Ingeniería en Computación

Taller de Proyecto I

Informe Final

Mano Robótica

Grupo 3:

Bosatta, Darío

Cabrera, Merlina

Pérez, Francisco

19/02/2024

Índice

1. Introducción 4
2. Objetivos 5
   2.1. Objetivos primarios 5
      2.1.1. Diseño mecánico 5
      2.1.2. Control de movimiento 7
      2.1.3. Programación 9
   2.2. Objetivos secundarios 9
      2.2.1. Movilidad de mayor resolución de cada dedo 9
      2.2.2. Movilidad del antebrazo 9
3. Requerimientos 10
   3.1. Funcionales 10
   3.2. No funcionales 10
   3.3. De producto 10
   3.4. Hardware a utilizar 10
4. Diseño de hardware 11
   4.1. Bloques del diseño 11
      4.1.1. Bloque de procesamiento de movimientos 11
      4.1.2. Bloque Mano Robótica 12
      4.1.3. Bloque Información para el Usuario 17
   4.2. Placa de circuito impreso (PCB) 18
      4.2.1. Diseño 18
      4.2.2. Fabricación 20
5. Diseño del firmware, simulación y depuración 23
   5.1. Bibliotecas utilizadas 23
      5.1.1. Detección en tiempo real 23
      5.1.2. Comunicación serial con placa CIAA 24
      5.1.3. Bibliotecas adicionales 24
   5.2. Funcionamiento del programa de detección 24
   5.3. Transferencia de información 30
   5.4. Placa EDU CIAA NXP 31
      5.4.1. Estados del sistema 31
   5.5. Placa EDU CIAA y Mano Robótica 34
      5.5.1. Control de los servomotores 34

6. Ensayos y mediciones 36
6.1. Pruebas previas a la PCB 36
6.2. Medición de voltaje y corriente 37
6.3. Prueba de funcionamiento de 5 servomotores a la vez 38
6.3.1. Fallo con trazos de cobre 38
6.3.2. Inconsistencias con la alimentación del circuito 39
6.4. Calibración de estructura física 39
6.5. Prueba de funcionamiento de pulsador y led RGB 40
6.6. Prueba final 40
7. Conclusiones 41
7.1. Cumplimiento de objetivos 41
7.2. Cumplimiento de objetivos 41
7.3. Observaciones finales 43
8. Bibliografía 46
9. Anexos 47
9.1. Circuito esquemático 47
9.2. Lista de materiales 50
9.3. Placa de circuito impreso (PCB) 51
9.3.1. General 51
9.3.2. TOP LAYER 51
9.3.3. BOTTOM LAYER 52
9.3.4. 3D VIEWER 52
9.4. Códigos 54
9.4.1. Código EDU CIAA NXP 54
9.4.2. Código Cámara 61

1. Introducción

La robótica combina principios de diversas disciplinas, y abarca una amplia gama de tareas. Un área muy característica es la creación de sistemas robóticos que imitan y extienden las capacidades motrices del cuerpo humano, teniendo un gran potencial para generar un impacto positivo en diversas áreas de la vida humana y la industria.

En particular, las extremidades robóticas han tenido una gran importancia en la vida de las personas con discapacidades, pudiendo así recuperar movilidad y funcionalidad perdida y mejorando la calidad de vida de los usuarios. Por otro lado, en la industria es importante para la automatización de procesos de fabricación, mejorando la precisión y eficacia, permitiendo obtener un mayor rendimiento y reducción de costos laborales.

Los mencionados son sólo algunos ejemplos del amplio abanico de funcionalidades que expone esta rama, que contribuye a la evolución e innovación de la tecnología.

Teniendo en cuenta la gran versatilidad que puede ofrecer la implementación y estudio de este campo, y la motivación de abordar un desafío técnico que comprende varios campos y ofrece una familiarización con la electrónica, programación, mecánica y control, se optó por elegir dicho camino.

El presente proyecto se centra en el diseño, desarrollo e implementación de una mano robótica haciendo uso de la placa EDU CIAA NXP, que proporciona un entorno ideal para explorar y materializar conceptos de robótica, entre otros.

2. Objetivos

La implementación de este proyecto conlleva una serie de objetivos primarios y secundarios, que impulsan su desarrollo y definen los límites del mismo.

2.1. Objetivos primarios

Se plantea como objetivo primario el diseño y puesta en marcha de una mano robótica que imite los movimientos de una mano humana real. Se busca lograr la movilidad de los 5 dedos que la componen en conjunto y por separado.

Se requiere un enfoque que abarque todos los aspectos. El proyecto se encuentra dividido en los siguientes módulos principales,

- Diseño mecánico
- Control de movimiento
- Programación

En ***Figura 2.1***, se presenta un diagrama en bloques del sistema final, y en las próximas secciones de desarrolla el contenido necesario más en detalle.

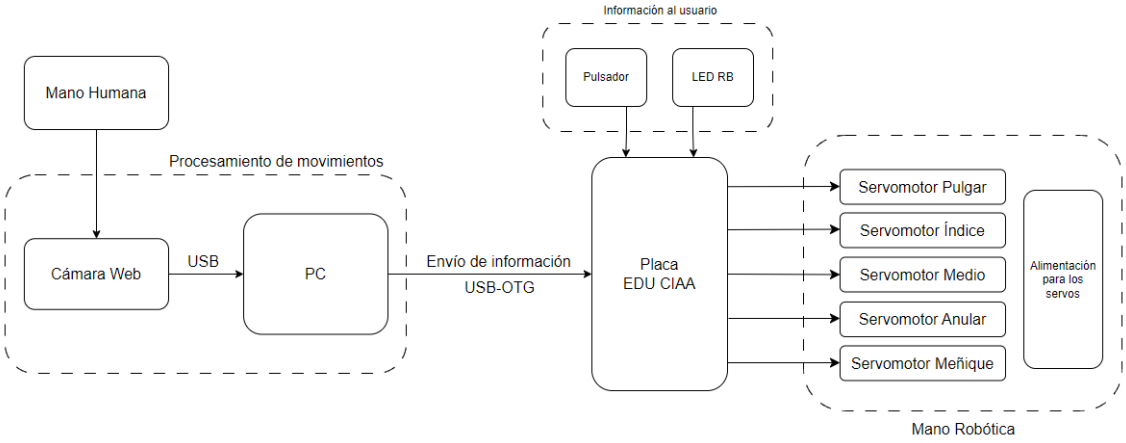


Figura 2.1.
Diagrama en bloques del sistema propuesto.

2.1.1. Diseño mecánico

Esta fase incluye el diseño de los componentes físicos de la mano robótica. Esto incluye el diseño de una estructura adecuada que simbolice y replique la flexibilidad y rango de movimiento característicos de una mano humana. Se deben considerar aspectos como la ergonomía, resistencia de los materiales, precisión de los movimientos y facilidad de montaje.

2.1.1.1. Estructura física

En cuanto a la estructura, se optó por recrearla en impresión 3D, inspirada en ***Figura 2.2***, abierto a modificaciones y/o cambios en su diseño a lo largo del desarrollo, por ejemplo, para hacer el sistema más realista o más eficiente físicamente para que ofrezca una movilidad óptima.

El modelado 3D se descargó del enlace ***[1]***, que además de copiar perfectamente la forma de la mano humana, está pensado para este tipo de proyecto, ya que tiene el espacio y las ranuras necesarias para incorporar los periféricos requeridos.

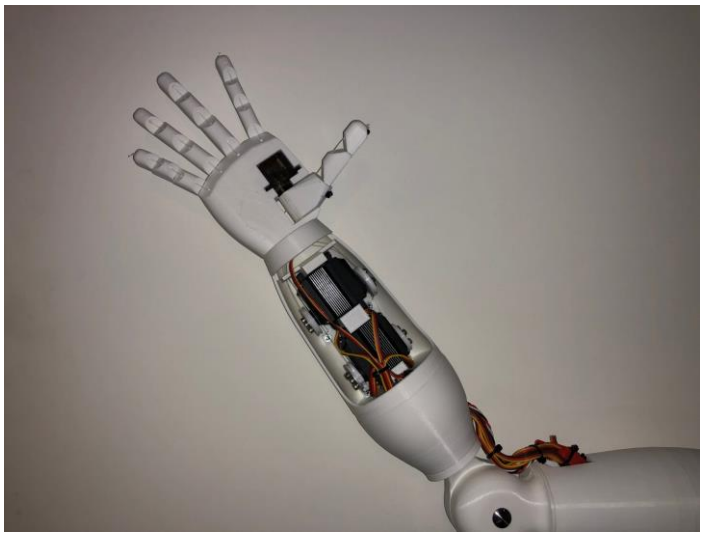


Figura 2.2.
Objetivo inicial de la mano robótica en 3D, hasta el antebrazo.

2.1.1.2. Servomotores

Los servomotores son un tipo de motor eléctrico diseñado para controlar el movimiento, la posición y la velocidad de un sistema, por eso son muy comunes en aplicaciones robóticas, donde se requiere un control preciso de la movilidad. Estos reaccionan ante una señal de control, típicamente PWM (modulación por ancho de pulso) pudiendo controlar la velocidad y posición deseadas.

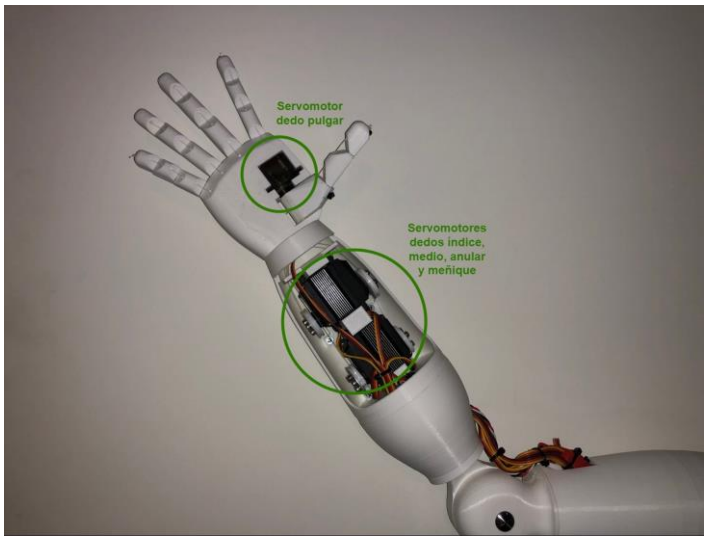


Figura 2.3.
Se destaca en la imagen la ubicación de los servomotores en la mano impresa.

La ***Figura 2.3*** señala donde van puntualmente ubicados los servomotores requeridos para satisfacer los objetivos primarios del proyecto.

2.1.1.3. Placa EDU CIAA NXP

Se trata de una plataforma diseñada para aplicaciones de sistemas embebidos, la placa brindada por la cátedra que se usará está basada en un microcontrolador LPC4337, con un núcleo Cortex M4f y otro Cortex M0, que proporcionan una buena potencia de procesamiento.

Ésta está esquipada con una gran variedad de interfaces de comunicación, pines de E/S y periféricos integrados. Debido a la naturaleza de este proyecto, algunos recursos de utilidad para el mismo son los siguientes,

- USB Debug

Existen diversas formas de realizar la comunicación entre la PC y la placa CIAA, tanto cableadas (como USB OTG, Debug o Ethernet) como inalámbricas (Wi-Fi, Bluetooth, radiofrecuencias), requiriendo de ciertos módulos

Esta conexión requiere de un paso adicional, que es asegurarse que en la configuración del proyecto la interfaz de depuración esté configurada para utilizar el USB adecuado, al que está conectado la placa en ese instante.

- PWM

El recurso PWM (modulación por ancho de pulso) ofrece una forma efectiva de enviar señales de control precisas a dispositivos periféricos.

- Puertos E/S

Puertos GPIO para la conexión con componentes, para colocar entradas o salidas del sistema.

2.1.2. Control de movimiento

Existen diversas maneras de implementar un sistema de control que permita la coordinación fluida y natural de los movimientos de las articulaciones.

Se filtraron algunas de las opciones consideradas viables (principalmente, debido al plazo disponible para desarrollar el proyecto), las cuales varían en calidad de componentes periféricos necesarios y, obviamente, el costo de los mismos. A continuación se describen los distintos métodos estudiados,

- *Acelerómetros o sensores de flexibilidad*

Consta de realizar una especie de guante con acelerómetros o sensores (mínimo uno por dedo) y los datos que genera el dispositivo son enviados a la placa, para que esta envíe señales a los servomotores ubicados en la mano impresa en 3D.

En el caso de los acelerómetros, generan un valor de aceleración lineal, y en el caso de los sensores un valor de resistencia variable (a mayor flexión, mayor resistencia).

Esta opción fue descartada por posibles complicaciones para adquirir la cantidad de dispositivos requeridos, problemas y complejidad de la calibración de dispositivos y los sensores son costosos.

- *Potenciómetros*

Los potenciómetros o resistencias variables, son dispositivos electrónicos usados para medir y ajustar la resistencia eléctrica. Al poder variar la resistencia, esta variación puede ser medida por un periférico ADC, traduciendo en un valor digital. Este valor digital puede ser usado para representar el movimiento del dedo a partir del uso del potenciómetro.

Esta opción también fue descartada ya que la motivación del grupo es realizar la implementación a través del movimiento de una mano real, y la aplicación de un potenciómetro limita y complejiza la situación.

- *Cámara web*

Consta de una cámara web, que pueda captar la imagen de una mano humana que se posiciona enfrente y sus puntos clave. El software de la cámara debe detectar los movimientos de la mano real captada, y comunicarse con la placa para que esta envíe señales a los servomotores ubicados en la mano impresa en 3D.

Finalmente, se optó por la cámara web para reproducir el control de movimiento de la mano robótica, debido a que es la solución más económica, ya que requiere de un único componente adicional, y además ofrece una interesante implementación.

2.1.2.1. Cámara Web

Se optó por esta opción para reproducir el control de movimiento de la mano robótica, ya que resulta la solución más económica, requiere de un único componente adicional y además ofrece una interesante implementación.

Puede usarse cualquier tipo de cámara web integrada o mediante conexión USB. A partir de las imágenes capturadas por la misma, con ayuda de un software adicional de detección para depurar los datos necesarios, se podrán transmitir los datos filtrados a la CIAA mediante un canal establecido.

2.1.3. Programación

En este apartado, se destacan dos ámbitos esenciales: un programa de detección y un programa de ejecución de la placa EDU CIAA NXP.

- Programa de detección

La necesidad de este programa surge a partir de la elección para controlar el movimiento, ***sección 2.1.2***. Es el encargado de analizar fotograma a fotograma la posición de la mano de la persona frente a la cámara web. A partir de este análisis, filtrar los datos necesarios y comunicarlos a las partes que lo requieran para lograr el cometido de este proyecto.

- Programa de ejecución

El programa cargado en la memoria del EDU CIAA NXP es el que requiere los datos procesados por el programa de detección, para procesarlos correspondientemente, generando las frecuencias adecuadas de PWM, encender leds RGB, entre otras tareas.

2.2. Objetivos secundarios

Una vez lograda la movilidad y coordinación de los 5 dedos de la mano, se podrán implementar los siguientes agregados o mejoras al sistema, para hacerlo más completo y armónico.

2.2.1. Movilidad de mayor resolución de cada dedo

En caso de ser posible se prevé obtener uno o varios grados más de resolución en cada dedo, es decir, la posibilidad de que los dedos puedan adoptar una posición intermedia o semi-abierta, en lugar de limitar a abrirlos y cerrarlos.

2.2.2. Movilidad del antebrazo

Se considera la posibilidad de añadir un nuevo movimiento, extendiendo la rotación del antebrazo, permitiendo tener mayor movilidad y precisión de la mano completa y no solo de los dedos.

3. Requerimientos

3.1. Funcionales

I. La cámara web debe ser capaz de capturar imágenes en tiempo real y enviarlas mediante un software a la placa para su procesamiento.
II. Enviar el valor “1” cuando la cámara detecta un dedo se detecta en estado “abierto”, y el valor “0” si lo detecta en estado “cerrado”.
III. Movimiento individual de cada dedo de la mano impresa en 3D.
IV. Iniciar el sistema con los 5 dedos cerrados.
V. Encender el led RGB en color azul cuando el sistema esté iniciado y listo para usar.
VI. Parpadear el led RGB en color verde cuando la cámara web detecte una mano humana y comience a enviar señales.
VII. El botón “stop” deshabilita el movimiento de todos los servomotores.
VIII. Si el movimiento de los servomotores está deshabilitado, y es presionado el botón “stop” nuevamente, se reanuda su movimiento.
IX. Si el movimiento de los servomotores está deshabilitado, se enciende el led RGB en color rojo.
X. Pulsación del botón bloqueante.
XI. Evitar la recepción de información redundante a los servomotores.
XII. Establecer frecuencia y ciclo de trabajo de PWM mayor a 50Hz para los servomotores a utilizar.

3.2. No funcionales

I. Movimiento suave de la mano robótica.
II. Limitación mínima de cantidad de movimientos por segundo.
III. Respuesta de la mano robótica a partir de la mano real en un tiempo aceptable.

3.3. De producto

- Imitación en cuasi tiempo real de la mano robótica de la mano detectada.
- Modelo impreso semejante a una mano real.

3.4. Hardware a utilizar

I. Placa EDU CIAA NXP.
II. Cámara web.
III. 4 servomotores de tamaño estándar.
IV. 1 micro servomotor.
V. 1 porta pilas y 2 pilas de litio.
VI. Resistencias, capacitores y transistores.
VII. Impresión 3D del modelado de una mano.
Incluyendo los materiales para el ensamblado.
VIII. Pulsador.
IX. Led RGB.
X. Regulador de tensión.

4. Diseño de hardware

4.1. Bloques del diseño

En base al diagrama de bloques de la ***Figura 2.1***, en esta sección se desarrollan los componentes que conforman cada bloque y la interconexión entre ellos, haciendo foco en el diseño esquemático desarrollado para este proyecto, diseño y fabricación del circuito impreso que lo hace funcional.

4.1.1. Bloque de procesamiento de movimientos

Este bloque está compuesto de una cámara web estándar para capturar imágenes de la mano y sus movimientos, y una PC que actúa como unidad central de procesamiento para el sistema, encargándose de la captura y procesamiento inicial de las imágenes de la cámara. Ambos dispositivos conectados mediante un cable USB, o en su defecto la cámara integrada de una portátil, en caso de usar dicha computadora.

A su vez, la PC se encuentra conectada mediante el puerto USB Debug a la placa EDU CIAA, ***Figura 4.1***. Esta conexión no solo facilita la comunicación entre la PC y la placa, sino que también proporciona la alimentación necesaria para la placa. El estándar USB 1.1 especifica una corriente máxima de 500mA a 5V y la placa consume aproximadamente 200mA, por lo que se puede asegurar una alimentación suficiente.

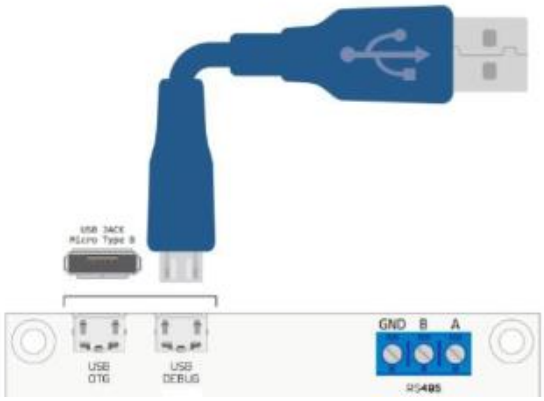


Figura 4.1.

Conexión USB Debug entre la placa EDU CIAA y la PC.

En resumen, este bloque establece la conexión física y funcional entre la cámara, la PC y la placa, destacando la importancia del software para la visualización y la transferencia de datos necesarios entre los dispositivos que conforman este canal para el procesamiento de movimientos de la mano robótica, lo cual se detalla en profundidad en la ***sección 5***.

4.1.2. Bloque Mano Robótica

Se trata de la mano robótica que imita los movimientos de una real. Este bloque, físicamente está compuesto de una mano impresa en 3D, un conjunto de servomotores y otros componentes internos que se agregan al circuito por cuestiones técnicas o de eficiencia en el sistema.

4.1.2.1. Base de la estructura

La base funcional de la estructura, además del armazón 3D de la mano, son los servomotores. Estos son los responsables del movimiento independiente de cada dedo de la mano y se incorporan dentro de la estructura física de la mano para garantizar comodidad y facilitar el diseño integrado general. ***Figura 4.2***.

En el modelado presentado, los dedos pueden reproducir movimientos gracias a la disposición de 5 servomotores controlados por señales enviadas desde la placa, a través de su programación.

Figura 4.2.

Impresión de mano en 3D.

En la imagen derecha se puede apreciar un esquema de las posiciones de los 5 servomotores.

Se utilizan 4 servomotores de tamaño estándar ($40mm\ x\ 20mm\ x\ 37mm$) que ofrecen la fuerza suficiente para controlar los dedos índice, anular, medio y meñique. Además, un micro servomotor para el dedo pulgar, ya que tiene una falange menos que el resto de los dedos y puede ser manejado con un poco menos de fuerza, y este se puede encastrar directamente en la palma de la mano.

La comunicación de cada servomotor a la placa se realiza mediante una conexión PWM (Modulación por Ancho de Pulso) enviada desde la CIAA para

controlar la posición del servo, lo que permite manejar el movimiento de cada dedo de manera precisa. Es importante destacar que la alimentación de los servomotores es independiente de la placa, y se detalla en la ***sección 4.1.2.2***.

Lógica de la estructura física

Como se mencionó, el armazón de la mano robótica es una impresión 3D de un modelo descargado online. El montaje de la misma con los respectivos componentes periféricos es fundamental.

En el caso de los 4 dedos ligados a servomotores estándar, un hilo de pesca atraviesa desde la punta de los dedos, pasando por la palma, hasta alcanzar el antebrazo donde se instalan los servos. El modelo 3D incluye poleas para añadir a cada servo, por la que se pasa el hilo de pesca, permitiendo que al girar este se enrolle y las falanges del dedo se contraigan. A su vez, las falanges son atravesadas por un elástico que llega hasta la palma, con el objetivo que cuando el hilo de pesca se contraiga para retraer el dedo, el elástico se tense, y cuando se comience a soltar, el dedo regrese a su forma original, es decir, se estire. ***Figura 4.3***. Por otro lado, el micro servo que controla el pulgar no requiere hilo de pesca ni elástico, simplemente la polea propia del servomotor se encastra en una falange del dedo.

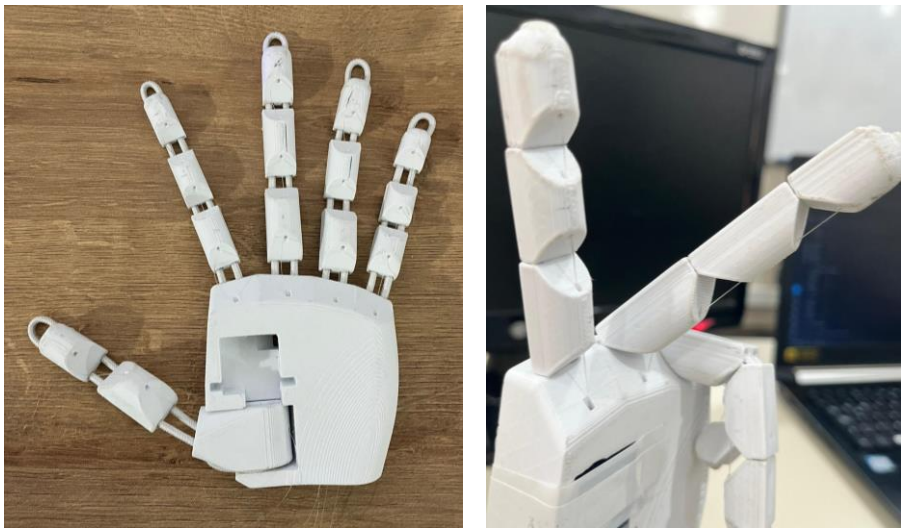


Figura 4.3.

Impresión de mano en 3D. Se puede observar la conexión entre las falanges con cordón elástico e hilo de pesca.

Es necesario aclarar que, tanto las medidas de los hilos de pesca y elásticos como las posiciones iniciales de los servomotores deben ser calibradas minuciosamente. Esto se discute en la ***sección 6***.

4.1.2.2. Alimentación de dispositivos

A la estructura básica mencionada en la sección previa, se le debe otorgar funcionalidad. Cada servomotor tiene 3 cables o conexiones que hacen posible su operatividad, ***Figura 4.4***.

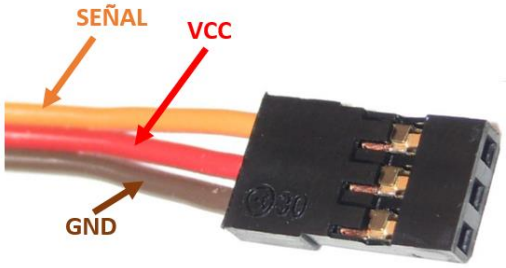


Figura 4.4.

Pines de conexión de un servomotor.

Para el diseño de este proyecto, se optó por conexionar dichos pines de cada servomotor de la forma en que se observa en ***Figura 4.5***.

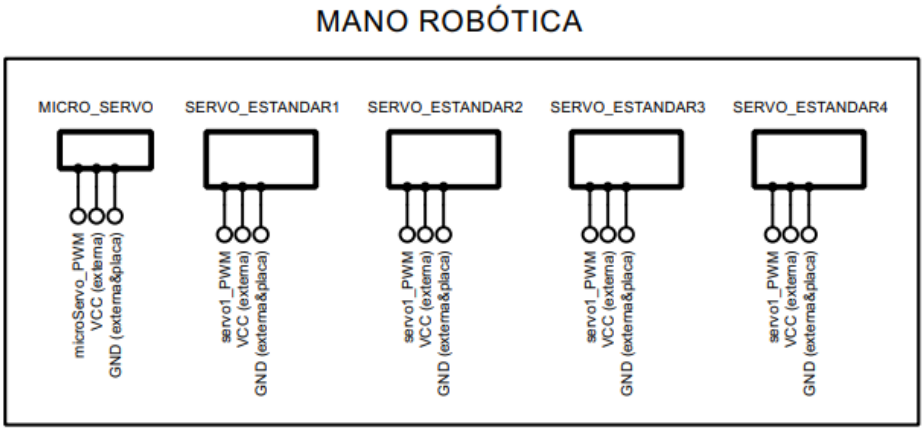


Figura 4.5.

Esquema de conexión de cada servomotor.

- Señal PWM (comunicación)

Se utilizan los pines de control de la placa EDU CIAA para enviar señales de control a cada uno de los servomotores, en especial a los que están diseñados para enviar señales PWM que controlarán a estos, y teniendo en cuenta que los servos elegidos son compatibles con señales PWM.

Es necesario asegurarse de que la frecuencia de la señal PWM esté en el rango de $50Hz$, que es el estándar para los servos, y modular el ciclo de trabajo de la señal para controlar la posición adecuadamente.

- VCC (alimentación)

La placa EDU CIAA es alimentada a través del puerto USB Debug cuando se conecta a la PC, ya que la computadora puede proporcionar la energía necesaria para su funcionamiento. Sin embargo, La capacidad de alimentación está limitada por la de la PC y la corriente máxima permitida por el puerto USB ($500mA$ a $5V$).

Debido a que otros dispositivos, como los servomotores, también se conectan a la placa y requieren una corriente significativa ($5V$), la potencia suministrada por la PC no es suficiente, a causa de que la placa suele tener un regulador de voltaje interno que toma la entrada de $5V$ del puerto USB y proporciona una salida de voltaje estable para alimentar algunos componentes, considerando que el regulador típicamente reduzca la tensión a un nivel más bajo $3.3V$, que será el voltaje en los pines GPIO.

Por lo tanto, se necesita una fuente de alimentación externa. El proyecto fue diseñado para utilizar dos pilas de litio en un porta-pilas y un regulador de tensión, y así conseguir la energía suficiente requerida, haciendo de alguna forma más portable el sistema completo, ganando independencia al utilizar las pilas. Cada pila ofrece $3.7V$ y $7800mAh$, y se conectan en serie en el portapilas, entonces el bloque podría alcanzar ofrecer, teóricamente, $7800mAh$ a $7.4V$.

Sin embargo, debido a que surgió un contratiempo, esta solución no pudo ser utilizada. Los detalles y alternativa propuesta se discuten en ***Sección 6*** y ***Sección 7***, aun así cabe destacar que la solución propuesta originalmente puede ser implementada sin problemas.

Tensión y corriente del circuito

Cada servo requiere de una alimentación promedio de $5V$. Al conectar estos dispositivos en paralelo, los cinco servomotores requerirán una tensión total de $5V$.

Es necesario regular la tensión que ofrece la fuente de alimentación (es decir, las baterías de litio), ya que proporciona una tensión fuera del valor nominal de los servomotores, los cuales no están diseñados para trabajar con valores excedentes a estos. Para esto, se usa un regulador de tensión conmutado (módulo regulador de voltaje) para hacer uso de la batería de manera segura, ajustando la tensión de salida a un valor constante y así garantizar que los servomotores funcionen dentro de sus especificaciones, protegiendo a los mismos de posibles daños provocados por una tensión excesiva y consolidando un suministro de energía constante y confiable.

Se incluyó este elemento en el diseño del sistema para que ofrezca beneficios como la estabilidad de la tensión de salida, protección contra sobretensiones y compatibilidad con diferentes fuentes de alimentación, se optó por un porta-pilas con pilas de litio, pero gracias a este dispositivo el sistema podría adaptarse para funcionar con diferentes fuentes de alimentación, permitiendo un sistema más versátil y operable en una variedad de condiciones.

A su vez, a la salida del porta-pilas se agrega un capacitor para ayudar a estabilizar la alimentación de los servomotores, ya que estos generan picos de corriente que la podrían alterar. Capacitancia:

$$C = \frac{\Delta V . \Delta t}{I}$$

- El porta-pilas teóricamente ofrece $7.8A$.
- El cambio de voltaje depende de la tolerancia de voltaje aceptable durante los picos de corriente, asumiendo que los servos pueden tolerar una caída de voltaje de hasta $0.5V$ (considerando un valor aproximado y contemplando tanto lo que consumen los servomotores estándar como el micro servomotor, para no sobre exigir a ninguno de ellos).
- El tiempo, que depende de cuánto dura el pico de corriente. Considerando un valor arbitrario, se puede decir que $10ms$.

$$C = \frac{0.5V . 0.01s}{7.8A} \approx 64uF$$

Teniendo en cuenta las capacitancias de capacitores electrolíticos comerciales, es una posibilidad redondear el valor obtenido a $100uF$.

Entonces, se usa un capacitor a la salida del regulador. Pero, para mayor seguridad y estabilizar adecuadamente la alimentación de energía proporcionada a cada motor, se decidió sumar un capacitor adicional para cada servomotor, basta con una capacitancia menor así que se utilizó uno de $10uF$.

- GND (tierra)

Simplemente para garantizar seguridad en caso de un cortocircuito o fallo en el sistema y para reducir el riesgo de descargas eléctricas, la conexión a tierra proporciona una referencia de voltaje común para el circuito, lo que es esencial para asegurar que los componentes, en este caso los servomotores, tengan una base común desde la cual operar, evitando que los niveles de voltaje fluctúen y causen mal funcionamiento del sistema. A su vez, la conexión a tierra actúa como una vía para disipar corrientes de sobretensión causadas por descargas eléctricas o interferencias, ayudando a proteger el servomotor.

Por estos motivos, es importante que los componentes tengan una referencia a tierra común. En este caso se puede conectar tanto al negativo del regulador de tensión (conectado al porta-pilas) como a algún punto GND de la placa, generando una referencia a tierra común.

Se conectan todos ellos al mismo GND, resulta más conveniente simplemente reducir la complejidad entre los trazos de cobre de la PCB, ya que sólo es necesario unir todos los GND en un mismo punto.

4.1.3. Bloque Información para el Usuario

Para ofrecer un intercambio de información con el usuario, se utilizan los siguientes recursos adicionales.

4.1.3.1. Pulsador

El pulsador está conectado a un pin de propósito general de la EDU CIAA, también a un puerto GND que ofrece una referencia a tierra y otro VCC para alimentarlo a $3.3V$, por lo que se mencionó en la ***Sección 4.1.2.2***, es decir, el pulsador es alimentado directamente por la placa.

La energía de la placa podría estar limitada por ser alimentada por la PC, porque es algo que depende de las características de la computadora en particular, pero el tipo de pulsador utilizado es capaz de funcionar con valores bastante bajos ($1V$), por ende se considera viable esta conexión.

El pulsador se usa con un resistor en Pull Up, para generar un estado lógico alto (High) cuando está inactivo y un estado bajo (Low) cuando está activo. Se calcula la resistencia con datos aproximados de lo que consume un pulsador estándar,

$$R_{pull-up} = \frac{V_{cc} - V_{boton\ presionado}}{I_{boton\ presionado}} = \frac{3.3V - 0V}{0.01A} = 330\Omega$$

Este valor puede adoptar un valor más comercial, pudiendo estimar el valor a ser mayor a $1k$ pero menor que $10k$ por simplicidad, además no se generará ninguna alteración mayor al circuito por este redondeo, se optó por una resistencia de $10k$.

4.1.3.2. LED RGB

El LED RGB se compone de un terminal para manipular la intensidad de cada color: rojo, verde y azul, y otro pin para la alimentación del dispositivo.

Se planteó el uso de transistores, ya que estos pueden funcionar como interruptores, de manera que un solo color esté encendido a la vez, lo que permitiría un mayor control sobre el LED. Pero, finalmente se descartó esta opción debido a que existe la posibilidad de utilizar algún color intermedio.

Entonces, simplemente con una resistencia limitadora para cada color del LED RGB teóricamente es suficiente. Se calculan las intensidades máximas de corriente para cada color, considerando que la corriente de salida se limita a un valor máximo $I = 6mA$, y que son alimentadas por un pin VCC de la placa, por ende se calcula que disponen de $V_{cc} = 3{,}3V$.

- $R_{rojo} = \frac{V_{cc} - V_{LED}}{I} = \frac{3.3V - 2.0V}{6mA} = 216{,}7\Omega$
- $R_{verde} = \frac{V_{cc} - V_{LED}}{I} = \frac{3.3V - 2.81V}{6mA} = 81{,}7\Omega$

- $R_{azul} = \frac{V_{cc} - V_{LED}}{I} = \frac{3.3V - 2.87V}{6mA} = 71{,}7\Omega$

Si se aproxima estos resultados a valores comerciales, se podría redondear a resistencias de 560Ω, considerando que esta aproximación no afectaría el funcionamiento del circuito más que en la intensidad de brillo que pueda proporcionar el Led, pero no se considera como un requerimiento crucial para este proyecto.

4.2. Placa de circuito impreso (PCB)

En un sistema embebido, la PCB desempeña un papel fundamental. Se trata de una placa plana fabricada de un material no conductor en la que se alojan y conectan distintos componentes electrónicos esenciales del sistema, creando así un circuito eléctrico funcional. El circuito impreso permite que los elementos se dispongan de forma organizada, facilitando la fabricación, ensamblaje y mantenimiento del sistema embebido. Además, un buen diseño es una forma de reducir los posibles errores de conexión y mejorando la confiabilidad del sistema.

En el marco de este proyecto, se emplea de una placa virgen de una sola capa y dimensiones estándar ($100\ x\ 150mm$), que es la forma más básica de PCB, cuyo tamaño es apropiado para las dimensiones de la EDU CIAA, permitiendo incluso la posibilidad de ajustarla a las medidas específicas de esta ($85{,}4\ x\ 137mm$).

Una vez identificados los componentes necesarios y se ha desarrollado un diseño de circuito coherente para el proyecto, ***sección 4.1***, en cuestiones de hardware, se procede al diseño y fabricación de la PCB.

4.2.1. Diseño

Para diseñar la PCB se utilizó la base del diseño esquemático previamente realizado, asignando a cada componente de la lista de materiales una huella adecuada, algunas facilitadas por el entorno de diseño Proteus y otras creadas midiendo minuciosamente el respectivo componente utilizando las funciones que provee el entorno, o bien tomando una huella de Proteus y realizando modificaciones. En ***Figura 4.6*** y ***Figura 4.7*** se muestra un poco del entorno, las funciones y demás cuestiones útiles.

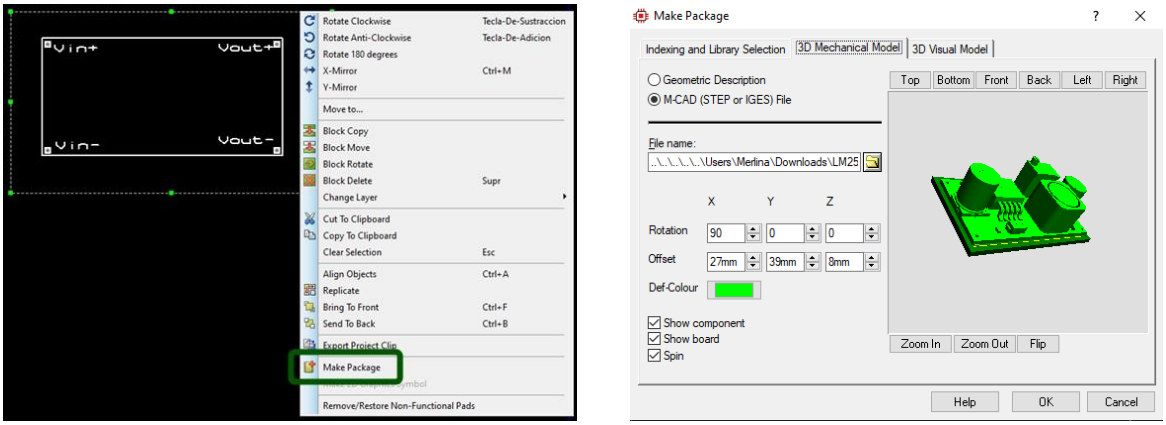


Figura 4.6.

Función Make Package de Proteus e importación de modelo 3D.

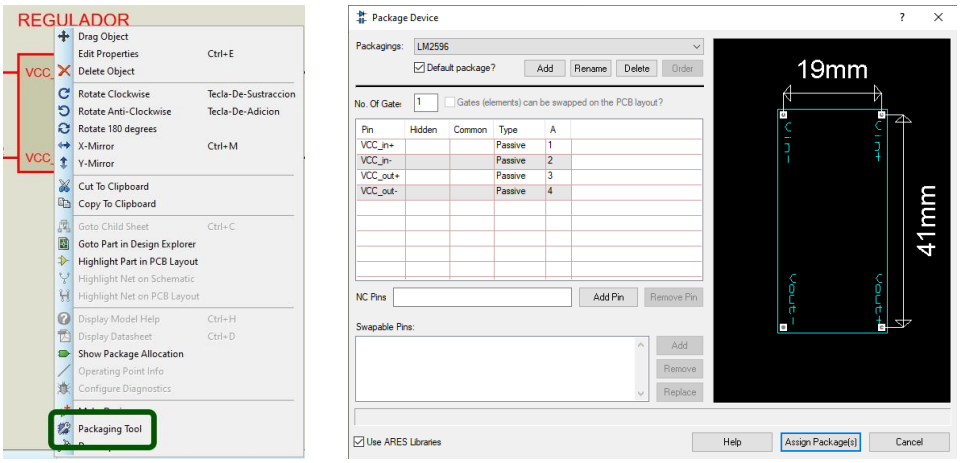


Figura 4.7.

Función Packing Tool de Proteus y asignación de pines según esquemático.

En ***Figura 4.8*** se aprecia el diseño de huella creado para el led RGB, que resultó el modelo más particular, debido a la corta distancia entre los pines del dispositivo. Se optó por buscar una distribución diferente pero más eficiente al momento de soldar el componente.

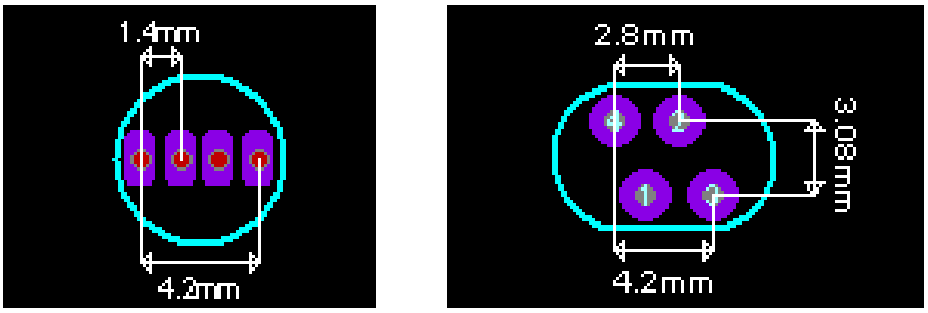


Figura 4.8.

Footprint de Led RGB

Del lado izquierdo se puede observar la huella creada originalmente, pero como los agujeros pasantes estaban tan próximos, se creó la huella de la derecha.

Se tuvieron en cuenta algunos requerimientos para el diseño, como agujeros pasantes de mínimo 0.7mm (modelo *C-70-30*) para asegurar sus dimensiones reales y no se superpongan con otros, ***Figura 4.9***, o los trazos de cobre de un trazo mínimo de *T30* en Proteus, en su mayoría se utilizó *T40,* siempre que alcance el espacio ya que es más grueso, pero proporciona una mayor capacidad de corriente y mejor disipación del calor.

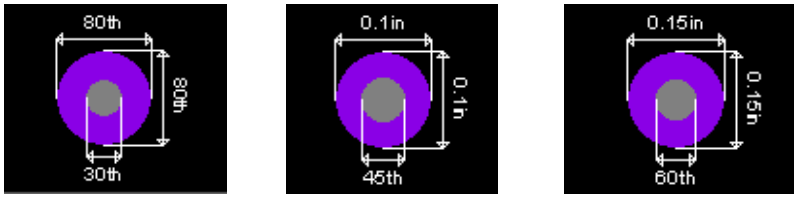


Figura 4.9.

Dimensiones del agujero pasante.

C-80-30, C-100-45, C-150-60

A partir de estas consideraciones, se dispusieron los elementos dentro de los límites de las medidas de la CIAA para que la PCB y la placa encastren perfectamente, ***Figura 4.10***, buscando una distribución eficiente, funcional y conforme al diseño, para garantizar el correcto funcionamiento, desempeño y comprensión del sistema embebido en su totalidad.

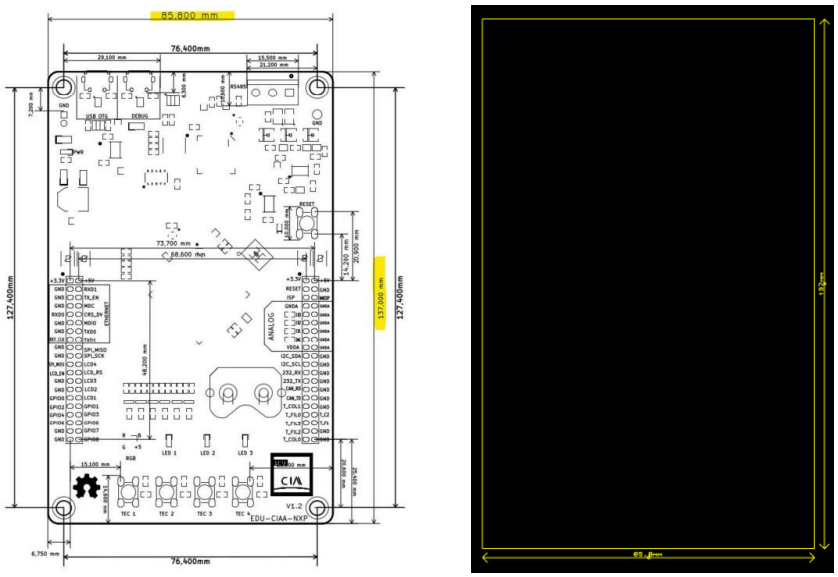


Figura 4.10.

Dimensiones del borde de la placa.

4.2.2. Fabricación

Una vez considerado completo el diseño de la PCB, se llevó a cabo la fabricación de la misma. Partiendo de una impresión a escala del diseño realizado, sobre un papel satinado para favorecer la transferencia del tóner de la imagen impresa sobre la placa virgen.

En la ***Figura 4.11*** se puede observar la placa virgen con la imagen del diseño de PCB transferida mediante la técnica de planchado. Se distinguen algunas partes de color negro, fueron los puntos donde la imagen no se logró transferir adecuadamente y se pintaron con una tinta negra con el objetivo que simule de tóner.

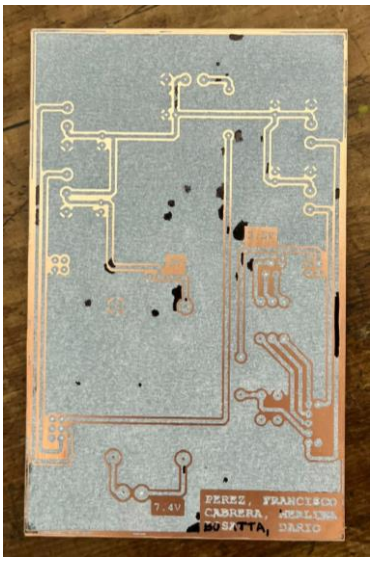


Figura 4.11.

Placa virgen con imagen transferida.

Luego, se procedió a atacar la placa con la imagen transferida y corregida con ácido, cuyo fin es carcomer los trazos que no están recubiertos por tóner o tinta negra para darle vida a la placa. En ***Figura 4.12*** se aprecia la placa después de ser atacada en ácido.

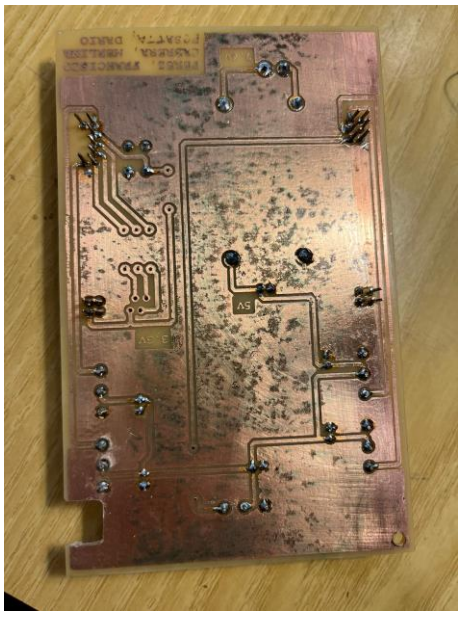


Figura 4.12.

Placa virgen con imagen transferida (se ven componentes soldados porque la imagen fue tomada en tal instancia).

Por último, se realizaron las perforaciones y recortes necesarios, para proceder a soldar manualmente los componentes en sus respectivas posiciones. En ***Video 4.1*** se encuentran distintos acercamientos del proceso y resultado final de la soldadura de los componentes.

https://drive.google.com/drive/folders/1LSowSgYvcj4ojAYX_1c0wHjzseQO3aeH?usp=drive_link

Video 4.1.
Soldadura de componentes a PCB.

En ***Video 4.1***, en una de las imágenes adjuntas se puede observar 4 cables en las puntas que correspondería soldar al regulador de tensión, esta medida fue tomada debido a que hubo algún error con las medidas exactas del componente al crear su huella, y los agujeros pasantes quedaron levemente más abiertos. Por ende, se recurrió a soldar un filamento de cobre por agujero pasante, así poder apenas doblarlos para que encastre el regulador y pueda ser soldado a la PCB como corresponde.

Finalmente, se adjunta imagen en ***Figura 4.13*** de la PCB con todos los componentes soldados.

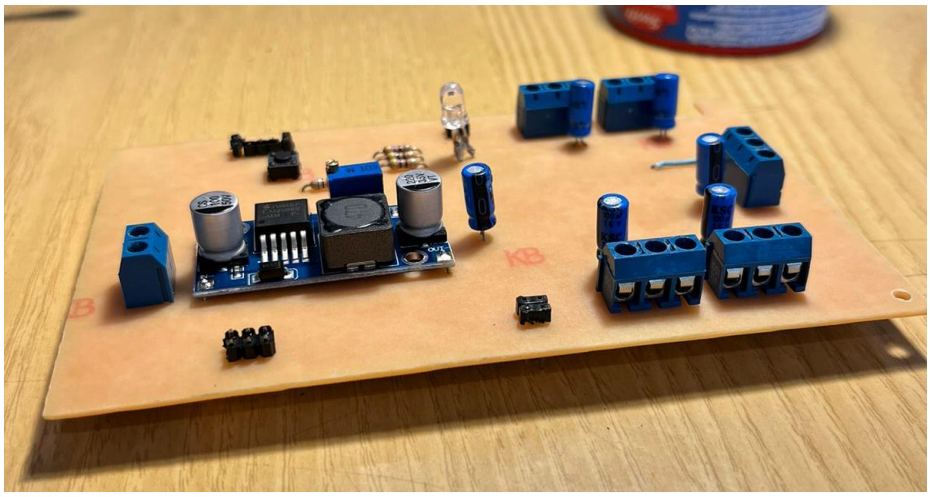


Figura 4.13.
PCB final.

5. Diseño del firmware, simulación y depuración

Esta sección se enfoca en el diseño y desarrollo de un software con el objetivo principal de aprovechar la potencia de la "visión de la computadora" y la capacidad de comunicación serie, utilizando una cámara web para la detección de una mano humana y establecer una comunicación robusta con la placa EDU CIAA NXP.

5.1. Bibliotecas utilizadas

Dentro del ámbito de la programación, las bibliotecas son conjuntos de código pre-escrito y funciones que pueden ser reutilizadas por otros programas. Estas bibliotecas contienen rutinas y funciones que abordan tareas comunes o específicas. Proporciona una forma eficiente de acceder a funcionalidades complejas sin tener que volver a escribir todo el código desde cero. El software diseñado hace uso de librerías matemáticas, Computer Vision y conexión serial con la CIAA.

La Computer Vision (visión por computadora) es un campo de la inteligencia artificial, y la informática que se centra en la capacitación de las computadoras para interpretar y comprender el mundo visual. En esencia, se trata de enseñar a las máquinas a "ver" y extraer información útil de imágenes o vídeos, similar al funcionamiento humano.

Se hace uso de las siguientes bibliotecas, todas facilitadas en ***Bibliografía***.

5.1.1. Detección en tiempo real

- Biblioteca "CVZONE", ***[3]***

Se trata de una biblioteca basada en Open CV (Open Source Computer Vision Libreary). Open CV es una de las máquinas más populares y ampliamente usadas para la CV (Computer Vision), que proporciona una amplia gama de funciones y algoritmos para procesar imágenes y videos, incluyendo:

- **Lectura y escritura de imágenes y vídeos:** Permite cargar imágenes y vídeos desde archivos y guardar los resultados procesados en diferentes formatos.
- **Procesamiento de imágenes:** Open CV ofrece funciones para realizar una variedad de operaciones en imágenes, como filtrado, transformaciones geométricas, suavizado, segmentación, extracción de bordes, entre otros.
- **Detección de objetos y seguimiento:** Permite utilizar algoritmos de detección de objetos, como Haar Cascades o el detector de características HOG, así como realizar seguimiento de objetos en secuencias de vídeo.
- **Reconocimiento facial:** OpenCV incluye un módulo de reconocimiento facial que permite detectar y reconocer rostros en imágenes y vídeos.
- **Calibración de cámaras:** Se puede utilizar para calibrar cámaras y realizar correcciones geométricas en imágenes.

- **Aprendizaje automático:** OpenCV se integra con otras bibliotecas de aprendizaje automático como TensorFlow y PyTorch para aplicaciones más avanzadas de visión por computadora.

- Biblioteca Mediapipe, ***[4]***

Es una biblioteca de código desarrollada por Google, que facilita el desarrollo de apliaciones de vision por computadora y procesamiento de medios.

Su objetivo es facilitar el desarrollo de aplicaciones que involucran el análisis de medios (como imágenes y vídeos) y la extraccion de información específica, como detección de objetos, seguimiento facial, estimación de poses humanas y más. Dentro de la comunidad de desarrolladores, se destaca por su enfoque en el aprendizaje automático y la interferencia en tiempo real en dispositivos móviles y sistemas embebidos.

5.1.2. Comunicación serial con placa CIAA

- Biblioteca “PySerial”, ***[8]***

Cuya funcionalidad consiste en una clase para abstraer un dispositivo conectado a través de USB Serial. Es una de las más populares para la comunicación serie en Python gracias a su utilidad, simplicidad y estabilidad. Otorga una amplia gama de funciones para diversos usos en el entorno de la comunicación serial:

- Detección de puertos disponibles.
- Configuración del puerto serie.
- Lectura y escritura de datos.
- Control de flujo.
- Temporización y espera.

5.1.3. Bibliotecas adicionales

Adicionalmente, se han utilizado las bibliotecas Math, una estándar del lenguaje Python y Numpy para disintos cáclulos matemáticos cuyos resultados serán enviados a la placa EDU CIAA NXP.

A partir de las herramientas descritas, se logra desarrollar un programa que capte los movimientos en tiempo real de cualquier parte del cuerpo. En este caso particular, se busca la detección de una mano, que se usa como referencia para el movimiento de unidad robótica.

5.2. Funcionamiento del programa de detección

A través de la utilización de los módulos CvZone y Mediapipe, se logra la detección de la mano que aparezca en frente a la cámara web a la velocidad de fotogramas del dispositivo de grabación. Para este proyecto, se utiliza una cámara de 30 fotogramas por segundo.

A su vez, estos modulos generan un resultado en pantalla para saber qué es lo que el programa está analizando en cada una de esas imágenes, como se muestra en ***Figura 5.1*** y ***Figura 5.2***.

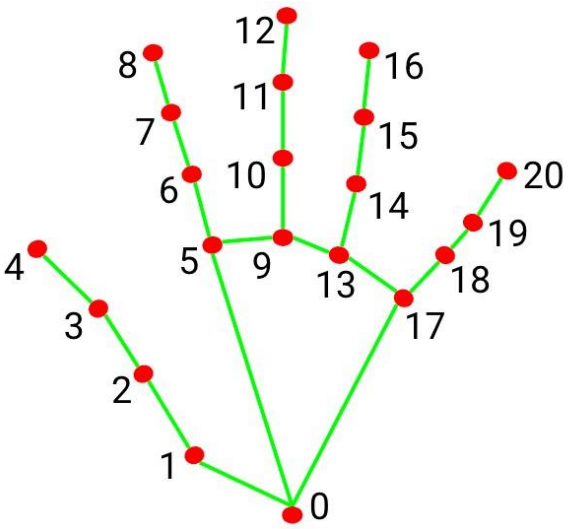


Figura 5.1.

Puntos y referencias tomadas por el programa.

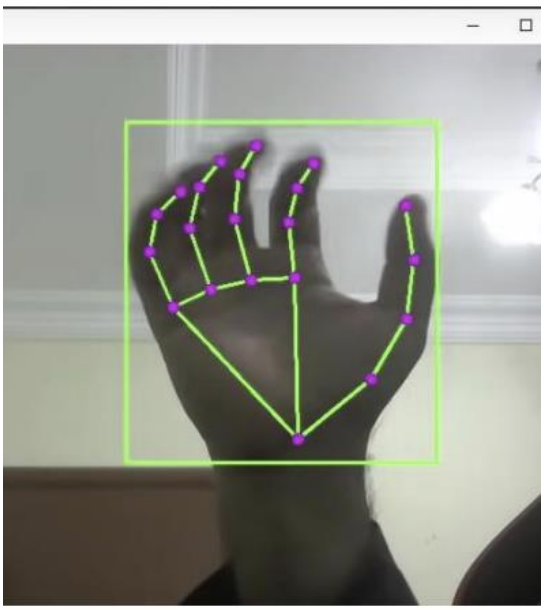


Figura 5.2.

Muestra en pantalla el fotograma procesador al captar una mano.

Una vez conseguida la correcta detección de la mano real, se prosigue con la metodología de analizar la posición y apertura de cada dedo de la misma. Se buscó implementar el software de detección de tal forma que pueda ser escalable desde algo simple como “dedo arriba” y “dedo abajo”, como también algo capaz de medir movimientos intermedios entre estos estados.

Para lograrlo, se ideó la implementación de un algoritmo que mida las distancias entre un punto extremo y un punto de referencia, siento las falanges distales los extremos y, la cabeza del metacarpiano del dedo meñique y la muñeca los ejes

para el pulgar y los demás dedos respectivamente, como se muestra en la ***Figura 5.3***.

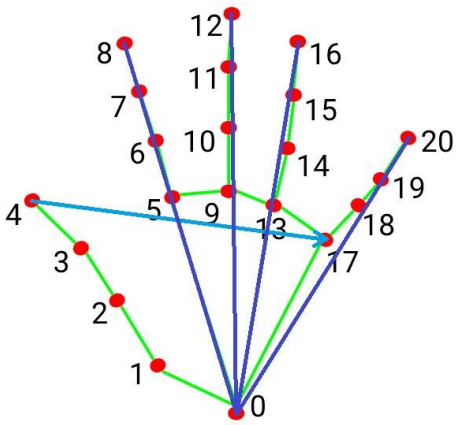


Figura 5.3.
Referencia de distancias.

Las bibliotecas utilizadas para la detección de la mano ofrecen la funcionalidad de extraer la ubicación de cualquiera de estos puntos en base a coordenadas X e Y en referencia a la resolución de ventana configurada. Partiendo de esta información, puede calcularse la distancia entre un punto extremo y su eje de referencia utilizando el teorema de Pitágoras, que es implementado a través de la biblioteca Math, que posee la función *hypot()*, facilitando la escritura del código, y funciona de la siguiente forma,

$$Distancia = math.hypot(x[extremo] - x[referencia], y[extremo] - y[referencia])$$

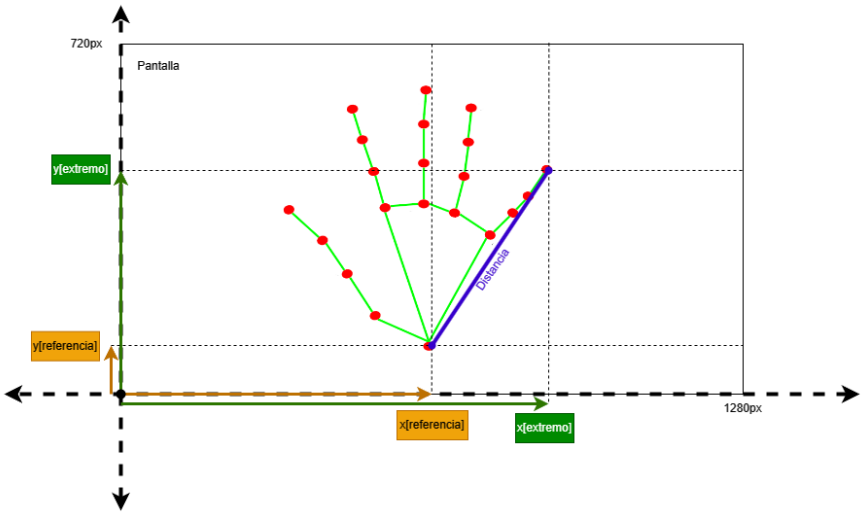


Figura 5.4.
Ejemplo de medición de distancia.

Se muestra un ejemplo gráfico en la ***Figura 5.4***. Luego, en ***Figura 5.5*** y ***Figura 5.6*** se muestran los resultados de un programa de prueba.

Se puede observar que los valores brutos rondan desde 127 hasta 690 aproximadamente, aunque como se mencionó previamente, estos son medidos de acuerdo al tamaño de la mano en relación a la ventana donde se muestra la imagen, por lo que si la mano se aleja o acerca, estos valores serán diferentes. Para simplificar, se trabaja con valores aproximados a los que se observan en ***Figura 5.5*** y ***Figura 5.6***.

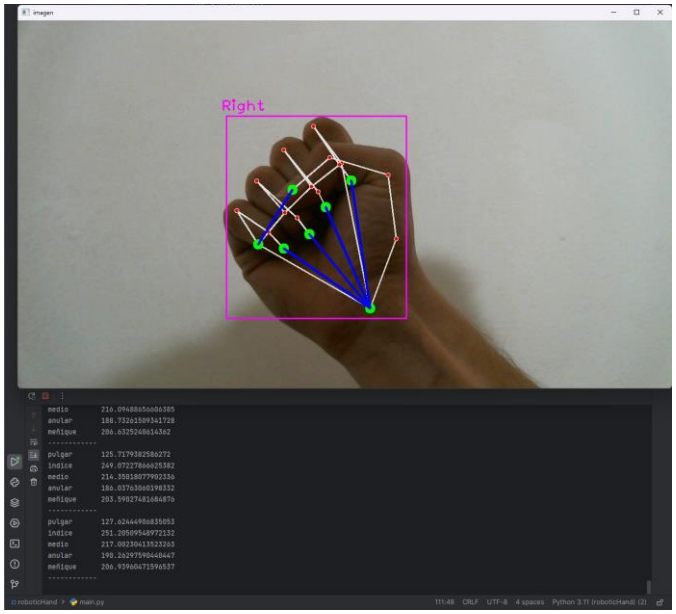


Figura 5.5.

Distancias con dedos abajo.

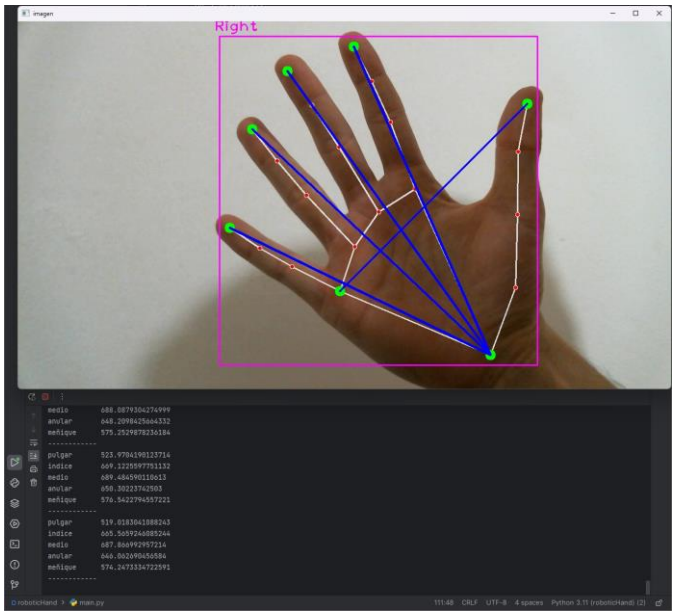


Figura 5.6.

Distancias con dedos arriba.

En base a las distancias obtenidas, se busca obtener valores más estables, facilitando la visualización y el procesamiento de los mismos por parte del

software de la placa CIAA. Aquí es donde entra en juego la biblioteca Numpy, que es utilizada para interpolar linealmente los valores gracias a la función *interp()*, logrando así un rango acotado en los valores definidos por el usuario. Adicionalmente, a estos valores se los convierte en enteros usando la función *round()*, que es inherente al lenguaje.

Dicho algoritmo funciona de la siguiente manera, en base a lo mostrado en ***Figura 5.5*** y ***Figura 5.6***, y buscando 2^n niveles de precisión,

$$DatoListo = round(numpy.interp(ValorBruto, [127, 690], [0, (2_n - 1)]))$$

Las distancias, una vez calculadas e interpoladas, son almacenadas en un arreglo de 5 elementos de tipo carácter, listos para su envío a la placa. En ***Figura 5.7***, ***Figura 5.8*** y ***Figura 5.9*** se muestran ejemplos con diferentes niveles de resolución.

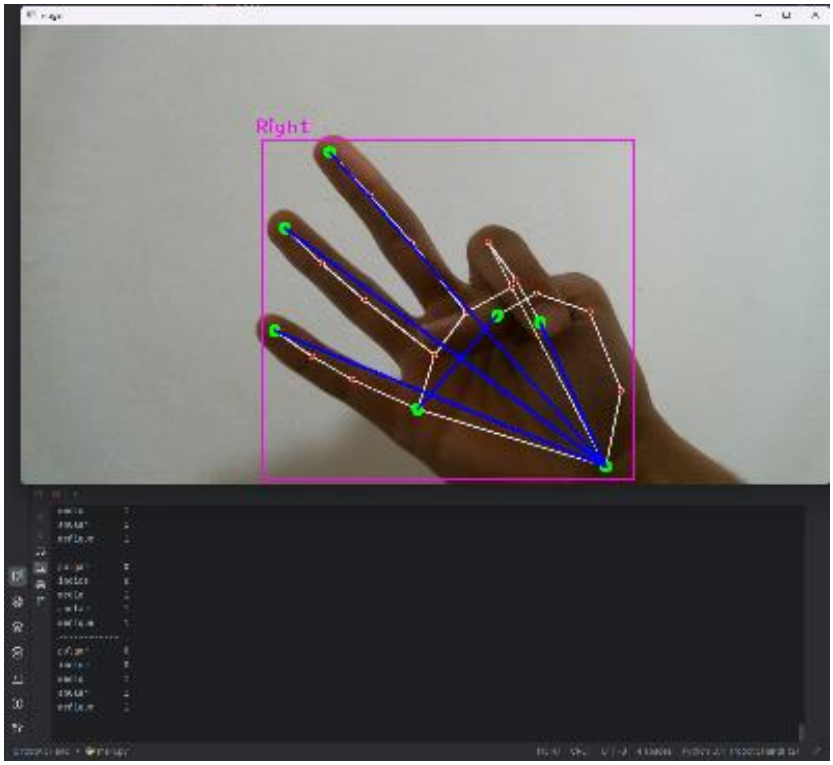


Figura 5.7.
2 Niveles de resolución

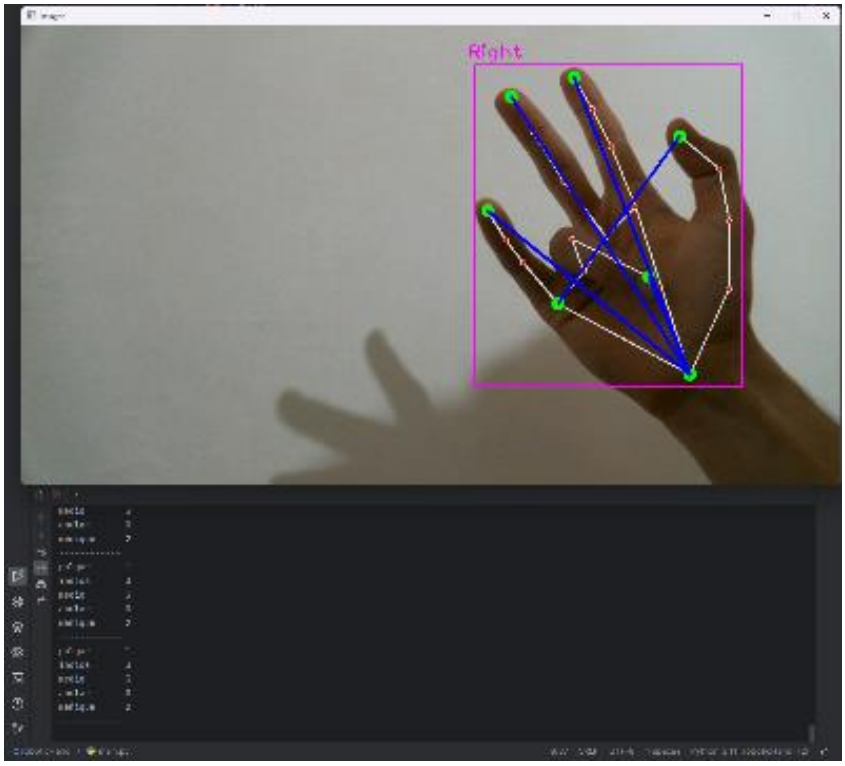


Figura 5.8.

4 Niveles de resolución

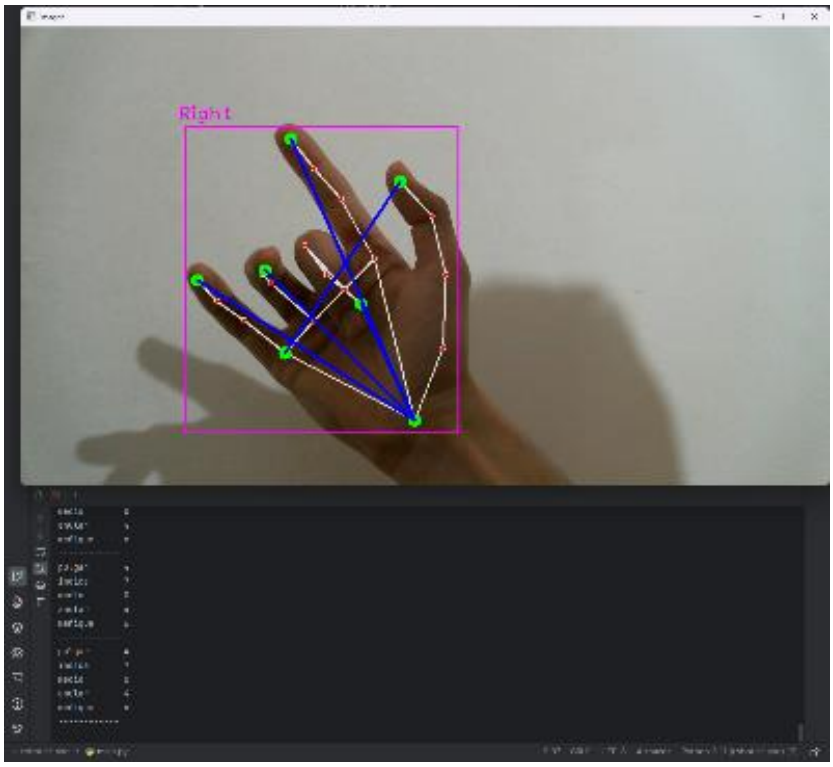


Figura 5.9.

8 Niveles de resolución

5.3. Transferencia de información

La ultima tarea del software de detección es llevada a cabo por la biblioteca PySerial, que se encarga de transferir el arreglo de datos a la placa CIAA haciendo uso del puerto DEBUG de la misma, utilizando de medio un cable USB.

Esta biblioteca será utilizada para el envio y recepción de datos y mensajes de sincronización en protocolo serie por medio de caracteres en la comunicación entre la placa y la PC. La particularidad de los caracteres, es que cada uno de ellos ocupan un byte de espacio (u ocho bits), por lo que la configuración más conveniente es la 8N1: ocho bits de datos, sin paridad, y un bit de parada.

La comunicación será bidireccional a una alta velocidad (en el contexto de la comunicación serial) de 115200 baudios, lo que posibilita una alta precisión en tiempo real y goza de una amplia compatibilidad con la mayoría de los dispositivos y microcontroladores modernos, simplificando la configuración y garantizando la interoperabilidad. Adicionalmente se implementarán medidas de sincronización adecuadas en ambos extremos.

Para establecer la conexión con el puerto correspondiente con la configuración antes mencionada, se utiliza la función *Serial()* de la siguiente manera,

$$conexion = serial.Serial(\text{“}COMx\text{”}, bytesize = 8, parity = serial.PARITY\_NONE, stopbits = serial.STOPBITS\_ONE, baudrate = 115200)$$

Siendo necesario reemplazar la ‘x’ en “COMx”, para conectarse al puerto adecuado.

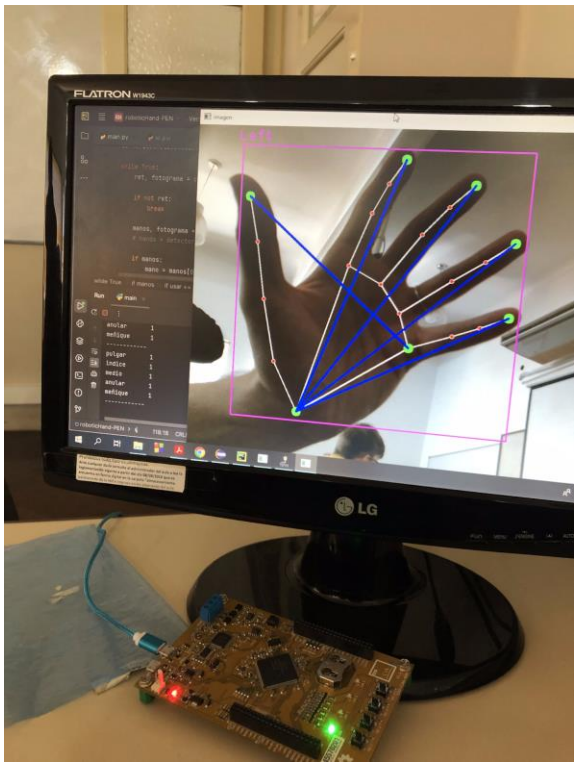


Figura 5.10.

Placa lista para recibir datos.

Una vez realizada la conexión, el programa pide al usuario que establezca el nivel de resolución inicial, este valor llamado N, es el encargado de configurar el programa para generar 2 N pasos entre la extensión mínima y máxima de un dedo. Así, el programa envía la instrucción "aN" a la CIAA para sincronización y, a partir de ese instante, la placa se encuentra lista para recibir los cinco datos, y como se muestra en ***Figura 5.10***. Estos son enviados uno por uno, por lo que se necesita un total de 5 envíos para que el microcontrolador pueda comenzar a operar con los mismos.

5.4. Placa EDU CIAA NXP

En este proyecto, se decidió que el firmware esté basado en la biblioteca SAPI que provee el proyecto EDU CIAA, siendo esta una biblioteca de código abierto diseñada para facilitar el desarrollo de aplicaciones embebidas en placas como la EDU CIAA, que provee una colección de funciones que simplifican la interacción entre el hardware y los periféricos a utilizar.

Esta biblioteca ayudará en la gran mayoría del desarrollo del sistema, ya que contiene funciones elementales en el objetivo del proyecto.

Por otro lado, se usa una arquitectura Time Triggered cooperativo en donde se debe definir el periodo en el cual se recibirán los datos, se procesarán y en base a los resultados, el movimiento de los servomotores.

5.4.1. Estados del sistema

El sistema contiene inicialmente 6 estados, varios de ellos relacionados con la comunicación con la PC mediante USB, y otros para esperar la obtención completa de los datos requeridos, para en base a estos, controlar los diferentes dispositivos que componen la mano robótica. *Figura 5.11*.

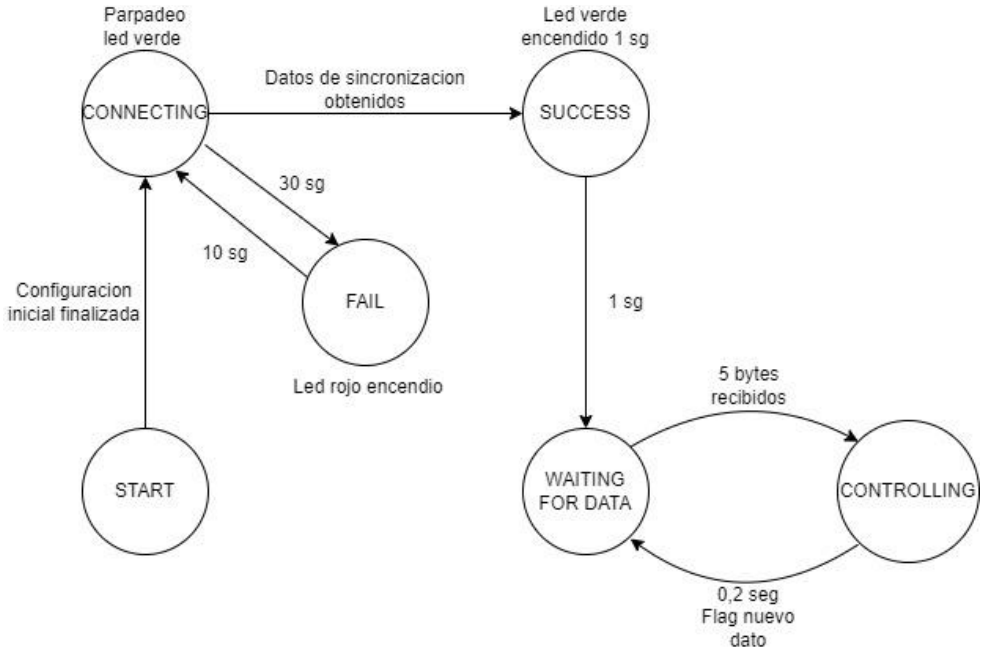


Figura 5.11.
Diagrama de estados.

A continuación se realiza una descripción de los estados que se observan en ***Figura 5.11***,

- **START**: es el estado inicial del sistema, inicializando cada módulo, pines y periféricos, como la inicialización de la UART.
- **CONNECTING**: en este estado, la placa espera un mensaje de 2 bytes para la sincronización y para definir los niveles de resolución de los datos. Como primer byte, espera un carácter “a”, seguido de un segundo byte definido como *Nivel*, que es un número entre 0 y 9. Este byte, es utilizado para luego convertir el número recibido por el software de reconocimiento en un ángulo correcto.

  Si dentro de 30 segundos no recibe los 2 bytes esperados, transiciona al estado FAIL. Caso contrario, al estado SUCCESS. ***Pseudocódigo 5.2***.

```
mientras no se reciban todos datos de sincronizacion{
      esperar (no bloqueante)
      hacer parpadear el led azul
      si pasaron 30 sg {
           transicionar a estado FAIL
      }
}
definir globalmente el nivel de resolucion luego de recibir los
datos
transicionar al estado SUCCESS
```

Pseudocódigo 5.2.

- **FAIL**: en este estado, se le informa al usuario mediante el LED en color rojo que la comunicación no pudo ser establecida. Luego de 10 segundos, transiciona automáticamente al estado START. ***Pseudocódigo 5.3***.

```
encender el led rojo
si pasaron 10 sg {
     transicionar al estado CONNECTING
}
```

Pseudocódigo 5.3.

- **SUCCESS**: este estado pondrá el LED en color verde, notificando que la comunicación fue exitosa. Luego de un par de segundos, transiciona al estado WAITING FOR DATA. ***Pseudocódigo 5.4***.

```
encender el led RGB de color verde
si pasó 1 segundo {
          transiciona a WAITING FOR DATA
}
```

Pseudocódigo 5.4.

- **WAITING FOR DATA**: durante este estado, el sistema esperará por parte de la PC, datos relacionados a la posición de los dedos (captados por cámara). La cantidad total de bytes que espera recibir son 5, donde cada byte en particular contiene información de la posición de los dedos de la mano real. El byte 1 corresponde al dedo pulgar, el byte 2 al índice, y así siguiendo con todos los dedos. Luego de recibir los 5 bytes, transiciona al estado CONTROLLING. Si se pulsa el botón, este espera a recibir los datos desde la PC y transiciona al estado CONNECTING ***Pseudocódigo 5.5***.

```
mientras que no se reciban los 5 bytes por UART {
esperar (no bloqueante)
}
si se recibieron los bytes {
           si se presiono el botón {
                transiciona a CONNECTING
           }
           transiciona a CONTROLLING
}
```

Pseudocódigo 5.5.

- **CONTROLLING**: este estado se encarga de enviar los ángulos correspondientes a cada servo, y de esperar cierto tiempo antes de avisar por comunicación que le envíen nueva información en relación al software, corriendo en paralelo. Esto es necesario, ya que el movimiento de los servomotores tarda algunos milisegundos en ser ejecutado y alcanzar el ángulo deseado. Una vez que los 5 bytes fueron recibidos exitosamente, envía a la PC una notificación de que así ha sido. ***Pseudocódigo 5.6***.

```
verificar para cada servo si el ángulo en el que está el servo es
igual al nuevo valor recibido {
      si es distinto { mover servo }
      si es igual { no enviar la acción de mover servo }
      }
si pasó 0.2 segundos {
     envía flag para comunicar que espera 5 nuevos bytes
           transiciona a WAITING
}
```

Pseudocódigo 5.6.

Como se puede observar en los estados, es importante la definición de un protocolo de comunicación para el correcto intercambio de mensajes, ya que de otra forma, la PC además de no saber en qué instante comenzar a enviar datos, enviaría de manera indiscriminada y sobrecargando el buffer de la recepción por parte de la placa, ya que el software es mucho más rápido que lo que implica el procesamiento de los datos recibidos y el movimiento mecánico de los servomotores.

Se realizaron pruebas con un único servomotor, ya que aún no se disponen la totalidad de componentes necesarios para realizar pruebas con todos los servomotores al mismo tiempo, para verificar la correcta conversión de los niveles de resolución. ***Video 5.2***.

https://drive.google.com/drive/folders/1CRKDr1bkJyAz45kxgyx7mFy_Cj32axGV

Video 5.2.

Pruebas en un servomotor al recibir datos mediante distintos niveles de resolución.

Se puede observar, que la placa espera los 2 bytes de sincronización como ‘a1’, ‘a2’ o ‘a3’, y cómo dependiendo de esto, el servo realiza movimientos entre 0 y 180 con más pasos. Al utilizar una mayor resolución a 1, la polea del servomotor ya no solo abre y cierra, sino que también puede estar entreabierta.

Por otro lado, también se verificó que un servomotor, en este caso el dedo pulgar, imite con resolución 1 (abierto y cerrado) lo que el software externo está procesando y comunicando. ***Video 5.3***.

https://drive.google.com/file/d/1Hs8Q4EMPlbSYpqusVegYVLSBjPALUwYm/view?usp=drive_link

Video 5.3.

Prueba sobre micro servomotor respondiendo a las señales que envía la PC.

5.5. Placa EDU CIAA y Mano Robótica

En esta sección se describen los tipos de conexión y las señales que envía la placa EDU CIAA a los componentes de la mano robótica para que esta sea funcional, en base a los datos que esta recibe por parte de la cámara.

5.5.1. Control de los servomotores

Se hará uso de una de las bibliotecas que provee la EDU CIAA, una de ellas es la que se encarga de diferentes tareas de los servos tales como iniciar al servomotor, moverlo, leer su estado, entre otras. La biblioteca precisamente es ***sapi_servo.h***, algunas funciones que ofrece y serán de ayuda en este proyecto son,

- ServoInit: inicializa el servomotor especificado.
- ServoRead: devuelve el valor actual del servomotor en 0° y 180°.
- ServoWrite: cambia el valor del servomotor entre 0° y 180°.

Es importante destacar que utiliza los temporizadores 1, 2 y 3 del microcontrolador para generar las señales.

Por otra parte, fue necesario el desarrollo e implementación de ciertas funcionalidades que, en base al dato de configuración, mencionado en la ***sección 5.2*** como N, que establece el nivel de resolución, y el dato recibido, genera un ángulo de movimiento adecuado del servo correspondiente.

Estas funciones, en conjunto, calculan el ángulo de giro de la siguiente forma, siendo necesario cuantificarlo para cada valor recibido e individualmente para cada servomotor,

$$angulo = redondeo\ (\frac{valorRecibido * 180}{2^N - 1})$$

En ***Figura 5.12***, se muestra una tabla con los ángulos adecuados para los primeros cuatro niveles de resolución.

N=	1		2		3		4	
	ÁNGULO	DATO	ÁNGULO	DATO	ÁNGULO	DATO	ÁNGULO	DATO
	0	0	0	0	0	0	0	0
	180	1	60	1	26	1	12	1
			120	2	51	2	24	2
			180	3	77	3	36	3
					103	4	48	4
					129	5	60	5
					154	6	72	6
					180	7	84	7
							96	8
							108	9
							120	10
							132	11
							144	12
							156	13
							168	14
							180	15

Figura 5.12.

Tabla que muestra los ángulos para cada nivel.

6. Ensayos y mediciones

Debido a la naturaleza de este proyecto, durante el desarrollo y la fabricación de la PCB no fue posible llevar a cabo pruebas exhaustivas para detectar fallos en el código o la estructura, aparte de algunas pruebas básicas realizadas con el micro servomotor. Esto se debió a limitaciones técnicas, como la incapacidad de la placa para alimentar más de un servomotor a la vez, y a problemas de rendimiento cuando se probó con un servomotor estándar, debido a la carga que generaba en la CIAA.

Hasta el momento, se ha verificado que las señales PWM puedan ser enviadas a cada servomotor según el pin PWM conectado a la placa, lo que asegura la comunicación y el correcto recibimiento de las señales.

Entonces, una vez fabricada la PCB, se procedió a realizar las pruebas necesarias para validar su funcionamiento y asegurar que cumpla con los requisitos de este proyecto.

6.1. Pruebas previas a la PCB

Estas pruebas fueron realizadas previo a tener la PCB finalizada y sin la fuente de alimentación necesaria para poder conectar y probar el funcionamiento de todos los servomotores en simultáneo. La CIAA puede alimentar con seguridad el micro servomotor, por lo que se pone a prueba de forma individual, utilizando los pines VCC y GND que ofrece la placa.

A continuación se muestra un video donde se ve la interacción entre ambos componentes,

- Prueba de conexión con $N = 3$, ***Video 6.1***.

https://drive.google.com/file/d/1caYXcHvZQs6tqGm9-kHxgN7-IFczN--4/view?usp=drive_link

Video 6.1.

Prueba sobre micro servomotor respondiendo a las señales que envía la PC, con resolución N=3.

- Prueba de conexión con $N = 2$, ***Video 6.2.***

https://drive.google.com/file/d/1cVOPwnyh7xOcrE51v5xqOV3esp5HBB1E/view?usp=drive_link

Video 6.2.

Prueba sobre micro servomotor respondiendo a las señales que envía la PC, con resolución N=2.

- Prueba de conexión con $N = 1$, ***Video 6.3.***

https://drive.google.com/file/d/1Hs8Q4EMPlbSYpqusVegYVLSBjPALUwYm/view?usp=drive_link

Video 6.3.

Prueba sobre micro servomotor respondiendo a las señales que envía la PC, con resolución N=1.

6.2. Medición de voltaje y corriente

Previo a conectar la PCB a la CIAA para comenzar a realizar pruebas generales, se llevó a cabo una serie de comprobaciones para asegurar su correcto funcionamiento.

Se verificó que el regulador de tensión cumpla su función y al recibir $7.4V$, como se puede observar en ***Figura 6.1***, mediante una pequeña configuración, entregue $5V$. Se utilizó una fuente externa configurada para brindar dicha energía.

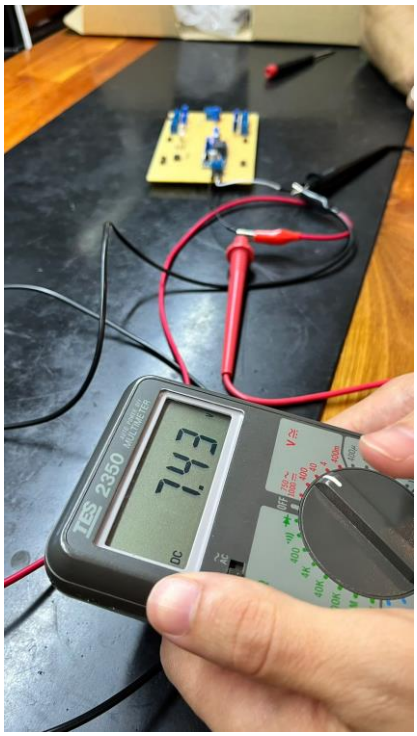


Figura 6.1.

Voltaje que recibe el regulador de tensión de la fuente externa.

A partir de esto, se inspeccionaron cuidadosamente los trazos de cobre en la PCB para asegurar que estuvieran completos, sin cortocircuitos ni discontinuidades, para corregirlas previo a poner en marcha el sistema, y asegurar que la alimentación proporcione suficiente corriente para todos los elementos que componen la placa. Se revisaron las interconexiones nuevamente, para verificar el correcto diseño y realización, y una inspección visual de los componentes montados, correcta soldadura y ubicación.

https://drive.google.com/drive/folders/1etoEVv7ESKzKeY0lYjDTWZVk_a06cwsu?usp=sharing

Video 6.4.

Pruebas iniciales del funcionamiento de la PCB y sus trazos.

En los videos adjuntos en ***Video 6.4***, se pueden observar algunas de las medidas realizadas con ayuda de un multímetro. Estas se enfocan en la energía que recibe y envía el regulador de tensión, se muestra cómo es configurado el dispositivo. Y también, chequear que las borneras a las que van conectados los servomotores reciban la alimentación que brinda el regulador.

6.3. Prueba de funcionamiento de 5 servomotores a la vez

Una vez conectados todos los componentes y alimentado el circuito, se probó que los cinco servomotores funcionen a la vez, es decir, que la energía provista sea suficiente. Como en ***Figura 6.2***, así se ven los servomotores conectados a las borneras y la placa siendo alimentada por una fuente externa.

Efectivamente no hubo problemas mayores, todos los servomotores funcionaron perfectamente.

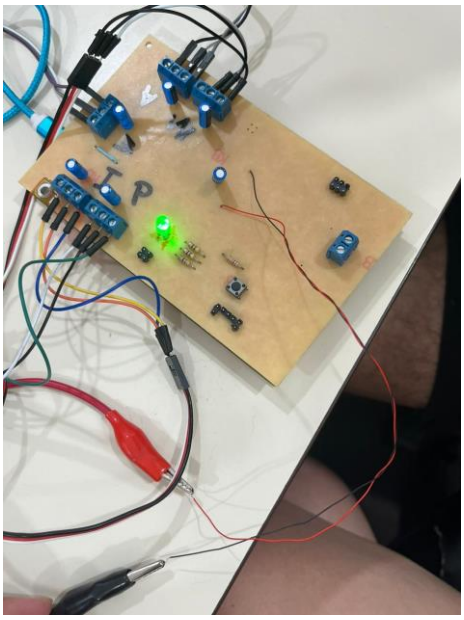


Figura 6.2.
Todos los componentes conectados a la PCB, incluyendo la CIAA, y alimentada mediante una fuente.

6.3.1. Fallo con trazos de cobre

Como se menciona en la ***Sección 4***, el planchado y la transferencia en la fabricación de la PCB no fue perfecta. Previo a esta prueba, se detectó que uno de los trazos de cobre que relaciona la señal PWM que envía la placa a uno de los servomotores estaba afectado.

Ante la duda, se realizó un puente con un cable con el objetivo de fortalecer este trazo inconsistente. ***Figura 6.3***. Esto se discute mejor en la ***Sección 7***.

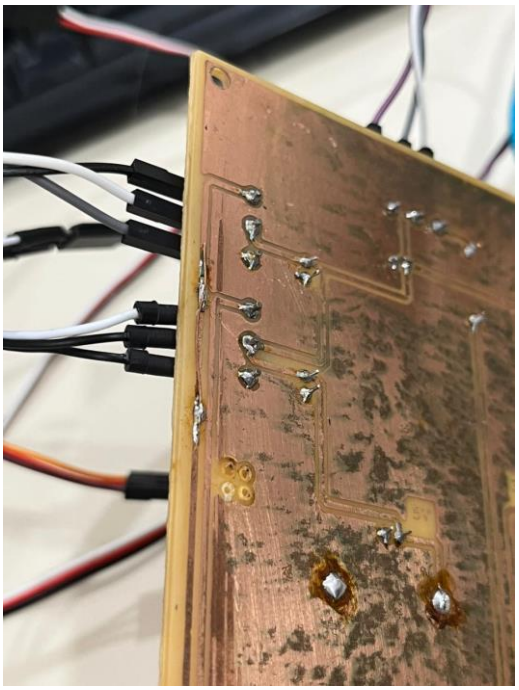


Figura 6.3.

Puente realizado ante un trazo de cobre débil.

6.3.2. Inconsistencias con la alimentación del circuito

Como se mencionó en esta y en la sección previa, las pruebas se realizaron utilizando una fuente externa como fuente de alimentación del circuito. El motivo es que al realizar pruebas con el porta-pilas y las pilas de litio como se ideo originalmente, había inconsistencias con la energía entregada, incluso en algunas ocasiones la carga era consumida a los pocos minutos.

Por esta razón y para proteger el circuito, se desistió del uso del porta-pilas. Además, luego de la pérdida del regulador de tensión, lo que se detalla en la ***Sección 7***, la mejor opción fue usar la fuente externa.

6.4. Calibración de estructura física

Luego de corroborar que los servomotores funcionan en conjunto y por separado, a prueba y error se buscó calibrar el giro de los servomotores, intentando hasta dar con el largo de hilo de pesca y tensión del elástico justos, para obtener un movimiento fluido y con fuerza al accionar el servomotor.

En el ***Video 6.5*** hay material que permite apreciar la forma en que los hilos de pesca conectan con los respectivos servomotores, y el movimiento que logra un buen calibrado en los dedos de la mano.

https://drive.google.com/drive/folders/1lGe_vWihRmgZlc3x8z_erkKwH9g8V-kX?usp=sharing

Video 6.5.

Calibre de los dedos.

6.5. Prueba de funcionamiento de pulsador y led RGB

Luego de realizadas las pruebas sobre los componentes más relevantes del circuito, es decir, los servomotores, se procedió a chequear el resto de componentes que cumplen una función secundaria.

Después de algunos retoques en el código que los controla, se logró que tanto el pulsador como el led RGB cumplan sus cometidos dentro del proyecto. Se puede observar en ***Video 6.6***.

https://drive.google.com/drive/folders/1MNiXmoYsuUcFHjY7Q_87RUDYbVqHM_4x?usp=sharing

Video 6.6.

Funcionamiento del pulsador y led.

6.6. Prueba final

Finalmente, con la PCB funcional en base a las pruebas realizadas, se conectan todos los dispositivos y componentes necesarios entre sí para llevar a cabo una prueba final y en conjunto.

En ***Video 6.7*** se encuentra el material que documenta las pruebas finales, desde el funcionamiento del software que acciona la cámara, el movimiento de los dedos de la mano robótica y la imitación fluida de los dedos que capta la cámara. También utilización del pulsador y el led.

Algo a destacar de ***Video 6.7***, es que también se hace foco a la fuente externa, mostrando cuanta energía consume el circuito en reposo y cuanta energía consume funcionando. Se destaca que al poner un objeto o interferir en el movimiento de los dedos cuando estos intentan funcionar, el sistema consume mucha más energía, a que si se lo deja funcionando sin restricciones.

https://drive.google.com/drive/folders/1IGGeLoL_PKUErNY2YvH-wF-XbOAOeKVq?usp=sharing

Video 6.7.

Sistema final.

7. Conclusiones

7.1. Cumplimiento de objetivos

Como principal objetivo del proyecto se planteó que la mano robótica pueda abrir y cerrar los dedos individualmente, sienta esta manejada por un software externo. Y como objetivo secundario, que cada dedo tenga ciertos grados intermedios entre "cerrado" y "abierto". También, se planteó la posibilidad de darle rotación al brazo. ***Sección 2***.

Luego de unos meses, se pudo concretar el objetivo de poder detectar la mano con el software externo y que este mueva los servomotores. Debido a que no se contaba con los materiales en su totalidad, se limitó a hacer pruebas con un solo servomotor.

Una vez alcanzado el principal objetivo, se comenzó a desarrollar la posibilidad añadir grados intermedios, es decir, uno de los objetivos planteados como secundarios. A las pocas semanas dicha meta fue lograda. Pero, en cuanto al objetivo secundario de la muñeca, este fue descartado ya que se utilizó el tiempo restante del proyecto en perfeccionar o solucionar imprevistos.

En principio, el funcionamiento de la PCB en general era correcto. Pero, al conectar todos los servomotores y probar el sistema, se observó que un trazo cerca del borde de la placa impresa estaba sufrió demasiada corrosión al momento de aplicarle el ácido a la PCB. Esto se corrigió realizando un puente con un filamento de cobre. Esto también es comentado en la ***Sección 6***.

Una vez solucionado este error, todos los servomotores funcionaban correctamente. No obstante, en pruebas posteriores se cometió un error al conectar las polaridades de las pilas del regulador de tensión, provocando que este se quemara. Debido a la falta de tiempo, se optó por utilizar una fuente de alimentación externa con voltaje y amperaje regulables.

A su vez, este regulador de tensión funcionaba como un puente entre la tierra de la CIAA y la tierra de la placa impresa, por lo que se tuvo que corregir agregando estaño, unificándose así nuevamente.

7.2. Cumplimiento de objetivos

Durante las semanas de duración del proyecto, se realizaron pruebas para el cumplimiento de los requerimientos, siendo estos los siguientes,

Requerimientos funcionales

1. La cámara web debe ser capaz de capturar imágenes en tiempo real y enviarlas mediante un software a la placa para su procesamiento

Este requerimiento fue el primer objetivo a realizar, ya que sin este, la mano no tenía el sentido que se le había dado, que, mediante algún método, poder mover los servomotores e imitar el movimiento de una mano humana. Por lo tanto, a las dos semanas, ya estaba en funcionamiento.

2. Movimiento individual de cada dedo de la mano impresa en 3D.

Se armó la mano impresa con todo su sistema de movilidad explicado en incisos anteriores, siendo este uno de los últimos requerimientos realizados ya que se necesitaba todos los materiales del proyecto, incluido el PCB.

3. Iniciar el sistema con los 5 dedos cerrados

Se completó satisfactoriamente.

4. Encender el led RGB en color azul cuando el sistema esté iniciado y listo para usar.

La funcionalidad de esto era poder dar un feedback al usuario que utilizaba el sistema para poder saber en que estado estaba.

Hubo un problema el cual no funcionaba el color azul del LED RGB, solo funcionaban el rojo y el verde, por lo cual, por temas de tiempo, se utilizaron estos dos colores. De todas formas, se pudo lograr el poder indicarle al usuario el estado del sistema.

5. Parpadear el led RGB en color verde cuando la cámara web detecte una mano humana y comience a enviar señales.

Se decidió que el led se prenda y apague cuando se haya recibido un nuevo dato, por lo que parpadea de forma rápida si recibe muchos datos. Esto fue así ya que había que diferenciar el parpadeo del led cuando este estaba buscando conexión a que cuando recibía nuevos datos.

6. El botón “stop” deshabilita el movimiento de todos los servomotores

Durante el desarrollo del proyecto, se cambió la funcionalidad del botón “stop” en donde su nueva funcionalidad es para reiniciar el sistema y, así poder cambiar los grados de movilidad de los dedos sin necesidad de desconectar la alimentación de la placa

7. Si el movimiento de los servomotores está deshabilitado, se enciende el led RGB en color rojo.

Como su funcionalidad es distinta como se mencionó, la pulsación del botón reinicia el sistema. La forma de poder saber si no se está recibiendo nuevos datos y moviendo los servomotores, es visualizar si el led RGB está estático.

8. Pulsación del botón bloqueante.

Se logró que la pulsación del botón fuera no bloqueante.

9. Evitar la recepción de información redundante a los servomotores

Para poder cumplir con este requerimiento, se verifica si el nuevo dato que se recibe es distinto al anterior, dedo por dedo, lo cual en el caso de generar el mismo ángulo, no mandar esta acción al servomotor.

Requerimientos no funcionales

1. Movimiento suave de la mano robótica.

Las primeras pruebas realizadas con el software externo y servomotor se hicieron con un movimiento cada 1 segundo, al finalizar el proyecto se logró una imitación con únicamente 200ms de latencia.

2. Respuesta de la mano robótica a partir de la mano real en un tiempo aceptable

Como se mencionó, la mano robótica imita en tiempos de los 200 ms.

7.3. Observaciones finales

Como conclusión final, luego de la realización de los requerimientos y los objetivos alcanzados, se apreció la división de tareas de forma que cada integrante tuviese el tiempo necesario para realizarlas, siendo importante la comunicación entre los miembros y también la cooperación en caso de problemas. Otro asunto a destacar, es que la división de tareas no generó que cada uno trabaje por su parte aislado del trabajo del compañero, si no que se planteaba la solución en grupo y todos intervenían en cierto grado en todos los aspectos del proyecto.

Algo importante, es haber seguido o intentar seguir el cronograma propuesto, ya que nos centrábamos en la etapa del proyecto sin que la ansiedad a poder hacer de más nos generará más problemas. También se aprecia la disponibilidad de consulta, siendo esta fundamental en la finalización del proyecto.

En conclusión, fue una gran satisfacción completar esta materia donde se comienza a trabajar de forma un poco más independiente y se permite aplicar los conocimientos adquiridos a lo largo de las materias previas. Además, a destacar el acompañamiento de los profesores y ayudantes.

En un principio pareció un proyecto algo ambicioso, sin saber hasta qué punto se llegarían a cumplir los objetivos planteados. Pero finalmente, con paciencia y trabajo se concretaron los requerimientos suficientes para hacer funcional este sistema, y se valora que a pesar de cierta incertidumbre con este trabajo, se recibió acompañamiento y ayuda ante los imprevistos surgidos.

Por último, en ***Figura 7.1*** y ***Figura 7.2***, se adjunta el cronograma y seguimiento de las tareas realizadas aproximadamente. También el tiempo invertido por alumno, aunque en este proyecto en particular, no es estrictamente así, ya que como se mencionó se intentó que todos los integrantes aprendan y participen de la mayoría de las tareas. Por más que no esté plasmado que alguien fue responsable o colaborador directamente, estuvo al tanto y tiene conocimiento suficiente al respecto.

	17-08	24-08	31-08	7-09	14-09	21-09	28-09	7-10	12-10	19-10	26-09	2-11	9-11	16-11	23-11	30-11	7-12	14-12	5-02	12-02
Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	1	2
Etapa de investigación																				
Análisis de herramientas disponibles	✓																			
Elección del tema del proyecto		✓																		
Estudio de opciones de diseño mecánico			✓																	
Estudio de tipos de implementación				✓																
Investigación de implementación: cámara				✓																
Etapa de diseño																				
Ensamble preliminar de la mano 3D					✓															
Estudio de conexionado: componentes - placa						✓														
Análisis y cálculos: alimentación del circuito						✓														
Pruebas comunicación placa - PC							✓													
Diseño de circuito esquemático								✓												
Código de cámara + detección de mano								✓												
Comunicación: cámara - PC - placa								✓												
Estudio de la programación de servomotores													✓							
Implementación de código preliminar													✓							
Estudio y análisis de PCB												✓								
Retoques al esquemático para realizar PCB												✓								
Diseño de circuito impreso PCB													✓							
Pruebas preliminares de código													✓							
Rediseño de biblioteca SAPI servo													✓							
Etapa final																				
Fabricación PCB														✓	✓					
Ensamble final de mano robótica																✓				
Ensayos de verificación																	✓			
Pruebas de validación																	✓	✓		
Muestra grupal																		✓		
Entregas formales																				
Informe inicial				✓																
Informe de avance 1								✓												
Informe de avance 2													✓							
Informe final																				✓

Figura 7.1.

Cronograma de realización del proyecto.

Tarea	Semanas dedicadas	Responsable	Colaborador
Definición de componentes	2	Darío Bosatta Merlina Cabrera Francisco Pérez	-
Especificación de software de PC	1	Darío Bosatta	Francisco Pérez
Diseño e impresión de mano 3D	1	Merlina Cabrera	-
Algoritmos de procesamiento de imagen	3	Darío Bosatta	-
Desarrollo de software para PC	3	Darío Bosatta	-
Estudio de recursos y funcionalidades de placa EDU CIAA	2	Francisco Pérez	Darío Bosatta Merlina Cabrera
Ensamble de mano robótica con componentes	1	Merlina Cabrera	-
Programación de EDU CIAA y servomotores	4	Francisco Pérez	Darío Bosatta
Comunicación entre PC y placa EDU CIAA	1	Darío Bosatta Francisco Pérez	-
Análisis de bibliotecas y programación de servomotores	4	Francisco Pérez	Darío Bosatta
Estudio y diseño de esquema de conexión	3	Merlina Cabrera	Francisco Pérez
Corrección de diseño esquemático y estudio de circuito impreso	2	Merlina Cabrera	Darío Bosatta
Diseño de PCB	2	Merlina Cabrera Darío Bosatta	-
Testeo de conexiones (PC – EDU CIAA – componentes)	1	Darío Bosatta	Francisco Pérez
Correcciones a PCB y estudio de fabricación de la misma	1	Merlina Cabrera	-
Confección de poncho - Ensamble	1	Merlina Cabrera	Darío Bosatta
Informes de avance	2	Merlina Cabrera	Darío Bosatta Francisco Pérez
Informe final	2	Darío Bosatta Merlina Cabrera Francisco Pérez	-

Figura 7.2.

División de tareas acordado por el grupo.

8. Bibliografía

[1] Humanoid Robotic Hand 3D. Disponible en:

- https://www.myminifactory.com/object/3d-print-humanoid-robotic-hand-34508

[2] Hoja de datos MPU-6050. Disponible en:

- https://www.alldatasheet.com/datasheet-pdf/pdf/517744/ETC1/MPU-6050.html

[3] CVZNE. Disponible en:

- https://github.com/cvzone/cvzone
- GitHub - cvzone/cvzone: This is a Computer vision package that makes its easy to run Image processing and AI functions. At the core it uses OpenCV and Mediapipe libraries. | https://github.com/cvzone/cvzone
- Computer Vision Zone | https://www.computervision.zone/

[4] Mediapipe. Disponible en:

- https://github.com/google/mediapipe
- MediaPipe | Google for Developers | https://developers.google.com/mediapipe/

[5] PySerial. Disponible en:

- Releases · pyserial/pyserial | https://github.com/pyserial/pyserial/releases
- pySerial — pySerial 3.4 documentation | https://pyserial.readthedocs.io/en/latest/pyserial.html

[6] Math. Disponible en:

- math — Mathematical functions — Python 3.12.0 documentation | https://docs.python.org/3/library/math.html

[7] Numpy. Disponible en:

- NumPy | https://numpy.org/
- NumPy - Wikipedia, la enciclopedia libre | https://es.wikipedia.org/wiki/NumPy

[8] serialDevice. Disponible en:

- https://pypi.org/project/serialDevice/

[9] Hoja de datos del servomotor. Disponible en:

- https://www.alldatasheet.com/datasheet-pdf/pdf/1131873/ETC2/MG996R.html

[10] Referencias de medidas de componentes

- https://grabcad.com/library/buck-converter-lm2596-1
- https://drive.google.com/drive/folders/1K758Dn1bQ9G2QYDxeh-yV2AN0NL6U5dw

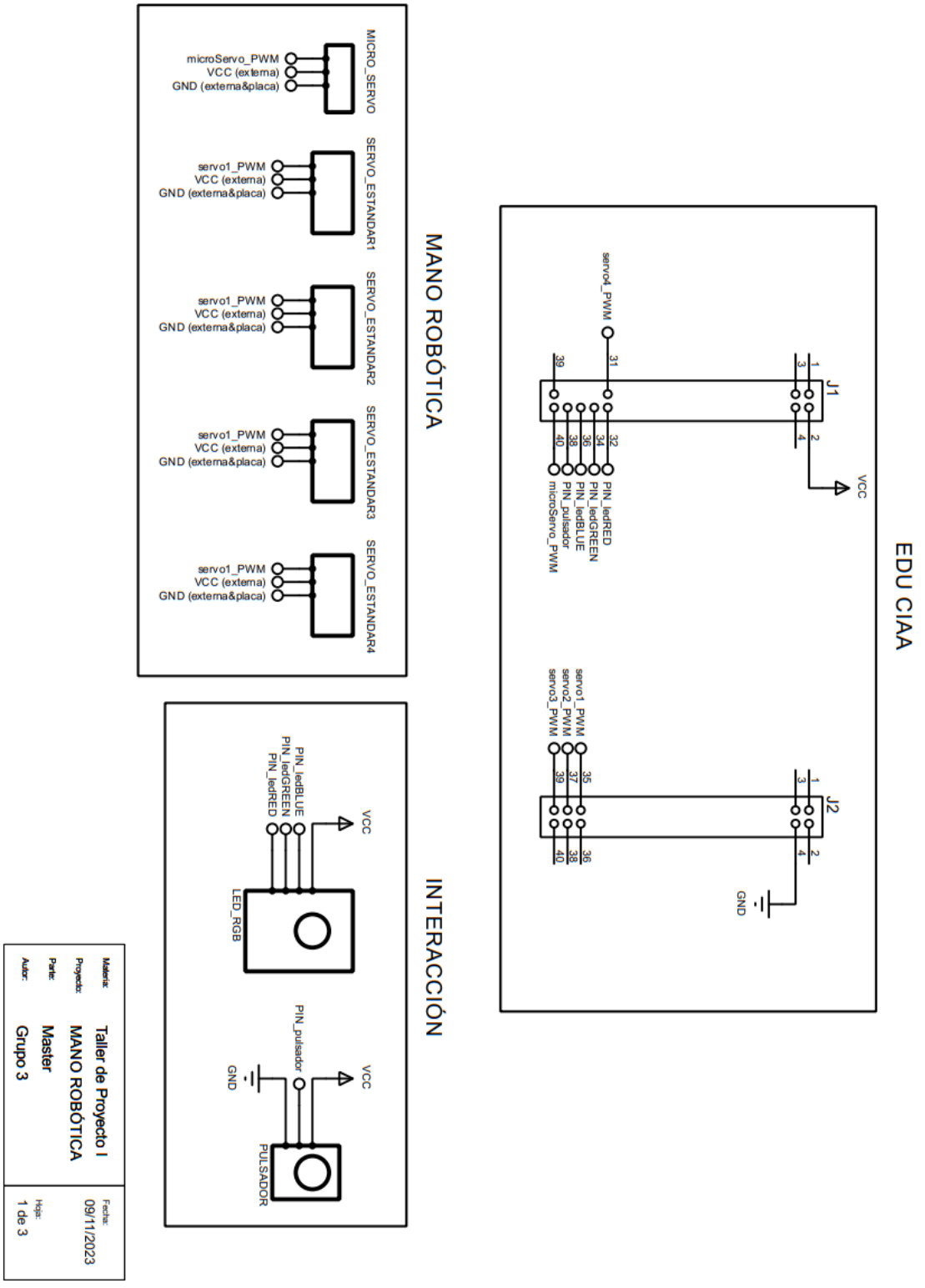
[11] Encapsulados

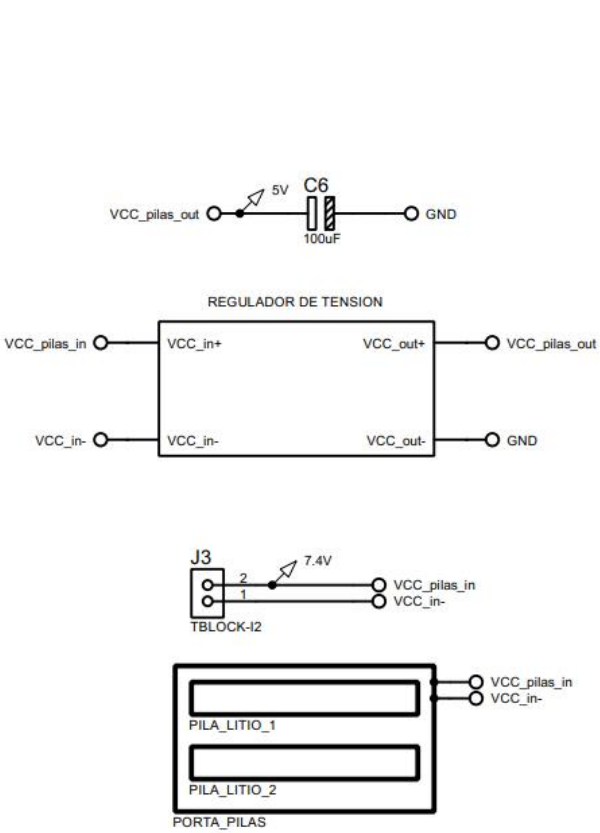
- https://grabcad.com/library/led-rgb-5mm-1/details?folder_id=4477261
- https://grabcad.com/library/dc-dc-lm2596-2
- https://grabcad.com/library/button-6x6x4-3-6x6x5-6x6x6-1
- https://grabcad.com/library/degson-dg306-series-5-0-terminal-block-pack-1/details?folder_id=9670019

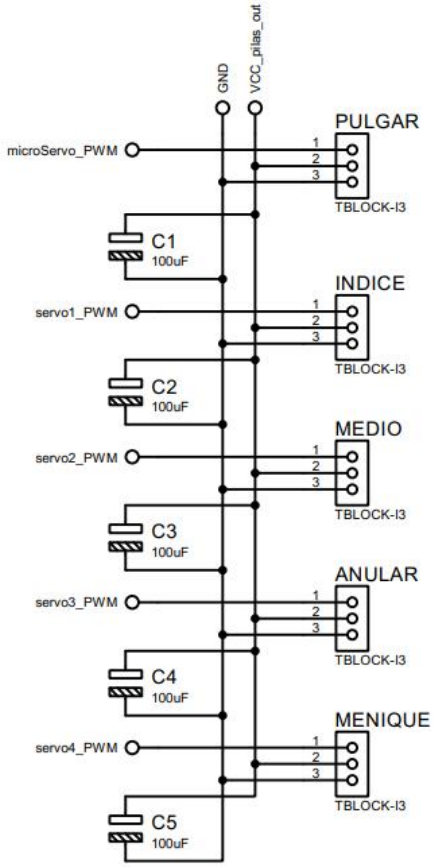
9. Anexos

Además de los anexos a continuación, re realizó un PDF aparte con un manual de usuario junto con este informe.

9.1. Circuito esquemático

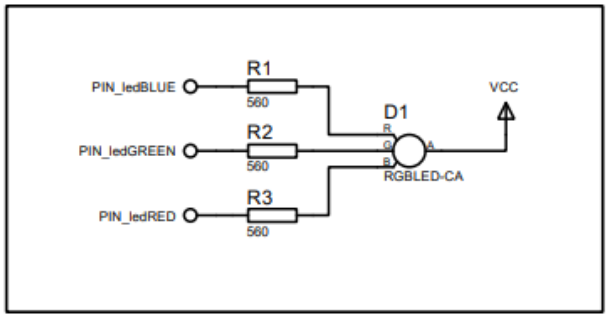




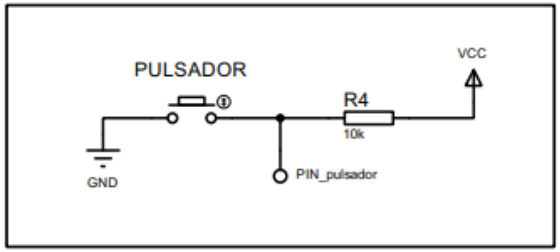


Materia:	Taller de Proyecto I	Fecha:	09/11/2023
Proyecto:	MANO ROBÓTICA		
Parte:	Master	Hoja:	2 de 3
Autor:	Grupo 3		

LEDs RGB



Pulsador (c/ resistencia Pull Up)



Materia:	Taller de Proyecto I	Fecha: 09/11/2023
Proyecto:	INTERACCION	
Parte:	Master	Hoja: 3 de 3
Autor:	Grupo 3	

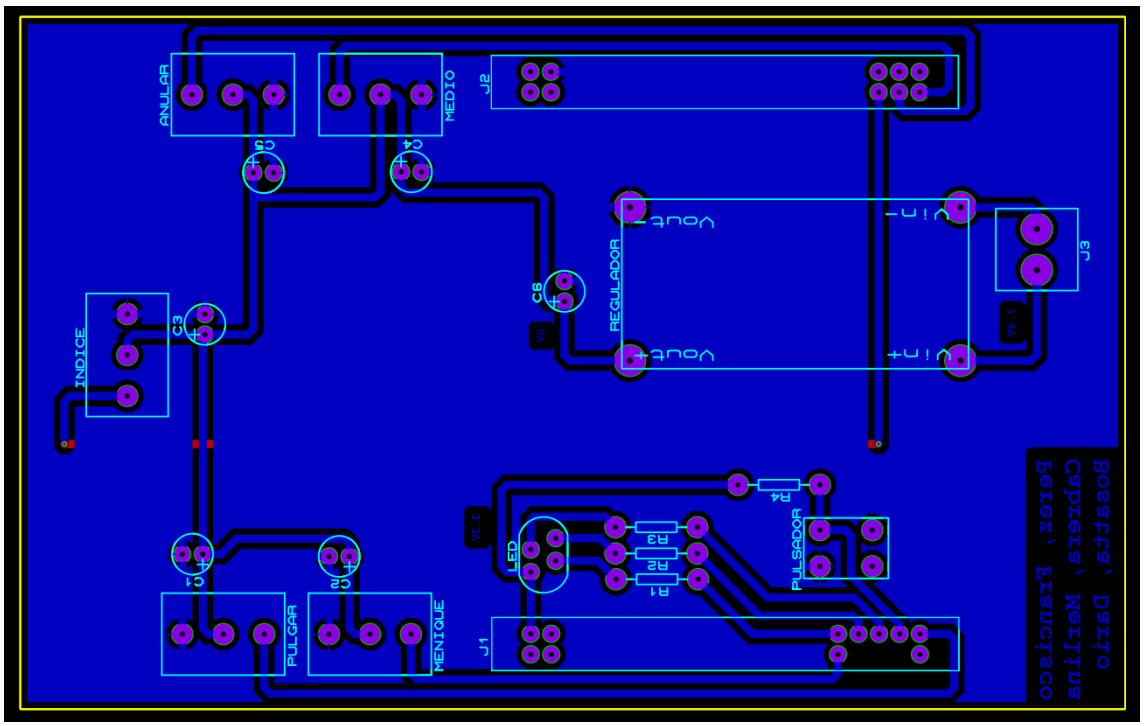
9.2. Lista de materiales

A continuación, se presenta una lista de los componentes adquiridos para llevar a cabo este proyecto.

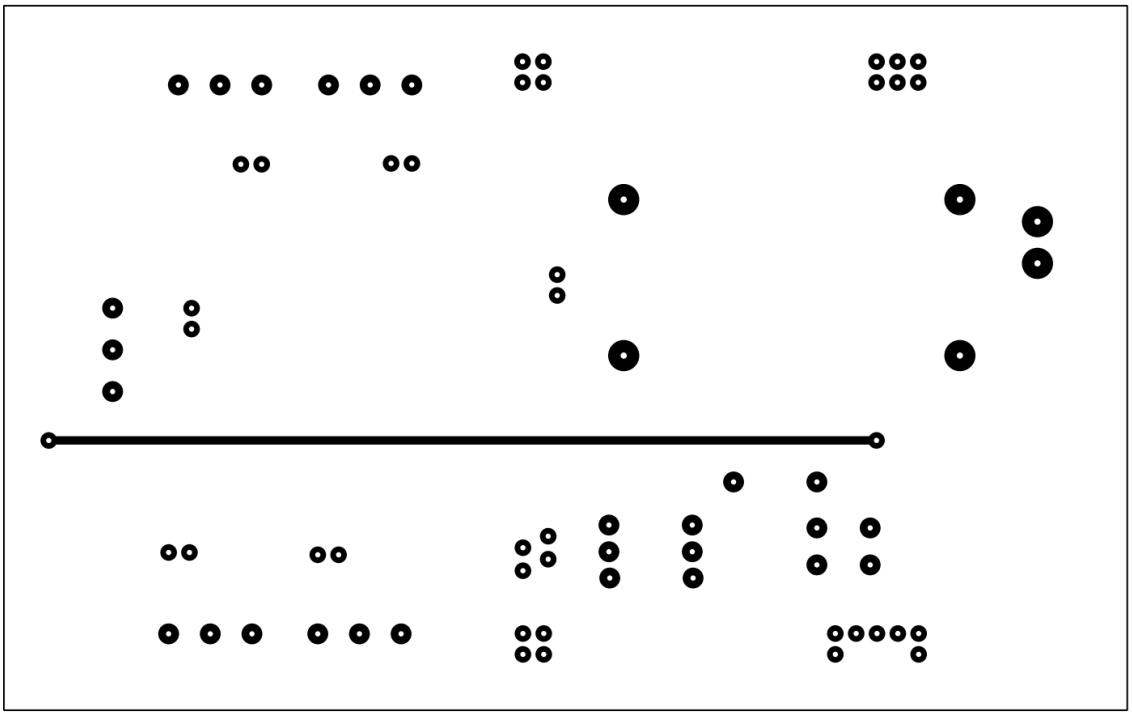
Lista de Materiales					
Elemento	Cantidad	Referencia (Esquemático)	Valor	Código/Nombre	Huella – PCB
Capacitor	6	C1, C2, C3, C4, C5, C6	$100uF$	CD268	ELEC-RAD10
Resistor	4	R1, R2, R3, R4	560Ω	560R ¼ W	RES40
Diodo LED	1	D1	RGB LED	Led RGB 5mm Ánodo Común	LEDRGB (creada manual)
Bornera 3	5	PULGAR, INDICE, MEDIO, ANULAR, MENIQUE	3 pines	Bornera BLOCK 3	TBLOCK-I3
Bornera 2	1	J3	2 pines	Bornera BLOCK 2	TBLOCK-I2
Conector	2	J1, J2	2x20 pines	Conector macho 2x20	CON40_2X20_US_FCI (CONECTOR_1, CONECTOR_2, modificados manual)
Pulsador	1	PULSADOR	6x6x8mm	Switch 6x6x8mm	PULSADOR (creada manual)
Mano Robótica					
Micro Servo	1	MICRO_SERVO	180°	SG90	No requiere
Servo	4	SERVO_ESTANDAR1, SERVO_ESTANDAR2, SERVO_ESTANDAR3, SERVO_ESTANDAR4	180°	S5010	No requiere
Pila Litio	2	PILA_LITIO1, PILA_LITIO2	2 pilas x 18650	Porta Pilas 18650	No requiere
Porta Pilas	1	PORTA_PILAS	3,7V, 7800mAh	Pila Litio 18650	No requiere
Regulador	1	REGULADOR	5V-35V, 4A	LM2596	REGULADOR (creada manual)
Total	29	-	-	-	-

9.3. Placa de circuito impreso (PCB)

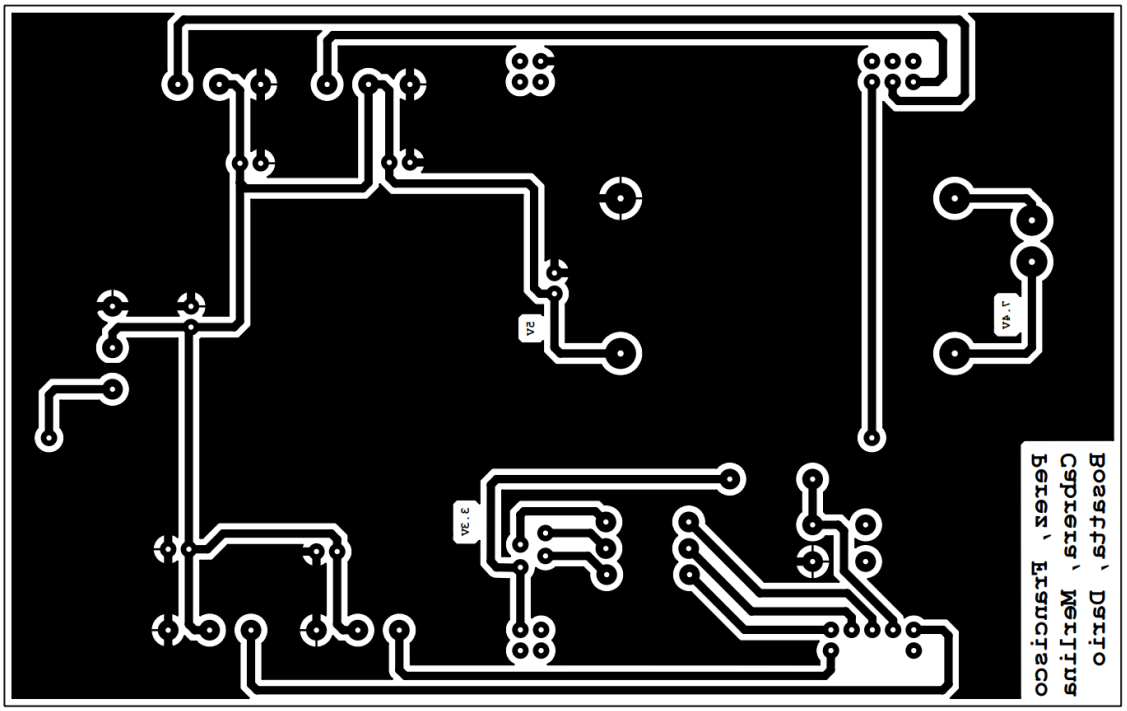
9.3.1. General



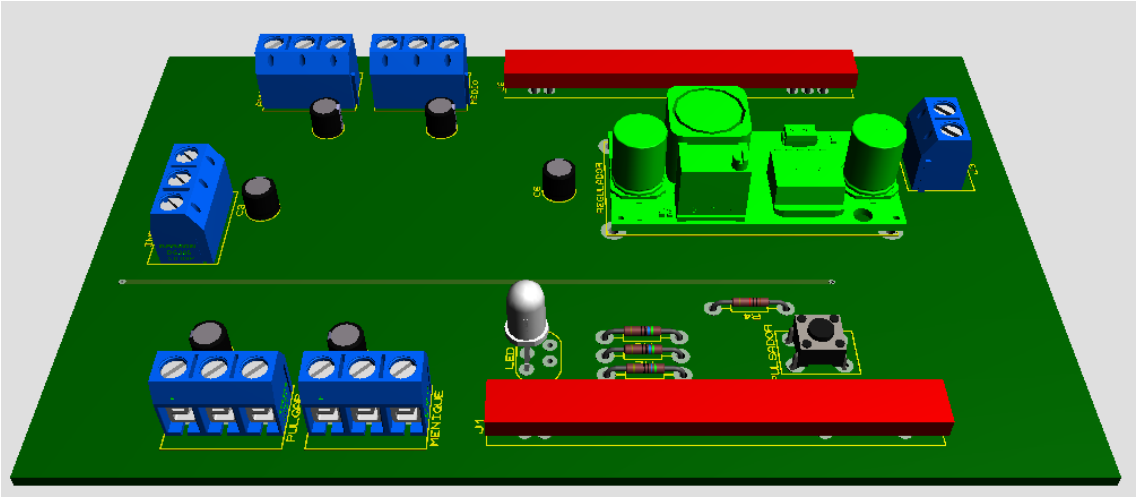
9.3.2. TOP LAYER

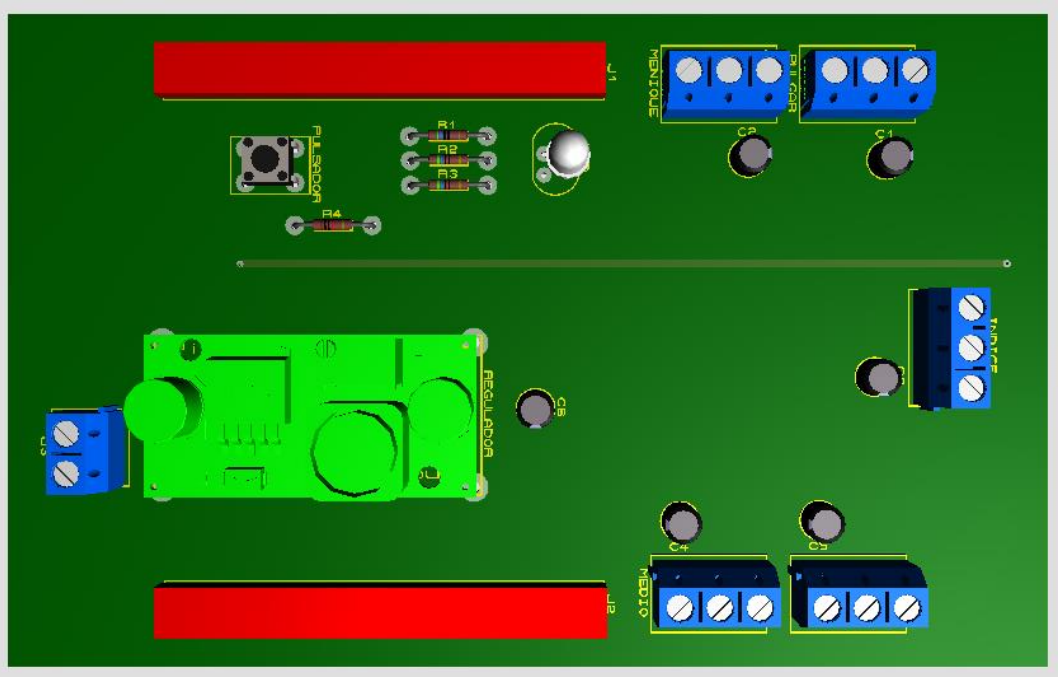


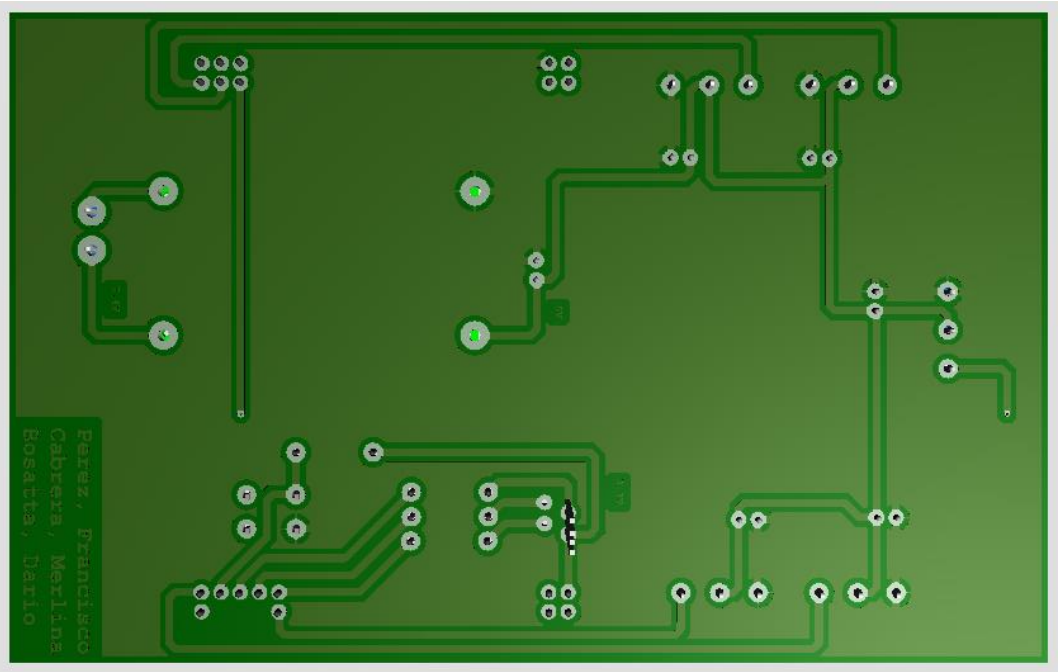
9.3.3. BOTTOM LAYER



9.3.4. 3D VIEWER




J1
J2
J3
J4
R1
R2
R3
R4
C1
C2
C3
C4
C5
REGULADOR
MEDIO


Perez, Francisco
Cabrera, Merlina
Bosatta, Dario

9.4. Códigos

En esta sección se adjuntan los códigos mencionados en ***Sección 5***.

9.4.1. Código EDU CIAA NXP

Se encuentra subdivido en módulos.

9.4.1.1. MEF

```
#include "sapi.h"
#include "mef.h"
#include "usb.h"
#include "resolucion.h"

#define LED_ROJO GPIO1
#define LED_VERDE GPIO3
#define LED_AZUL GPIO7

uint8_t WAIT_FLAG = 0;
uint8_t BUTTON_FLAG = 0;
char initMsg, levelMsg;
char dataFromUart [5];
char aux [5] = "00000";
bool_t estado_boton;
states actState;



// Defno los niveles de resolucion y el angulo que girará el servo
// EL final será 2^level niveles distintos de giro
uint8_t level = 0, angulo = 0;

void girarServo(uint8_t servo, uint8_t angulo){
	switch (servo){
		case 0:
			servoWrite(SERVO8, angulo);
			break;
		case 1:
			servoWrite(SERVO4, angulo);
			break;
		case 2:
			servoWrite(SERVO1, angulo);
			break;
		case 3:
			servoWrite(SERVO2, angulo);
			break;
		case 4:
			servoWrite(SERVO3, angulo);
			break;
```

```
	}
}



void MEF_init(){
	actState=START;
	 gpioConfig(LED_ROJO, GPIO_OUTPUT);
	 gpioConfig(LED_VERDE, GPIO_OUTPUT);
	 gpioConfig(LED_AZUL, GPIO_OUTPUT);
	gpioConfig(GPIO7, GPIO_INPUT);
}

void MEF_update(){		//Cada 1 ms
static uint16_t call_count_led = 0;
static uint16_t call_count_mef = 0;
static uint16_t call_count = 0;
static bool_t LEDB_state = OFF;
static uint8_t i=0, first_time = 1, j=0;


	switch (actState){
		case START:
			USB_init();
			actState = CONNECTING;

		break;
		case CONNECTING:
			if(++call_count_led == 500){ //0.5 s
				gpioToggle(LED_VERDE);
				call_count_led = 0;
			}
			if(++call_count_mef == 30000){	//30 s
				call_count_mef=0;
				actState = FAIL;
			}
			if(uartReadByte(UART_USB, &initMsg)){	//Espero mensaje de
comienzo de comunicacion
				if(initMsg == 'a'){
					// se deberá esperar un aX (donde X = 1, 2, 3, 4,
5, 6, 7, 8)
					if(uartReadByte(UART_USB, &levelMsg)){
						level = definirNiveles(levelMsg);
						gpioWrite(LED_AZUL, OFF);
						call_count_mef = 0;
						actState = SUCCESS1;
						uartWriteByte(UART_USB, 'c');
					}
```

```
                }
        }

break;
case SUCCESS1:
        gpioWrite(LED_VERDE, ON);

        if(++call_count_mef == 1000){
                call_count_mef = 0;
                gpioWrite(LED_VERDE, OFF);
                actState = WAITING;
        }

break;
case FAIL:
        gpioWrite(LED_ROJO, ON);
        if(++call_count_mef == 10000){
                call_count_mef = 0;
                gpioWrite(LED_ROJO, OFF);
                actState = CONNECTING;
        }

break;
case WAITING:
        gpioWrite(LED_AZUL, ON);
        if(uartReadByte(UART_USB, &dataFromUart[i])){
                i++;
        }
        estado_boton= gpioRead(GPIO7);
        if(!(estado_boton)){
                BUTTON_FLAG = 1;
                //actState = CONNECTING;
                //gpioWrite(LED_AZUL, OFF);
        }
        if(i == 5){
                gpioWrite(LED_AZUL, OFF);
                call_count_mef = 0;
                i=0;
                if(BUTTON_FLAG){
                        actState = CONNECTING;
                        BUTTON_FLAG = 0;
                } else {
                        actState = CONTROLLING;
                }
        }

break;
```

```
		case CONTROLLING:
			gpioWrite(LED_VERDE, ON);
			/*if(first_time){
				first_time = 0;
				while (j < 5){
					if(aux[j] != dataFromUart[j]){
						uartWriteByte(UART_USB,
dataFromUart[j]);
						uartWriteByte(UART_USB, aux[j]);
						angulo =
AnguloDeGiro(dataFromUart[j],0,level);
						girarServo(j, angulo);
						aux[j] = dataFromUart[j];
					}
					j++;

				}
				j = 0;

			}*/
			if(first_time){
				first_time = 0;
				for(uint8_t j=0; j<5; j++){
					if (j == 0) {
						angulo =
AnguloDeGiroPulgar(dataFromUart[j],0,level);
					} else {
						angulo =
AnguloDeGiro(dataFromUart[j],0,level);
					}
					girarServo(j, angulo);
				}
			}
			if(++call_count_mef == 200){
				first_time = 1;
				gpioWrite(LED_VERDE, OFF);
				call_count_mef = 0;
				uartWriteByte(UART_USB, 'c');
				// A modo de ejemplo, solamente tomo en cuenta el
primero de los 5 elementos

				actState = WAITING;
			}
		break;
		}
	}
```

9.4.1.2. APP

```
#include "app.h"       // <= Su propia cabecera
#include "sapi.h"      // <= Biblioteca sAPI
#include "timer.h"
#include "mef.h"

char dataArray[5]= {0};
uint8_t flagWait=0;
bool_t tec3Value = OFF;

#define SERVO_N SERVO0
/*
  SERVO0 <---> T_FIL1 de EDU-CIAA-NXP
  SERVO1 <---> T_COL0 de EDU-CIAA-NXP
  SERVO2 <---> T_FIL2 de EDU-CIAA-NXP
  SERVO3 <---> T_FIL3 de EDU-CIAA-NXP
  SERVO4 <---> GPIO8 de EDU-CIAA-NXP
  SERVO5 <---> LCD1 de EDU-CIAA-NXP
  SERVO6 <---> LCD2 de EDU-CIAA-NXP
  SERVO7 <---> LCD3 de EDU-CIAA-NXP
  SERVO8 <---> GPIO2 de EDU-CIAA-NXP
*/

#define LED_ROJO GPIO1
#define LED_VERDE GPIO3
#define LED_AZUL GPIO7

// FUNCION PRINCIPAL, PUNTO DE ENTRADA AL PROGRAMA LUEGO DE ENCENDIDO O RESET.
int main( void )
{
  // Configuraciones

  // Inicializar y configurar la plataforma

        boardConfig();


  servoConfig(4, SERVO_ENABLE);
  servoConfig(8, SERVO_ENABLE);
  servoConfig(1, SERVO_ENABLE);
  servoConfig(2, SERVO_ENABLE);
  servoConfig(3, SERVO_ENABLE);


  servoConfig(SERVO4, SERVO_ENABLE_OUTPUT);
```

```
  servoConfig(SERVO8, SERVO_ENABLE_OUTPUT);
  servoConfig(SERVO1, SERVO_ENABLE_OUTPUT);
  servoConfig(SERVO2, SERVO_ENABLE_OUTPUT);
  servoConfig(SERVO3, SERVO_ENABLE_OUTPUT);

  //Inicializar MEF y la temporizacion del programa
  MEF_init();
  TIMER_Init(1);

  gpioWrite(LED_ROJO,ON);

  uint16_t call_count=0;
  uint8_t i=0;


  //Bucle infinito
  while( TRUE ) {
        TIMER_Dispatch_Task();

  }
  return 0;
}
```

9.4.1.3. RESOLUCION

```
#include "sapi.h"


uint8_t potencia(uint8_t b, uint8_t e){
        uint8_t aux = b;
        for (int i = 1; i<e; i++){
                b = b*aux;
        }
        return b;
}

uint8_t redondeo(double x){
        double entero = (int)x;
        double fraccion = x-entero;

        if (fraccion < 0.5){
                return entero;
        } else {
                return entero + 1;
        }
```

```
}

uint8_t definirNiveles(char n){
	uint8_t n_numero = (int)n - '0';
	return (potencia(2,n_numero)-1);
}

uint8_t AnguloDeGiro(char dedo, uint8_t min, uint8_t max){
	uint8_t dedo_numero = (int)dedo -'0';
	if (dedo_numero > max){
		return 180;
	} else {
		return redondeo(dedo_numero*180/max);
	}
}

uint8_t AnguloDeGiroPulgar(char dedo, uint8_t min, uint8_t max){
	uint8_t dedo_numero = (int)dedo -'0';
	if (dedo_numero > max){
		return 90;
	} else {
		return redondeo(dedo_numero*90/max);
	}
}
```

9.4.1.4. USB

```
#include "usb.h"
#include "sapi.h"

uint8_t FLAG_WAIT;
//char data_array[5] = {0};

void USB_init(){

	FLAG_WAIT=0;
	uartInit(UART_USB, 115200);

}

uint8_t USB_receiveArray(char* arreglo){
	static uint16_t dataCount=0;
	static uint8_t i=0;
	if( uartReadByte(UART_USB, &arreglo[i++]) ){
		dataCount++;
	}
```

```
	if(dataCount==5){
		i=0;
		dataCount=0;
		return 1;
	} else {
		return 0;
	}
}
void USB_sendFlagWait(){
}
```

9.4.1.5. TIMER

```
#include "timer.h"
#include "mef.h"
#include "sapi.h"

volatile uint8_t FLAG_T;

void myTickHook(void *ptr){
	FLAG_T = 1;
}

void TIMER_Init(uint8_t ms){
	tickConfig(ms);
}

uint8_t TIMER_Dispatch_Task(){
	tickCallbackSet( myTickHook, (void*)NULL );
	if(FLAG_T){
		FLAG_T = 0;
		MEF_update();
	}
}
```

9.4.2. Código Cámara

```
from cvzone.HandTrackingModule import HandDetector

import cv2
import math
import numpy
import serial
import io
import interpolacion

# ----------------------- SETUP CAMARA -----------------------
```

```
print("CARGANDO...")
cap = cv2.VideoCapture(0) # Abir la cámara (index 0)

#Si no se abre/detecta correctamente, insiste en abrir la cámara
while not cap.isOpened():
   input("conecte la cámara y presione ENTER para reintentar")
   cap = cv2.VideoCapture(0)

print("Cámara conectada!")
print("CARGANDO...")
print("Presione ENTER para cerrar el programa")
print("Presione N para cambiar el nivel de resolución mientras se ejecuta")

# Definir tamaño de imagen
anchoCamara, altoCamara = 1280, 720
cap.set(3, anchoCamara)
cap.set(4, altoCamara)
# ------------------------- FIN CAMARA -------------------------



# ----------------------- SETUP DETECCION -----------------------
detector = HandDetector(detectionCon=0.7, maxHands=1) # Fidelidad y cantidad de manos a
detectar

falanges = [0, 4, 8, 12, 16, 20, 17] # muñeca, pulgar, indice, medio, anular, meñique,
baseMeñique
x = [0] * 7
y = [0] * 7
dedos = ["pulgar ", "indice ", "medio  ", "anular ", "meñique"]
distancias = [0] * 5
# ------------------------ FIN DETECCION ------------------------

# --------------------- SETUP ENVIO EN SERIE ---------------------
listo = False
conexionSerie = False
recepcion = ''

ser = serial.serial_for_url('loop://', timeout=1)  # crea una instancia de serial.Serial que simula
un puerto serie en memoria sin conexión física



usar = input("Quiere conectar en serie? (si | no): ")
if usar == "si":
   ser.close() # Cerra conexion falsa
```

```
    conexionSerie = True

    comUsuario = input("Ingresar puerto serie:")

    serial_port = "COM"+comUsuario
    try:
        # Abrir serial port
        ser = serial.Serial(serial_port, bytesize = 8, parity = serial.PARITY_NONE, stopbits=serial.
STOPBITS_ONE , baudrate = 115200)

        print(f"Conectado a {serial_port} a 115200 baudios") # Aviso al usuario

    except serial.SerialException as e:
        print(f"Error: {e}")
        input("Presiona Enter para reintentar...")
# ---------------------- FIN ENVIO EN SERIE ----------------------



# ------------------ SETUP MANIPULACION DE DATOS ------------------
# Con esto se setea la resolución (niveles de presición)
nivelMuestreo = input("Setee el nivel de resolución (niveles de presición)\n[1, 2, 3, 4, 5, 6, 7,
8]: ")
datosAEnviar = ['']*5  # Inicializo arreglo con caracteres vacíos
pasos = pow(2,int(nivelMuestreo))

# ------------------- FIN MANIPULACION DE DATOS -------------------


# ----------------------- INICIO DE PROGRAMA -----------------------
print("Iniciando cámara...")
hola = "a"+nivelMuestreo
if conexionSerie:
    ser.write(hola.encode())

while True:
    ret, fotograma = cap.read() # Capturar un fotograma de la cámara

    if not ret:
        break

    manos, fotograma = detector.findHands(fotograma) # Encontrar Mano en el fotograma
    # hands = detector.findHands(img, draw=False)  # sin dibujito

    if manos:
```

```
    mano = manos[0]

    contenedor = mano["bbox"] # Cuadro alrededor de la mano (coordenadas x,y,w,h)
    puntos = mano["lmList"] # puntos de las articulaciones

    # circulos de puntos de referencia
    for i in range(7):
      x[i] = puntos[falanges[i]][0]
      y[i] = puntos[falanges[i]][1]
      cv2.circle(fotograma, (x[i], y[i]), 10, (0, 255, 0), cv2.FILLED)

    # medicion de distancias entre los dedos y el punto de referencia
    for i in range(5):

      if i != 0:
        # Si el dedo no es el pulgar, uso la muneca como referencia
        cv2.line(fotograma, (x[0], y[0]), (x[i + 1], y[i + 1]), (255, 0, 0), 3)
        distancias[i] = math.hypot(x[i + 1] - x[0], y[i + 1] - y[0])
      else:
        # Si el dedo es el pulgar, uso la base del menique como referencia
        cv2.line(fotograma, (x[6], y[6]), (x[i + 1], y[i + 1]), (255, 0, 0), 3)
        distancias[i] = math.hypot(x[i + 1] - x[6], y[i + 1] - y[6])

      datosAEnviar[i] = chr(round(numpy.interp(distancias[i], [175, 425], [0, (pasos - 1)])) + 48)
      # numpy.interp Transforma un rango en otro
      # int, transforma en entero
      # chr, transforma en char (necesario porque sino no se guarda el valor como caracter)
      # + 48, ASCII, '0' = 48, ..., '9' = 57
      print(dedos[i],"    ",datosAEnviar[i])

    print("------------")

    # En caso de usar comunicacion en serie, se espera la recepcion para comenzar a enviar
datos
    if conexionSerie:
      if ser.in_waiting > 0:
        recepcion = ser.read(1).decode()
        if recepcion == 'b':
          print("b recibido, enviando")
          for i in range(5):
            # print(dedos[i], "enviado!")
             ser.write(datosAEnviar[i].encode())  # Envía el dato

  # Mostrar el fotograma procesado
  cv2.imshow("imagen", fotograma)
```

```
    # Si se presiona la tecla ENTER, se cierra la ventana de la cámara
    tecla = cv2.waitKey(1) # Captura la tecla presionada
    if tecla == 13:  # 13 es el código ASCII de la tecla "ENTER"
        ser.close() # Cerrar conexión con puerto serie
        print("FIN DE PROGRAMA")
        break

    if tecla == 78 or tecla == 110:
        print("CAMARA EN PAUSA\n")
        ingreso = input("Ingrese nuevo nivel de muestreo: ")
        hola = "a"+ingreso
        pasos = pow(2,int(ingreso))

    if tecla == 114:
        ser.close()
        ser = serial.Serial(serial_port, bytesize=8, parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE,
                            baudrate=115200)
        ser.write(hola.encode())

cap.release() # Cerrar el acceso a cámara
cv2.destroyAllWindows() # Cerrar la ventanas
# ------------------------ FIN DE PROGRAMA ------------------------
```