACIONAL**  
**Escuela Superior de Ingeniería Mecánica y Eléctrica**  
**Unidad Zacatenco**



**UNIDAD DE APRENDIZAJE:** Microprocesadores

**PROFESOR:** Roberto Galicia Galicia

**ALUMNO:** Díaz Estrada Jesús Adrián

**GRUPO:** 6CM4

**TRABAJO:** Proyecto Final Brazo para Dibujo

## **Objetivo general.**

El objetivo general de la realización de este proyecto es lograr hacer un brazo robótico capaz de moverse en dos dimensiones, simulando las articulaciones de un brazo real al dibujar implementando lo visto en la materia de microprocesadores.

## **Objetivo particular.**

El objetivo particular en la realización de este proyecto es poder implementar el uso de la raspberry para un proyecto innovador, en este caso específico utilizando los instrumentos proporcionados por raspberry para controlar los servomotores del brazo y lograr que éste lleve a cabo rutinas específicas para dibujar, a través de un servidor.

## **Introducción.**

El brazo robótico realizado para este proyecto está conformado por servomotores, que son actuadores rotativos que ofrecen un control preciso en términos de posición angular, aceleración y velocidad, capacidades que un motor eléctrico común y corriente simplemente no puede igualar, también incluyendo estructuras de madera para su correcto armado.

Se implementó lo aprendido en la materia para poder controlar el brazo a través de comandos introducidos directamente a la raspberry. Además se llevó a cabo la utilización de la raspberry Pico w para la creación de un servidor capaz de controlar el mismo brazo con ayuda de sliders, debido a que la pico w cuenta con módulo wifi y facilita la creación de un servidor para ayudarnos a controlar los servomotores.

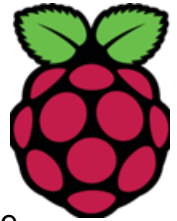
Para la realización de proyectos se utilizó programación en distintos lenguajes, en este caso se utilizó principalmente python, con su entorno de desarrollo Thonny, html, implementado principalmente para la creación de interfaces de la páginas web y javascript para la conexión entre cliente y servidor, que fueron los pilares necesarios para la conformación del proyecto.

## **Marco Teórico.**

Raspberry.

La Raspberry Pi es una computadora de bajo costo y con un tamaño compacto, del porte de una tarjeta de crédito, puede ser conectada a un monitor de computador o un TV, y usarse con un mouse y teclado estándar. Es un pequeño computador que corre un sistema operativo linux. Es capaz de hacer la mayoría de las tareas típicas de un computador de escritorio, desde navegar en internet, reproducir videos en alta resolución, manipular documentos de ofimática, hasta reproducir juegos.

Además la Raspberry Pi tiene la habilidad de interactuar con el mundo exterior, puede ser usada en una amplia variedad de proyectos digitales, desde reproductores de música y video, estaciones meteorológicas, hasta cajas de aves con cámaras infrarrojas.



La Raspberry Pi fue creada en febrero del 2012 por la Raspberry Pi Foundation, originalmente pensado para promover y enseñar las ciencias básicas de la computación en las escuelas y universidades de Reino Unido. Originalmente lanzaron dos modelos, el Modelo A y el Modelo B. Al poco tiempo de su lanzamiento ya había una comunidad formada por miles de “locos por la tecnología” que compraron una Raspberry para empezar a experimentar con nuevos proyectos.

Gran parte de la popularidad del producto fue debido a su bajo costo, a su versatilidad y facilidad de modificar para diferentes proyectos y a la capacidad de ejecutar el sistema operativo Linux, el cual es un sistema operativo muy popular entre los desarrolladores por ser de software libre.

#### Especificaciones Técnicas:

Raspberry Pi 4 es una actualización mayor de lo que podemos ver a primera vista, el cambio de procesador a un ARM Cortex-172 con cuatro núcleos a 1,5 GHz también implicaba pasar de los 40 nm a los 28 nm. En consecuencia, todos los componentes y la potencia del dispositivo han cambiado.



--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

### *Especificaciones Técnicas*

RASPBERRY PI 4	
PROCESADOR	ARM Cortex-A72
FRECUENCIA DE RELOJ	1,5 GHz
GPU	VideoCore VI (con soporte para OpenGL ES 3.x)
MEMORIA	1 GB / 2 GB / 4 GB LPDDR4 SDRAM
CONECTIVIDAD	Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet
PUERTOS	GPIO 40 pines 2 x micro HDMI 2 x USB 2.0 2 x USB 3.0 CSI (cámara Raspberry Pi) DSI (pantalla tácil) Micro SD Conector de audio jack USB-C (alimentación)

Además de mejorar su potencia, un cambio interesante y a tener en cuenta en esta nueva Raspberry Pi 4 es la puesta al día de sus conexiones. Raspberry Pi 4 viene con Bluetooth 5.0 y Wi-Fi 802.11ac para las conexiones inalámbricas. También se ha cambiado el conector microUSB de alimentación por un USB-C que suma 500 mA extra de energía para alcanzar un total de 1.2 A.

Algunos detalles extra a tener en cuenta son por ejemplo el soporte para doble monitor con resolución 4K. La opción para reproducir vídeo 4K a 60 fps en HEVC o los gráficos VideoCore VI, compatibles con OpenGL ES 3.x ,además de que cuenta que Raspberry Pi 4 es compatible con todos los productos de Raspberry Pi anteriores.

#### Servidores.

Un servidor es un sistema que proporciona recursos, datos, servicios o programas a otros ordenadores, conocidos como clientes, a través de una red. En teoría, se consideran servidores aquellos ordenadores que comparten recursos con máquinas cliente. Existen muchos tipos de servidores, como los servidores web, los servidores de correo y los servidores virtuales.

Un sistema individual puede, al mismo tiempo, proporcionar recursos y usar los de otro sistema. Esto significa que todo dispositivo podría ser a la vez servidor y cliente.

Los primeros servidores eran mainframes o microcomputadoras, que se denominan así por ser mucho más pequeñas que los equipos de mainframe. Sin embargo, conforme progresaba la tecnología, terminaron superando en tamaño a los ordenadores de sobremesa, por lo que el término microcomputadora resultaba un tanto inapropiado.

Inicialmente, dichos servidores estaban conectados a clientes que no realizaban ninguna computación real, y se les conocía como terminales. Estos terminales (también llamados dumb terminals), existían simplemente para aceptar entradas a través de un teclado o lector de tarjetas y devolver los resultados de cualquier cálculo a una pantalla o impresora. La computación real se efectuaba en el servidor.

Más tarde, los servidores pasaron a ser sistemas individuales de gran potencia que se conectaban a un conjunto de ordenadores cliente menos potentes a través de una red. A esta arquitectura de red se la conoce como el modelo cliente-servidor, en el que tanto el ordenador cliente como el servidor poseen potencia computacional pero determinadas tareas se delegan a los servidores.

La definición del concepto de servidor ha ido evolucionando con el avance de la tecnología. Hoy en día, un servidor puede no ser más que un software que se ejecuta en uno o más dispositivos informáticos físicos. A tales servidores se les suele adjetivar como virtuales. Originalmente, los servidores virtuales se usaban para incrementar el número de características que un servidor individual de hardware podía efectuar. Actualmente, los servidores virtuales se suelen ejecutar en la nube, es decir, dentro de un hardware que pertenece a un tercero y al que se puede acceder a través de internet.

Un servidor puede estar diseñado para realizar una sola tarea, como un servidor de correo, que acepta y almacena mensajes de correo electrónico y, luego, se los proporciona a un cliente que los solicita. Los servidores también pueden realizar más de una tarea, como un servidor de archivos e impresión que almacena archivos y acepta trabajos de impresión de los clientes, para luego enviarlos a una impresora conectada a la red.

Cómo funciona un servidor.

Para que un dispositivo trabaje como un servidor, debe estar configurado para escuchar las solicitudes de los clientes en un entorno de red. Esta funcionalidad puede existir como parte del sistema operativo: en forma de aplicación instalada, un rol o una combinación de ambos.

Por ejemplo, el sistema operativo Windows Server de Microsoft proporciona características necesarias para escuchar y responder solicitudes de los clientes. Además, los roles o servicios instalados incrementan el número de tipos de solicitudes del cliente a los que puede responder el servidor.

Cuando un cliente pide datos o una funcionalidad de un servidor, lo hace enviando una solicitud a través de la red. El servidor recibe dicha solicitud y responde con la información correspondiente. Este es el modelo de solicitud y respuesta de la conexión cliente-servidor, lo que también se conoce como el modelo de llamada y respuesta.

Hay muchos tipos de servidores que realizan diferentes funciones. En la mayoría de las redes podemos encontrar al menos uno de los tipos de servidores más comunes:

**Servidores de archivos:** Almacenan y distribuyen ficheros que varios clientes o usuarios pueden compartir. El almacenamiento centralizado de archivos ofrece soluciones de copia de seguridad o tolerancia a fallos de forma más sencilla que tratar de proporcionar seguridad e integridad a los archivos en todos y cada uno de los dispositivos de la organización.

**Servidores de impresión:** Permiten la gestión y distribución de la funcionalidad de imprimir documentos. Hoy en día, algunas impresoras de alta gama y gran tamaño vienen con su propio servidor de impresión incorporado, ahorrando la necesidad de instalar uno en un equipo separado. Este servidor de impresión interno hace que la impresora responda también a las solicitudes de impresión de los clientes conectados.

**Servidores de aplicaciones:** Este tipo de servidores sirve para ejecutar aplicaciones de forma remota, en lugar de que los equipos cliente lo hagan localmente. Los servidores de aplicaciones a menudo ejecutan software que hace un uso intensivo de los recursos, y lo comparten para una gran cantidad de usuarios.

**Servidores DNS:** Servidores del sistema de nombres de dominio (DNS) son servidores de aplicaciones que proporcionan funcionalidades de resolución de nombres a los equipos cliente. La resolución de nombres consiste en convertir nombres fácilmente comprensibles por los humanos en direcciones IP legibles por las máquinas. El sistema DNS es una base de datos ampliamente distribuida de nombres y otros servidores DNS a los que se puede consultar para obtener un nombre de equipo desconocido.

**Servidores de correo:** Un tipo muy común de servidor de aplicaciones. Los servidores de correo reciben mensajes de correo electrónico que se remiten a un usuario y los almacenan hasta que un cliente los solicite en nombre de dicho usuario.

**Servidores web:** De los tipos de servidores más abundantes en el mercado. Un servidor web es un tipo especial de servidor de aplicaciones que aloja programas y datos solicitados por los usuarios a través de internet o en una intranet. Los servidores web responden a las solicitudes de páginas web u otros servicios basados en la web que llegan de los navegadores que se ejecutan en los ordenadores cliente. Entre los servidores web que podemos encontrar más frecuentemente tenemos servidores Apache, Microsoft Internet Information Services (IIS) y Nginx.



**Servidor de base de datos:** La cantidad de datos utilizados por empresas, usuarios y otros servicios es sobrecogedora. Estas bases de datos deben poder ser accesibles por parte de múltiples clientes en cualquier momento y, generalmente, exigen cantidades extraordinarias de espacio de almacenamiento. Estos servidores ejecutan aplicaciones y responden a numerosas solicitudes de clientes. Los servidores de bases de datos más frecuentes son Oracle, Microsoft SQL Server, DB2 e Informix.

**Servidores virtuales:** A diferencia de los servidores tradicionales, que se instalan como una dupla de sistema operativo y máquina de hardware, los servidores virtuales solo existen según los parámetros establecidos en un software especializado denominado hipervisor. Cada hipervisor puede ejecutar cientos o incluso miles de servidores virtuales a la vez. El hipervisor presenta el hardware virtual al servidor como si de una máquina física se tratase.

**Servidores proxy:** Actúa como intermediario entre un cliente y un servidor. A menudo se emplean para aislar a clientes o servidores por motivos de seguridad. Un servidor proxy toma la solicitud del cliente pero, en lugar de responderle directamente, traslada la solicitud a otro servidor o proceso. El servidor proxy recibe la respuesta del segundo servidor y, luego, responde al cliente original como si lo hiciera por sí mismo. De este modo, ni el cliente ni el servidor que se comunican realmente se conectan entre sí.

**Servomotores.**

Estos elementos son actuadores rotativos que ofrecen un control preciso en términos de posición angular, aceleración y velocidad, capacidades que un motor eléctrico común y corriente simplemente no puede igualar. En resumen, un servomotor toma un motor estándar y lo eleva a un nuevo nivel de rendimiento al integrarlo con un sensor para la retroalimentación de posición.

Los servomotores no son una categoría específica de motor, son una combinación ingeniosa de componentes específicos, que incluyen un motor de corriente continua o alterna. Estos componentes se ensamblan de tal manera que el servomotor resultante es perfectamente adecuado para su uso en un sistema de control de bucle cerrado.

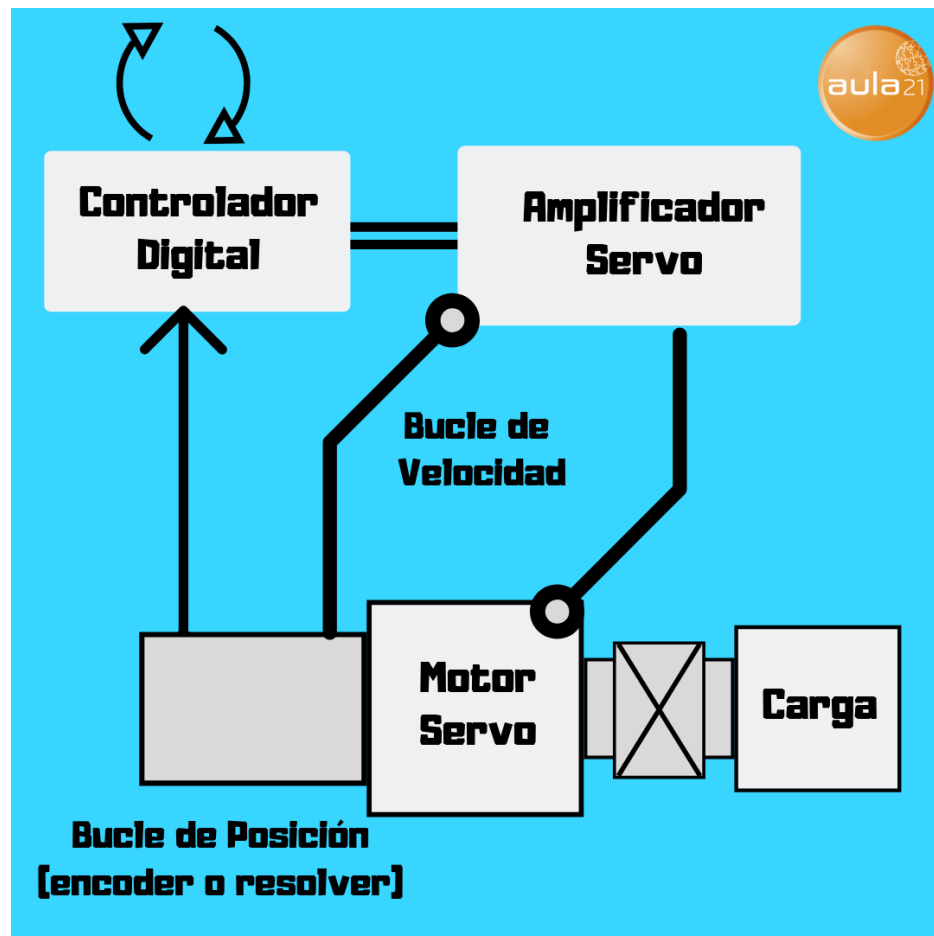
Existen varios tipos de servomotores, algunos de los cuales utilizan motores de corriente continua y detectan posiciones a través de un potenciómetro. Estos servomotores también utilizan un control de gran potencia, lo que significa que el motor se mueve a la velocidad máxima hasta que se detiene en la posición designada.

Por otro lado, los servomotores destinados al uso industrial están equipados con sensores de posición y velocidad, así como con algoritmos de control proporcional-integral-derivativo. Esto permite que el motor alcance su posición de manera rápida y eficiente.

Desde la robótica industrial hasta la fabricación con sistemas de automatización y las aplicaciones de mecanizado de control numérico (CNC) por ordenador, los servomotores están allí, listos para salvar el día.



- **Industrias:** Los servomotores son los caballos de batalla en las máquinas herramienta, embalaje, automatización de fábricas, manipulación de materiales, conversión de impresión y líneas de ensamblaje. Si hay una tarea exigente, puedes apostar que hay un servomotor trabajando duro para hacerla realidad.
- **Robótica:** Los servomotores son los músculos de los robots, proporcionando un encendido y apagado suave y un posicionamiento preciso. Ya sea en un brazo robótico en una línea de montaje o en un robot explorador en Marte, los servomotores hacen posible el movimiento.
- **Aeroespacial:** En la industria aeroespacial, los servomotores son los guardianes de los sistemas hidráulicos, manteniendo el fluido hidráulico en su lugar y asegurando que todo funcione sin problemas.
- **Juguetes Controlados por Radio:** Los servomotores son los que hacen que los juguetes controlados por radio cobren vida, permitiendo movimientos precisos y controlados.
- **Electrónica de Consumo:** En dispositivos como DVD y reproductores de discos, los servomotores son los que hacen que las bandejas de discos se abran y cierren con un movimiento suave y preciso.
- **Automóviles:** En los vehículos, los servomotores son los que mantienen la velocidad constante, permitiendo un viaje suave y seguro.





Los servomotores se manejan enviando un pulso eléctrico de ancho variable, también conocido como modulación de ancho de pulso (PWM), a través del cable de control. Este pulso puede variar en duración, con un pulso mínimo, un pulso máximo y una frecuencia de repetición.

En términos generales, un servomotor sólo puede girar  $90^\circ$  en cualquier dirección, lo que da un movimiento total de  $180^\circ$ .

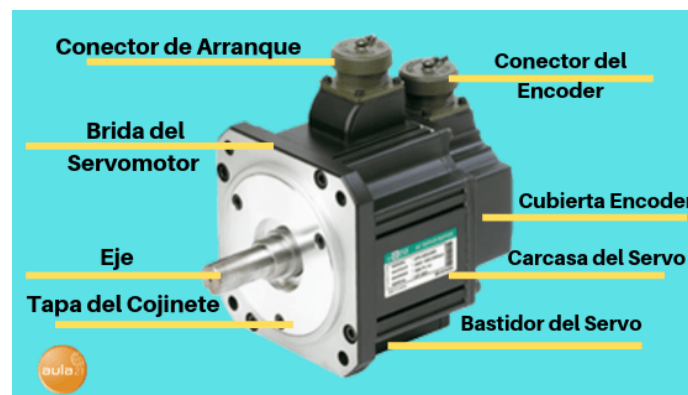
El PWM que se envía al motor y determina la posición del eje, y se basa en la duración del pulso enviado a través del cable de control. El servomotor espera ver un pulso cada 20 milisegundos (ms). La longitud de este pulso determinará hasta dónde gira el motor.

Si el tiempo del pulso es inferior a 1,5 ms, el servo se moverá en sentido contrario a las agujas del reloj hacia la posición de 0. Y si el tiempo del pulso es superior a 1,5 ms, el servo girará en sentido de las agujas del reloj hacia la posición de  $180^\circ$ .

Cuando se les da la orden de moverse, los servos se moverán a la posición y mantendrán esa posición. Si una fuerza externa empuja contra el servo mientras mantiene una posición, el servo se resistirá a salir de esa posición.

La cantidad máxima de fuerza que puede ejercer el servo se denomina par de torsión del servo. Sin embargo, los servos no mantendrán su posición indefinidamente; el pulso de posición debe repetirse para indicar al servo que se mantenga en posición.

*Partes de un Servomotor*



El brachiograph es un dispositivo similar a un brazo robótico capaz de moverse únicamente en dos dimensiones, emulando los movimientos del brazo de una persona al dibujar.

Este dispositivo está conformado por tres servomotores tower pro micro servo SG90, Qué cuenta con control de posición de 180 grados con velocidad de operación de 60 grados y con un par de torsión de 1.6 kg por centímetro.

Los ciervos están conectados entre sí con ayuda de piezas de madera pegadas y atornilladas a los mismos, qué sigue en el movimiento de rotación del servo, se utilizó una pieza de madera al estilo de una pinza para sujetar el instrumento con el que se va a dibujar, en este caso una pluma.

Con ayuda de entornos de desarrollo virtuales y programación en python se lograron utilizar las librerías necesarias para el funcionamiento del brazo para de esta manera automatizar las rutinas necesarias para los dibujos que realiza.



Micro servo con control de posición angular de 180 grados. Ideal para proyectos pequeños de robótica o mecatrónica que no requieren gran par de torsión.

Totalmente compatible con Arduino, utiliza la librería Servo.h para tus proyectos con servos en Arduino.

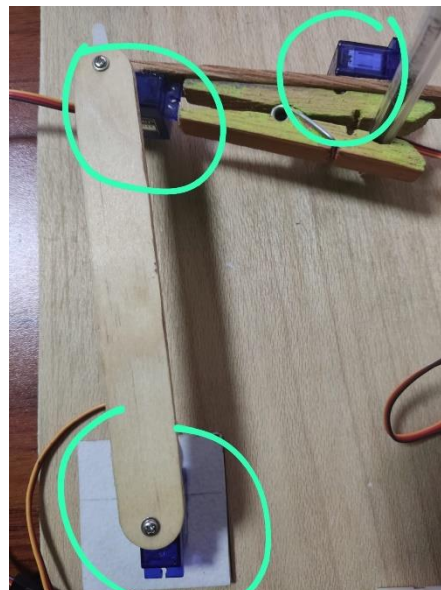
Es el servo más utilizado para crear hexapodos, robots que imiten animales, sistemas animatrónicos y algunas aplicaciones de aeromodelismo.

#### ESPECIFICACIONES

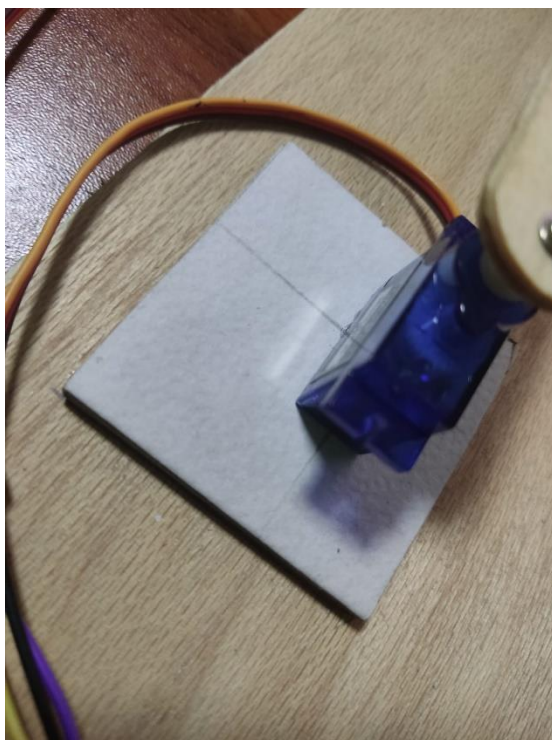
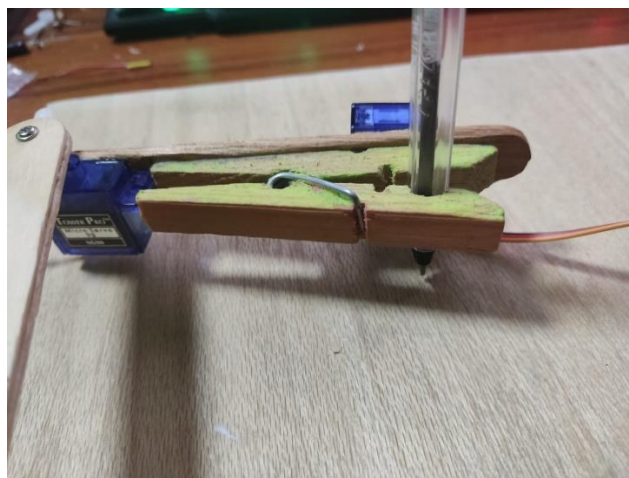
- Velocidad de operación sin carga:
  - 0.12 segundos / 60 grados (4.8V)
  - 0.10 segundos / 60 grados (6.0V)
- Par de torsión: 1.6kg · cm (4.8V)
- Temperatura de funcionamiento: -30 a +60 grados Celsius
- Ajustes de zona muerta: 5 microsegundos.
- Voltaje de funcionamiento: 3.5V ~ 6V
- Longitud del cable: 25cm
- Tamaño: 23x12.2x29mm
- Peso: 13g
- Función: Para Barco / Coche / Avión / Helicóptero / Robot
- Servo: Servo analógico.

## Desarrollo.

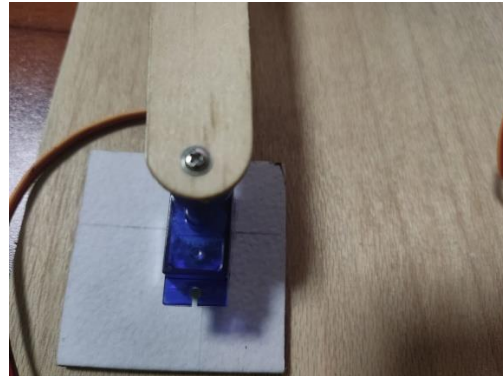
Para la realización del proyecto en físico se utilizaron los tres servos sg90, cada uno para una articulación diferente, se utilizó uno como la articulación del hombro, otro como la articulación del codo y otro como la articulación de la muñeca.



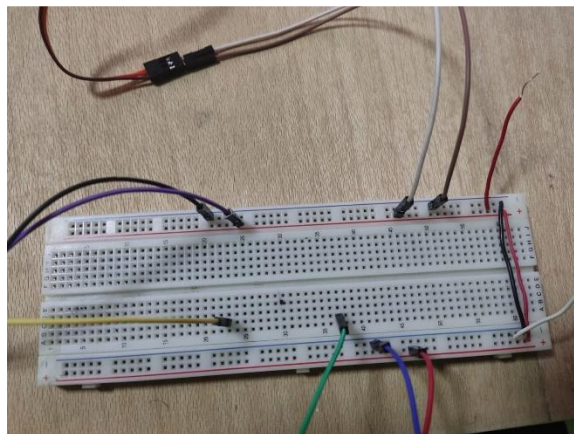
También se utilizaron piezas de madera estilo abate lenguas de una medida similar y también en forma de pinza, papel cascarón para colocarlo como base para el servomotor correspondiente a la parte del hombro del brazo, con la intención de elevar un poco el brazo en general y lograr que la parte de la muñeca se mantenga elevada hasta que el servomotor correspondiente reciba la instrucción de bajar la pluma o el instrumento para dibujar.



Para poder armar correctamente la estructura del brazo completa se utilizó pegamento, en este caso resistol 5000 para fijar las piezas de madera a los servomotores, además de utilizar los tornillos correspondientes a dichos servomotores.



Posteriormente se utilizó una protoboard para aterrizar en este punto las terminales de alimentación para los servos y también los pines correspondientes a la recepción de datos referentes al PWM o a la modulación de ancho de pulso.



Para evitar problemas con la alimentación y la raspberry, se alimentaron de manera independiente los servomotores, solamente unificando ambas tierras utilizadas, la proporcionada directamente de la raspberry y la proporcionada por la fuente externa.

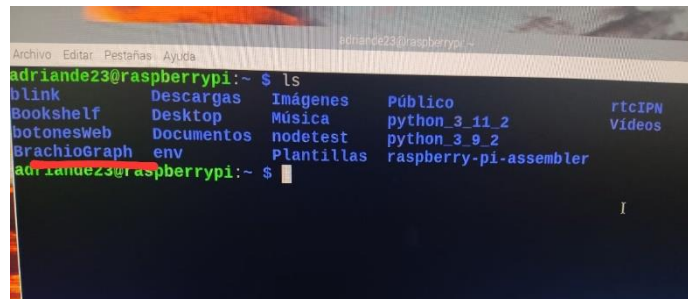




Dentro de la terminal de la raspberry se creó un nuevo directorio en donde colocar los programas y en dónde instalar también las siguientes librerías:

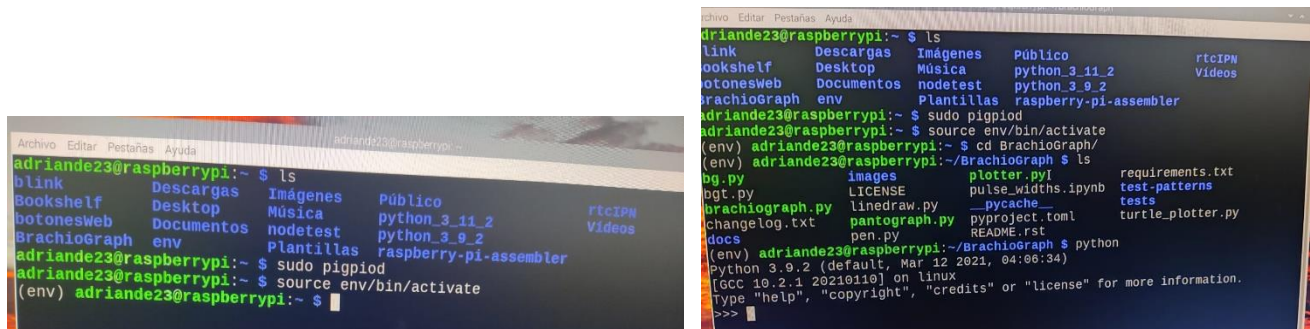
- python3-venv
- python3-tk
- pigpiod
- libbig0
- libjpeg-dev
- liblcms2-2
- libopenjp2-7
- libtiff5
- libwebp6
- libwebpdemux2
- libwebpmux3
- libzstd1
- libatlas3-base
- libgfortran5

Una vez agregadas correctamente las librerías, se creó un entorno virtual de desarrollo dentro de la misma terminal, para poder agregar más complementos sobre el proyecto.



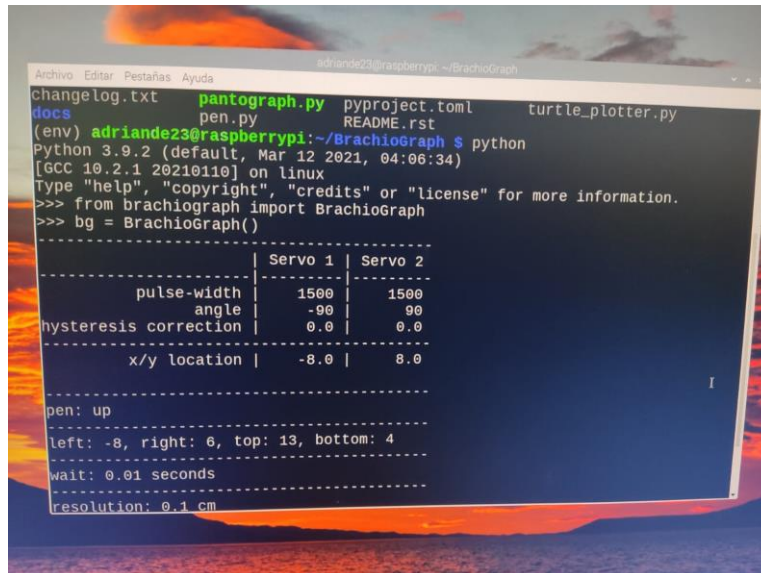
```
adriande23@raspberrypi:~$ ls
blink      Descargas  Imágenes   Público    python_3.11.2  rtcIPW
Bookshelf  Desktop    Música      python_3.9.2  Videos
botonesWeb Documentos  nodetest   python_3.9.2
BrachioGraph env         Plantillas raspberry-pi-assembler
adriande23@raspberrypi:~$
```

Con ayuda del entorno virtual, se utilizó un entorno de desarrollo de python, hacia el que importamos las librerías anteriormente instaladas que nos permiten comenzar a utilizar de manera correcta el brazo.



```
adriande23@raspberrypi:~$ ls
blink      Descargas  Imágenes   Público    python_3.11.2  rtcIPW
Bookshelf  Desktop    Música      python_3.9.2  Videos
botonesWeb Documentos  nodetest   python_3.9.2
BrachioGraph env         Plantillas raspberry-pi-assembler
adriande23@raspberrypi:~$ sudo pigpiod
adriande23@raspberrypi:~$ source env/bin/activate
(env) adriande23@raspberrypi:~$ cd BrachioGraph/
(env) adriande23@raspberrypi:~/BrachioGraph$ ls
bg.py      images      plotter.py  requirements.txt
bgt.py     LICENSE     pulse_widths.ipynb  test-patterns
brachioGraph.py  linedraw.py  __pycache__  tests
changelog.txt  pantograph.py  pyproject.toml  turtle_plotter.py
docs          pen.py        README.rst
(env) adriande23@raspberrypi:~/BrachioGraph$ python
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

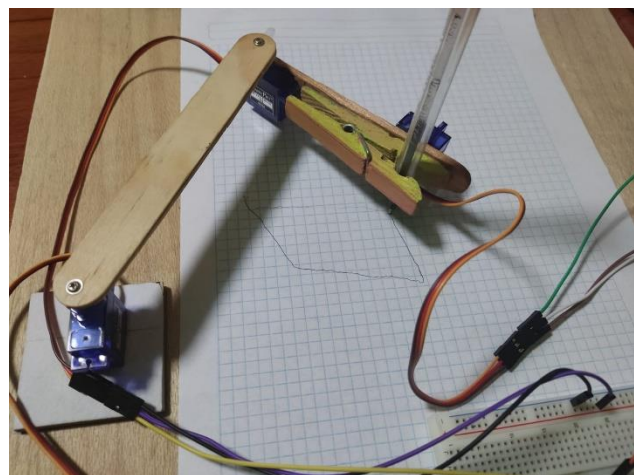
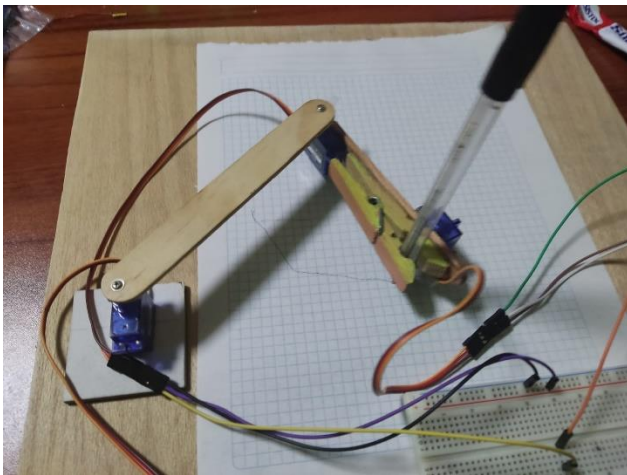
Se utilizan las funciones previamente establecidas en el código correspondiente al BrachioGraph para realizar distintos tests, crear varias figuras, e incluso con ayuda de un programa llamado Linedraw convertir nuevas imágenes a código que puede ser procesado y dibujado por el proyecto.



```
adriande23@raspberrypi: ~/BrachioGraph
change log.txt  pantograph.py  pyproject.toml  turtle_plotter.py
docs            pen.py          README.rst
(env) adriande23@raspberrypi:~/BrachioGraph $ python
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from brachioGraph import BrachioGraph
>>> bg = BrachioGraph()

-----
                Servo 1 | Servo 2
-----
pulse-width      1500   | 1500
angle            -90    | 90
hysteresis correction 0.0 | 0.0
-----
x/y location    -8.0   | 8.0
-----

pen: up
-----
left: -8, right: 6, top: 13, bottom: 4
wait: 0.01 seconds
-----
resolution: 0.1 cm
```



El código está completamente realizado en lenguaje python.

```
brachio graph.py X
brachio graph.py

from time import sleep
import readchar
import math
import numpy
from plotter import Plotter

class BrachioGraph(Plotter):
    """A shoulder-and-elbow drawing robot class."""

    def __init__(
        self,
        virtual: bool = False, # a virtual plotter runs in software only
        turtle: bool = False, # create a turtle graphics plotter
        turtle_coarseness=None, # a factor in degrees representing servo resolution
        # ----- geometry of the plotter -----
        bounds: tuple = (-8, 4, 6, 13), # the maximum rectangular drawing area
        inner_arm: float = 8, # the lengths of the arms
        outer_arm: float = 8,
        # ----- naive calculation values -----
        servo_1_parked_pw: int = 1500, # pulse-widths when parked
        servo_2_parked_pw: int = 1500,
        servo_1_degree_ms: int = -10, # milliseconds pulse-width per degree
        servo_2_degree_ms: int = 10, # reversed for the mounting of the shoulder servo
        servo_1_parked_angle: int = -90, # the arm angle in the parked position
    ):
        # ----- hysteresis -----
        hysteresis_correction_1: int = 0, # hardware error compensation
        hysteresis_correction_2: int = 0,
        # ----- servo angles and pulse-widths in lists -----
        servo_1_angle_pws: tuple = [], # pulse-widths for various angles
        servo_2_angle_pws: tuple = [],
        # ----- servo angles and pulse-widths in lists (bi-directional) -----
        servo_1_angle_pws_bidi: tuple = [], # bi-directional pulse-widths for various angles
        servo_2_angle_pws_bidi: tuple = [],
        # ----- the pen -----
        pw_up: int = 1500, # pulse-widths for pen up/down
        pw_down: int = 1100,
        # ----- physical control -----
        wait: float = None, # default wait time between operations
        angular_step: float = None, # default step of the servos in degrees
        resolution: float = None, # default resolution of the plotter in cm

        # set the geometry
        self.inner_arm = inner_arm
        self.outer_arm = outer_arm
```

Creamos la función init en dónde creamos diferentes graficadores. También asignamos de manera fija valores para los vértices de las figuras que se van a realizar y asignamos el valor de la medida de las extensiones entre los servos. Además de que asignamos los valores en pwm para los diferentes ángulos de los servomotores.

```
brachio graph.py X
brachio graph.py

79
80
81 def setup_turtle(self, coarseness):
82     from turtle_plotter import BrachioGraphTurtle
83     self.turtle = BrachioGraphTurtle(
84         inner_arm=self.inner_arm, # the length of the inner arm (blue)
85         outer_arm=self.outer_arm, # the length of the outer arm (red)
86         shoulder_centre_angle=90, # the starting angle of the inner arm, relative to straight
87         elbow_centre_angle=90, # the centre of the outer arm relative to the inner arm
88         elbow_sweep=180, # the arc covered by the elbow motor
89         window_size=850, # width and height of the turtle canvas
90         speed=10, # how fast to draw
91         machine=self,
92         coarseness=coarseness,
93     )
94
95     self.turtle.draw_grid()
96     self.t = self.turtle
97
98
99 def test_arcs(self):
100     self.park()
101     elbow_angle = 120
102     self.move_angles(angle_2=elbow_angle)
103
104     for angle_1 in range(-135, 15, 15):
105         self.move_angles(angle_1=angle_1, draw=True)
```

Tratamos de corregir la histéresis en el posicionamiento de los servomotores agregando y restando un valor de 16 a los valores originales.

Utilizamos métodos trigonométricos para calcular la hipotenusa entre el servomotor del hombro y el servomotor del codo.

Importamos las bibliotecas necesarias para la creación del programa.

En este caso importamos la instancia sleep de la biblioteca time, que nos permite generar delays o retardos, la instancia plotter de la biblioteca plotter, Qué es una biblioteca basada en matplotlib que simplifica la graficación de datos en python, y las bibliotecas Readchar, math y numpy.

```
py
servo_2_degree_ms: int = 10, # reversed for the mounting of the shoulder servo
servo_1_parked_angle: int = -90, # the arm angle in the parked position
servo_2_parked_angle: int = 90,
# ----- hysteresis -----
hysteresis_correction_1: int = 0, # hardware error compensation
hysteresis_correction_2: int = 0,
# ----- servo angles and pulse-widths in lists -----
servo_1_angle_pws: tuple = [], # pulse-widths for various angles
servo_2_angle_pws: tuple = [],
# ----- servo angles and pulse-widths in lists (bi-directional) -----
servo_1_angle_pws_bidi: tuple = [], # bi-directional pulse-widths for various angles
servo_2_angle_pws_bidi: tuple = [],
# ----- the pen -----
pw_up: int = 1500, # pulse-widths for pen up/down
pw_down: int = 1100,
# ----- physical control -----
wait: float = None, # default wait time between operations
angular_step: float = None, # default step of the servos in degrees
resolution: float = None, # default resolution of the plotter in cm

):
    # set the geometry
    self.inner_arm = inner_arm
    self.outer_arm = outer_arm
```

Creamos la función setup\_turtle y en esta asignamos los ángulos que van a tomar los servomotores en su valor inicial.

```
brachio graph.py X
brachio graph.py

102 self.move_angles(angle_2=elbow_angle)
103
104 for angle_1 in range(-135, 15, 15):
105     self.move_angles(angle_1=angle_1, draw=True)
106
107     for angle_2 in range(elbow_angle, elbow_angle + 16):
108         self.move_angles(angle_2=angle_2, draw=True)
109     for angle_2 in range(elbow_angle + 16, elbow_angle - 16, -1):
110         self.move_angles(angle_2=angle_2, draw=True)
111     for angle_2 in range(elbow_angle - 16, elbow_angle + 1):
112         self.move_angles(angle_2=angle_2, draw=True)
113
114 # ----- trigonometric methods -----
115
116 def xy_to_angles(self, x=0, y=0):
117     """Return the servo angles required to reach any x/y position."""
118
119     hypotenuse = math.sqrt(x**2 + y**2)
120
121     if hypotenuse > self.inner_arm + self.outer_arm:
122         raise Exception(
123             f"Cannot reach {hypotenuse}; total arm length is {self.inner_arm + self.outer_arm}."
124         )
125
126     hypotenuse_angle = math.asin(x / hypotenuse)
```



```

# ----- reporting methods -----
def report(self):
    print(f"-----")
    print(f"Servo 1 | Servo 2")
    print(f"-----")

    h1, h2 = self.hysteresis_correction_1, self.hysteresis_correction_2
    print(f"hysteresis (h1:>2.1f) | (h2:>2.1f)")

    pw_1, pw_2 = self.get_pulse_widths()
    print(f"pulse-width (pw_1:>4.0f) | (pw_2:>4.0f)")

    angle_1, angle_2 = self.angle_1, self.angle_2
    if angle_1 and angle_2:
        print(f"angle (angle_1:>4.0f) | (angle_2:>4.0f)")

    print(f"-----")
    print(f"min max mid | min max mid")
    print(f"-----")

```

Creamos métodos de reporte, en los que el programa muestra en pantalla los valores en los que se encuentran los servomotores, para obtener un análisis general del brazo.

```

print(f"min max mid | min max mid")
print(f"-----")

if (
    self.angles_used_1
    and self.angles_used_2
    and self.pulse_widths_used_1
    and self.pulse_widths_used_2
):
    min1 = min(self.pulse_widths_used_1)
    max1 = max(self.pulse_widths_used_1)
    mid1 = (min1 + max1) / 2
    min2 = min(self.pulse_widths_used_2)
    max2 = max(self.pulse_widths_used_2)
    mid2 = (min2 + max2) / 2

    print(f"pulse-widths (min1:>4.0f) (max1:>4.0f) (mid1:>4.0f) | (min2:>4.0f) (max2:>4.0f) (mid2:>4.0f)")

    min1 = min(self.angles_used_1)
    max1 = max(self.angles_used_1)
    mid1 = (min1 + max1) / 2
    min2 = min(self.angles_used_2)
    max2 = max(self.angles_used_2)
    mid2 = (min2 + max2) / 2

```

## Programa Servidor

Realicé el código para el servidor utilizando la raspberry Pico w, para obtener un código más compacto, ya que trabajando desde esta placa de desarrollo en el entorno de Thonny se pueden combinar los códigos tanto para cliente como para servidor, en html y en python.

```

1 import network
2 import socket
3 from time import sleep
4 import slow_servo
5 import machine
6
7 ssid='...'
8 password='...'
9
10 def webpage():
11     #Template HTML
12     html = f"""
13
14     <!DOCTYPE html>
15     <html>
16     <head>
17     <meta name="viewport" content="width=device-width, initial-scale=1">
18
19     </head>
20     <body>
21
22     <h1>Brazo de Dibujo Slider</h1>
23
24     <p>Servo del hombro</p>
25
26     <div class="slidecontainer">
27     <input type="range" min="0" max="180" value="90" class="slider" id="myRange">
28     <p>Valor Inicial: <span id="demo2"></span></p>
29     <p>Cambiar Valor: <span id="demo3"></span></p>
30     </div>
31
32     <p>Servo del codo</p>
33
34     <div class="slidecontainer">
35     <input type="range" min="0" max="180" value="90" class="slider" id="myRange2">
36     <p>Valor Inicial: <span id="demo2"></span></p>
37     <p>Cambiar Valor: <span id="demo2"></span></p>
38     </div>
39
40     <p>Servo de la pluma</p>
41
42     <div class="slidecontainer">
43     <input type="range" min="0" max="180" value="90" class="slider" id="myRange3">
44     <p>Valor Inicial: <span id="demo3"></span></p>
45
46     </div>
47
48     </body>
49
50     </html>
51
52     """
53
54     return str(html)

```

Se importaron las bibliotecas necesarias. En este caso se importa la biblioteca network hizo que son utilizadas para la creación del servidor, next sleep para generar delays o retardos, la biblioteca slow servo necesaria para la utilización de este programa, y la típica biblioteca machine para utilizar los pines de la raspberry Pico.

También se asignan a las variables ssid y password el nombre de la red a la que nos conectamos y la contraseña respectivamente.

Utilizamos la función webpage para utilizar una plantilla tipo html, de esta manera escribimos el código en html entre comillas y se lo asignamos a una variable, en este caso la variable html.

```

<body>
<h1>Brazo de Dibujo Slider</h1>
<p>Servo del hombro.</p>
<div class="slidecontainer">
<input type="range" min="0" max="180" value="90" class="slider" id="myRange">
<p>Valor Inicial: <span id="demo2"></span></p>
<p>Cambiar Valor: <span id="demo3"></span></p>
</div>
<br>
<p>Servo del codo.</p>
<div class="slidecontainer">
<input type="range" min="0" max="180" value="90" class="slider" id="myRange2">
<p>Valor Inicial: <span id="demo2"></span></p>
<p>Cambiar Valor: <span id="demo2"></span></p>
</div>
<br>
<p>Servo de la pluma.</p>
<div class="slidecontainer">
<input type="range" min="0" max="180" value="90" class="slider" id="myRange3">
<p>Valor Inicial: <span id="demo3"></span></p>
</div>
</body>
</html>

```

```

var slider3 = document.getElementById("myRange3");
var output3 = document.getElementById("demo3");
var output3 = document.getElementById("demo3");
output3.innerHTML = slider3.value;

slider3.oninput = function() {
    output3.innerHTML = this.value;
}

slider3.onchange = function() {
    output3.innerHTML = this.value;
    var xhr3 = new XMLHttpRequest();
    xhr3.open("GET", "/slider?" + this.value, true);
    xhr3.send();
}

</script>
</body>
</html>

return str(html)

def connect():
    # Connect to the network

```

```

101     return str(html)
102
103 def connect():
104     #Connect to WLAN
105     wlan = network.WLAN(network.STA_IF)
106     wlan.active(True)
107     wlan.connect(ssid, password)
108     while wlan.isconnected() == False:
109         print('Waiting for connection...')
110         sleep(1)
111     ip = wlan.ifconfig()[0]
112     print(f'Connected on {ip}')
113     return ip
114
115 def open_socket(ip):
116     # Open a socket
117     address = (ip, 80)
118     connection = socket.socket()
119     connection.bind(address)
120     connection.listen(1)
121     return connection
122
123 def serve(connection):

```

Creamos la función connect que nos sirve para conectarnos a la red wi-fi, en este caso utilizamos las funciones provenientes de la instancia network y pasamos como parámetros los valores que asignamos como ssid y password.

Con la función open\_socket asignamos el valor de la dirección IP de nuestra raspberry pico w y utilizamos la biblioteca socket para realizar la conexión, además de usar el metodo listen para esperar la conexión.

```

101     return str(html)
102
103 def connect():
104     #Connect to WLAN
105     wlan = network.WLAN(network.STA_IF)
106     wlan.active(True)
107     wlan.connect(ssid, password)
108     while wlan.isconnected() == False:
109         print('Waiting for connection...')
110         sleep(1)
111     ip = wlan.ifconfig()[0]
112     print(f'Connected on {ip}')
113     return ip
114
115 def open_socket(ip):
116     # Open a socket
117     address = (ip, 80)
118     connection = socket.socket()
119     connection.bind(address)
120     connection.listen(1)
121     return connection
122
123 def serve(connection):

```

```

    connection.listen(1)
    return connection
def serve(connection):
    #Start a web server
    servo1 = slow_servo.Slow_Servo(0) #create servo object on pin 0
    servo2 = slow_servo.Slow_Servo(1)
    servo3 = slow_servo.Slow_Servo(2)
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        print (request)
        try:
            request = request.split()[1]
        except IndexError:
            pass
        if request.find('slider1') > -1:
            slider_val1 = request.split('?')[1]
            print (slider_val1)
            servo1.set_angle(slider_val1,1000)

```

```

        request = request.split()[1]
    except IndexError:
        pass
    if request.find('slider1') > -1:
        slider_val1 = request.split('?')[1]
        print (slider_val1)
        servo1.set_angle(slider_val1,1000)

    if request.find('slider2') > -1:
        slider_val2 = request.split('?')[1]
        print (slider_val2)
        servo2.set_angle(slider_val2,1000)

    if request.find('slider3') > -1:
        slider_val3 = request.split('?')[1]
        print (slider_val3)
        servo3.set_angle(slider_val3,1000)

    html = webpage()
    client.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
    client.send(html)
    client.close()

```

En la función serve creamos los objetos correspondientes para los héroes y les asignamos el pin con el que van a trabajar recibiendo los datos, verificamos la conexión del cliente, limitamos la cantidad de información que podemos recibir y la implementamos como una cadena de caracteres.

Utilizamos condicionales if para separar los datos que llegan a la petición.Finalmente terminamos la ejecución el programa en html y después del programa completo.

## **Conclusión.**

En conclusión la realización de este proyecto fue una gran manera de implementar los conocimientos adquiridos con respecto a la raspberry, ya que se pudo utilizar la raspberry que es un microprocesador como el controlador de este proyecto.

Debido a que el proyecto se trabaja dentro de un entorno virtual en python desde la terminal de la raspberry, surgieron complicaciones para compaginar este hecho con un servidor desde la raspberry, sin embargo el proyecto es totalmente funcional.

La utilización del brazo se puede extender a otras áreas e incluso otros procesos industriales, con la utilización y la correcta implementación de las rutinas necesarias.

## **Referencias.**

[1] <https://raspberrypi.cl/que-es-raspberry/>

[2] <https://www.xataka.com/ordenadores/raspberry-pi-4-caracteristicas-precio-ficha-tecnica>

[3] <https://www.paessler.com/es/itexplained/server#:~:text=Un%20servidor%20es%20un%20sistema,comparten%20recursos%20con%20m%C3%A1quinas%20cliente.>

[4] <https://www.cursosaula21.com/que-es-un-servomotor/>

[5] <https://github.com/evildmp/BrachioGraph/tree/master/docs>