

---

# Optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab de HUMANA - Fase III

---

Francisco Estephan Portales Okrassa





UNIVERSIDAD DEL VALLE DE GUATEMALA  
Facultad de Ingeniería



**Optimización de la herramienta de procesamiento de imágenes  
para el sistema Brainlab de HUMANA - Fase III**

Trabajo de graduación presentado por Francisco Estephan Portales  
Okrassa para optar al grado académico de Licenciado en Ingeniería  
Mecatrónica

Guatemala,

2024

Vo.Bo.:

(f)   
\_\_\_\_\_

Dr. Luis Alberto Rivera Estrada

Tribunal Examinador:

(f)   
\_\_\_\_\_

M. Sc. Andrés Monterroso

(f)   
\_\_\_\_\_

M. Sc. Guilmar Escobar

(f)   
\_\_\_\_\_

M. Sc. Carlos Esquit

Fecha de aprobación: Guatemala, 13 de enero de 2024



<b>Lista de figuras</b>	<b>VI</b>
<b>Lista de cuadros</b>	<b>VII</b>
<b>Resumen</b>	<b>IX</b>
<b>Abstract</b>	<b>XI</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Antecedentes</b>	<b>2</b>
2.1. HUMANA . . . . .	2
2.2. Brainlab . . . . .	4
2.3. Investigaciones externas a la Universidad del Valle de Guatemala . . . . .	5
2.4. Investigaciones en Universidad del Valle de Guatemala . . . . .	5
<b>3. Justificación</b>	<b>8</b>
<b>4. Objetivos</b>	<b>9</b>
4.1. Objetivo general . . . . .	9
4.2. Objetivos específicos . . . . .	9
<b>5. Alcance</b>	<b>10</b>
<b>6. Marco teórico</b>	<b>11</b>
6.1. Procesamiento digital de imágenes . . . . .	11
6.1.1. Dominio del tiempo o espacio . . . . .	11
6.1.2. Filtrado en el dominio del espacio . . . . .	11
6.1.3. Dominio de la frecuencia . . . . .	12
6.1.4. Filtrado en el dominio de la frecuencia . . . . .	12
6.2. Visión por computadora . . . . .	12
6.3. Reconocimiento Óptico de Caracteres (OCR) . . . . .	13
6.4. Motores para reconocimiento de caracteres . . . . .	13
6.5. Open CV . . . . .	14

6.6.	Tesseract . . . . .	14
6.7.	Python . . . . .	14
6.8.	Protocolo de Control de Transmisión (TCP) . . . . .	15
6.9.	Universal Asynchronous Receiver-Trasmmitter (UART) . . . . .	15
6.10.	Sistemas embebidos . . . . .	15
6.11.	Raspberry Pi . . . . .	16
6.12.	Tarjetas capturadoras de video . . . . .	17
<b>7.</b>	<b>Visión por computadora</b>	<b>19</b>
7.1.	Módulos implementados . . . . .	19
7.1.1.	Pruebas preliminares Tesseract . . . . .	20
7.1.2.	Pruebas preliminares Asprise . . . . .	22
7.1.3.	Pruebas preliminares OCRspace . . . . .	25
7.1.4.	Resultados de pruebas con motores seleccionados . . . . .	25
7.2.	Validación de captura de pantalla . . . . .	26
7.3.	Preprocesamiento a imágenes . . . . .	28
7.4.	Procesado de la imagen . . . . .	29
7.4.1.	Reconocimiento de ángulos de configuración . . . . .	30
7.4.2.	Creación de modelo <i>Machine Learning</i> para recortes a identificación de juntas . . . . .	32
<b>8.</b>	<b>Protocolo de comunicación TCP/IP</b>	<b>34</b>
8.1.	Pruebas preliminares con protocolo . . . . .	34
8.2.	Codificación y decodificación de datos . . . . .	36
<b>9.</b>	<b>Interfaz gráfica</b>	<b>38</b>
9.1.	Módulo implementado . . . . .	39
9.2.	Herramientas implementadas en interfaz . . . . .	40
9.3.	Resultado captura de pantalla . . . . .	41
9.4.	Resultado envío y recepción de datos . . . . .	42
<b>10.</b>	<b>Sistema embebido</b>	<b>44</b>
10.1.	Implementación de sistema operativo . . . . .	44
10.2.	Resultado procesamiento de imágenes . . . . .	45
<b>11.</b>	<b>Comunicación con brazo robótico</b>	<b>48</b>
11.1.	Pruebas preliminares . . . . .	48
11.2.	Estructura de envío de datos hacia sistema robótico . . . . .	49
11.3.	Envío y recepción de datos . . . . .	49
<b>12.</b>	<b>Conclusiones</b>	<b>52</b>
<b>13.</b>	<b>Recomendaciones</b>	<b>53</b>
<b>14.</b>	<b>Bibliografía</b>	<b>55</b>

---

## Lista de figuras

---

1.	Robot NeuroMate. [2]	2
2.	Sistema Brainlab en HUMANA.	3
3.	Calibración <i>Varioguide</i> a sistema Brainlab en HUMANA.	4
4.	Fotografía por Gopro Hero 8. [7]	6
5.	Resultado de aplicación de filtros. [7]	7
6.	Recorte y binarización a imagen. [7]	7
7.	Ejemplo Kernel aplicado a píxeles de una imagen. 8	12
8.	Ejemplo de aplicación de Visión por computadora. 8	13
9.	Raspberry Pi PICO. [18]	16
10.	Arduino UNO. [19]	16
11.	Raspberry Pi 4. [18]	17
12.	Imagen de prueba para reconocimiento de letras.	20
13.	Resultados reconocimiento Tesseract solo letras.	20
14.	Imagen de prueba para reconocimiento de números.	21
15.	Resultado reconocimiento Tesseract solo números.	21
16.	Imagen de prueba para reconocimiento números y letras.	21
17.	Resultado reconocimiento Tesseract números y letras.	22
18.	Imagen prueba solo letras con Asprise.	22
19.	Resultado prueba solo letras con Asprise.	23
20.	Imagen prueba solo números con Asprise.	23
21.	Resultado prueba solo números con Asprise.	24
22.	Imagen prueba combinación números y letras con Asprise.	24
23.	Resultado prueba combinación números y letras Asprise.	25
24.	Ejemplo de capturas de pantalla realizadas a sistema Brainlab.	26
25.	GUI realizando captura de pantalla a interfaz de Brainlab.	28
26.	Resultado de preprocesamiento a recortes de ángulos	29
27.	Ejemplo del recorte a la identificación de la junta.	30
28.	Ejemplo de sección correspondiente a Joint 1 en interfaz de Brainlab.	30
29.	Ejemplo de sección correspondiente a Joint 2 en interfaz de Brainlab.	31
30.	Ejemplo de sección correspondiente a Joint 3 en interfaz de Brainlab.	31
31.	Ejemplo de recorte a junta con decimales.	32

32.	Ejemplo de recorte a junta con decimales y número negativo. . . . .	32
33.	Ejemplo de recorte a junta con decimales y decenas. . . . .	32
34.	Entrenamiento del modelo de <i>Machine Learning</i> para recorte de juntas. . . . .	33
35.	Resultado 1 uso modelo generado para hacer recorte a identificador. . . . .	33
36.	Resultado 2 uso modelo generado para hacer recorte a identificador. . . . .	33
37.	Resultado de reconocimiento de caracteres de modelo con Tesseract. . . . .	33
38.	Resultado de pruebas preliminares TCP/IP. . . . .	35
39.	Diagrama funcionamiento de protocolo TCP/IP. . . . .	36
40.	Ejemplo de codificación final de las imágenes. . . . .	37
41.	Propuesta de diseño para la interfaz. . . . .	39
42.	Propuesta de diseño para la interfaz todo en uno. . . . .	40
43.	Apartado correspondiente a establecer comunicación con sistema robótico y servidor TCP/IP. . . . .	40
44.	Apartado de la interfaz correspondiente a capturas y procesamiento de las imágenes. . . . .	41
45.	Apartado de la interfaz correspondiente visualizar los datos transcritos e interacción con sistema robótico. . . . .	41
46.	Resultado de implementación de espacio para visualizar captura de pantalla. . . . .	42
47.	Resultado de implementación de espacio para visualizar los datos reconocidos. . . . .	43
48.	Resultado de instalación de sistema operativo en Raspberry Pi 4. . . . .	45
49.	Resultado de ejecución de sistema de reconocimiento de ángulos en Raspberry Pi 4. . . . .	46
50.	Resultado de recortes realizadas por parte de servidor. . . . .	46
51.	Resultados del procesamiento enviados a cliente. . . . .	47
52.	Estructura de envío de datos por medio de UART. . . . .	49
53.	Recepción de datos implementada con Arduino. . . . .	49
54.	Recepción de datos exitosa con Arduino. . . . .	50
55.	Impresión de datos y transmisión exitosa. . . . .	50
56.	Recepción de datos exitosa con Arduino. . . . .	51

---

Lista de cuadros

---

1. RendimientoMotoresReg . . . . . 25



La visión por computadora es una herramienta que permite a las computadoras y a los sistemas informáticos obtener información útil contenida en imágenes digitales, videos y otras entradas digitales. Esta herramienta ha sido utilizada en diversos proyectos enfocados en la obtención de caracteres correspondientes a ángulos de configuración para el posicionamiento de un sistema robótico.

El proyecto que se trabajó se enfoca en la optimización de algoritmos para el reconocimiento óptico de caracteres que ya habían sido desarrollados en fases anteriores. Se centró en el reconocimiento de los ángulos de configuración proporcionados por el sistema de Brainlab, utilizado por HUMANA. Se creó una interfaz diseñada para facilitar al usuario el uso de las herramientas de procesamiento e identificación de los ángulos de configuración.

Además de la interfaz, se desarrolló un sistema de servidor-cliente que conecta la computadora principal que ejecuta el programa de Brainlab con un sistema embebido encargado de recortar las imágenes que contienen los datos de interés. Posteriormente, estos datos se transcriben y se transfieren al cliente. El cliente, junto con la interfaz, tiene la capacidad de realizar capturas de pantalla y recortar la ventana del programa de Brainlab. Además, incluye una sección dedicada a la comunicación con el brazo robótico.

Para la transmisión de datos hacia el brazo robótico, se implementó una estructura utilizando el protocolo de comunicación UART, la comunicación es de dos vías, por lo que el cliente puede recibir una confirmación de que se ha realizado exitosamente la transmisión de datos.

Como herramienta principal para el reconocimiento de caracteres, se empleó el motor Tesseract, el cual es de código abierto y ofrece una amplia gama de aplicaciones, incluyendo la identificación de letras, números, estos, en una gran variedad de idiomas. Como herramienta secundaria, se implementó Asprise OCR, la cuál es una solución comercial y conlleva un costo por su uso. Se llevaron a cabo pruebas preliminares con ambas herramientas para verificar su buen funcionamiento y fidelidad para detectar los datos de configuración a pesar de las distorsiones que se pueden presentar en las imágenes capturadas.

Las pruebas preliminares se realizaron con imágenes que presentan diversos textos, el

primero con texto que contenía exclusivamente números, el segundo con texto únicamente en letras y símbolos de puntuación, y finalmente, se realizaron pruebas con la combinación de ambos tipos de caracteres. Los resultados obtenidos fueron satisfactorios para ambas herramientas, por lo que se decidió incluirlas en el diseño final del proyecto.

Además, se evaluó el rendimiento de una tercera herramienta llamada OCR Space. Sin embargo, los resultados obtenidos no fueron satisfactorios, por lo que esta herramienta se incluyó en el trabajo únicamente con el propósito de proporcionar antecedentes a personas interesadas en explorar su uso en el reconocimiento de caracteres en futuras investigaciones.

Se realizaron 4 interfaces, las primeras dos diseñadas para contener todo el procesamiento e identificación de datos de configuración de manera que se ejecuten en la misma interfaz en la computadora, con la que se hace uso del programa de Brainlab, una de estas utiliza el motor de Tesseract y la otra el motor de Asprise OCR. En las dos versiones restantes, cada una implementa un motor diferente antes mencionado, con la diferencia que estas están diseñadas para trabajar con el servicio de servidor-cliente implementado con el sistema embebido.

Computer vision is a tool that allows computers and computer systems to extract useful information from digital images, videos, and other digital inputs. This tool has been used in various projects focused on obtaining characters corresponding to configuration angles for the positioning of a robotic system.

The project at hand focuses on optimizing algorithms for optical character recognition that had already been developed in previous phases. It concentrated on recognizing the configuration angles provided by the Brainlab system, used by HUMANA. An interface was created to facilitate the user in using the processing and identification tools for the configuration angles.

In addition to the interface, a client-server system was developed that connects the main computer running the Brainlab program with an embedded system responsible for cropping images containing the data of interest. Subsequently, this data is transcribed and transferred to the client. The client, along with the interface, has the ability to take screenshots and crop the Brainlab program window. Additionally, it includes a section dedicated to communication with the robotic arm.

For data transmission to the robotic arm, a structure using the UART communication protocol was implemented. The communication is two-way, so the client can receive confirmation that the data transmission has been successfully completed.

As the main tool for character recognition, the Tesseract engine was employed, which is open-source and offers a wide range of applications, including the identification of letters, numbers, in a variety of languages. As a secondary tool, Asprise OCR was implemented, which is a commercial solution and incurs a cost for its use. Preliminary tests were conducted with both tools to verify their proper functioning and fidelity in detecting configuration data despite distortions that may occur in captured images.

Preliminary tests were conducted with images containing various texts, the first with text exclusively containing numbers, the second with text only in letters and punctuation symbols, and finally, tests were conducted with a combination of both types of characters. The results obtained were satisfactory for both tools, so it was decided to include them in

the final project design.

Additionally, the performance of a third tool called OCR Space was evaluated. However, the results obtained were not satisfactory, so this tool was included in the work solely for the purpose of providing background information to individuals interested in exploring its use in character recognition in future research.

Four interfaces were developed, the first two designed to contain all the processing and identification of configuration data so that they run in the same interface on the computer that uses the Brainlab program. One of these uses the Tesseract engine, and the other uses the Asprise OCR engine. In the remaining two versions, each implements a different aforementioned engine, with the difference that these are designed to work with the client-server service implemented with the embedded system.

# CAPÍTULO 1

---

## Introducción

---

El presente trabajo de graduación se enfoca en la optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab en HUMANA. Se proponen soluciones para el reconocimiento de ángulos de configuración mediante el uso de visión por computadora y motores de reconocimiento óptico de caracteres. Estos ángulos de configuración son fundamentales para determinar el posicionamiento óptimo de un sistema de brazo robótico destinado a asistir en cirugías cerebrales.

Para mejorar los algoritmos y herramientas previamente desarrollados en fases anteriores, se implementa una alternativa al uso de cámaras web para capturar imágenes del sistema de calibración de Brainlab. La alternativa consiste en realizar capturas de pantalla directamente desde el equipo de cómputo. Este enfoque evita problemas como el reflejo de luz en la pantalla, la búsqueda del ángulo adecuado para la cámara web y otros obstáculos que dificultan el proceso de reconocimiento de caracteres, además de generar costos adicionales en la implementación de la herramienta de reconocimiento de caracteres.

El trabajo de graduación comienza proporcionando antecedentes sobre investigaciones externas a la Universidad del Valle de Guatemala, seguidos de antecedentes de las fases previas a esta investigación. Estos antecedentes sirven como una sólida, guía base para la optimización de la herramienta de reconocimiento de ángulos de configuración.

El trabajo se divide varios capítulos, cada uno de los cuales describe una etapa del proceso seguido para alcanzar los objetivos principales y secundarios del proyecto.



El reconocimiento de imágenes ha sido una rama que ha llevado muchos años de estudio y perfeccionamiento. Se ha buscado implementar numerosos algoritmos tanto para el preprocesar de las imágenes como para el reconocimiento de características específicas contenidas en estas para diversas aplicaciones como lo son los sistemas de control.

En la Universidad del Valle de Guatemala se han realizado diversos estudios relacionados con la visión por computadora y *machine learning*. A lo largo de los estudios se ha buscado poder optimizar y perfeccionar todo el proceso que conlleva realizar el procesamiento de imágenes. Un ejemplo de estos estudios es el trabajo realizado por S. Galicia [1] el cual complementa de gran manera el detectar características de calibración para sistemas robóticos en HUMANA [2], el sistema robótico con el que se ha trabajado es el Neuromate, este se puede observar en la Figura 1.



Figura 1: Robot NeuroMate. [2]

### 2.1. HUMANA

El centro de epilepsia HUMANA [2] se especializa en el diagnóstico y tratamiento de la epilepsia, HUMANA se enfoca en utilizar tecnología avanzada para de esta manera poder mejorar la atención de los pacientes. Una tecnología clave en estos procesos es la inteligencia

artificial, esta permite analizar grandes cantidades de datos para la identificación de patrones y predecir resultados. La visión por computadora también es utilizada en este centro para poder analizar las imágenes cerebrales y brindar información detallada sobre la ubicación y la extensión de las lesiones cerebrales.



Figura 2: Sistema Brainlab en HUMANA.



Figura 3: Calibración *VarioGuide* a sistema Brainlab en HUMANA.

## 2.2. Brainlab

Brainlab [3] es una empresa líder en tecnología en el ámbito médico quienes incorporan la tecnología de los sistemas robóticos actuales con inteligencia artificial para poder mejorar los procesos de diagnóstico y tratamiento de los pacientes. Estos ofrecen robots quirúrgicos junto con soluciones de inteligencia artificial para poder proporcionar una mayor precisión y control de las intervenciones quirúrgicas. Realizan procesamiento de imágenes en tiempo real con herramientas como ExacTrac.

## 2.3. Investigaciones externas a la Universidad del Valle de Guatemala

En el trabajo de graduación de Andrea Builes y Esteban Palacio [4] se presenta una simulación de la integración entre un sistema de visión por computadora y un manipulador robótico serial con el objetivo de posicionar cuerpos con formas geométricas simples que se encuentran sobre una banda transportadora en movimiento. Se utilizó el software de MATLAB para extraer el centroide de los objetos mediante el procesamiento de imágenes usando como herramienta principal el *toolbox* de *image processing*. Se presenta una herramienta para calibrar la cámara que se desea utilizar, esta herramienta es *Computer Vision System*. Para este trabajo se utilizó la cámara Logitech HD C615 la cuál cumplió con los requerimientos para la aplicación deseada. A pesar de presentar un proceso bastante completo para el correcto procesamiento de imágenes no se realizaron pruebas con diferentes dispositivos y software para poder brindar una comparación que pueda ser útil para posteriores investigaciones.

En la tesis de Enrique Palafox [5] se presentan principalmente algoritmos y técnicas para el procesamiento de imágenes. Se muestran métodos espaciales en los que se incluyen diferentes tipos de filtros tanto para mejorar la nitidez de las imágenes como para la detección de bordes. Se presentan métodos frecuenciales con los que es posible realizar compresiones y expansiones de las imágenes con el objetivo de tener archivos que sean más livianos y sean mucho más sencillos de procesar, esto mediante herramientas matemáticas. Se buscó poder implementar las diferentes herramientas mencionadas en un DSK 6211, y se presenta el proceso con el cual se puede realizar la primera comunicación con este y cómo transferirle todos los datos que son necesarios para el proceso implementado de procesado de imágenes.

En la tesis de Jonathan González [6] se presentan los resultados del estudio sobre el uso de la librería OpenCV en Python para el reconocimiento de imágenes. El procesamiento de imágenes se realizó incluyendo la carga de imágenes, el filtrado y la segmentación de objetos, como resultado de este procesamiento se implementó una detección de objetos utilizando algoritmos de aprendizaje automático como SVM y redes neuronales. Como resultado de la investigación se determinó que OpenCV y Python cumplen con los requerimientos necesarios para el procesamiento de imágenes y la detección de objetos en estas. Se encontró que la precisión que se tiene en la detección depende mucho del conjunto de datos utilizados y de la complejidad de los objetos y características que se intentan detectar.

## 2.4. Investigaciones en Universidad del Valle de Guatemala

La Universidad del Valle de Guatemala, a lo largo de los años, ha mantenido la relación de investigación con el Centro de Epilepsia y Neurocirugía Funcional HUMANA [2] en la cual se han desarrollado numerosos proyectos que aportan de gran manera a la optimización de todos los procesos que se llevan a cabo en este centro, desde el control mecánico de los instrumentos hasta procesamiento de resultados de los diferentes estudios que ahí se realizan.

En el trabajo de graduación de Juan González [7] se presentan los resultados de un sistema de detección de ángulos de configuración del sistema VarioGuide de Brainlab [3].

Los datos detectados tienen como propósito final que un sistema robótico pueda replicarlos y así maximizar la precisión al realizar procesos quirúrgicos. Esta es una fase del proyecto del brazo robótico de HUMANA [2], los proyectos involucrados se dividen en software y hardware. La implementación realizada por González cuenta con 7 fases de procesamiento de las imágenes, la implementación tuvo como resultado reconocer satisfactoriamente los ángulos de configuración, esto dentro de los márgenes de error establecidos por Brainlab. Parte del proyecto consistió en validar diferentes sensores ópticos, el primero fue el que posee la cámara web Logitech 1080p, el segundo fue con la cámara GoPro Hero 8 con la cual se obtuvieron los mejores resultados debido a su mejor resolución. En la Figura 4 se observa la fotografía tomada por la GoPro Hero 8, en la Figura 5 se observa el resultado de la aplicación de los filtros para finalmente recortar y someter a binarización el ángulo de configuración deseado que se observa en la Figura 6.

En el trabajo de graduación de Santiago Galicia [1] se presenta una investigación bastante amplia en el procesamiento de imágenes, mediante distintos algoritmos, estos se encargan en preparar las imágenes para su posterior análisis y reconocimiento de caracteres. Aplicando diferentes filtros a las imágenes para mejorar la implementación y resultados, se llegó a la conclusión sobre software más óptimo para el procesamiento de imágenes. Se investigó sobre el mejor hardware para la implementación tanto como dispositivos que capturen las imágenes como las bases utilizadas para colocar los dispositivos para realizar las capturas. Los resultados de la investigación indican que, en cuanto al dispositivo de captura, lo más importante entre muchos otros es el ángulo de visión ya que entre más amplio, el error de procesamiento será mayor. Se recomienda realizar pruebas con mejores cámaras y buscar alternativas de procesamiento para poder implementar en HUMANA [2].

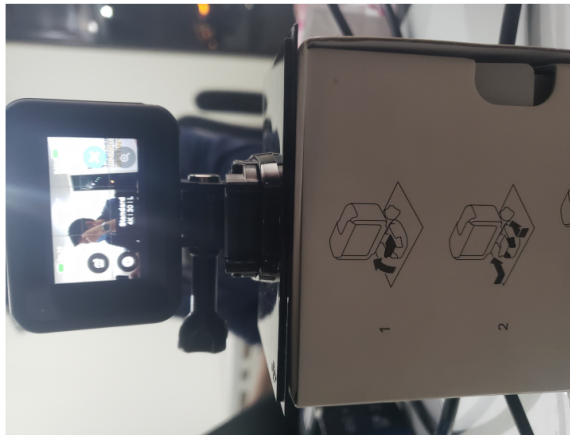


Figura 4: Fotografía por GoPro Hero 8. [7]



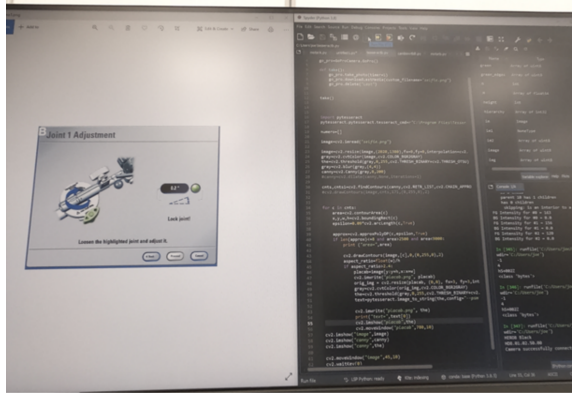


Figura 5: Resultado de aplicación de filtros. [7]



Figura 6: Recorte y binarización a imagen. [7]

El objetivo principal de este trabajo de graduación es mejorar y optimizar los algoritmos previamente empleados en proyectos de graduación anteriores, específicamente los desarrollados por Gonzalez y Galicia. Además, se pretende validar la posibilidad de trasladar estas herramientas a un sistema embebido con el fin de mejorar la eficiencia y con esto no depender del equipo de cómputo disponible en el entorno de implementación.

En proyectos previos, se ha enfrentado el desafío de no contar con equipos eficientes, tanto en HUMANA como en el área de implementación, lo que puede afectar el rendimiento y la correcta ejecución de los algoritmos. La utilización de una cámara web para las capturas de imágenes generó problemas con los que había que lidiar como reflejos de luz en la pantalla, iluminación deficiente en el área de trabajo y dificultades con la posición de la cámara. Para abordar estos problemas, se propone la adopción de un sistema de captura de pantalla, con el objetivo de obtener imágenes más claras y reducir de manera significativa el margen de error al aplicar los algoritmos de procesamiento y obtener los datos de los ángulos deseados.

Mediante este trabajo, se logró la optimización tanto de los algoritmos de reconocimiento de caracteres como de la implementación física de la herramienta diseñada para la captura de pantalla, procesamiento y reconocimiento de la información de los ángulos que se desean extraer del sistema de Brainlab.

### 4.1. Objetivo general

Optimizar la herramienta de procesamiento de imágenes y reconocimiento de caracteres para el sistema Brainlab de HUMANA, para poder generar correctamente los comandos de control para un brazo robótico de asistencia en cirugías.

### 4.2. Objetivos específicos

- Validar la herramienta de procesamiento de imágenes y reconocimiento de caracteres desarrollada en la fase anterior con más imágenes, obtenidas con cámaras, del sistema Brainlab de HUMANA.
- Evaluar el rendimiento de la herramienta con imágenes obtenidas como capturas de pantalla.
- Ajustar los algoritmos para mejorar el reconocimiento de caracteres logrado en la fase anterior.
- Evaluar la posibilidad de migrar la herramienta a un sistema embebido y su integración al sistema de Brainlab.
- Implementar un protocolo para el envío de comandos al sistema de control del brazo robótico que se está desarrollando para HUMANA.

El sistema optimizado para el procesamiento y reconocimiento de datos de configuración, para la calibración del sistema Brainlab de HUMANA, se limitó a validar la efectividad de dos motores para el reconocimiento de ángulos de configuración, estos fueron Tesseract y Asprise OCR. Se trabajó con estas dos herramientas ya que demostraron una alta efectividad y asertividad al momento de detectar los ángulos de configuración. No obstante, Tesseract fue la herramienta principal en la mayoría de los procesos, ya que esta es de código abierto por lo que la accesibilidad para trabajar en esta es mayor que Asprise OCR. Sin embargo, Asprise OCR se implementó como una herramienta auxiliar, pero debido a que es de uso comercial, es necesario efectuar un pago por el uso extendido.

Se logró adecuar protocolos de comunicación como lo son TCP/IP y UART. TCP/IP se utilizó para la transferencia de datos a distancia entre el servidor y el cliente dedicados al procesamiento de imágenes y al reconocimiento de caracteres. UART se implementó para la comunicación y el envío de información sobre los ángulos de configuración al sistema del brazo robótico. A pesar de haber implementado un protocolo para trasladar información al sistema robótico, este trabajo de graduación se limitó a desarrollar una estructuración base con el que se pueda dar seguimiento al (*software*) que se encarga de controlar el brazo robótico en desarrollo.

Se desarrolló una interfaz con la cual es posible que el usuario final pueda interactuar de una forma sencilla y eficiente, esta interfaz tiene 4 versiones de las cuales 2 pertenecen al sistema que integra todas las herramientas dentro de la interfaz y 2 que pertenecen al sistema servidor-cliente que se integra en un sistema embebido. Las últimas dos versiones, incorporan una sección con la cual es posible introducir una dirección IP para conectarse y comunicarse con el servidor. En todas las versiones se integró una sección para conectarse y comunicarse con el sistema robótico, en este se muestran los ángulos que se reconocen.

Se limitó a validar las herramientas disponibles para pre-procesamiento de imágenes, así como validar el trabajar con capturas de pantallas obtenidas por el sistema de Brainlab.

## 6.1. Procesamiento digital de imágenes

El procesamiento digital de imágenes se utiliza en muchas aplicaciones, estas pueden ser con propósitos industriales, ciencias médicas, biométricas, agricultura, ganadería y satélites de observación terrestre.

Una imagen puede definirse como el conjunto de puntos de colores los cuales generan una sucesión coherente de puntos que conforman una matriz de información para el posterior uso digital. Todos estos puntos individualmente son nombrados como "Píxel".

Un filtro sobre una imagen digital es la operación que se le aplica a los píxeles para poder optimizarla, esto se utiliza tanto para suavizar aspectos no deseados en la imagen como para resaltarlos en caso de ser requerido. El proceso de filtrado puede llevarse a cabo tanto sobre dominios de frecuencia y/o espacio, por esto mismo la herramienta fundamental en estos procesos es la transformada de Fourier. [8]

### 6.1.1. Dominio del tiempo o espacio

El dominio del tiempo es un término que se utiliza para describir el análisis de funciones matemáticas o señales respecto al tiempo. [8]

### 6.1.2. Filtrado en el dominio del espacio

Este tipo de filtros trabaja directamente sobre los píxeles. Existen filtros que se clasifican como lineales, los cuales son basados en kernel o máscaras de convolución, y otros que son clasificados como no lineales.

El kernel o máscara de convolución es prácticamente una matriz de coeficientes la cual,



al ser aplicada a un píxel objetivo, se obtiene una transformación en el píxel objetivo y en sus vecinos. [8]

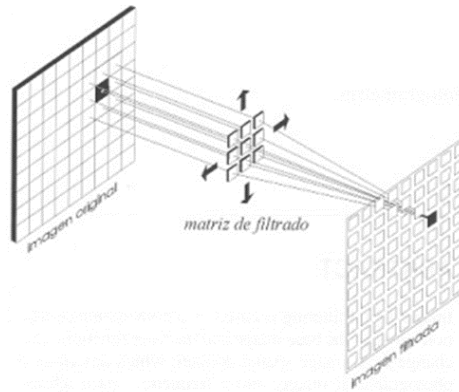


Figura 7: Ejemplo Kernel aplicado a píxeles de una imagen. 8

### 6.1.3. Dominio de la frecuencia

El dominio de la frecuencia es un término usado para describir el análisis de funciones matemáticas, señales o movimientos periódicos respecto a su frecuencia. [8]

### 6.1.4. Filtrado en el dominio de la frecuencia

Este tipo de filtros trabajan modificando la imagen siguiendo el teorema de convolución que corresponde. Se empieza aplicando la transformada de Fourier, posteriormente se multiplica por la función del filtro que se desea aplicar. Finalmente, para recuperar la imagen con el filtro aplicado se realiza una transformada inversa de Fourier para regresar al dominio del tiempo o espacial. [8]

## 6.2. Visión por computadora

La visión artificial mediante computadoras permite que los dispositivos computacionales puedan percibir y distinguir información relevante presente en imágenes digitales, videos y otras entradas visuales, lo cual les permite tomar acciones con base en esa información. Un ejemplo de esto se puede observar en la Figura 8

Para lograr que la visión artificial funcione, es necesario inicialmente contar con una gran cantidad de imágenes para entrenar y capacitar el modelo para reconocimiento de elementos importantes presentes en dichas imágenes.

Dos tecnologías comúnmente utilizadas son el aprendizaje profundo (*deep learning*) y las redes neuronales convolucionales. El aprendizaje profundo utiliza modelos algorítmicos que permiten que una computadora aprenda sobre el contexto de los datos visuales. Si se

dispone de suficientes datos, la computadora puede observar y aprender a diferenciar entre distintas imágenes o características y objetos.

Por otro lado, las redes neuronales convolucionales realizan convoluciones y evalúan la precisión de sus predicciones en múltiples iteraciones hasta que estas se acerquen a la realidad. Luego de este proceso, la computadora adquiere la capacidad de reconocer e interpretar imágenes de manera similar a los seres humanos. [9]

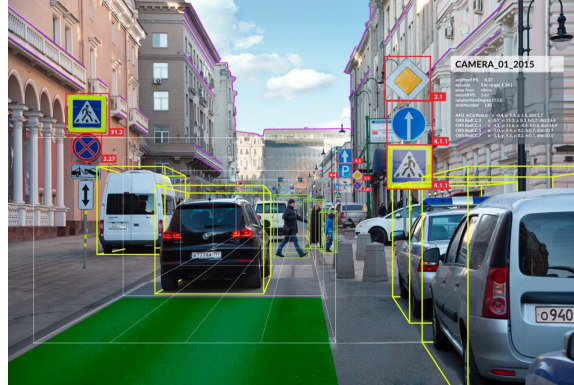


Figura 8: Ejemplo de aplicación de Visión por computadora. 8

### 6.3. Reconocimiento Óptico de Caracteres (OCR)

El Reconocimiento Óptico de Caracteres (OCR) es un software que identifica el texto presente en una imagen y lo convierte en cadenas de caracteres para ser guardado en un formato utilizable en programas de edición de texto. Son sistemas capaces de leer datos escritos y reconocerlos a altas velocidades, de manera individual por carácter. Sin embargo, la mayoría de los sistemas OCR aún no pueden reconocer caracteres o palabras escritas a mano ni documentos que presenten deterioro físico. En la actualidad, los sistemas OCR realizan varios procesos para procesar una imagen, como organizarla para obtener los caracteres, extraer los caracteres y clasificar los caracteres extraídos.

Cuando se tiene una imagen, ya sea una fotografía o un documento escaneado, el texto que contiene es parte integral de la imagen, al igual que cualquier otro elemento visual, como dibujos o esquemas. Si se desea extraer ese texto para editarlo, se necesita un programa de OCR que pueda reconocer y convertir el texto en una cadena de caracteres, ya sea en formato ASCII o Unicode. Posteriormente, se puede copiar esta cadena en un programa de edición de texto para trabajar con ella, lo que nos ahorra tiempo al no tener que teclear manualmente dicho texto. [10]

### 6.4. Motores para reconocimiento de caracteres

Algunos de los motores más populares utilizados en el mercado son Tesseract, el cual es un motor de código abierto que fue desarrollado por google. Soporta más de 100 idiomas,

y es conocido por su alta precisión y capacidad de poder trabajar con diferentes tipos de fuentes y diseños de texto. [11]

ABBY FineReader también es conocida por su alta precisión y capacidad de reconocer textos de varios idiomas, sin embargo, este también ofrece algunas funciones avanzadas como detección de tablas, preservación del formato y conversión de formatos que se pueden editar. [12]

## 6.5. Open CV

OpenCV es una librería de código abierta enfocada a la visión por computadora y *machine learning*. Esta librería contiene más de 2500 algoritmos optimizados, estos se pueden utilizar para detectar rostros, objetos, clasificar acciones humanas en videos, etc. Se tiene soporte para distintas plataformas y lenguajes de programación como C++, Python y MATLAB, aunque nativamente está escrita en C++. [13]

## 6.6. Tesseract

Tesseract es un motor código abierto de reconocimiento óptico de caracteres, este se encarga de extraer el texto impreso de documentos como archivos PDF o imágenes para convertirlo en texto editable. Originalmente fue desarrollado por Hewlett-Packard y posteriormente fue adquirido por Google. Uno de los aspectos que destacan a esta herramienta es que es compatible con muchos lenguajes de programación y muchas estructuras que hacen uso de *wrappers* como Pytesseract.

Especializado en visión por computadora, destaca por su capacidad para detectar caracteres en diversas condiciones gracias a mejoras constantes en precisión mediante aprendizaje automático y redes neuronales. Ofrece soporte multilingüe, permitiendo el reconocimiento en varios idiomas, y puede modificarse para entrenar en fuentes o idiomas específicos. Tesseract es ampliamente adoptado como solución confiable para detección de caracteres y procesamiento de imágenes. [11]

## 6.7. Python

Python es un lenguaje de programación versátil, multiplataforma y multiparadigma el cual se ha destacado por su código legible y limpio. Este es un lenguaje de programación con licencia de código abierto lo cual permite que se pueda utilizar en distintos contextos de forma gratuita.

El objetivo principal de Python es la automatización de procesos lo que hace de las tareas algo mucho más simple. Uno de los usos de Python es la inteligencia artificial, ya que al ser un lenguaje simple, escalable y robusto permite implementar las ideas complejas en unas cuantas líneas de código. [14]

## 6.8. Protocolo de Control de Transmisión (TCP)

El protocolo TCP/IP es un conjunto de reglas que permiten la comunicación efectiva entre dispositivos en redes de computadoras. Su eficiencia radica en la segmentación y reensamblaje de datos en paquetes, garantizando la entrega ordenada y confiable de información. Funciona mediante una estructura de capas, donde la capa de transporte asegura la integridad y el control de flujo de datos, mientras que la capa de internet se encarga del enrutanamiento y direccionamiento de paquetes. TCP/IP es ampliamente utilizado debido a su interoperabilidad y adaptabilidad a distintas tecnologías de red.

En el contexto de aplicaciones en Python, el uso de sockets es altamente beneficioso para la transmisión de datos. Los sockets son interfaces de programación que permiten la comunicación entre procesos en diferentes dispositivos a través de una red. En Python, los sockets ofrecen una manera conveniente y eficaz para enviar y recibir datos entre aplicaciones, ya que aprovechan la robustez de TCP/IP para asegurar la entrega confiable de información. La versatilidad de los sockets Python permite la implementación de diversas soluciones de comunicación, como clientes y servidores, chat en tiempo real o transferencia de archivos, brindando control sobre la conexión y transmisión de datos. [15]

## 6.9. Universal Asynchronous Receiver-Transmitter (UART)

El protocolo UART es un estándar de comunicación serie utilizado para transmitir datos de manera eficaz y eficiente entre dispositivos electrónicos. Su eficacia radica en su simplicidad y bajo consumo de recursos, lo que hace adecuado para conexiones de corta distancia. Funciona mediante la transmisión secuencial de bits, donde los datos se envían sin un reloj compartido, lo que simplifica la implementación. Este protocolo es comúnmente utilizado para la comunicación entre microcontroladores, sensores, actuadores y otros dispositivos electrónicos, especialmente cuando se necesita una comunicación directa y sencilla.

En el contexto de aplicaciones de programación, el uso de protocolo UART puede ser beneficioso para la comunicación entre dispositivos, especialmente cuando se requiere una comunicación serie sin complicaciones y con baja latencia. En lenguajes de programación como Python, se pueden utilizar bibliotecas y módulos para gestionar la comunicación UART, lo que facilita la implementación de conexiones serie entre dispositivos. Esto es particularmente útil en proyectos de electrónica y robótica, donde la comunicación directa y en tiempo real entre componentes es esencial. [16]

## 6.10. Sistemas embebidos

Los sistemas embebidos son sistemas de computación diseñados para cumplir funciones específicas, y suelen tener sus componentes integrados en una placa base. En el campo del *machine learning* y la visión por computadora, es relevante destacar que estos sistemas embebidos pueden ser utilizados para llevar a cabo tareas específicas en estos campos. Por ejemplo, se pueden emplear microcontroladores con capacidades limitadas pero dedicados a

tareas de reconocimiento de imágenes o análisis de datos visuales. En las figuras 9 y 10 se puede observar ejemplos de sistemas embebidos. [17]

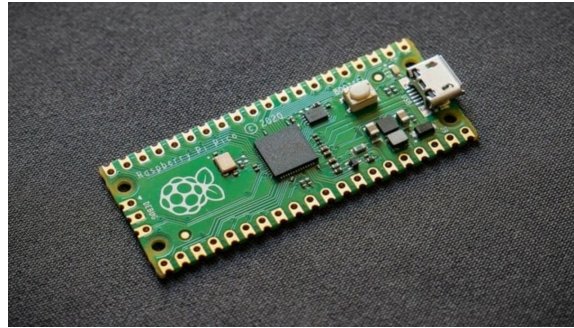


Figura 9: Raspberry Pi PICO. [18]



Figura 10: Arduino UNO. [19]

## 6.11. Raspberry Pi

La Raspberry Pi, un económico y compacto ordenador del tamaño de una tarjeta de crédito, se destaca por su capacidad para ser conectado a un monitor, teclado y ratón. A pesar de su tamaño reducido, esta potente plataforma es capaz de llevar a cabo una amplia gama de tareas similares a las de una computadora convencional. Sin embargo, lo que la distingue aún más es su capacidad de interactuar con su entorno físico a través de periféricos adicionales.

En el contexto de *machine learning* y visión por computadora, la Raspberry Pi se ha convertido en una opción popular debido a su versatilidad. Utilizando el lenguaje de programación Python y aprovechando el sistema operativo Linux, este dispositivo se convierte

en un recurso invaluable para implementar algoritmos de aprendizaje automático y llevar a cabo análisis visual. Sus capacidades de procesamiento de datos y su capacidad para interactuar con sensores y cámaras le permiten realizar tareas complejas como el reconocimiento de imágenes, la detección de objetos y otras aplicaciones relacionadas con la visión por computadora. [18]



Figura 11: Raspberry Pi 4. [18]

## 6.12. Tarjetas capturadoras de video

Las tarjetas capturadoras de video desempeñan un papel fundamental en la visión por computadora al permitir la adquisición de señales de video provenientes de fuentes externas. Estas tarjetas están diseñadas para capturar y procesar la información visual transmitida a través de conexiones como HDMI.

Una de las aplicaciones más comunes de las tarjetas capturadoras de video es la captura de juegos en consolas. Estas tarjetas reciben de manera continua los datos generados por la GPU de la computadora o consola de videojuegos. La información visual, que incluye imágenes en tiempo real y secuencias de video, es capturada en tiempo real por la tarjeta capturadora.

Una vez que los datos de video son capturados, se pueden realizar varias acciones según la configuración establecida por el usuario. La captura de video puede ser almacenada en un archivo para su posterior reproducción, grabación o análisis. Además, la tarjeta capturadora puede aplicar codificación a los datos capturados, lo que permite comprimir el contenido de video para su transmisión o almacenamiento eficiente.

Es importante destacar que las tarjetas capturadoras de video son altamente versátiles y pueden ser utilizadas en diversas aplicaciones de visión por computadora más allá de la captura de juegos. Estas tarjetas son utilizadas en la adquisición de video para el procesamiento

y análisis de imágenes, detección y seguimiento de objetos, reconocimiento de patrones, entre otras aplicaciones relacionadas con la visión por computadora.[20]

En este proyecto, la herramienta principal utilizada fue la visión por computadora, aprovechando diversas funcionalidades de la librería OpenCV. Esta librería ofrece un amplio conjunto de herramientas que abarcan desde la capacidad de leer y almacenar imágenes hasta la aplicación de diversos filtros para su procesamiento.

Para tareas más complejas, que requieren el reconocimiento de características fundamentales, como la detección de números, letras, caracteres y signos, se trabajó con Tesseract como la herramienta principal y Asprise OCR como secundaria. Además, se incorporaron herramientas alternas que complementan de gran manera la funcionalidad principal del proyecto, como la obtención de imágenes a partir de capturas de pantalla para, de esta manera optimizar completamente proceso de reconocimiento de ángulos de configuración.

### 7.1. Módulos implementados

Se realizaron pruebas con varios motores de reconocimiento óptico de caracteres con el objetivo de validar cada uno de ellos y, posteriormente, seleccionar el principal a utilizar. Estas pruebas involucraron tres tipos de imágenes: una con solo números, solo letras y, finalmente, una imagen que contenía la combinación de ambos.

Las herramientas que se utilizaron en las pruebas fueron las siguientes:

- Tesseract con Pytesseract
- Asprise
- OCRspace



### 7.1.1. Pruebas preliminares Tesseract

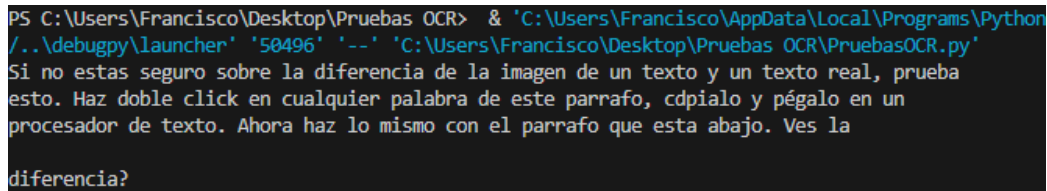
Tesseract es un motor desarrollado específicamente para el reconocimiento óptico de caracteres, esta es una herramienta valiosa debido a su capacidad para ajustar conjuntos de configuración que definen lo que se espera que el motor identifique. Sin embargo, para obtener un rendimiento óptimo de Tesseract, es esencial aplicar un procesamiento previo al reconocimiento a las imágenes que se desean procesar.

En una fase inicial de pruebas, Tesseract se utilizó para realizar el reconocimiento de caracteres en imágenes que contenían exclusivamente texto compuesto por letras, imágenes que contenían únicamente números y, finalmente, imágenes que contienen una combinación de ambos. En estas pruebas iniciales, no se aplicó ningún procesamiento previo a las imágenes con el objetivo de evaluar el rendimiento de la herramienta en condiciones sin alguna modificación a las imágenes.

Para las pruebas realizadas con imágenes que contienen únicamente texto, se empleó la imagen que se muestra en la Figura 12. Esta imagen también incluye caracteres especiales como tildes y signos de puntuación. El resultado del reconocimiento de esta cadena de caracteres se observa en la Figura 13. Como se puede observar, se logró reconocer la gran mayoría de las letras, incluyendo aquellas que contienen caracteres especiales.

Si no estás seguro sobre la diferencia de la imagen de un texto y un texto real, prueba esto. Haz doble click en cualquier palabra de este párrafo, cópialo y pégalo en un procesador de texto. Ahora haz lo mismo con el párrafo que está abajo. ¿Ves la diferencia?

Figura 12: Imagen de prueba para reconocimiento de letras.



```
PS C:\Users\Francisco\Desktop\Pruebas OCR> & 'C:\Users\Francisco\AppData\Local\Programs\Python\Python38-32\python.exe' -u -i -l spa --oem 1 --psm 11 -- -- 'C:\Users\Francisco\Desktop\Pruebas OCR\PruebasOCR.py'
Si no estas seguro sobre la diferencia de la imagen de un texto y un texto real, prueba
esto. Haz doble click en cualquier palabra de este parrafo, cdpialo y pégalo en un
procesador de texto. Ahora haz lo mismo con el parrafo que esta abajo. Ves la
diferencia?
```

Figura 13: Resultados reconocimiento Tesseract solo letras.

En las pruebas preliminares realizadas utilizando imágenes que contenían únicamente números, a pesar que algunos números presentaron cambios en el color de fondo, se obtuvieron buenos resultados. La imagen utilizada para estas pruebas se observa en la Figura 14. En la Figura 15, se observa el resultado del reconocimiento de caracteres, en donde la impresión de caracteres en la consola muestra una gran cantidad de líneas que fueron procesadas y transcritas exitosamente. Esto demuestra la capacidad de la herramienta para reconocer el texto a pesar de que no se le presenten imágenes uniformes.

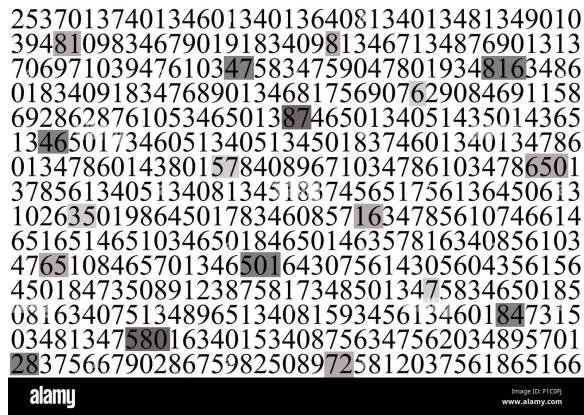


Figura 14: Imagen de prueba para reconocimiento de números.

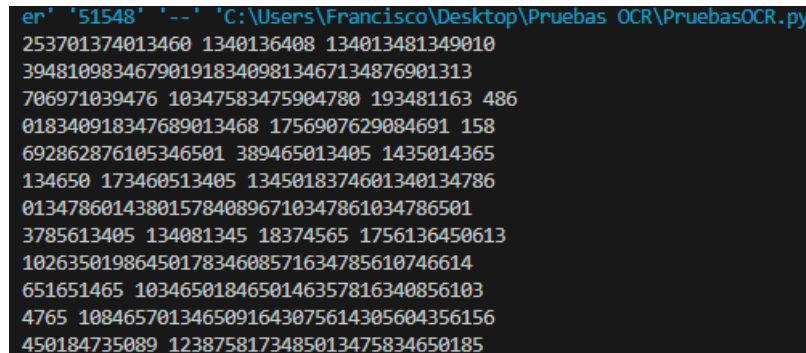


Figura 15: Resultado reconocimiento Tesseract solo números.

Luego de los resultados obtenidos en las pruebas con imágenes que contienen únicamente números y letras, se llevaron a cabo pruebas adicionales utilizando una combinación de ambos conjuntos de caracteres. El objetivo fue determinar y evaluar si el motor de reconocimiento era propenso a confundir letras con números y viceversa. Se utilizó como prueba la imagen mostrada en la Figura 16 y, como se puede observar en la Figura 17, la impresión en consola muestra la transcripción realizada, en esta se obtuvieron buenos resultados, ya que logró reconocer todos los caracteres del texto sin confusiones. Esto enriquece de gran manera el hecho de decidir utilizar Tesseract como herramienta principal para el reconocimiento de ángulos y datos de configuración.

**3N UN LU64R D3 L4 M4NCH4 D3 CUY0  
 N0MBR3 N0 QU13R0 4C0RD4RM3, N0 H4  
 MUCH0 71EMP0 QU3 V1V14 UN H1D4L60 D3  
 LOS D3 L4NZ4 3N 4S71LL3R0, 4D4R64 4N716U4,  
 R0C1N FL4C0 Y 64L60 C0RR3D0R.**

Figura 16: Imagen de prueba para reconocimiento números y letras.

```
python311\python.exe 'C:\Users\Francisco\.vscode\extensions\ms-python.p  
er' '54948' '--' 'C:\Users\Francisco\Desktop\Pruebas OCR\PruebasOCR.py'  
3N UN LUG64R D3 L4 M4NCH4 D3 CUY0  
NOMBR3 NO QU13R0 4CORD4RMS3, NO H4  
MUCHO) 71EMP0 QU3 V1V14 UN H1D4L60 D3  
LOS D3 L4NZ4 3N 4571LL3R0, 4D4R64 4N716U4,  
ROCIN FL4C0 Y 64L60 CORR3DOR.
```

Figura 17: Resultado reconocimiento Tesseract números y letras.

### 7.1.2. Pruebas preliminares Asprise

Asprise es una tecnología especializada en el reconocimiento óptico de caracteres, es desarrollada por una empresa dedicada a soluciones de software para la gestión de documentos y la automatización de procesos. Esta herramienta representa una alternativa al uso de Tesseract. Asprise proporciona un servicio de API que permite el envío y recepción de datos, y aunque implica un costo por uso prolongado de la herramienta, también ofrece una versión de prueba, esta se utilizó para validar las pruebas preliminares.

En estas pruebas iniciales, al igual que con Tesseract, se emplearon tres tipos de imágenes: una con solo letras y símbolos, solo números, y finalmente, una imagen que contiene la combinación de ambos tipos de caracteres. El propósito de esta metodología fue evaluar el rendimiento de la herramienta al aislar y combinar diferentes tipos de caracteres, con el objetivo de verificar que el motor de reconocimiento no los confundiera.

En la primera prueba, se utilizó la imagen representada en la Figura 18, la cual, se envió sin realizar ningún procesamiento previo, esto con el fin de evaluar el rendimiento de la herramienta. Los resultados de esta prueba se presentan en la Figura 19. Estos resultados indican que el algoritmo y el motor de reconocimiento detectan de manera precisa tanto las letras como los símbolos que acompañan al texto, como los signos de puntuación. Esta herramienta demuestra ser un método eficiente y fácil de implementar.

En un lugar de la Mancha, de cuyo  
nombre no quiero acordarme, no ha  
mucho tiempo que vivía un hidalgo de  
los de lanza en astillero, adarga antigua,  
rocín flaco y galgo corredor. Una olla de  
algo más vaca que carnero, salpicón las  
más noches, duelos y quebrantos los  
sábados, lantejas los viernes, algún palo-  
mino de añadidura los domingos, con-  
sumían las tres partes de su hacienda.

Figura 18: Imagen prueba solo letras con Asprise.

```

python311\python.exe' 'c:\Users\Francisco\.vscode\exten
er' '56658' '--' 'C:\Users\Francisco\Desktop\Pruebas O
En un lugar de la Mancha, de cuyo
nombre no quiero acordarme, no ha
mucho tiempo que vivía un hidalgo de
los de lanza en astillero, adarga antigua,
rocín flaco y galgo corredor. Una olla de
algo más vaca que carnero, salpicón las
más noches, duelos y quebrantos los
sábados, lantejas los viernes, algún palo
mino de añadidura los domingos, con
sumían las tres partes de su hacienda.
PS C:\Users\Francisco\Desktop\Pruebas OCR>

```

Figura 19: Resultado prueba solo letras con Asprise.

La imagen utilizada en esta prueba presenta un formato de tabla de datos, que se puede apreciar en la Figura 20, y contiene una serie de números. Se pudo verificar que el motor de reconocimiento de caracteres identifica exclusivamente números y no los confunde con letras. Además, reconoce la separación entre los números, comprendiendo que cada uno pertenece a un número distinto al anterior. Esto, se ilustra claramente en la Figura 21, en donde se muestra que el motor identifica la cadena completa de números presente en la imagen de manera precisa.

	2	3	5	7	11	13	17	19	23
29	31	37	41	43	47	53	59	61	67
71	73	79	83	89	97	101	103	107	109
113	127	131	137	139	149	151	157	163	167
173	179	181	191	193	197	199	211	223	227
229	233	239	241	251	257	263	269	271	277
281	283	293	307	311	313	317	331	337	347
349	353	359	367	373	379	383	389	397	401
409	419	421	431	433	439	443	449	457	461
463	467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593	599
601	607	613	617	619	631	641	643	647	653
659	661	673	677	683	691	701	709	719	727
733	739	743	751	757	761	769	773	787	797
809	811	821	823	827	829	839	853	857	859
863	877	881	883	887	907	911	919	929	937
941	947	953	967	971	977	983	991	997	

Figura 20: Imagen prueba solo números con Asprise.

```
er' '56746' '--' 'C:\Users\Francisco\Desktop\Prueba
2 3 5 7 11 13 17 19 23
29 31 37 41 43 47 53 59 61 67
71 73 79 83 89 97 101 103 107 109
113 127 131 137 139 149 151 157 163 167
173 179 181 191 193 197 199 211 223 227
229 233 239 241 251 257 263 269 271 277
281 283 293 307 311 313 317 331 337 347
349 353 359 367 373 379 383 389 397 401
409 419 421 431 433 439 443 449 457 461
463 467 479 487 491 499 503 509 521 523
541 547 557 563 569 571 577 587 593 599
601 607 613 617 619 631 641 643 647 653
659 661 673 677 683 691 701 709 719 727
733 739 743 751 757 761 769 773 787 797
809 811 821 823 827 829 839 853 857 859
863 877 881 883 887 907 911 919 929 937
941 947 953 967 971 977 983 991 997
```

Figura 21: Resultado prueba solo números con Asprise.

En la prueba final para el motor de reconocimiento Asprise, al igual que con el motor de reconocimiento Tesseract antes validado, se utilizó la imagen que se muestra en la Figura 22. Esta imagen contiene tanto caracteres numéricos como letras, y el propósito fue validar la capacidad de la herramienta para identificar cada diferente tipo de carácter sin confundirlos. Como se puede apreciar en la Figura 23, se logró identificar con precisión toda la cadena de texto, reconociendo letras y números según corresponde.

**MUCH05 4Ñ05 D35PU35, FR3N73 4L P3L070N D3  
FU51L4M13N70, 3L C0R0N3L 4UR3L14N0 8U3ND14  
H48R14 D3 R3C0RD4R 4QU3LL4 74RD3 R3M074 3N QU3  
5U P4DR3 L0 LL3V0 4 C0N0C3R 3L H13L0. M4C0ND0 3R4  
3N70NC35 UN4 4LD34 D3 V31N73 C4545 D3 84RR0 Y  
C4Ñ48R4V4 C0N57RU1D45 4 L4 0R1LL4 D3 UN R10 D3  
46U45 D14F4N45 QU3 53 PR3C1P17484N P0R UN L3CH0  
D3 P13DR45 PUL1D45, 8L4NC45 Y 3N0RM35 C0M0  
HU3V05 PR3H1570R1C05. 3L MUNDO 3R4 74N R3C13N73,  
QU3 MUCH45 C0545 C4R3C14N D3 N0M8R3, Y P4R4  
M3NC10N4RL45 H4814 QU3 53Ñ4L4RL45 C0N 3L D3D0.**

Figura 22: Imagen prueba combinación números y letras con Asprise.

```

python311\python.exe' 'c:\Users\Francisco\.vscode\exter
er' '64911' '--' 'C:\Users\Francisco\Desktop\Pruebas C
MUCH05 4Ñ05 D35PU35, FR3N73 4L P3L070N D3
FU51L4M13N70,3L CORON3L 4UR3L14N0 8U3ND14
H48R14 D3 R3CORD4R 4QU3LL4 74RD3 R3M074 3N QU3
5U P4DR3 L0 LL3V0 4 CONOC3R 3L H13L0. M4C0ND0 3R4
3N70NC35 UN4 4LD34 D3 V31N73 C4545 D3 84RRO Y
C4Ñ48R4V4 CON57RU1D45 4 L4 0R1LL4 D3 UN R10 D3
46U45 D14F4N45 QU3 53 PR3C1P17484N POR UN L3CHO
D3 P13DR45 PUL1D45,8LANC45 Y 3NORM35 COMO
HU3V05 PR3H1570R1C05.3L MUNDO 3R4 74N R3C13N73,
QU3 MUCH45 C0545 C4R3C14N D3 NOM8R3, Y P4R4
M3NC10N4RL45 H4814 QU3 53Ñ4LARL45 CON 3L D3D0.

```

Figura 23: Resultado prueba combinación números y letras Asprise.

Este algoritmo se consideró como una alternativa principal, ya que ofrece una respuesta sólida en la identificación de caracteres. Sin embargo, es importante tener en cuenta que se trata de una alternativa de pago por lo que su uso prolongado tiene un costo mensual.

### 7.1.3. Pruebas preliminares OCRspace

Se evaluó y validó una tercera herramienta como una segunda alternativa al uso de Tesseract, OCRspace. Este motor de reconocimiento óptico de caracteres, al igual que Asprise OCR, es de uso comercial y, por lo tanto, conlleva un costo por su uso. No obstante, ofrece un API de prueba que permite alrededor de 500 solicitudes al día. Al igual que con los dos motores presentados anteriormente, se realizaron validaciones de la herramienta utilizando imágenes sin algún procesamiento previo, con el objetivo de probar su eficiencia y efectividad.

Los resultados obtenidos en la validación de la herramienta demostraron una baja efectividad, ya que solamente en algunos casos se logró obtener el resultado deseado. En la mayoría de los casos, se obtuvo una respuesta incorrecta o una respuesta en blanco, lo que indica que no se pudo reconocer satisfactoriamente los caracteres.

### 7.1.4. Resultados de pruebas con motores seleccionados

Prueba	Tesseract	Asprise OCR
Prueba 1 (Solo números)	98.86 %	100 %
Prueba 2 (Solo letras)	100 %	100 %
Prueba 3 (Combinación de ambas)	100 %	100 %

Cuadro 1: Porcentaje de caracteres identificador correctamente en pruebas preliminares.

Como se observa en el Cuadro 1, el porcentaje de casos exitosos en el reconocimiento de caracteres es en su mayoría del 100 %, lo que demuestra que las dos herramientas validadas

funcionan muy bien cuando no se les aplica ningún tipo de preprocesamiento a las imágenes. Esto significa que al trabajar con los recortes de los ángulos de configuración, no es necesario aplicar una gran cantidad de filtros o realizar modificaciones en las imágenes. En el caso de Tesseract, es necesario aumentar el tamaño de la imagen original, ya que los recortes son muy pequeños por lo que aplicar procesamiento previo a estas es recomendable para aumentar la efectividad del reconocimiento. En cambio, con la herramienta alternativa, Asprise OCR, no es necesario realizar ningún tipo de modificaciones a los recortes, ya que es una herramienta optimizada que realiza esas modificaciones y filtraciones de forma automática.

Sin embargo, para el caso de OCRspace, no se obtuvo un porcentaje aceptable en ninguno de los casos, por lo que no se incluyó en el cuadro de resultados. Si se desea utilizar OCRspace en trabajos posteriores, es importante considerar que es necesario implementar una serie de filtros para mejorar la calidad de la imagen y permitir que la herramienta realice el reconocimiento de manera adecuada.

## 7.2. Validación de captura de pantalla

En fases anteriores a este proyecto, se trabajó con imágenes obtenidas a través de capturas realizadas con una cámara web. Sin embargo, independientemente de la calidad de la cámara utilizada, se enfrentaron desafíos relacionados con factores externos que dificultaron en gran manera el proceso, estos fueron el reflejo de la luz en la pantalla, posicionamiento incorrecto de la cámara y la calidad del sensor óptico de la cámara, estos complicaron el proceso de reconocimiento de caracteres. En busca de una solución más eficiente, se propuso experimentar y validar obtener las imágenes mediante capturas de pantalla. En la Figura 24 se observa la captura realizada como reemplazo al uso de cámara web.



Figura 24: Ejemplo de capturas de pantalla realizadas a sistema Brainlab.

En este sentido, se desarrolló una interfaz junto con un algoritmo que permite al usuario obtener las imágenes que contienen los datos de configuración de manera rápida y eficaz. Esta herramienta facilita la captura de pantalla de forma automática y realiza el recorte de la sección específica del programa Brainlab que contiene la información requerida para que el sistema robótico se posicione correctamente. Este recorte se puede modificar fácilmente en la programación correspondiente a la GUI, en caso de que se realicen modificaciones en el tamaño de las ventanas del sistema de Brainlab.

Para verificar que la captura de pantalla se realizó correctamente, se incluyó una sección en la GUI donde se puede visualizar la captura realizada. Es importante destacar que las dimensiones de esta ilustración no corresponden al tamaño original, ya que se realiza una modificación de tamaño para adecuarla a las dimensiones de la interfaz. En la Figura 25 se muestra la sección de la interfaz donde se puede observar la captura de pantalla.

Esta sección encargada de mostrar una pre visualización de la captura de pantalla, se actualiza constantemente cuando se realiza la captura de pantalla, esto al momento de oprimir el botón correspondiente a dicha tarea. Se programó para que la ventana de la interfaz se oculte automáticamente al momento de realizar la captura, esto para evitar problemas para capturar la ventana del programa de Brainlab.

La interfaz cuenta con dos botones principales y únicos para esta tarea. El primero se encuentra en la esquina inferior izquierda y se utiliza para realizar la captura de pantalla. El segundo botón, al ser presionado, ejecuta secuencialmente todas las funciones necesarias para realizar el reconocimiento de los ángulos de configuración correspondientes a cada junta identificada. Desde la interfaz del sistema de Brainlab, se obtiene una imagen por cada junta, teniendo 1 dato para la primera junta, 2 datos para la segunda junta y 1 dato para la junta final.





Figura 25: GUI realizando captura de pantalla a interfaz de Brainlab.

### 7.3. Preprocesamiento a imágenes

Debido a los buenos resultados obtenidos con las herramientas de reconocimiento óptico de caracteres sin necesidad de aplicar un preprocesamiento a las imágenes, se procedió a validar si el procesamiento previo era necesario para los recortes de ángulos que se realizan en el proceso de obtención de datos. Se realizaron pruebas preliminares con los recortes que se realizan en las imágenes correspondientes a las juntas. Los resultados de estas pruebas fueron satisfactorios para algunos casos, pero en otros, la herramienta confundió caracteres debido a las dimensiones de los recortes. Para solucionar esto, se validó la implementación de algunos filtros en los recortes para maximizar el rendimiento de la herramienta. Los filtros que se aplican son los siguientes:

- Aumento de la imagen de 200 hasta 2000 %
- Transformación a escala de grises
- 2 filtros Gaussianos con kernel  $5 \times 5$  y  $3 \times 3$

Con estos filtros, fue posible mejorar la eficiencia del reconocimiento de los dígitos correspondientes a los ángulos de configuración mediante Tesseract. Como se mencionó anteriormente, en el caso de Asprise OCR no es necesario aplicar estos filtros, pero se aplican para ser consistentes con la secuencia de procesamiento de las imágenes.



Figura 26: Resultado de preprocesamiento a recortes de ángulos

Con el resultado de la aplicación de estos filtros y la modificación de las dimensiones de los recortes, se logró mejorar el rendimiento de la herramienta. En la Figura 26 se observa el resultado del preprocesamiento que se realiza a los recortes que posteriormente se procesan para extraer los datos correspondientes a las configuraciones para el brazo robótico.

#### 7.4. Procesado de la imagen

Para el procesamiento final de los recortes de ángulos de configuración, se implementaron los motores de reconocimiento óptico de caracteres, con los cuales se logró obtener buenos resultados. Estos motores fueron Tesseract y Asprise OCR, siendo el primero de código abierto y, por lo tanto, gratuito, y el segundo una alternativa de uso comercial con un costo asociado.

En el caso de Tesseract, para ajustar adecuadamente la herramienta al reconocimiento de los ángulos de configuración, fue necesario realizar ajustes a través de pruebas y errores en las configuraciones establecidas para los filtros y el redimensionamiento de los recortes. Esto se hizo con el fin de obtener los mejores resultados posibles.

Luego del trabajo de prueba y error, se encontraron las configuraciones que proporcionaron el rendimiento deseado de la herramienta. Estas configuraciones son las siguientes:

- Aumento con rango 200 % a 2000 %
- 2 filtros Guassianos con Kernel de  $5 \times 5$  y  $3 \times 3$

En el caso de Asprise, la situación fue diferente, ya que esta herramienta está muy bien desarrollada y optimizada. Implementar el algoritmo en la interfaz del programa para el reconocimiento de ángulos de configuración fue más sencillo que en el caso de Tesseract, ya que se hace uso de APIs que permiten el envío de los recortes al servicio en la nube y este retorna la información contenida en formato JSON, de dicho recorte. No fue necesario

realizar preprocesamiento de la imagen ni redimensionamiento. Sin embargo, para aprovechar mejor la herramienta y ser consistentes con lo que se trabajó en Tesseract, se aplicaron los mismos filtros a los recortes que se envían al servicio de Asprise cuando se trabaja con esta herramienta.

#### 7.4.1. Reconocimiento de ángulos de configuración

Antes de realizar los recortes de cada junta, se efectúa un recorte en el texto que indica a qué junta pertenece la imagen. Esto se hace para poder llevar a cabo los recortes posteriores de los ángulos en la ubicación adecuada para cada uno. Un ejemplo de este recorte se muestra en la Figura 27.



Figura 27: Ejemplo del recorte a la identificación de la junta.

Para cada una de las juntas, se tiene una cantidad diferente de datos de configuración. En el caso de la junta 1, se tiene 1 ángulo de configuración, como se muestra en la Figura 28. Para la junta 2, se tiene 1 ángulo de configuración y 1 junta lineal ajustable, como se muestra en la Figura 29. Finalmente, para la junta 3, se observa la configuración con 1 ángulo, como se muestra en la Figura 30.

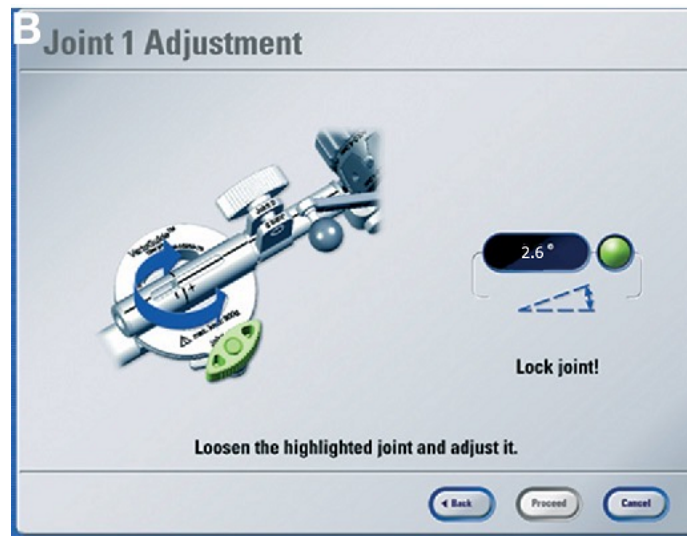


Figura 28: Ejemplo de sección correspondiente a Joint 1 en interfaz de Brainlab.

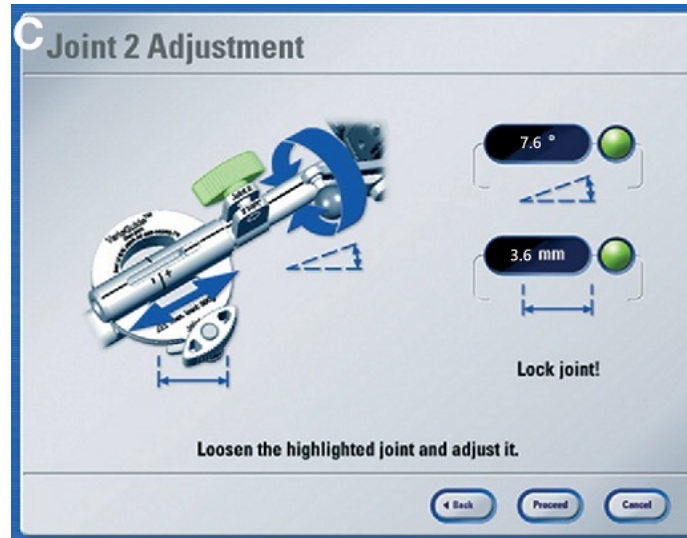


Figura 29: Ejemplo de sección correspondiente a Joint 2 en interfaz de Brainlab.

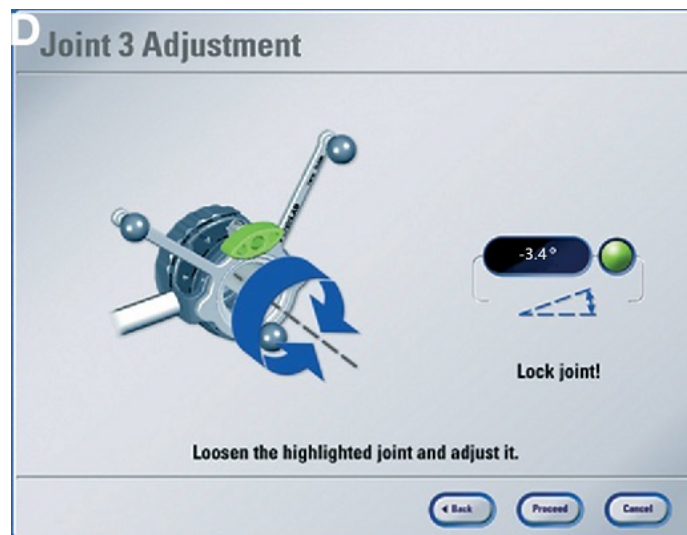


Figura 30: Ejemplo de sección correspondiente a Joint 3 en interfaz de Brainlab.

Como observa, estos ángulos y otras configuraciones se pueden representar como números positivos, números negativos y con decimales. Por esta razón, se realizó la validación del reconocimiento de números, letras y la combinación de estas con símbolos de puntuación.

Los recortes que se realizan a los ángulos de configuración, sin modificaciones previas, las cuales se realizan para cada una de las juntas se observan en las figura 31, que corresponde a un número con decimales y número positivo, en la Figura 32 se muestra un ejemplo con decimales y número negativo, y finalmente, en la Figura 33 se observa el ejemplo de ángulo positivo, con decimales y decenas. Las Figuras mostradas de los recortes originales no representan el tamaño real, ya que se han incrementado las dimensiones con fines de visualización.

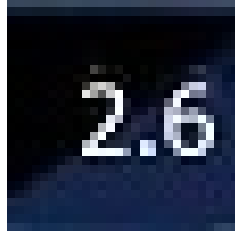


Figura 31: Ejemplo de recorte a junta con decimales.



Figura 32: Ejemplo de recorte a junta con decimales y número negativo.



Figura 33: Ejemplo de recorte a junta con decimales y decenas.

#### 7.4.2. Creación de modelo *Machine Learning* para recortes a identificación de juntas

Se trabajó en la validación de la creación de un modelo de *Machine Learning* para realizar los recortes en la identificación de cada junta. Esto se realizó como una optimización al proceso de ubicar y realizar el recorte de cada identificación de la junta. Aunque esta funcionalidad no se implementó en la versión final del proyecto, se espera que pueda ser incorporada en fases posteriores del proyecto.

En la Figura 34, se observa el entrenamiento del modelo utilizando capturas de pantalla de cada junta. El proceso se realizó utilizando la librería Keras para crear el modelo con 500 épocas y una capa densa con activación relu. Se realizaron pruebas con una mayor cantidad de épocas, sin embargo, no se observó un cambio significativo en la cantidad de pérdida, por lo que se determinó que 500 épocas son suficientes para la creación del modelo.

```
7/7 [=====] - 0s 33ms/step - loss: 10.5063
Epoch 492/500
7/7 [=====] - 0s 39ms/step - loss: 9.9388
Epoch 493/500
7/7 [=====] - 0s 49ms/step - loss: 14.3711
Epoch 494/500
7/7 [=====] - 0s 50ms/step - loss: 10.1100
Epoch 495/500
7/7 [=====] - 0s 38ms/step - loss: 6.3858
Epoch 496/500
7/7 [=====] - 0s 35ms/step - loss: 8.0092
Epoch 497/500
7/7 [=====] - 0s 35ms/step - loss: 7.1776
Epoch 498/500
7/7 [=====] - 0s 32ms/step - loss: 9.8882
Epoch 499/500
7/7 [=====] - 0s 33ms/step - loss: 7.2298
Epoch 500/500
7/7 [=====] - 0s 33ms/step - loss: 9.1289
1/1 [=====] - 0s 231ms/step - loss: 6.6014
```

Figura 34: Entrenamiento del modelo de *Machine Learning* para recorte de juntas.

Los resultados del uso del modelo generado para realizar los recortes en el identificador de la junta correspondiente a las imágenes proporcionadas por el sistema Brainlab se observan en la Figura 35 y en la Figura 36. Como se observa, los recortes realizados mediante el modelo muestran una identificación muy precisa que puede ser utilizada posteriormente para determinar a qué junta pertenece la imagen. Un ejemplo de esto se muestra en la Figura 37, en la cual se realizó el reconocimiento del texto mediante la herramienta principal, Tesseract. Estas pruebas se llevaron a cabo utilizando imágenes obtenidas a partir de capturas realizadas por una cámara web, las cuales fueron adquiridas en la fase anterior. Es importante tener en cuenta que el reconocimiento puede no ser completamente preciso dependiendo de la calidad de la imagen, ya que factores externos como el brillo de la luz en la pantalla y la posición de la cámara pueden afectar el reconocimiento de caracteres.



Figura 35: Resultado 1 uso modelo generado para hacer recorte a identificador.



Figura 36: Resultado 2 uso modelo generado para hacer recorte a identificador.

```
C:\Users\Francisco\Desktop\Documentacion Tesis\Pruebas
C:\Users\Francisco\Desktop\Documentacion Tesis\Pruebas inde
1/1 [=====] - 0s 180ms/step
Joint 3
```

Figura 37: Resultado de reconocimiento de caracteres de modelo con Tesseract.

---

## Protocolo de comunicación TCP/IP

---

Se buscó un método práctico para enviar y recibir datos entre el servidor y el cliente, sin requerir de una interconexión física de los dispositivos, de manera que se permita una mayor portabilidad del sistema. Se determinó que el protocolo de comunicación TCP/IP satisface esta característica. Este protocolo de comunicación cuenta con una serie de pasos y verificaciones que facilitan y aseguran la transmisión de datos. En caso que se pierda información durante la transmisión, el receptor notificará al emisor para que reenvíe los paquetes faltantes, garantizando así, una comunicación confiable.

### 8.1. Pruebas preliminares con protocolo

Las pruebas preliminares consistieron en el envío unidireccional constante de imágenes, simulando una transmisión en tiempo real. Se registró el tiempo en el que las imágenes fueron enviadas y el tiempo en el que fueron recibidas para evaluar la eficiencia de este protocolo de comunicación. Se observa un ejemplo de estas pruebas en la Figura 38. El código utilizado para estas pruebas fue de utilidad para procesos implementados en la interfaz, como lo es el tener un servidor remoto para procesamiento de las imágenes.

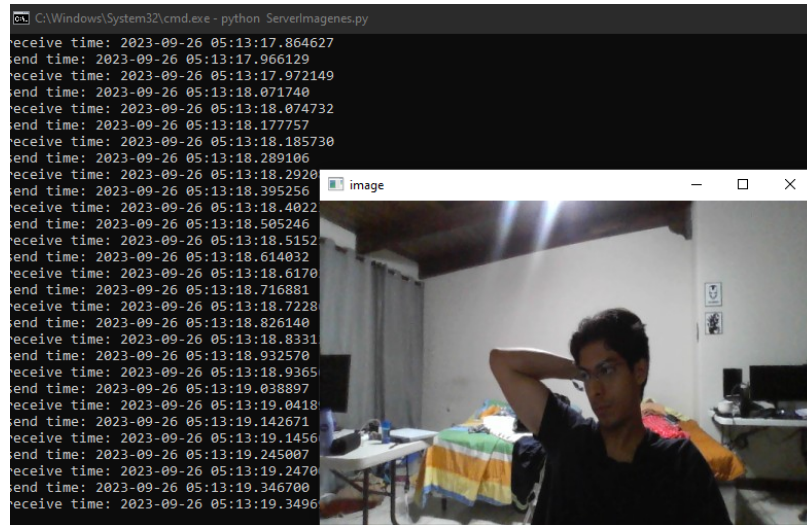


Figura 38: Resultado de pruebas preliminares TCP/IP.

Finalmente, se implementó una transmisión bidireccional en la que el cliente envía las capturas de pantalla al servidor. Estas capturas se recortan para incluir solo el área que contiene la ventana del programa de Brainlab. El servidor, una vez que recibe las capturas, realiza los recortes en las áreas de interés, como lo es el identificador de la junta y los ángulos de configuración. Luego, se lleva a cabo el procesamiento previo de cada recorte según corresponda, se identifican los caracteres y símbolos en cada recorte, se envían de vuelta al cliente a través de una cadena estructurada con identificadores y sus valores correspondientes para cada ángulo de las juntas. Los identificadores enviados en la cadena de datos dependen de la junta que se identifica y se componen de la siguiente manera:

- JAxx
- JBxx,xx
- JCxx

Donde las equis corresponden al valor de cada ángulo de configuración. Como ya se había mencionado anteriormente, estos valores pueden ser positivos, negativos y con decimales. Para poder separar los números de los identificadores. Cuando se recibe la cadena de datos por parte del servidor, el identificador se busca mediante la función *find()*, ya que esta función retorna la posición de dicho indicador, se recorre la cadena de caracteres hasta llegar al final y luego se realiza la conversión a un número del tipo *double* o *float*, debido a que la cantidad de decimales es poca, se realiza la conversión a números de tipo flotante.

En la Figura 39 se muestra un diagrama que describe el proceso por el que pasan los datos para llegar a su destino. El protocolo TCP/IP se encarga de asegurar que los paquetes lleguen de manera segura a su destino, desde la codificación de los datos hasta la verificación y recuperación de los paquetes, con el fin de asegurar que toda la información llegue a su destino de manera efectiva.



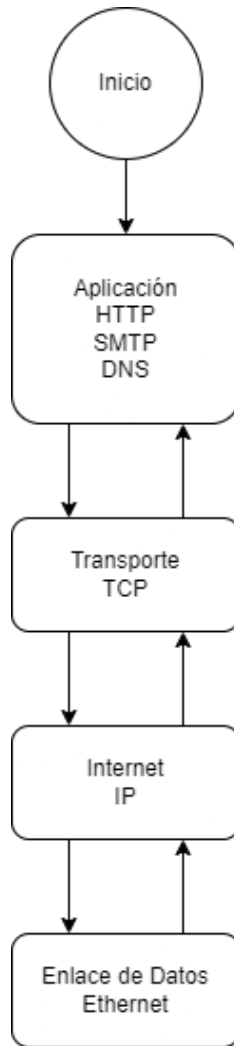


Figura 39: Diagrama funcionamiento de protocolo TCP/IP.

## 8.2. Codificación y decodificación de datos

Para el proceso de codificación utilizado en la transmisión de datos mediante el protocolo de comunicación TCP/IP por parte del cliente, se inicia mediante el uso de la librería OpenCV. En primer lugar, se define una constante que representa la calidad de compresión JPEG. En este proyecto, se utilizó un valor de 90, que proporciona una calidad de imagen lo suficientemente buena para identificar los datos de configuración en cada imagen obtenida de Brainlab, previo a enviar los datos, realiza una compresión de datos para mejorar la transmisión. Luego, se utiliza la función `cv2.imencode` para realizar la codificación de los datos correspondientes a la imagen. Esta función toma como argumentos la imagen que se desea codificar y el parámetro mencionado anteriormente que define la calidad de compresión deseada. Finalmente, se convierte el resultado en un arreglo utilizando la biblioteca `numpy`. Esto es útil para manipular los datos de la imagen de manera más sencilla y, finalmente, se realiza una codificación en base64. Esta codificación se utiliza comúnmente para transmitir

imágenes a través de la red, ya que las cadenas codificadas en base64 son más fáciles de enviar y decodificar en el extremo receptor.

En la Figura 40, se muestra un ejemplo del resultado de la codificación final que tienen las imágenes al ser enviadas al receptor, que en este caso es el servidor.

```
Client socket is connected with Server socket [ TCP_SERVER_IP: localhost, TCP_SERVER_PORT: 8080 ]
b' /9j/4AAQSkZJRgABAQAAQABAAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQgQoHBwYIDAoMDAsKCwsNDhIQDQ4RDgsLEBYQERMUFRUVDA8XGBYUGBj
QUFBT/wAARCAE7AeADASTAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUE
lqC3R1dnd4eXqDhIwGH4iJiPKtLJlWl15iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytL1NXw19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwE
EIFEKRobHBBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzsg50kNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJn
D8vHALcHIoJ49+1IhI60Nkng1z26HrJ3V7biYyS04700ctyc4pADnNBwW54qt2ZWsG578UoGM+1Ju4IzSk8DHmkYN+Y0Zz16VKhLHpUeSfpT1Ug8UmOm3
IZCFazxVbjfgdKmUADcDyDUsfMmrlyM1wy+vvWgu4xhRhgB0z2rNhYD5hzkdKuWTASZyemPX0rnle5rFp6tmzaysq0qnPJP6n/ABrb02ZRDGh711z2yS
Iz/Qn8687sJRDGyOw3DDAnIBxt/qtdLpty4jaNmXIG3ap9A4BHscV5coa7b61kz0rSr5JLZR5gaXYAQD1h8j9a9H0HVo33D01jggkKz/eccf8AAh+VeK6
RHDvuGQDuznKH90leAaBrJQRp+7xkIAQcg5YDYSJ4rTyEUzFFJ4B00cDufpWTgmr2HzWdj3KPxiQwVJAKgA3EjI4P/wCumDxkUIjXaPnQDYdwHzMK8cg
IfdHBP6VzU/iQPtwWjAMy40BjzCP61z8mqImiZpkwPs40Tz/rHFbRoxu49QUhVGNqWtxyRwXJIGLMndz2b/EV5/eX/m3KgNsTyomgDqROf8a1vNZjIsl
/nw1quosUunjKLSiumwRyPmB/rWHfagy3LAMHFzkJ+br1PP6V0R00W5K330R1uwiFtGBjaEZg3qd1cZf2Rjd1IiwcF4V3V70JbVwZA4w2cZHY7+D+X865
GL02EBNNY8jiMPLgmojMc4NVysObowFrShuc55qsJCSM1Isw5pcrW5asTAhjkD5qVuCMnp60xTkcvJKYGRmpF6Dd/JwKczfL/OmgBjk05UGaRSa2Yag8Y
hSOK6Dw3Yy386QxJJK7MFVUBJJPQAcSOKLkDB617H8CfEU3hTU7g2t1bzXV3CbcTToGaEE/eT+63QZ9Cay17sw2XFt056f4R+AcGnw0dz4kupLaaUjBzV
8FvCmNkBaad9mYDho5CMV1ToV5y57wMKuKilyl1c+VT+xB41trAtY3mmXnyAoyTMhcfiuP1ryrxx+zH4980h5Ljw7f526ruaW2j89VHq5mf1r9DG8Mand
tcvibh2ba2/Lk8cV+137YH7R9j3qgK68Mx+G1bU5omSa91CFWNoS0TADzu9zjhVx5d+Ipp3zEZ0SeSfWwMmqal7rue/hJyes1Y8Z/hppJDZpMYHwKAMf
```

Figura 40: Ejemplo de codificación final de las imágenes.

La interfaz se diseñó de manera que para el usuario fuese práctica y sencilla utilizar. En ella se implementaron herramientas que permiten interactuar con lo siguiente:

- Búsqueda de puertos serial.
- Conectar/Desconectar puertos serial.
- Hacer *Screenshot*.
- Enviar imagen a servidor para procesar.
- Visualización de ángulos de configuración.
- Conectar/Desconectar con sistema robótico.
- Mandar datos a sistema robótico.

Como se mencionó anteriormente, para la implementación de la interfaz propuesta en este trabajo de graduación, se implementaron funciones que se encargan de ocultar la interfaz en el tiempo en que se realiza la captura de la imagen. De esta manera, se obtiene solo el espacio de interés sin necesidad de realizar una edición adicional que complique el proceso de reconocimiento de caracteres.

Se implementaron 4 versiones de la interfaz, dos de estas versiones implementan todos los algoritmos, tanto de captura de pantalla como procesamiento y reconocimiento de caracteres correspondientes a los ángulos de configuración, esto, ya que si el usuario posee un equipo de cómputo con capacidad de ejecutar estos algoritmos simultáneamente con los programas de captura de señales utilizados en HUMANA, puede optar por utilizar estas, las

dos versiones restantes se implementan en el sistema embebido, como se mencionó anteriormente, se implementó como herramienta principal para reconocimiento el motor Tesseract y como auxiliar Asprise, se implementó una por cada par de versiones. De igual manera se implementó para el sistema embebido.

## 9.1. Módulo implementado

Se eligió Tkinter como la herramienta para crear la interfaz debido a su simplicidad y practicidad en la edición de diseño. Tkinter permite crear la interfaz mediante código en lugar de interactuar de forma gráfica, lo que facilita la modificación de aspectos de la interfaz simplemente ajustando unas líneas de código.

Tkinter proporciona una amplia variedad de *widgets* con los que es posible implementar todas las herramientas necesarias para que el usuario pueda procesar imágenes, visualizar los ángulos de configuración y luego enviarlos al sistema robótico de manera sencilla.

En la Figura 41 se observa la versión para sistemas embebidos. La versión que incluye todo en la interfaz es mucho más simplificada ya que no es necesario conectar a un servidor TCP/IP, por lo tanto, los campos correspondientes a esto se eliminaron. Dicha interfaz que incluye todo en uno se observa en la Figura 42.

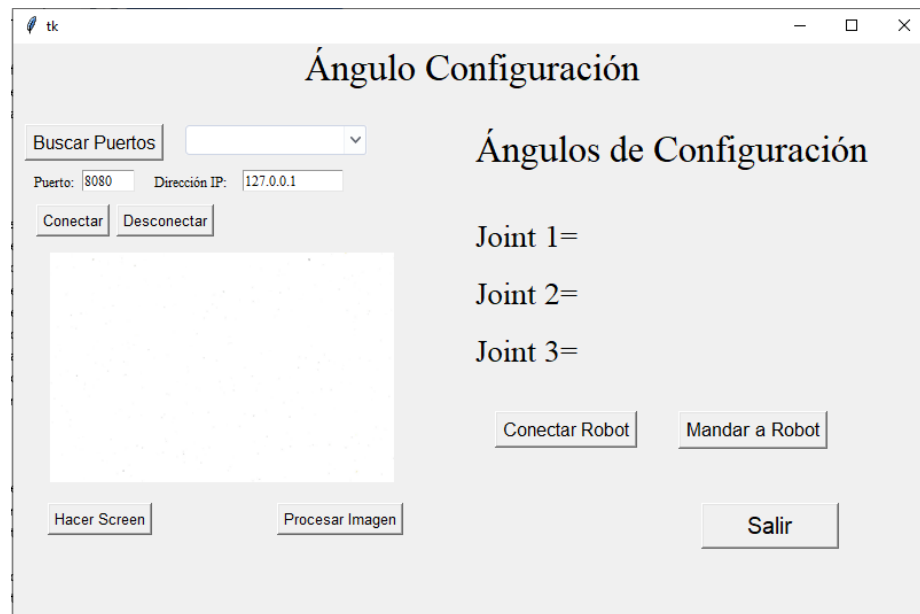


Figura 41: Propuesta de diseño para la interfaz.

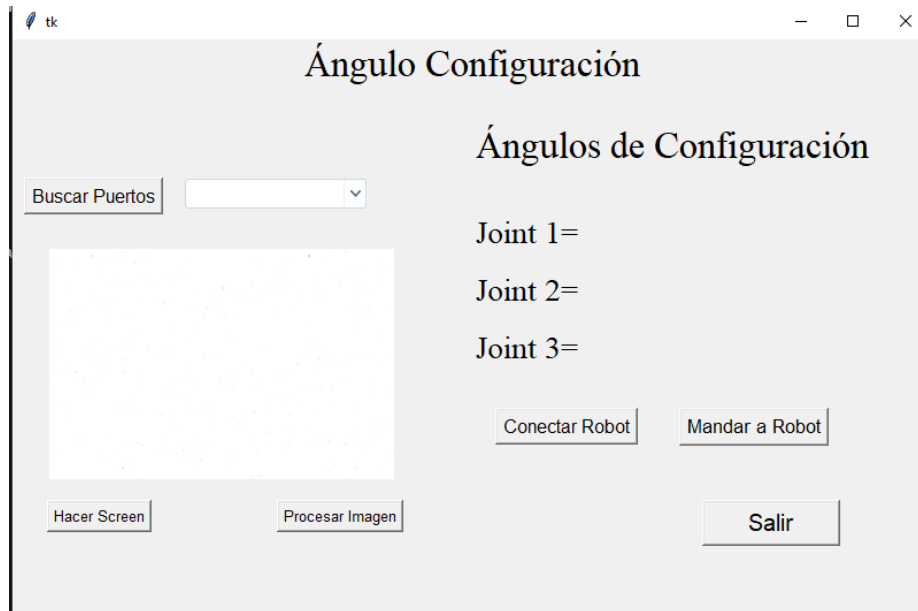


Figura 42: Propuesta de diseño para la interfaz todo en uno.

## 9.2. Herramientas implementadas en interfaz

El diseño de la interfaz incluye un apartado para establecer la comunicación tanto con el servidor TCP/IP como con el sistema robótico. Se observa este apartado en la Figura 43. Al ser la conexión con el servidor mediante TCP/IP, es necesario introducir el puerto y la dirección IP de este. Para conectarse con el sistema robótico, se debe saber el puerto correspondiente asignado a este, y seleccionarlo en el apartado superior que se observa en dicha Figura 43.

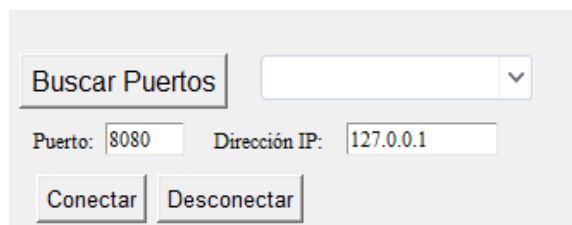


Figura 43: Apartado correspondiente a establecer comunicación con sistema robótico y servidor TCP/IP.

En segundo lugar, la interfaz cuenta con una sección dedicada a realizar capturas de pantalla. En esta sección se encuentra un espacio para visualizar la captura, lo que permite verificar que esta se haya hecho correctamente. Además, se incluye un botón para enviar la imagen al servidor y que esta sea procesada. Se observa esta sección en la Figura 44.

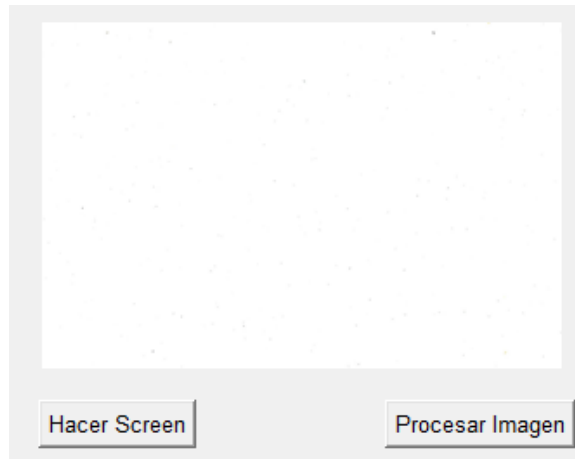


Figura 44: Apartado de la interfaz correspondiente a capturas y procesamiento de las imágenes.

Tercero, se dispone de una sección que permite visualizar los ángulos y distancias lineales de configuración extraídos de las imágenes obtenidas a través del sistema de Brainlab. Esto facilita la verificación que los datos reconocidos se transcriben correctamente. En esta misma sección, se encuentran los botones que permiten establecer la comunicación con el sistema robótico, enviarle todas las configuraciones encontradas. Se observa esta parte en la Figura 45.

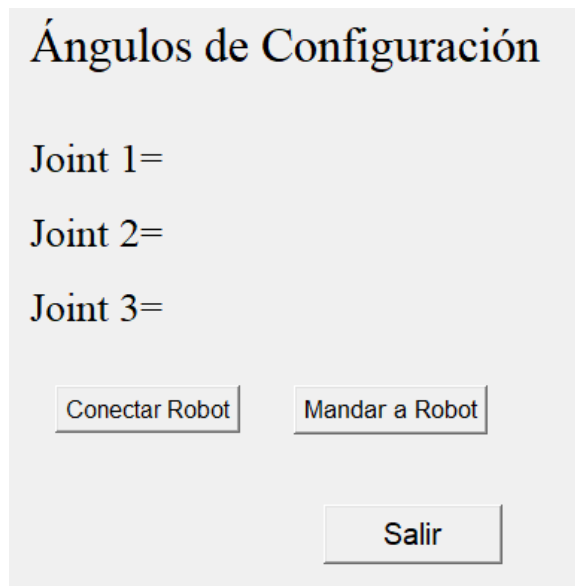


Figura 45: Apartado de la interfaz correspondiente visualizar los datos transcritos e interacción con sistema robótico.

### 9.3. Resultado captura de pantalla

La captura de pantalla se realiza inicialmente a toda la pantalla. Luego, como se mencionó previamente, el espacio que contiene la interfaz de Brainlab se encuentra definido y

predeterminado, lo que permite realizar un recorte de la sección de interés y establecer estos parámetros como predeterminados. Este recorte se guarda para ser enviado al servidor de procesamiento mediante TCP/IP y, así mismo, se muestra en una sección de la interfaz para visualizar y verificar el buen estado de la captura realizada. Se observa el resultado de esta implementación en la Figura 46.



Figura 46: Resultado de implementación de espacio para visualizar captura de pantalla.

#### 9.4. Resultado envío y recepción de datos

El resultado de la identificación de los datos de configuración se muestran en la Figura 47. Estos datos se actualizan cada vez que se ejecuta todo el proceso de captura y procesamiento de los recortes. Cada junta que es identifica tiene una cantidad específica de ángulos o datos, de esta manera, como se observa, la junta 1 y 3 solamente tienen un ángulo de configuración, mientras que la junta 2 posee un ángulo y un componente lineal ajustable.

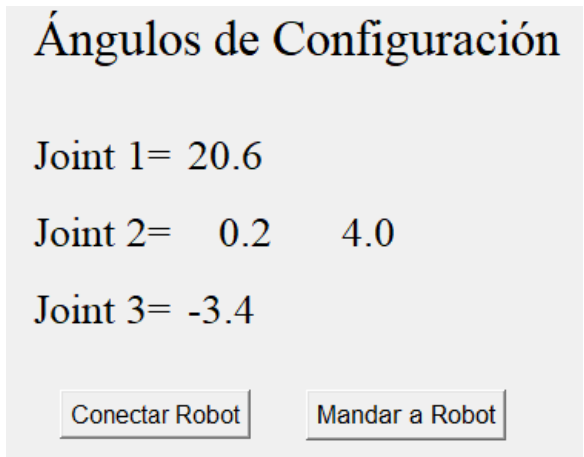


Figura 47: Resultado de implementación de espacio para visualizar los datos reconocidos.



Los sistemas embebidos se utilizan para establecer la conexión entre diversos sistemas, permitiendo que cada uno realice tareas específicas que contribuyan al control y monitoreo de un sistema más complejo. Al emplear sistemas embebidos, se garantiza la ejecución adecuada de cada tarea.

Para validar el uso de un sistema embebido, se implementó una Raspberry Pi 4, ya que esta plataforma ofrece características que favorecen la implementación exitosa del programa de reconocimiento de ángulos de configuración. Algunas de estas características son:

- Memoria RAM 4Gb
- 32 Gb EVO+

### 10.1. Implementación de sistema operativo

El sistema operativo utilizado fue una distribución de Linux diseñada específicamente para trabajar con el ecosistema de la Raspberry Pi. En la Figura 48, se observa el sistema instalado en una tarjeta micro-USB. Este sistema operativo se conoce como Raspbian y está adaptado para la arquitectura de 32 bits de la Raspberry Pi.

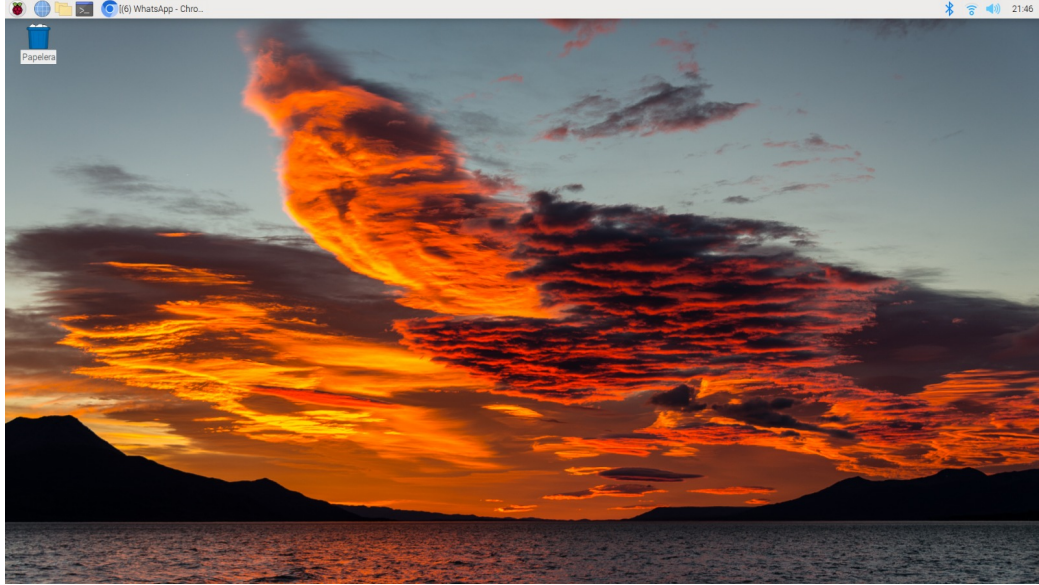


Figura 48: Resultado de instalación de sistema operativo en Raspberry Pi 4.

## 10.2. Resultado procesamiento de imágenes

Como parte de los objetivos de este proyecto de graduación, se buscó migrar el sistema central de procesamiento de imágenes a un sistema embebido. Esto se hizo ya que los sistemas en los que se implementa esta herramienta, HUMANA, no cuentan con la capacidad necesaria para ejecutar aplicaciones pesadas y complejas. Por lo tanto, se decidió validar la implementación de esta herramienta en un sistema embebido para no sobrecargar los sistemas computacionales del área y evitar problemas en el futuro cuando se necesite ejecutar más aplicaciones en los mismos equipos. Esto es debido a que estos equipos actualmente se encargan del monitoreo de señales cerebrales de pacientes con epilepsia, por lo que al sobrecargar los equipos, se corre riesgo de perder información si en caso se presenta una falla.

Este sistema de reconocimiento de ángulos de configuración se compone de un cliente, encargado de tomar las capturas de pantalla y enviar la información al brazo robótico, y un servidor, que realiza el procesamiento y reconocimiento de los ángulos de configuración. El servidor se implementó en el sistema embebido.

Dado que los algoritmos desarrollados se trabajaron con Python, fue sencillo migrarlos de un sistema *Microsoft Windows* a una distribución de Linux. Se realizaron pruebas para validar el funcionamiento del servidor en la Raspberry Pi 4 y se obtuvo el mismo rendimiento que la versión que implementa todas las herramientas dentro de la misma interfaz. En la Figura 49, se observa el servidor ejecutándose en la Raspberry Pi 4, en la Figura 51, se muestra el resultado del procesamiento realizado.

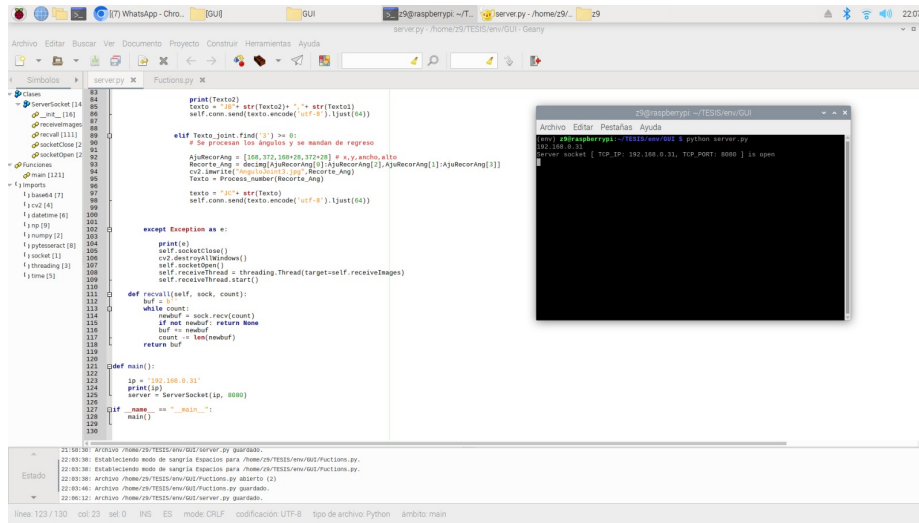


Figura 49: Resultado de ejecución de sistema de reconocimiento de ángulos en Raspberry Pi 4.

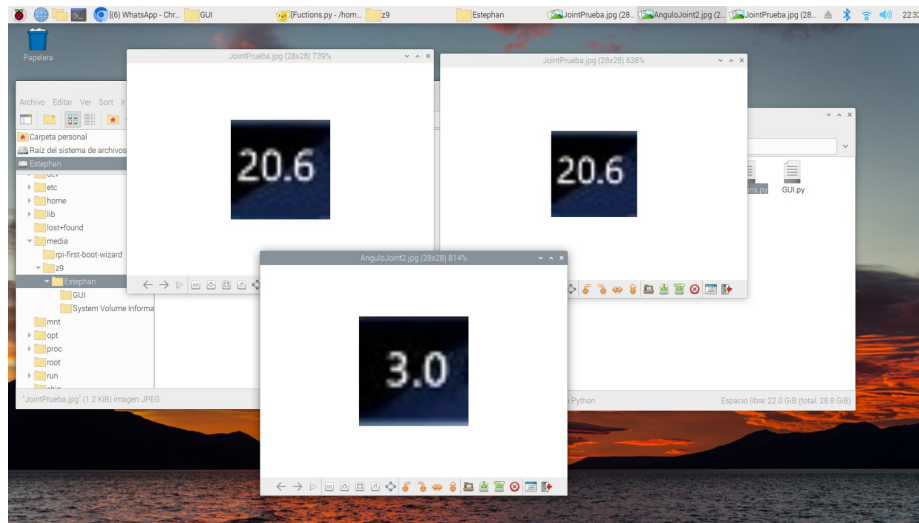


Figura 50: Resultado de recortes realizadas por parte de servidor.

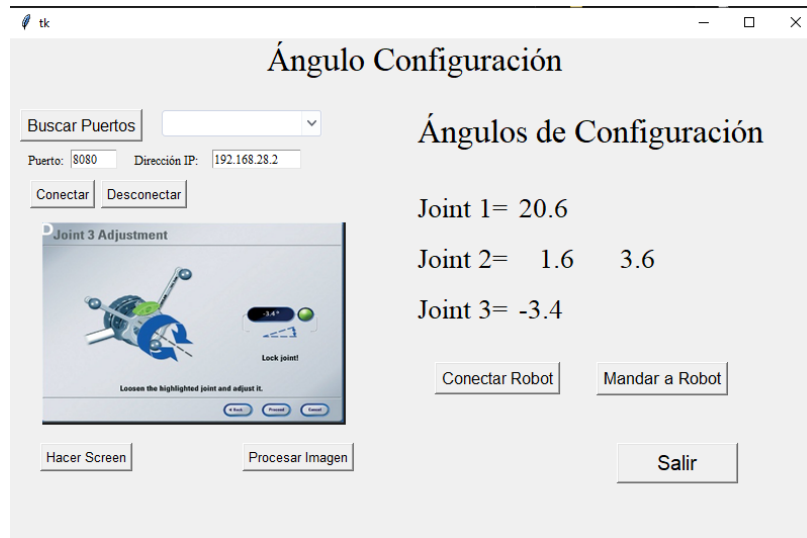


Figura 51: Resultados del procesamiento enviados a cliente.

Para la comunicación con el sistema robótico, se optó por utilizar un protocolo de comunicación sencillo y eficaz. Se propuso un protocolo ampliamente utilizado en aplicaciones que involucran sistemas electrónicos y robóticos debido a su simplicidad y practicidad en la configuración e implementación. Este protocolo es el *Universal Asynchronous Receiver-Transmitter* (UART), que, como su nombre indica, es asíncrono y puede utilizarse tanto para la transmisión como para la recepción de datos. La característica principal de un UART es que es asíncrono, lo que significa que no depende de una señal de reloj externa como ocurre con otros protocolos como el SPI o I2C. Esta característica lo hace más fácil de implementar y adecuado para muchas aplicaciones.

### 11.1. Pruebas preliminares

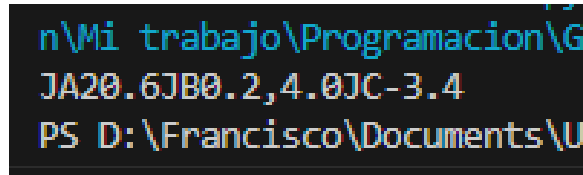
En las pruebas preliminares, se propuso una estructura para transmitir los datos de manera eficiente. Se buscó transmitir todos los datos en una sola cadena para asegurar que el usuario no olvide enviar uno de los ángulos de configuración. Solo se enviarán datos si todas las variables correspondientes a los ángulos tienen valores válidos.

Las características a considerar al utilizar este protocolo son:

- *Baudrate* de 115200
- Conexión física, por medio de USB
- Identificación por índices en cadena

## 11.2. Estructura de envío de datos hacia sistema robótico

Una vez ya se tienen todos los datos de configuración que corresponden a cada junta, se arma una cadena de datos que, con algunos caracteres, se indica a qué junta corresponde cada dato. Esta cadena de datos se observa en la Figura 52, se tienen los prefijos "JA", "JB" Y "JC" que identifican los datos de cada una de las juntas. En el caso de la segunda junta se tiene 2 datos separados por una coma, el primero corresponde al ángulo de configuración y el segundo al ajuste lineal de la junta.



```
n\Mi trabajo\Programacion\G
JA20.6JB0.2,4.0JC-3.4
PS D:\Francisco\Documents\U
```

Figura 52: Estructura de envío de datos por medio de UART.

## 11.3. Envío y recepción de datos

Para validar que los datos se trasladan correctamente hacia el sistema robótico, se utilizó un microcontrolador Arduino. En este, se implementó un algoritmo que es capaz recibir la cadena de datos proveniente del cliente. Luego de recibir dicha cadena de datos, se reconoce el identificador que corresponde de cada junta y se extrae la subcadena de caracteres que contiene la información del ángulo o dato de configuración. Finalmente, se realiza una conversión de ASCII a números de tipo flotante para que de esta manera se pueda hacer uso de esta información por parte del sistema robótico. Dado que el sistema robótico es accesible físicamente al cliente, se implementó de tal forma que es posible conectar vía alámbrica. La propuesta del protocolo implementado toma en cuenta la retroalimentación hacia el cliente, este es capaz de retornar datos hacia el cliente, se pretende que esta funcionalidad sea de utilidad cuando se desarrolle el sistema robótico.

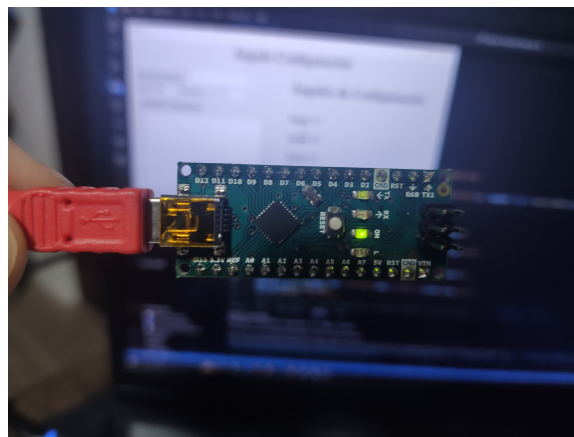


Figura 53: Recepción de datos implementada con Arduino.

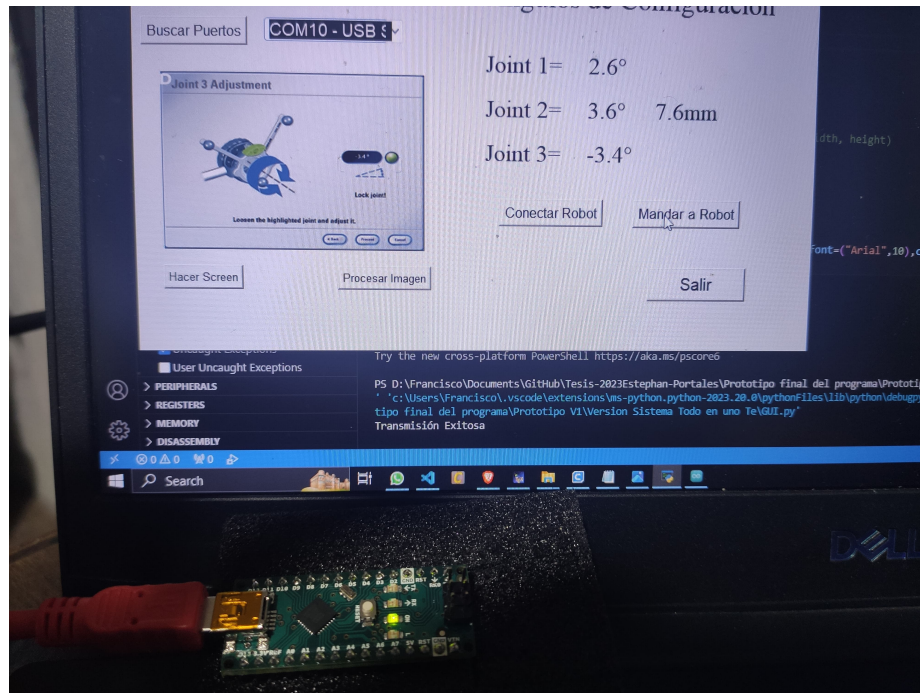


Figura 54: Recepción de datos exitosa con Arduino.

Para demostrar el funcionamiento de la transmisión, conversión y almacenamiento de los ángulos y datos de configuración en el sistema robótico, se imprimió la cadena de datos al momento de enviarlo una vez el microcontrolador del sistema robótico lo recibe, esto para validar que los datos se transmiten correctamente. En la Figura 52 se observa la cadena de datos que el cliente envía al sistema robótico y en la Figura 55 se observan los datos luego de haberse procesado y almacenado en el microcontrolador del sistema robótico.

Los resultados de la transmisión se observan en la Figura 56, la conexión serial vía alámbrica permite que los datos se transmitan de forma segura y rápida, esto permitiendo que la totalidad de datos llegue a su destino.

```

Transmisión Exitosa
PS D:\Francisco\Documents\GitHub\Tesis-2023Estephan-Portales\Prototipo final del programa\Prototipo V1\Version Sistema Todo en uno Te\GUI.py'
No se pudo conectar al puerto Serial
Transmisión Exitosa
JA20.60JB3.00,89.20JC-3.40;

```

Figura 55: Impresión de datos y transmisión exitosa.

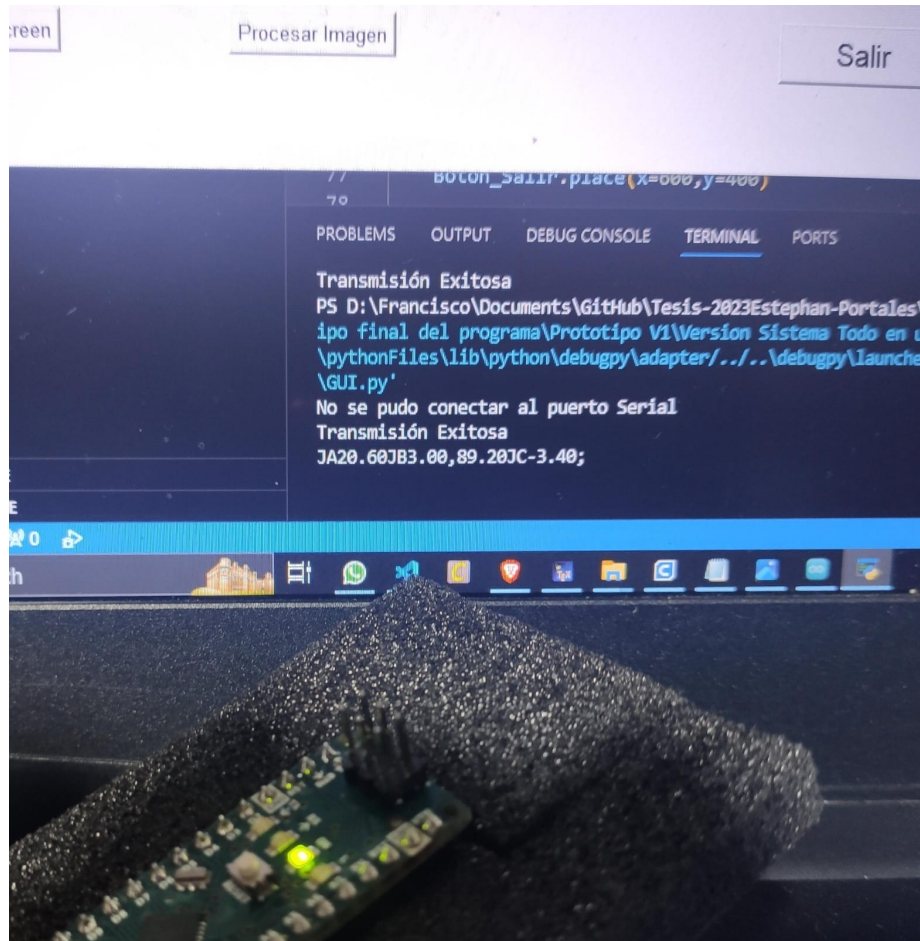


Figura 56: Recepción de datos exitosa con Arduino.



La mejor herramienta, validada en este trabajo, para el procesamiento óptico de caracteres de código abierto y por lo tanto accesible para desarrolladores, es Tesseract.

El uso de *screenshots* como método para capturar la pantalla del sistema de Brainlab resulta mucho más eficiente que el realizar captura mediante una cámara web.

El preprocesamiento aplicado a los recortes para su posterior transcripción no requiere de muchas capas de filtrado, ya que los motores utilizados presentan una alta eficiencia en ausencia de dichos filtros. Sin embargo, es recomendable para así aprovechar al máximo la capacidad de reconocimiento de la herramienta.

Migrar los algoritmos de preprocesamiento y procesamiento imágenes para extraer los datos de calibración no representa un consumo de recursos significativo al sistema embebido. Esto valida la viabilidad de la migración a un sistema embebido.

El lenguaje de programación Python fue fundamental para poder migrar las herramientas a un sistema embebido, esto debido a que ofrece compatibilidad con diferentes sistemas operativos.

El protocolo implementado para la transmisión de datos al sistema robótico, UART, resulta práctico y eficiente ya que su estructura de funcionamiento asíncrona no es compleja y no requiere de algoritmos complejos para su implementación. Esto respalda la elección del protocolo UART para la comunicación con el sistema robótico.

El protocolo de comunicación TCP/IP es muy eficiente ya que posee algoritmos de seguimiento de paquetes por lo que se logra una transmisión exitosa de todos los datos, además transmite los datos a una alta velocidad, por lo que resulta una muy buena opción para transmisión de imágenes.

En futuros trabajos que involucren el reconocimiento de características de configuración a partir de una imagen, se recomienda implementar más herramientas de código abierto que brinden un alto rendimiento y diferentes aplicaciones, ya que una de las limitaciones de Tesseract es que no ofrece una amplia gama de aplicaciones, como lo es reconocer texto escrito a mano.

Dado que no se tuvo a una persona trabajando en el sistema del brazo robótico, se recomienda validar la estructura propuesta en este trabajo, para la comunicación teniendo el brazo en funcionamiento.

Se recomienda indagar en la posibilidad de integrar la herramienta desarrollada al sistema que posee HUMANA de Brainlab, esto para facilitar y optimizar el proceso de reconocimiento de ángulos de configuración, para tener todo integrado en una sola interfaz.

Se recomienda trabajar con imágenes vectorizadas, de manera que no se pierda calidad al momento de ampliar los recortes.

Ante la posibilidad de ajustes en las ventanas del sistema Brainlab, en actualizaciones de dicho sistema, se recomienda la implementación de un modelo de inteligencia artificial o *machine learning*. Este modelo tendría la capacidad de identificar las áreas específicas de interés y realizar los recortes para su posterior reconocimiento, asegurando así la robustez del sistema frente a cambios en la disposición o tamaño de las ventanas. Con esta solución, se garantiza una detección precisa de datos de calibración mediante un motor de reconocimiento de caracteres, incluso en escenarios donde las dimensiones y ubicaciones de las ventanas experimenten modificaciones.

Los algoritmos desarrollados se pusieron a prueba mediante *screenshots* obtenidos con las herramientas que Varioguide ofrece en su interfaz, sin embargo, los algoritmos para captura de pantalla y reconocimiento de caracteres, se desarrolló como un sistema de lazo abierto. Se recomienda explorar los distintos puertos de salida de video y las herramientas que este ofrece Varioguide para así generar un sistema lazo cerrado y lograr un reconocimiento en

tiempo real, y de esta manera poder retroalimentar al sistema robótico sobre la posición actual de este.

- [1] S. Galicia, “Optimización de un sistema de procesamiento de imágenes para el reconocimiento de variables en la pantalla del sistema de Brainlab de HUMANA,” Trabajo de graduación de grado de licenciatura, Universidad del Valle de Guatemala, 2022.
- [2] HUMANA. “HUMANA.” (2023), dirección: [humanagt.org](http://humanagt.org) (visitado 2023).
- [3] Brainlab. “Brainlab transforma el sector de la salud mediante tecnología basada en software.” (2023), dirección: [brainlab.com](http://brainlab.com) (visitado 2023).
- [4] A. Builes y E. Palacio, “Aplicación del reconocimiento de imágenes para la manipulación robótica de objetos en movimiento,” Trabajo de grado para licenciatura en ingeniería mecánica, Universidad Pontificia Bolivariana, 2015.
- [5] L. E. Palafox, “Técnicas de procesamiento de imágenes utilizando un procesador digital de señales,” Tesis de postgrado, Instituto Politécnico Nacional, 2002.
- [6] F. J. González, “Reconocimiento de objetos utilizando OpenCV y Python en una Raspberry pi 2 en una tlapalería,” Trabajo de graduación de grado de licenciatura, Universidad Autónoma del Estado de Mexico, 2017.
- [7] J. G. González, “Diseño de un sistema de procesamiento de imágenes para la identificación de variables en una pantalla de sistema Brainlab del Centro de Epilepsia y Neurocirugía Funcional HUMANA,” Trabajo de graduación de grado de licenciatura, Universidad del Valle de Guatemala, 2022.
- [8] D. de Electrónica y Autonomía, *Procesamiento Digital de Imágenes*, <http://dea.unsj.edu.ar/imagenes/recursos/Capitulo1.pdf>, Accessed: 2023-6-15, 2023.
- [9] IBM. “¿Que es la visión artificial?” (2023), dirección: <https://www.ibm.com/mx-es/topics/computer-vision> (visitado 2023).
- [10] Microsoft. “OCR: reconocimiento óptico de caracteres.” (2023), dirección: <https://learn.microsoft.com/es-es/azure/cognitive-services/computer-vision/overview-ocr> (visitado 2023).

- [11] Klippa. “Tesseract OCR: ¿Qué es y por qué lo deberías elegir en el 2023?” (2023), dirección: <https://www.klippa.com/es/blog/informativo/que-es-tesseract-ocr/#:~:text=OCR%3A%20Klippa%20DocHorizon-,%C2%BFQu%C3%A9%20es%20Tesseract%3F,como%20%E2%80%9CGoogle%20Tesseract%20OCR%E2%80%9D>. (visitado 2023).
- [12] I. Library. “Introduction to OCR and Searchable PDFs.” (2023), dirección: <https://bytepace.medium.com/what-is-tesseract-and-how-it-works-dfff720f4a32> (visitado 2023).
- [13] OpenCV. “OpenCV.” (2023), dirección: <https://opencv.org/about/> (visitado 2023).
- [14] Capacitarte. “¿Qué es y para qué sirve Pyhon?” (2023), dirección: <https://www.capacitarte.org/blog/nota/que-es-y-para-que-sirve-python> (visitado 2023).
- [15] IMB. “Protocolos TCP/IP.” (2023), dirección: <https://www.ibm.com/docs/es/aix/7.1?topic=protocol-tcpip-protocols> (visitado 2021).
- [16] R&S®Essentials. “Qué es UART.” (2023), dirección: [https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart\\_254524.html#:~:text=UART%20\(universal%20asynchronous%20receiver%20%2F%20transmitter,y%20recibir%20en%20ambas%20direcciones](https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart_254524.html#:~:text=UART%20(universal%20asynchronous%20receiver%20%2F%20transmitter,y%20recibir%20en%20ambas%20direcciones). (visitado 2023).
- [17] S. Luchetti. “Sistemas embebidos y sus características.” (2021), dirección: <https://tech.tribalyte.eu/blog-sistema-embebido-caracteristicas> (visitado 2023).
- [18] R. P. Foundation. “What is a Raspberry Pi?” (2023), dirección: <https://www.raspberrypi.org/> (visitado 2023).
- [19] Arduino. “Arduino Hardware.” (2023), dirección: <https://www.arduino.cc/en/hardware> (visitado 2023).
- [20] Filmora. “Introducción Completa a las Tarjetas Capturadoras.” (2023), dirección: <https://filmora.wondershare.es/screen-recorder/introduction-to-capture-card.html> (visitado 2023).