UNIVERSIDAD TÉCNICA DE COTOPAXI

FACULTAD DE CIENCIAS DE LA INGENIERÍA Y APLICADAS

CARRERA DE INGENIERÍA ELECTROMECÁNICA

PROPUESTA TECNOLÓGICA

TITULO: DESARROLLO DE UN DISPOSITIVO DE IDENTIFICACIÓN DE NÚMEROS Y LETRAS EMPLEANDO HARDWARE Y SOFTWARE LIBRE ORIENTADO A LA ENSEÑANZA DE NIÑOS DE EDUCACIÓN BÁSICA.

Autores:

Hugo Livardo Alomía Landázuri

David John Cedeño Arcos

Tutor:

Ing. Byron Paúl Corrales Bastidas M.Sc.

Latacunga- Ecuador

Septiembre 2020

DECLARACIÓN DE AUTORÍA

Nosotros **Hugo Livardo Alomía Landázuri** y **David John Cedeño Arcos** declaramos ser los autores del presente proyecto de investigación**: Desarrollo de un dispositivo de identificación de números y letras empleando Hardware y Software libre orientado a la enseñanza de niños de Educación Básica**, siendo el **Ing. Byron Paúl Corrales Bastidas M.Sc.** tutor del presente trabajo; y eximimos expresamente a la Universidad Técnica de Cotopaxi y a sus representantes legales de posibles reclamos o acciones legales.

Además, certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

……. ……

Hugo Livardo Alomía Landázuri
CI. 1500890932

David John Cedeño Arcos
CI: 0503455644

AVAL DEL TUTOR DEL PROYECTO DE TITULACIÓN

En calidad de tutor del Trabajo de Investigación sobre el título:

"DESARROLLO DE UN DISPOSITIVO DE IDENTIFICACIÓN DE NÚMEROS Y LETRAS EMPLEANDO HARDWARE Y SOFTWARE LIBRE ORIENTADO A LA ENSEÑANZA DE NIÑOS DE EDUCACIÓN BÁSICA", de Alomía Landázuri Hugo Livardo y Cedeño Arcos David John, de la Carrera de Ingeniería Electromecánica, considero que dicho informe investigativo cumple con los requerimientos metodológicos y aportes científico-técnicos suficientes para ser sometidos a la evaluación del Tribunal de Validación de proyecto, que el Honorable Consejo Directivo de la Facultad de Ciencias de la Ingeniería y Aplicadas de la Universidad Técnica de Cotopaxi designe para su correspondiente estudio y calificación.

Latacunga, Septiembre 2020

Tutor de Titulación

_____

Ing. Byron Paúl Corrales Bastidas M.Sc.
CC:0502347768

APROBACIÓN DEL TRIBUNAL DE TITULACIÓN

En calidad de tribunal de lectores, aprueban el siguiente Informe de Investigación de acuerdo a las disposiciones reglamentarias emitidas por la Universidad Técnica de Cotopaxi, y por la Facultad de Ciencias de la Ingeniería y Aplicadas; por cuanto, los postulantes: Alomía Landázuri Hugo Livardo y Cedeño Arcos David John, con el título de proyecto de titulación:

"DESARROLLO DE UN DISPOSITIVO DE IDENTIFICACIÓN DE NÚMEROS Y LETRAS EMPLEANDO HARDWARE Y SOFTWARE LIBRE ORIENTADO A LA ENSEÑANZA DE NIÑOS DE EDUCACIÓN BÁSICA", han considerado las recomendaciones emitidas oportunamente y reúne los méritos suficientes para ser sometidos al acto de Sustentación de Proyecto.

Por lo antes expuesto, se autoriza realizar los empastados correspondientes, según la normativa institucional.

Latacunga septiembre 2020

Para constancia firman:

Lector 1
Ing. Verónica Paulina Freire Andrade
CC: 0502056229

Lector 2
Ing. Luigi Orlando Freire Martínez
CC:0502529589

Lector 3
Luis Miguel Navarrete López
CC: 1803747284

AGRADECIMIENTO

Agradezco a Dios por permitirme cumplir mis metas, a la Universidad Técnica de Cotopaxi por abrirme las puertas, en especial, a la Facultad de Ingeniería Electromecánica por ser parte de este proceso de formación para alcanzar mi meta profesional, a los docentes con los que tuve el privilegio de ser su alumno y que compartieron conmigo, grandes conocimientos, en especial a nuestro tutor, Ing. Paúl Corrales. por brindarnos su apoyo durante este último paso para cumplir este reto. A mis padres y abuelos, quienes fueron incondicionales para que pueda convertirme en un gran profesional. A mi compañero y amigo, David Cedeño, gracias por el apoyo y dedicación en nuestro proyecto de titulación que hoy nos permite cumplir esta meta.

HUGO

AGRADECIMIENTO

Agradezco primeramente a Dios, a la Virgen y al Niño Divino por cuidarme y bendecirme con su sabiduría y protección. Para mi madre el más grande de los agradecimientos por todo su esfuerzo para sacarme adelante, a mis familiares, a mis compañeros y a todos los ingenieros de la Universidad que han logrado inculcarme valores y conocimientos dentro y fuera de las aulas. A mis amigos Cristian, Faustico, Cajitas y a mi novia por sus consejos y apoyo incondicional.

Agradezco de todo corazón a mi compañero y amigo Hugo Alomía que con su esfuerzo, apoyo y dedicación logramos juntos terminar nuestro proyecto de titulación, le deseo mucha suerte en su vida profesional que siga cosechando éxitos en su vida día a día.

Agradezco enormemente al Ing. Paúl Corrales por apoyarnos desde el primer momento que lo elegimos como nuestro tutor para la realización de nuestra tesis; por brindarnos su valioso tiempo y aportarnos con su conocimiento en cada revisión y sobre todo por nunca negarnos su incondicional ayuda.

DAVID

DEDICATORIA

Este logro se lo dedico a mis padres Livardo Alomía y Julia Landázuri, por su apoyo incondicional, su paciencia y dedicación para que yo pueda alcanzar mi meta profesional. A mis abuelos Hugo y Julia quienes me inculcaron valores e hicieron de mí una persona de bien. A toda mi familia, amigos y quienes estuvieron junto a mi durante este proceso que, gracias a sus consejos oportunos y sinceros, hoy se reflejan en mí.

HUGO

DEDICATORIA

Dedico esta tesis a Dios por guiarme en lo correcto día a día, y de manera muy especial a mi madre Cenaida Arcos que siempre me apoyo incondicionalmente tanto en lo moral como en lo económico ya que fue un pilar fundamental para lograr mi anhelada meta. A mis abuelitos Rosa Silva y Marcial Arcos que siempre me dan su bendición. Por último y no menos importante a mi novia Lizbeth Claudio por su cariño y apoyo durante el transcurso de mi carrera universitaria.

DAVID

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

UNIVERSIDAD TÉCNICA DE COTOPAXI

FACULTAD DE CIENCIAS DE LA INGENIERÍA Y APLICADAS

TEMA: DESARROLLO DE UN DISPOSITIVO DE IDENTIFICACIÓN DE NÚMEROS Y LETRAS EMPLEANDO HARDWARE Y SOFTWARE LIBRE, ORIENTADO A LA ENSEÑANZA DE NIÑOS DE EDUCACIÓN BÁSICA.

Autores: Alomía Landázuri Hugo Livardo

Cedeño Arcos David John

RESUMEN

El presente proyecto desarrolla un dispositivo para identificar números y letras, utilizando Software y Hardware libre, orientado a la enseñanza de niños de Educación Básica, y está dirigido a poblaciones vulnerables desde los espacios de la Proyección Social, ya que según la constitución de la Republica del Ecuador, la educación es un derecho de las personas a lo largo de su vida, lo cual motivó a que se realice una investigación basada en esta necesidad. El dispositivo procesa visión artificial, enviando en tiempo real a una interfaz, lugar en donde la imagen es clasificada para reconocer números o letras según sea identificado.

Para ello se empleó redes neuronales, las mismas que fueron entrenadas utilizando software libre, tanto para letras mayúsculas, minúsculas y números, permitiendo así que el sistema no se sature con una sola red. Una vez obtenidas las ponderaciones dadas por el entrenamiento de las redes neuronales se implementa el algoritmo para su posterior validación.

De las pruebas realizadas se puede determinar que el módulo presenta una exactitud y confiabilidad del 95 a 98% para su uso, además se experimentó con docentes de Educación Básica ajustando a sus necesidades y requerimientos.

El dispositivo es de fácil uso para docentes, o personas que posean conocimientos básicos de computación, ya que su interfaz es muy similar a la de una computadora de escritorio, además presenta la ventaja de ser portátil y no depender de recursos adicionales como internet.

Palabras clave: Visión artificial, hardware libre, software libre, redes neuronales, sistemas embebidos.

TECHNICAL UNIVERSITY OF COTOPAXI

FACULTY OF SCIENCES AND APPLIED ENGINEERING

THEME: DEVELOPMENT OF A NUMBER AND LETTER IDENTIFICATION DEVICE USING HARDWARE AND FREE SOFTWARE, ORIENTED TO THE TEACHING OF CHILDREN OF BASIC EDUCATION.

Authors: Alomía Landázuri Hugo Livardo

Cedeño Arcos David John

ABSTRACT

Developing a device to identify numbers and letters, by free Software and Hardware, aimed at teaching children into elementary Education, and is aimed at vulnerable populations from the spaces of Social Projection was the aim of this research, since according to the Constitution of the Republic of Ecuador, education is a right of people throughout their lives, which led to an investigation based on this need. The device processes artificial vision, sending it in real time to an interface, a place where the image is classified to recognize numbers or letters as identified. For this, neural networks were used, the same ones that were trained using free software, for both uppercase and lowercase letters and numbers, thus allowing the system not to be saturated with a single network. Once the weights given by the training of the neural networks have been obtained, the algorithm is implemented for its subsequent validation. From the tests carried out it can be determined that the module has an accuracy and reliability of 95 to 98% for its use, in addition it was experimented with elementary Education teachers adjusting to their needs and requirements. The device is easy to use for teachers, or people who have basic computer knowledge, since its interface is very similar to that of a desktop computer, it also has the advantage of being portable and not depending on additional resources such as the internet.

Keywords: Artificial vision, free hardware and software, neural networks, embedded systems.

Universidad Técnica de Cotopaxi

CENTRO DE IDIOMAS

AVAL DE TRADUCCIÓN

En calidad de Docente del Idioma Inglés del Centro de Idiomas de la Universidad Técnica de Cotopaxi; en forma legal **CERTIFICO** que: La traducción del resumen del proyecto de Titulación al Idioma Inglés presentado por los señores Egresados de la Carrera de **Ingeniería Electromecánica** de la **Facultad de Ciencias de la Ingeniería y Aplicadas: Hugo Livardo Alomía Landázuri y David John Cedeño Arcos**, cuyo título versa **"Desarrollo de un dispositivo de identificación de números y letras empleando Hardware y Software libre orientado a la enseñanza de niños de Educación Básica"**, lo realizaron bajo mi supervisión y cumple con una correcta estructura gramatical del Idioma.

Es todo cuanto puedo certificar en honor a la verdad y autorizo a los peticionarios hacer uso del presente certificado de la manera ética que estimaren conveniente.

Latacunga, 21 de septiembre del 2020

Atentamente,

Mg. Lidia Rebeca Yugla Lema
DOCENTE CENTRO DE IDIOMAS
C.C. 0502652340

CENTRO DE IDIOMAS

INTRODUCCIÓN

La inteligencia artificial se ha definido de diversas maneras a lo largo de la historia, pero la definición que generalmente es aceptada es la capacidad de un ordenador o máquina de pensar o la capacidad de actuar inteligentemente. La primera de estas definiciones se centra en la idea de inteligencia humana, es decir, el proceso que se realiza para llegar a una conclusión, mientras que la segunda definición se centra en los resultados que se obtienen sin centrarse en el proceso seguido para obtener dichos resultados [1]. En el proyecto se explicará cómo se ha desarrollado un dispositivo para la detección de números y letras, mediante el uso de Software libre y Hardware libre, acompañado del entrenamiento de redes neuronales.

Las redes neuronales artificiales, son modelos matemáticos que intentan reproducir el funcionamiento del sistema nervioso, así como todo modelo neuronal; también conocido como red con aprendizaje conexionista y procesamiento de distribución paralela; además realizan una simplificación del sistema real que simulan y toman las características principales para una mejor interpretación del mismo para la resolución de una tarea determinada y especifica [2].

La ejecución del dispositivo para detectar números y letras, empleando software y hardware libre, orientado a ser usado en el aprendizaje de niños de educación básica, es de fácil funcionamiento para el usuario y usa tecnología de programación e información, lo cual provee de nuevos métodos de aprendizaje que incentivan a la enseñanza estudiantil.

El dispositivo se ha desarrollado dentro del ambiente académico y de investigación formativa, cuyos componentes principales son una tarjeta Raspberry PI 4, una cámara de conexión directa a Raspberry, una pantalla LCD para la visualización gráfica y el entrenamiento de redes neuronales para su posterior programación respectiva, para la detección de números o letras según sea lo requerido y está dirigido a la enseñanza de niños de educación básica de poblaciones vulnerables desde los espacios de la Proyección Social.

1. INFORMACIÓN BÁSICA

PROPUESTO POR:

- Hugo Livardo Alomía Landázuri
- David John Cedeño Arcos.

TEMA APROBADO: Desarrollo de un dispositivo de identificación de números y letras empleando Hardware y Software libre orientado a la enseñanza de niños de Educación Básica.

CARRERA: Ingeniería Electromecánica.

DIRECTOR DE LA PROPUESTA TECNOLÓGICA:

Ing. Byron Paúl Corrales Bastidas M.Sc.

EQUIPO DE TRABAJO:

- Hugo Alomía
- David Cedeño
- Ing. Byron Paúl Corrales Bastidas M.Sc.

LUGAR DE EJECUCIÓN: Región Sierra, Provincia Cotopaxi, Cantón Latacunga, Parroquia Eloy Alfaro, Universidad Técnica de Cotopaxi.

TIEMPO DE DURACIÓN DE LA PROPUESTA: 6 meses

FECHA DE ENTREGA: Septiembre 2020

LÍNEA DE INVESTIGACIÓN: Procesos Industriales.

SUBLÍNEA DE INVESTIGACIÓN: Automatización, control y protecciones de sistemas electromecánicos.

TIPO DE PROPUESTA TECNOLÓGICA: Tecnología e Innovación.

2. DISEÑO INVESTIGATIVO DE LA PROPUESTA TECNOLÓGICA

2.1. TÍTULO DE LA PROPUESTA TECNOLÓGICA

Desarrollo de un dispositivo de identificación de números y letras empleando Hardware y Software libre orientado a la enseñanza de niños de Educación Básica.

2.2. TIPO DE ALCANCE

Integrador: El presente proyecto, exige la aplicación coordinada de conocimientos adquiridos en el transcurso de la Carrera Universitaria, lo cual permitirá desarrollar un dispositivo de fácil uso y ayudará al estudiante de educación básica a reconocer números y letras dentro de su entorno de aprendizaje.

Se proporcionará información durante el desarrollo del documento que permitirá comprender el modo de funcionamiento de dicho dispositivo.

2.3. ÁREA DEL CONOCIMIENTO

Área: Ingeniería, Industria y Construcción (07)

Subárea de Conocimiento: Ingeniería y profesiones afines (071)

Subárea Específica de Conocimiento: Electrónica y automatización (0713)

2.4. SINOPSIS DE LA PROPUESTA TECNOLÓGICA

El presente proyecto pretende dotar a docentes de educación básica, un dispositivo que permita la identificación de números y letras, para el aprendizaje.

Mediante el entrenamiento de las redes neuronales y su posterior programación en software libre, el dispositivo tendrá la facultad de detectar y diferenciar entre números y letras, permitiendo a los estudiantes contar con tecnología de uso didáctico para su aprendizaje. Por lo tanto, los estudiantes serán los beneficiarios directos, los cuales podrán adquirir sus respectivos conocimientos con una herramienta tecnológica distinta a la convencional.

2.5. OBJETO DE ESTUDIO Y CAMPO DE ACCIÓN

2.5.1. Objeto de estudio

Dispositivo de identificación de números y letras.

2.5.2. Campo de acción

Electrónica, Programación, Sistemas de control, Instrumentación y Microcontroladores.

2.6. SITUACIÓN PROBLÉMICA Y PROBLEMA

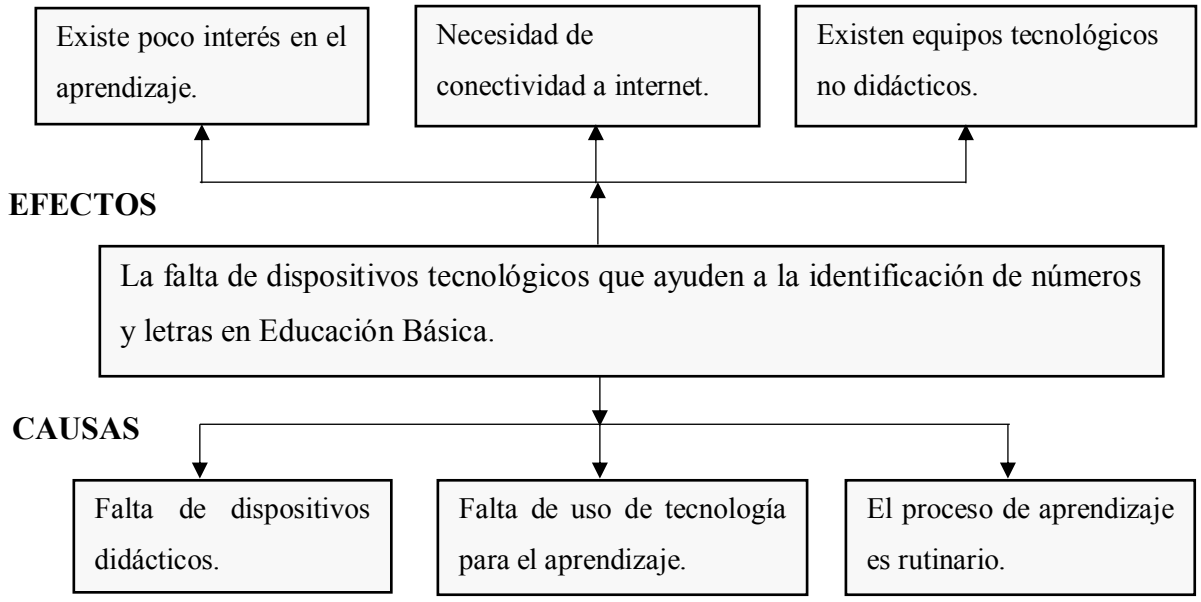
2.6.1. Situación Problémica

Los estudiantes de educación básica, no cuentan con un dispositivo didáctico que permita detectar números y letras para su aprendizaje, por lo cual, es necesario el uso de ciertos métodos tecnológicos de enseñanza que permitan a los docentes demostrar varias características y diferencias de números o letras y que los estudiantes reconozcan con mayor facilidad estos parámetros.

Este dispositivo constituye parte fundamental para el desarrollo de conocimientos básicos para los niños.

En la actualidad existen dispositivos de visión artificial, tanto en el ámbito industrial como en el ámbito educativo, los cuales han mejorado el trabajo y la eficiencia del aprendizaje, por lo cual es necesario definir tipos, diseño y dimensionamiento total en la ejecución de este dispositivo.

2.6.2. Matriz Causa - Efecto



2.6.3. Problema

La falta de dispositivos tecnológicos que ayuden a la identificación de números y letras en Educación Básica, hace que nazca la necesidad de utilizar nuevos métodos de aprendizaje empleando tecnología libre, que permitan a los estudiantes, obtener conocimientos fundamentales para su desarrollo estudiantil.

2.7. HIPÓTESIS O FORMULACIÓN DE LA PREGUNTA CIENTÍFICA

Con el desarrollo del dispositivo se podrá identificar números, letras mayúsculas y minúsculas a fin de que los estudiantes de educación básica puedan adquirir conocimientos de manera tecnológica e innovadora en tiempo real.

2.8. OBJETIVOS

2.8.1. Objetivo general

Desarrollar un dispositivo de identificación de números y letras empleando hardware y software libre, orientado a la enseñanza de niños de educación básica.

2.8.2. Objetivos específicos

- Investigar sobre sistemas de detección de objetos empleando visión artificial.
- Analizar las ventajas que presentan tanto el hardware como el software libre utilizado para el desarrollo de proyectos.
- Diseñar el sistema para la identificación de números y letras empleando visión artificial.
- Validar el correcto funcionamiento del dispositivo para la identificación de números y letras empleando visión artificial.

2.9. DESCRIPCIÓN DE LAS ACTIVIDADES Y TAREAS PROPUESTAS CON LOS OBJETIVOS ESTABLECIDOS

Tabla 2.1 Sistema de tareas por objetivos.

Objetivos	Actividades	Resultados	Medio de verificación
Investigar sobre sistemas de detección de objetos empleando visión artificial.	Investigación en fuentes bibliográficas sobre visión artificial.	Identificación de las condiciones necesarias para la detección de objetos.	Libros y artículos científicos.
	Análisis de los instrumentos y requerimientos para la detección de objetos empleando visión artificial.	Análisis de los métodos de recepción de información en dispositivos de detección.	Tesis, estudios relevantes sobre visión artificial, artículos científicos.
Analizar las ventajas que presentan tanto el hardware como el software libre utilizado para el desarrollo de proyectos.	Identificación del tipo de software que utilizan los sistemas de visión artificial.	Análisis de requerimientos de software necesarios para la detección de objetos y las librerías disponibles que ofrece Python para este tipo de dispositivos.	Consulta en catálogos y artículos científicos referente a hardware y software libre.
	Evaluación del tipo de hardware utilizado para dispositivos de detección.	Estudio de la funcionalidad del hardware en el proyecto.	Tesis o proyectos de detección de objetos con hardware libre.

Diseñar el sistema para la identificación de números y letras empleando visión artificial.	Aplicación de librerías de software libre sobre el hardware de Raspberry Pi.	Análisis de las características y funciones de las capas de entrada, ocultas y salidas.	Artículos científicos, paginas oficiales de descarga de software libre.
	Entrenamiento de las redes neuronales.	Entrenamiento de letras mayúsculas, minúsculas y números.	Librerías preinstaladas en la tarjeta de desarrollo como Python y TensorFlow
	Programación de las ponderaciones o pesos, de las redes entrenadas.	Importación de los pesos obtenidos por el entrenamiento de las redes neuronales en el mismo software.	Uso de Python, TensorFlow más funciones de activación.
Validar el correcto funcionamiento del dispositivo para la identificación de números y letras empleando visión artificial.	Verificación de la funcionalidad del hardware y software libre.	Correcto funcionamiento del dispositivo detección de números y letras.	Videos del funcionamiento del dispositivo

Fuente: Autores

3. MARCO TEÓRICO

3.1. Antecedentes.

Se han desarrollado trabajos similares de acuerdo al mencionado proyecto de titulación, los mismos se indican a continuación:

[3] "Monitoreo de variables analógicas usando Raspberry PI." En el desarrollo de este proyecto se implementó un módulo de monitoreo de variables analógicas, usando como modulo central la tarjeta Raspberry Pi, usando el sistema operativo Raspbian.

[4] "Diseño e implementación de sistema interactivo de información de Docentes, con Raspberry PI." El desarrollo de este proyecto se basa en un sistema que permite tener información de los horarios de clase de un docente universitario y el aula donde se da la materia, utilizando una tarjeta Raspberry y una pantalla LCD donde se aprecia la información del proyecto.

[1] "Sistema de reconocimiento facial basado en redes neuronales convolucionales sobre el dispositivo Raspberry Pi." En el proyecto se realiza el entrenamiento e implementación de redes neuronales utilizando el lenguaje Python y TensorFlow en una tarjeta Raspberry Pi para el reconocimiento facial.

[2] "Módulo didáctico de entrenamiento de redes neuronales para el reconocimiento de patrones de imágenes y voz con Raspberry Pi". El trabajo que se detalla se fundamenta en la necesidad de la enseñanza y el aprendizaje práctico por parte del profesor y el alumno, esto se realizó con el modelamiento de la red neuronal y el enlace con el medio externo.

[5] "Reconocimiento de imágenes y datos financieros en Streaming." El desarrollo de este proyecto está basado en redes neuronales y técnicas de Machine Learning, en una tarjeta Raspberry Pi 3, lo cual permite la clasificación de objetos mediante técnica de reconocimiento de imágenes y la predicción de valores futuros de datos financieros.

[6] "Detección de personal no autorizado en el departamento de TI utilizando redes neuronales convolucionales en tiempo real con Raspberry pi 3 b+." El proyecto detalla un sistema de reconocimiento facial y envió de alertas mediante correo electrónico, utilizando las librerías TensorFlow, Numpy y OpenCV, además de una Raspberry y una cámara Pi. El sistema estudia y aprende los rostros captados y los compara con los almacenados en una base de datos y da una alerta cuando detecta un rostro no encontrado.

3.2. Visión artificial

La visión artificial es un campo de la inteligencia artificial cuyo objetivo es programar un computador para que "entienda" una escena o las características de una imagen. Esta información es empleada para tomar decisiones o a su vez controlar un proceso. Aunque parece un término de moda al que únicamente hacen referencia los expertos, muchas personas ya lo utilizan en los lectores de huellas digitales o códigos de barras, Smartphone, redes sociales como Facebook, etc. En la actualidad, podemos encontrar sistemas de visión artificial implementados en diversos campos como [7]:

- **La industria:** automatización, metrología laser, identificación, cuantificación, manipulación, estudio de materiales, electroluminiscencia, corte automático, control de calidad.
- **Entretenimiento:** Kinect, Realidad virtual, Realidad aumentada, videoconsolas.
- **Lectura de códigos:** letras, números, código de barras, OCR, OCV.
- **Medicina:** imágenes diagnósticas, medicina forense, Escáner 3D, cirugía robotizada, ortopedia, tomografía axial computarizada, RX, resonancia magnética, medicina nuclear, Angiografía, endoscopía, retina artificial, industria farmacéutica.
- **Biometría:** reconocimiento de caras, posturas 3D, firmas, huellas, retina, iris, movimiento de los labios, deformación.
- **Robótica:** guiado de robots, control de brazos móviles, sensores, ensamblado de piezas, interacción sin mandos.
- **Inteligencia Artificial:** control de sistemas, planificación automática, reconocimiento de escritura, reconocimiento del habla, reconocimiento de patrones.
- **Psicología:** Neurociencia, aprendizaje, redes neuronales.
- **Biología:** Microscopía de fluorescencia, conteo de microorganismos y células, identificación de propiedades como el color, forma, tamaño, textura, reconocimiento de hojas, plantas, grado de floración, patrones.
- **Seguridad:** Telepresencia, radar, aplicaciones militares, vigilancia, seguimiento de actividades humanas, objetos abandonados.
- **Tráfico Vehicular:** Vehículos autónomos, Drones Semáforos inteligentes, control de tráfico, control de velocidad, identificación automática de placas y de señales de tránsito.
- **Teledetección:** análisis multiespectral, geología, fotogrametría, cartografía.

- **Astronomía:** misiones espaciales, detección de rayos gamma, radiotelescopios, Tracking.

En todos estos ámbitos, la visión artificial, no sólo tiene un futuro prometedor sino un campo de aplicación exponencial a medida que surjan nuevos servicios y demandas. Esta multitud de usos y áreas de investigación demanda profesionales que posean conocimientos en visión artificial [7].

La visión artificial abarca todas las aplicaciones industriales y no industriales en las que una combinación de hardware y software brinda un guiado operativo a los dispositivos en la ejecución de sus funciones de acuerdo con la captación y procesamiento de imágenes. La visión artificial aplicada a la industria utiliza los mismos algoritmos y enfoques que las aplicaciones académicas y gubernamentales de visión artificial, aunque con diferentes limitaciones [8].

Los sistemas de visión artificial pueden realizar medidas objetivas, como determinar la separación de los electrodos de una bujía u ofrecer información de ubicación que guíe a un robot para alinear piezas en un proceso de fabricación. [8].

3.2.1. Etapas

- **Escenario a analizar:** Es el área que queremos capturar, donde se encuentra la información que buscamos procesar [9].
- **Adquisición y digitalización:** Es el proceso de capturar una imagen y pasarla a algún formato digital, se utiliza una cámara para capturar la escena y después se envía a una unidad donde pueda ser procesada [9].
- **Procesamiento previo:** La cámara normalmente captura ruido, por lo que se necesita de un preprocesamiento para eliminar dicho ruido de la imagen mediante una gran variedad de filtros. En ocasiones también deben hacerse transformaciones geométricas como recortes o rotaciones [9].
- **Obtención de características:** En esta etapa se resalta las características de la imagen que son de nuestro interés. Las operaciones que comúnmente se realizan aquí son: detección de esquinas, colores, flujo óptico y formas, realce de bordes, entre otros [9].

  Etapa en la que se analiza el uso de un software con las características necesarias para ejecutar la programación del proyecto.
- **Reconocimiento e interpretación de información:** Se procesa la información obtenida en la etapa previa, se le da una interpretación, y se aplica una acción de acuerdo a lo analizado; dicha acción será la que controle a la aplicación [9].

- **Aplicación:** Es el objeto final que se controlará de acuerdo a la información que se procesó. Podría controlarse un brazo robótico, motores, drones, etc. [9].

3.2.2. Reconocimiento de objetos

Se requiere un dispositivo que provea imágenes del mundo real para procesarlas. Una vez instaladas las librarías necesarias para el funcionamiento del sensor óptico, se comienza a obtener imágenes del mismo [10].

La imagen del mundo real que provee el sensor óptico tiene tres canales de colores, Rojo, Verde y Azul (RGB del inglés Red, Green, Blue), cada canal tiene valores entre 0 y 255, el proceso de digitalización consiste pasar los valores RGB de la imagen de entrada a una matriz tridimensional de valores RGB [10].

3.2.2.1. Escala de Grises

Para procesar una imagen y reconocer sus patrones en la misma, se necesita que tenga un solo canal de color, es decir a escala de grises, el cual consiste en sumar los tres canales y dividirlo entre 3, para obtener el promedio del RGB [10].

Posteriormente se obtiene el histograma de la imagen en escala de grises. El histograma se debe calcular para obtener un umbralado automático de la imagen. Todos los métodos para umbralar imágenes, necesitan un parámetro entero, un límite y a partir de ese valor se asignarán valores según el caso, 255 (blanco) o 0 (negro) dependiendo. El valor entero lo proporciona el histograma, el cual no es más que la contabilización de los valores de cada pixel de la imagen, por ejemplo: Una imagen con resolución 640×480 tiene 307200 pixeles, el histograma cosiste en contar cuantos pixeles tiene con valor 0, cuantos con valor 1 y así sucesivamente hasta 255 [10].

Figura 3.1. Imagen referencial de RGB a escala de grises.

Fuente: [10]

En visión artificial, el histograma no solo se usa para umbralar (proceso para generar una imagen a escala de grises con valores de 0 y 255) una imagen ni mucho menos para mostrar gráficas, otra de su principal función es para obtener rasgos descriptivos de una imagen y guardar esas características en una base de conocimiento, a ese proceso se le conoce como entrenamiento de una máquina de aprendizaje [10].

3.2.2.2. Binarización

La binarización consiste en tomar el valor que más se repitió del histograma, es decir la moda y a partir de él todos los valores superiores se establecerán como pixeles blancos (255) y los menores como negros (0) [10].

3.2.2.3. Desenfoque Gaussiano

El siguiente paso es hacer un desenfoque para eliminar el ruido de la imagen. El desenfoque Gaussiano es un efecto de suavizado para mapas de bits, pero en visión artificial es un filtro que sirve para reducir zonas parasitas (ruido) a fin de que las regiones queden más definidas. Una región es un mapa de bits que contiene el mismo valor, es decir que están conectados sin ningún obstáculo (celdas con otro valor), un ejemplo de región y la más grande, es la mano, ya que el conjunto de valores 0 (blancos) están conectados por derecha, izquierda, arriba, abajo y diagonal hasta formar la misma [10].

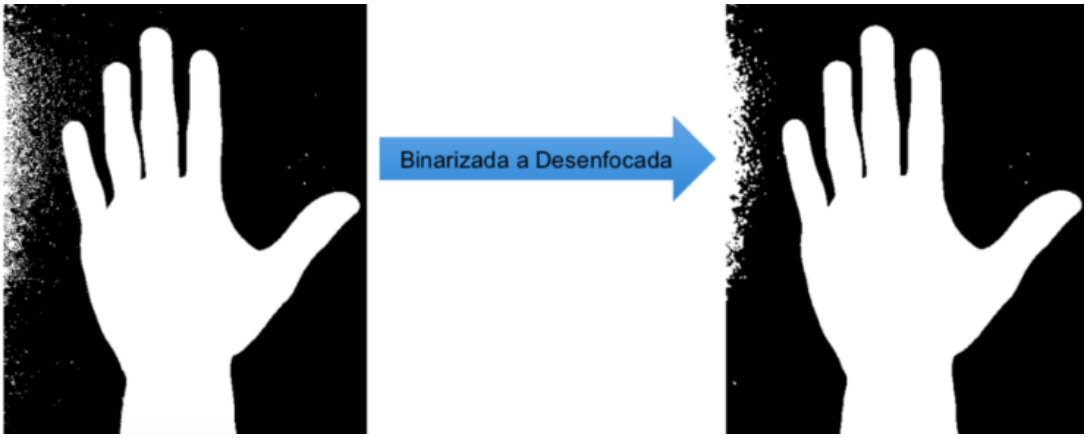


Figura 3.2. Desenfoque de una imagen binarizada.

Fuente: [10]

Como se observa en la imagen de la derecha los puntos pequeños blancos disminuyen, haciendo las regiones más grandes y definidas.

3.2.2.4. Erosión

Es una de las técnicas de lo morfología matemática que sirve para disminuir zonas parasitas se la puede definir de la siguiente manera:

Dada una imagen A, y un elemento estructural B, (ambas imágenes binarias con fondo blanco), la erosión de una imagen, A, por un elemento estructural, B, es el conjunto de todos los elementos (x) para los cuales B trasladado por (x) está contenido en A [10].

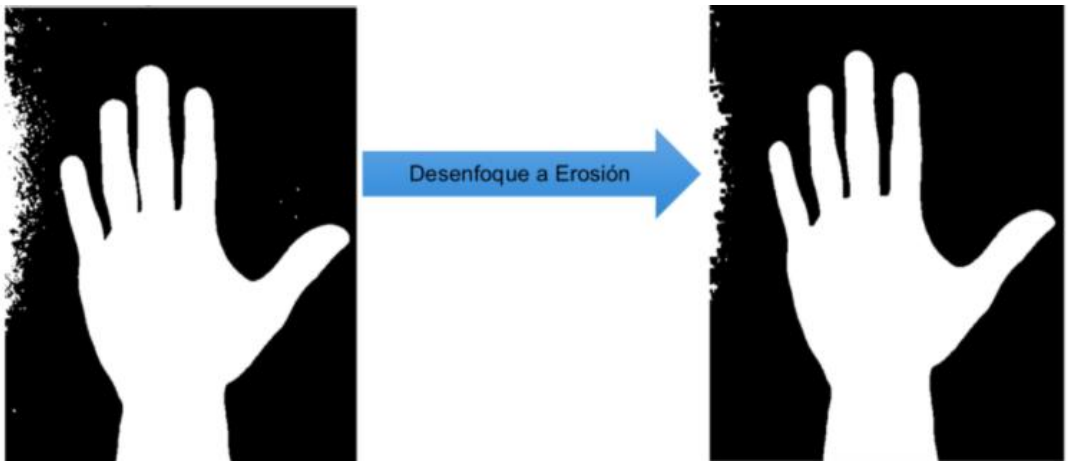


Figura 3.3. Proceso de Erosión a una imagen.

Fuente: [10]

La Erosión consiste en hacer decrecer en tamaño el conjunto de puntos o regiones originales. Como en cualquier otra operación de Morfología Matemática, la reducción es en función del elemento estructural utilizado [10].

3.2.2.5. Dilatación

Este proceso se ocupa para expandir las zonas blancas dentro de la imagen, en este caso es para hacer más grande el objeto que se desea detectar, pero la desventaja de usarlo es que también crecen las zonas parasitas, lo cual se puede solucionar con la detección de contornos [10].

3.2.2.6. Detección de Contornos

La detección de Contornos puede explicarse cómo una curva que une todos los puntos continuos a lo largo de la frontera, que tiene el mismo color o intensidad. Los contornos son una herramienta útil para el análisis de la forma y la detección de objetos y reconocimiento. Para una mayor precisión, se utilizan imágenes binarias. La búsqueda de contornos es cómo encontrar un objeto blanco de fondo negro. En otras palabras, se hace la sumatoria de los pixeles blancos para obtener el área de cada región y posteriormente extraer una matriz de puntos que equivalen a su perímetro, es decir todo el borde o contorno de cada región [10].

Figura 3.4. Proceso de dilatación al proceso de detección de contornos.

Fuente: [10]

3.2.3. Dispositivos utilizados para la detección de objetos

Para diseñar un sistema de visión artificial se debe definir el tipo de cámara a utilizar, esta elección va a depender de múltiples factores que se detallan a continuación:

- Cámara de color o monocromática: Si se utiliza una cámara de color o de escala de grises. Generalmente, se elegirá dependiendo del proceso que vayamos a realizar. La mayoría de los casos en los que se utiliza un sistema de visión artificial la adquisición se realiza mediante una cámara monocromática, siendo las ventajas principales económicas y de cómputo. Hay que darse cuenta que una imagen en color necesita mucho más tiempo de procesado y para procesos en tiempo real la velocidad de cálculo es un parámetro importante a considerar [11].
- La mayoría de los procesos se basan en el reconocimiento de formas a partir de la determinación de los contornos de los objetos, siendo innecesaria la información cromática de los objetos. Existen aplicaciones en las que la información de color es fundamental, por ejemplo, en control de calidad en la industria alimentaria donde se clasifica la fruta, verdura, etc.; según el color de cada elemento; o por ejemplo en la determinación del tipo de acero fundido o su temperatura según el color que irradia, o en piezas de diferentes colores, etc [11]. La imagen en color realmente proporciona una información más completa que puede servir en la etapa de segmentación y extracción de parámetros en detrimento de una mayor necesidad de tiempo de cómputo [11].

3.3. Redes neuronales

En la actualidad existen varias formas de definir a las redes neuronales artificiales que van desde definiciones cortas hasta generales y también las que intentan explicar más explícitamente qué son y que hacen las redes neuronales [2].

a) Son una nueva forma de tecnología, inspirada en modelos biológicos.

b) Es un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles y niveles ocultos.

c) Un sistema de control compuesto por un gran número de elementos simples y elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas al sistema [2].

d) Son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico [2].

Las estructuras básicas de redes neuronales artificiales se basan en el perceptrón diseñado por Rosenblatt en 1959. Esta neurona artificial o perceptrón, puede percibir variables de entradas (y), mediante su suma ponderada y una función de activación posterior, nos proporciona una salida que queremos que se aproxime a la deseada [1].

$$y = f(\sum(x_n w_n + b)) \tag{3.1}$$

Donde:

y = Salida o resultado de la función.

x_n = Entradas.

w_n = Pesos

f = Función de activación.

b = Bias

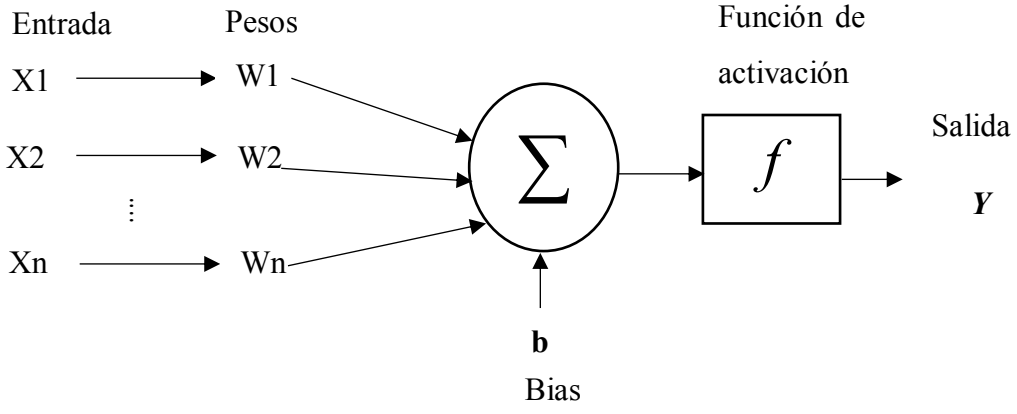


Figura 3.5. Modelo de neurona artificial.

Fuente: [1]

La precisión del perceptrón es limitada ya que solo permite clasificar problemas linealmente separables.

Posteriormente como una mejora del perceptrón, surgió el perceptrón multicapa diseñado en 1986 por Rumelhard y otros autores. Esta estructura presenta una mejora ya que permite la clasificación de problemas que no son linealmente separables a diferencia del perceptrón simple. Se basa en añadir capas intermedias entre la capa de entrada y la capa de salida a la estructura, denominadas capas ocultas con diferentes tipos con funciones de activación [1].

De esta forma se diseña una red de varias capas con nodos interconectados capaces de procesar la información que se les proporciona y devolviendo un resultado a la capa siguiente, hasta llegar a la última capa que devuelve un resultado basado en todo este proceso. El resultado que se obtiene en cada capa depende de las denominadas funciones de activación [1].

Normalmente las redes neuronales cuentan con más de una neurona por capa y con varias capas como el caso del perceptrón multicapa antes mencionado. Estas neuronas pueden tener relación entre ellas o solo con las neuronas de capas próximas y anteriores, dependiendo del tipo de capa en la que nos encontremos. Cada una de estas relaciones es decir los pesos entre las neuronas y los Bías son los parámetros que se entrenan en la red [1].

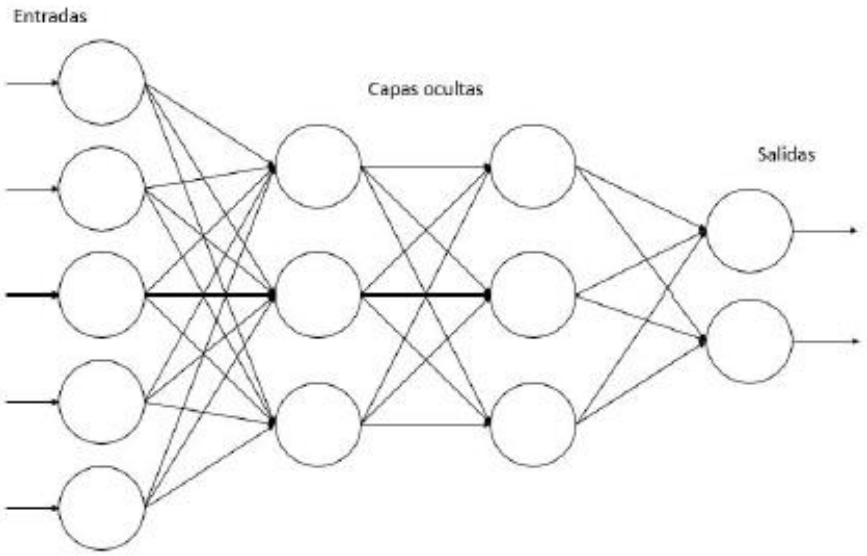


Figura 3.6. Modelo con varias capas de neuronas artificiales.

Fuente: [1]

Durante el proceso de aprendizaje, los valores de los pesos y el bias son corregidos de forma iterativa con el objetivo de, tras suficientes iteraciones, conseguir que el resultado obtenido sea el correcto para una tarea en concreto [1].

Uno de los procesos más populares para realizar esta operación es el llamado backpropagation, este método consiste en calcular las salidas de cada capa hasta llegar a la capa final o también llamada (etapa fordward), donde se obtiene la salida que se compara con el resultado deseado

para calcular el error e intentar minimizarlo mediante el proceso de backpropagation, el que se busca el descenso en gradiente de la función de coste o error [1].

La función de activación también tiene un papel fundamental, por lo que hay varias funciones entre las que se pueden elegir para intentar mejorar nuestro modelo como, por ejemplo: Relu o Leaky Relu. [1]

Tabla 3.1. Ventajas de las redes neuronales.

TIPO	CARACTERÍSTICAS
Aprendizaje adaptativo	• Capacidad de aprender tareas basadas en una experiencia inicial. • Capacidad de autoajuste y adaptables. • Siguen aprendiendo luego del entrenamiento. • Capacidad de discriminar patrones de aprendizaje mal aprendidos.
Auto - Organización	• Representan la información que recibe por etapas de entrenamiento. • Puede modificar la red neuronal completa para llegar al objetivo deseado. • Puede organizar la información para poder obtener información adecuada.
Tolerancia a fallos	• Las redes pueden aprender a reconocer patrones con ruido, distorsiones o incompletos. • Pueden almacenar datos a pesar de los errores del sistema, porque existe redundancia de datos en las conexiones de la red. • Almacenan información no localizada. • Operación en tiempo real. • Inserción dentro de tecnologías ya existentes.

Fuente: [2]

3.3.1. Tipos de aprendizaje

Los datos de entrada se procesan a través de la red neuronal con el propósito de lograr una salida. También se dice que las redes neuronales extraen generalizaciones desde un conjunto determinado de ejemplos anteriores de tales problemas de decisión. Una red neuronal debe

aprender a calcular la salida correcta para cada constelación arreglo o vector de entrada en el conjunto de ejemplos. Este proceso de aprendizaje se denomina: proceso de entrenamiento o acondicionamiento. El conjunto de datos o conjunto de ejemplos sobre el cual este proceso se basa es, por ende, llamado conjunto de datos de entrenamiento [2].

En otras palabras, el aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el mismo se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. En los sistemas biológicos existe una continua destrucción y creación de conexiones entre las neuronas. En los modelos de redes neuronales artificiales, la creación de una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de cero. De la misma manera, una conexión se destruye cuando su peso pasa a ser cero [2].

Un aspecto importante respecto al aprendizaje de las redes neuronales es el conocer cómo se modifican los valores de los pesos, es decir, cuáles son los criterios que se siguen para cambiar el valor asignado a las conexiones cuando se pretende que la red aprenda una nueva información [2].

Hay dos métodos de aprendizaje importantes que pueden distinguirse:

a) Aprendizaje supervisado.

b) Aprendizaje no supervisado.

Otro criterio que se puede utilizar para diferenciar las reglas de aprendizaje se basa en considerar si la red puede aprender durante su funcionamiento habitual o si el aprendizaje supone la desconexión de la red, es decir, su inhabilitación hasta que el proceso termine [2]. En el primer caso, se trataría de un aprendizaje (on line), mientras que el segundo es lo que se conoce como off line. Cuando el aprendizaje es off line, se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de entrenamiento y un conjunto de datos de test o prueba, que serán utilizados en la correspondiente fase. Además, los pesos de las conexiones permanecen fijos después que termina la etapa de entrenamiento de la red. Debido precisamente a su carácter estático, estos sistemas no presentan problemas de estabilidad en su funcionamiento [2].

Una generalización de la fórmula o regla para decir los cambios en los pesos es la siguiente:

Peso Nuevo = Peso Viejo + Cambio de Peso

Matemáticamente es:

$$wt + 1 = wt - \Delta w(t) \tag{3.2}$$

Donde:

t = hace referencia a la etapa de aprendizaje

Wij (t+1) = al peso nuevo

Wij (t) = al peso viejo

3.3.1.1. Aprendizaje supervisado

El aprendizaje supervisado se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada [2].

El supervisor controla la salida de la red y en caso de que ésta no coincida con la deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la deseada [2].

En este tipo de aprendizaje se suelen considerar, a su vez, tres formas de llevarlo a cabo, que dan lugar a los siguientes aprendizajes supervisados:

- Aprendizaje por corrección de error.

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos a la salida de la red, es decir, en función del error cometido en la salida [2].

- Aprendizaje por refuerzo.

Se trata de un aprendizaje supervisado, más lento que el anterior, que se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado, es decir, de no indicar durante el entrenamiento exactamente la salida que se desea que proporcione la red ante una determinada entrada [2].

- Aprendizaje estocástico.

Consiste básicamente en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad. [2]

3.3.1.2. Aprendizaje no supervisado

Conocido como auto supervisado no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas, la red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta [2].

En estas redes se deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presenten en su entrada. Existen algunas posibilidades en cuanto a la interpretación de la salida, que dependerían de su estructura y del algoritmo de aprendizaje utilizado.

En cuanto a los algoritmos de aprendizaje no supervisado, en general se suelen considerar dos tipos, que dan lugar a los siguientes aprendizajes [2]:

- Aprendizaje hebbiano.

Esta regla de aprendizaje es la base de muchas otras, la cual pretende medir la familiaridad o extraer características de los datos de entrada. El fundamento es una suposición bastante simple: si dos neuronas Ni y Nj toman el mismo estado simultáneamente activas o inactivas, el peso de la conexión entre ambas se incrementa [2].

- Aprendizaje competitivo y comparativo.

Se orienta a la (clusterización) o clasificación de los datos de entrada. Como característica principal de este tipo de aprendizaje se puede decir que, si un patrón nuevo se determina que pertenece a una clase reconocida previamente, entonces la inclusión de este nuevo patrón a esta clase matizará la representación de la misma. Si el patrón de entrada se determinó que no pertenece a ninguna de las clases reconocidas anteriormente, entonces la estructura y los pesos de la red neuronal serán ajustados [2].

3.3.2. Estructura de un sistema neuronal artificial

Las redes neuronales son modelos matemáticos que intentan reproducir el comportamiento del cerebro humano. El principal objetivo de este modelo es la construcción de sistemas capaces de presentar un cierto comportamiento inteligente. Esto implica la capacidad de aprender a realizar una determinada tarea. [2]

La neurona artificial es un elemento de procesamiento simple que a partir de un vector de entradas produce una única salida. En general podemos encontrar tres tipos de neuronas artificiales, donde cada una de las cuales tiene su contraparte en el sistema nervioso: [2]

- Las que reciben información desde el exterior, a las cuales se las denomina neuronas de entrada.
- Las que reciben información desde otras neuronas artificiales, a las cuales se las denomina neuronas ocultas. Es en estas neuronas, es donde se realiza la representación de la información almacenada.
- Las que reciben la información procesada y las devuelven al exterior, se las denomina neuronas de salida.

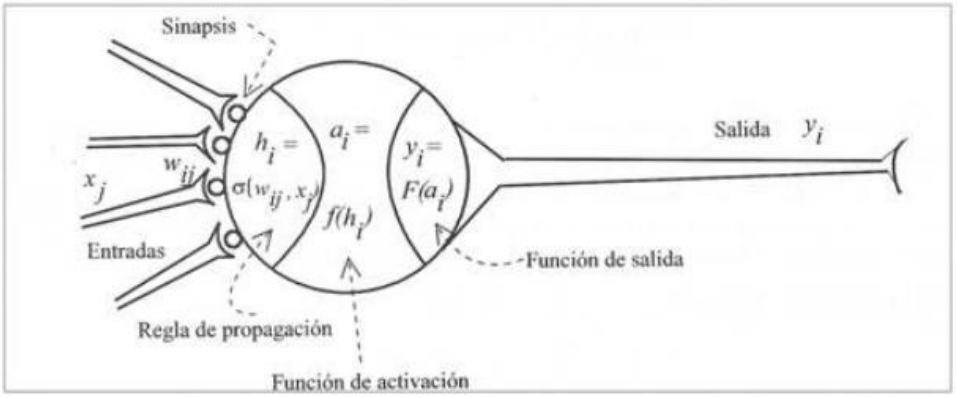


Figura 3.7. Neurona artificial.

Fuente: [2]

- Conjunto de entradas, xj (t). Estas pueden ser provenientes del exterior o de otras neuronas artificiales.
- Pesos sinápticos, wij. Representan el grado de comunicación entre la neurona artificial j y la neurona artificial i. Pueden ser excitadores o inhibidores.
- Regla de propagación, σi (wij, xj(t)). Integra la información proveniente de las distintas neuronas artificiales y proporciona el valor del potencial post sináptico de la neurona i.
- Función de activación, fi (ai(t-1), hi(t)). Provee el estado desactivación actual de la neurona (i).

De esta forma, la salida producida por una neurona i, para un determinado instante de tiempo (t) puede ser escrita en forma general de la siguiente manera [2]:

Función de salida, Fi (ai(t)).

$$yt = F(fat - 1, \sigma(w, x(t))) \tag{3.3}$$

Donde:

F(at)= Representa la salida de la actual neurona i

σ(w,x(t))= Integra la información proveniente de distintas neuronas.

3.3.2.1. Entradas y salidas

Las entradas y salidas de una neurona se clasifican en dos grupos las cuales pueden ser, binarias o continuas.

Las neuronas binarias (digitales) sólo admiten dos valores posibles. En general en este tipo de neurona se utilizan los siguientes dos alfabetos {0,1} o {-1,1} [2].

Las neuronas continuas (analógicas) admiten valores dentro de un determinado rango, que en general suele definirse como [-1, 1] [2].

La selección del tipo de neurona que se requiera utilizar depende de la aplicación y del modelo a construir [2].

3.3.2.2. Pesos sinápticos

El peso sináptico Wij define la fuerza de una conexión sináptica entre dos neuronas, la neurona pre sináptica i y la neurona pos sináptica j. Los pesos sinápticos pueden tomar valores positivos, negativos o cero. En caso de una entrada positiva, un peso positivo actúa mientras que un peso negativo actúa como inhibidor. En caso de que el peso sea cero, no existe comunicación entre el par de neuronas. Mediante el ajuste de los pesos sinápticos la red es capaz de adaptarse a cualquier entorno y realizar una determinada tarea [12]:

3.3.2.3. Regla de Propagación

Este método sirve para que una red neuronal aprenda una asociación que existere entre sus patrones de entrada y las clases correspondientes.

Conocido como backpropagation, propagación del error hacia atrás o retro propagación, y está basado en la regla de aprendizaje que es posible aplicar solo a modelos de redes multicapa. Una característica importante de este algoritmo es la representación interna del conocimiento que es capaz de organizar en la capa o capas intermedias, para conseguir cualquier correspondencia entre la entrada y la salida de la red [12].

3.3.2.4. Función de activación

La función de activación, es el estado de activación que presenta actualmente la neurona en base al potencial resultante (hi) y al estado de activación anterior de la neurona ai (t-1). El estado de activación de la neurona para un determinado instante de tiempo (t) es expresado de la siguiente manera [2].

$$ai(t) = fi(ai(t-1), hj(t)) \tag{3.4}$$

Donde:

ai(t-1) = Estado de activación anterior de la neurona.

hi = Potencial resultante.

t = Tiempo.

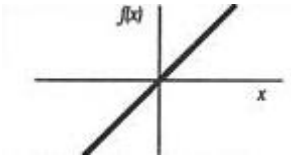
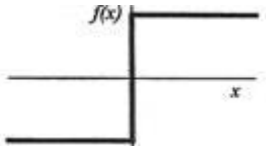
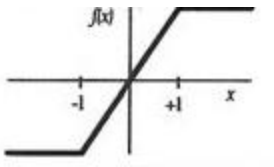
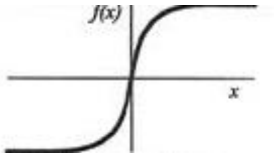
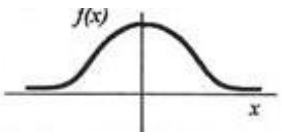
Sin embargo, en la mayoría de los modelos se suele ignorar el estado anterior de la neurona, definiéndose el estado de activación en función del potencial resultante **hi** [2]:

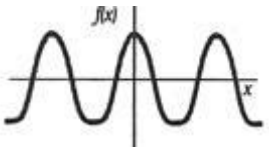
$$ai(t) = fi(hj(t)) \tag{3.5}$$

Los valores de entrada se multiplican por los pesos anteriormente ingresados a la neurona, por consiguiente, los pesos que generalmente no están restringidos cambian la medida de influencia que tienen los valores de entrada. Es decir, que permiten que un gran valor de entrada tenga solamente una pequeña influencia, si estos son lo suficientemente pequeños [2].

Para realizar el entrenamiento de las redes neuronales se pueden utilizar las funciones de activación de los distintos modelos de redes neuronales, en la siguiente tabla se detalla los distintos tipos de funciones:

Tabla 3.2. Funciones de activación de una red neuronal.

	Función	**Rango**	**Gráfico**
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = h(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{Bmatrix} -1, si\ x < -l \\ x, si + l \leq x \leq -l \\ +1, six > +l \end{Bmatrix}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	

Sinusoidal	$y = Asen(wx + \varphi)$	$[-1, +1]$	

Fuente: [13].

Para explicar de mejor manera el por qué se utilizan las funciones de activación se suele emplear la analogía a la aceleración de un automóvil; cuando un automóvil inicia su movimiento necesita una potencia elevada para comenzar a acelerar; pero al ir tomando velocidad, esta demanda un menor incremento de dicha potencia para mantener la aceleración. Al llegar a altas velocidades, nuevamente un amplio incremento en la potencia es necesario para obtener una pequeña ganancia de velocidad [2].

En conclusión, en ambos extremos del rango de aceleración del automóvil se demanda una mayor potencia para la aceleración que en la mitad de dicho rango.

3.3.2.5. Función salida

El último componente que una neurona necesita es la función de salida que es el valor resultante o la salida de la neurona i (outi); donde la función de salida determina qué valor se transfiere a las neuronas vinculadas. Si la función de activación está por debajo de un umbral determinado, ninguna salida se pasa a la neurona subsiguiente. Normalmente, no cualquier valor es permitido como una entrada para una neurona, por lo tanto, los valores de salida están comprendidos en el rango [0, 1] o [-1, 1]. También pueden ser binarios {0, 1} o {- 1, 1} [2].

Esta función proporciona el valor de la salida de la neurona, en base al estado de activación de la neurona. En la mayoría de los casos la identidad que es más utilizada es la identidad, la fórmula es:

$$yi(t) = Fi(ai(t) = aj(t)) \tag{3.6}$$

Donde:

F= Salida de la actual neurona.

ai(t)= estado de activaciones anteriores.

3.3.3. Arquitectura de una red neuronal

La arquitectura de una red neuronal, se define mediante una topología para su organización y disposición de las neuronas dentro de la red neuronal, esta viene dado por: el número de capas de la red, la cantidad de neuronas en cada capa, el grado de conectividad y el tipo de conexión entre neuronas [2].

3.3.3.1. Niveles o capas de una red neuronal

La distribución de neuronas dentro de la red se realiza formando niveles o capas, con un número determinado de dichas neuronas en cada una de ellas. A partir de su situación dentro de la red, se pueden distinguir tres tipos de capas [2]:

- De entrada: Capa que recibe directamente la información proveniente de las fuentes externas a la red.
- Ocultas: Son internas a la red y no tienen contacto directo con el entorno exterior. El número de niveles ocultos puede estar entre cero y un número elevado. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas topologías de redes neuronales [2].
- De salidas: transfieren información de la red hacia el exterior.

En la Ilustración se puede observar el ejemplo de la estructura de una red multicapa, en la que cada neurona está conectada con neuronas de un nivel superior; se puede observar que hay más conexiones que neuronas en sí, se dice que una red es totalmente conectada si todas las salidas desde un nivel llegan a todos y cada uno de las neuronas del nivel siguiente [2].

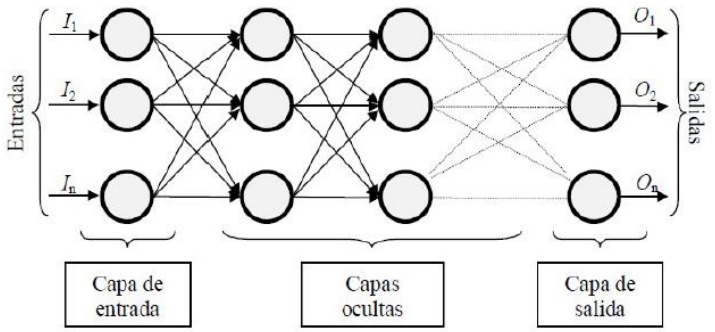


Figura 3.8. Red neuronal conectada.

Fuente: [2]

3.3.3.2. Tipos de neuronas artificiales.

Las neuronas artificiales se pueden clasificar de acuerdo a los valores que pueden tomar. Por ahora es suficiente distinguir entre dos tipos principales [2]:

• Neuronas binarias.

• Neuronas reales.

Las neuronas binarias solamente pueden tomar valores dentro del intervalo {0,1} o {-1, 1}, mientras que las neuronas reales pueden hacerlo dentro del rango [0, 1] o [-1, 1] [2].

Los pesos normalmente no están restringidos a un cierto intervalo, aunque para aplicaciones específicas puede ser esto necesario [2].

3.3.3.3. Técnicas de decisión.

El proceso de decisión puede ser caracterizado como se muestra en el diagrama.

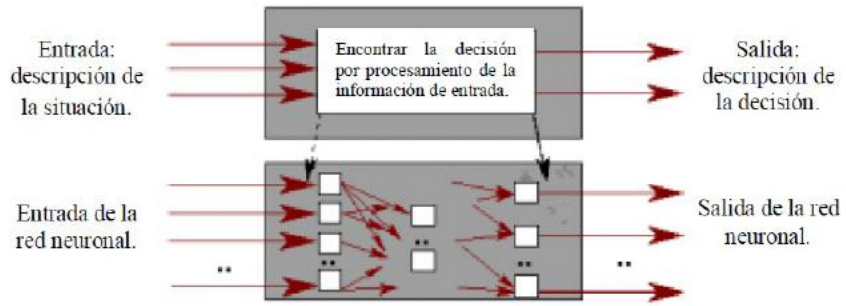


Figura 3.9. Esquema del proceso de decisión.

Fuente: [2]

3.3.3.4. Elección del conjunto inicial de pesos

Antes de empezar el proceso de entrenamiento es necesario determinar su estado inicial, lo que significa: escoger un conjunto inicial de pesos para las diversas conexiones entre las neuronas de la red neuronal. Esto puede realizarse por varios criterios; por ejemplo, uno de ellos es otorgar un peso aleatorio a cada conexión, encontrándose los mismos dentro de un cierto intervalo. Generalmente un intervalo del tipo [-n, n], donde n es un número natural positivo [13].

3.3.4. Topología de las redes neuronales

Al hablar de redes neuronales se debe definir su topología o arquitectura, que consiste en la organización y disposición de las neuronas, formando capas o subcapas de neuronas alejadas de la entrada y salida de dicha red. De acuerdo a esto los parámetros fundamentales de la red neuronal son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas [2].

3.3.4.1. Redes mono capa

Se establecen conexiones entre las neuronas que pertenecen a la única capa que constituye la red. Las redes mono capa se utilizan generalmente en tareas relacionadas con lo que se conoce como auto asociación, regenerar información de entrada que se presenta a la red de forma incompleta o distorsionada [2].

3.3.4.2. Redes multicapa

Son aquellas redes que disponen de un conjunto de neuronas agrupadas en varios niveles o capas. Una forma para distinguir la capa a la que pertenece una neurona, consistiría en fijarse en el origen de las señales que recibe a la entrada y el destino de la señal de salida. Normalmente, todas las neuronas de una capa reciben señales de entrada desde otra capa

anterior la cual se encuentra más cerca de la entrada de la red y envían señales de salida a una capa posterior que es la que se encuentra más cerca de la salida de la red. A estas conexiones se las denomina conexiones hacia adelante o feedforward [13].

En un gran número de estas redes también existe la posibilidad de conectar la salida de las neuronas de capas posteriores a la entrada de capas anteriores; a estas conexiones se las denomina conexiones hacia atrás o feedback [2].

3.3.4.3. Conexión entre neuronas.

La conectividad entre los nodos de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo o conexión auto recurrente [2].

Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles precedentes, la red se describe como de conexión hacia adelante Cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de conexión hacia atrás [2].

3.3.5. Redes neuronales convolucionales

Son un tipo de red neuronal artificial multicapa (feedfordward) en español, adelantar hacia adelante, especialmente diseñada para trabajar con imágenes. Estas neuronas trabajan de forma similar a las neuronas reales que podemos encontrar en la corteza visual primaria de un cerebro biológico [1].

Las imágenes no se perciben de igual forma por un ordenador o por un humano, aunque podamos decir que este tipo de red neuronal funciona de forma similar, ya que el ordenador percibe las imágenes como conjunto de matrices bidimensionales con valores relativos a la imagen en cada punto, es decir, pixeles [1].

Las redes neuronales convolucionales se componen de múltiples capas de filtros convolucionales y filtros de reducción de datos (average poolin, max pooling, etc). Después de cada capa convolucional, se añade una función para realizar un mapeo causal no-lineal. Hay distintas funciones, al igual que en las redes neuronales tradicionales, entre las que se debe elegir con el fin de obtener el mejor resultado posible [1].

Según se avanza en estas capas / filtros se suele reducir el tamaño de las imágenes o matrices de salida de los filtros a la vez que se aumentan el número de capas, es decir, si partimos de una imagen RGB por norma general en cada filtro se irán reduciendo las dimensiones y aumentando el número de capas [1]:

$$3(capas\ RGB)x150x150 \rightarrow 16x100x100 \rightarrow 32x85x85 \quad (3.7)$$

En el último paso en estas redes de neuronas artificiales, se encuentran neuronas de perceptrón o (fully connected) para realizar la clasificación final sobre las características extraídas mediante los filtrados previos [1].

Debido a las convoluciones que se realizan en estas redes neuronales, este tipo de red neuronal es apta para poder clasificar todo tipo de datos donde estos estén distribuidos de forma continua en el mapa o imagen de entrada, y a su vez sean estadísticamente parecidos a lo largo de la imagen de entrada. Esto se debe a su invarianza a traslaciones dentro de la imagen, es decir, el resultado no se basa en la posición exacta de la característica que estudia, ya que subdivide la imagen en pequeños conjuntos que estudia de forma separada, permitiendo así obtener el resultado deseado pese a posibles traslaciones de la región de interés dentro de la imagen, siempre que se haya entrenado adecuadamente. Por esta razón, son especialmente eficaces para clasificar imágenes, así como tareas de percepción por computador [1].

3.3.5.1. Capa convolucional

Proceso en el que se realiza la convolución del filtro con la imagen de entrada de ese filtro. La convolución consiste en una serie de multiplicaciones y sumas de los parámetros del filtro convolucional y los parámetros de las matrices bidimensionales o imágenes [1].

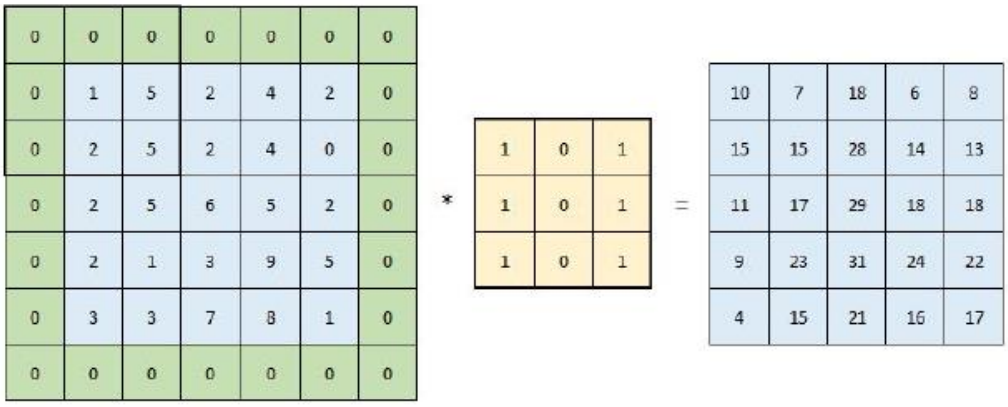


Figura 3.10. Proceso de convolución.

Fuente: [1]

En este proceso se puede elegir si se quiere que la matriz resultante sea del mismo tamaño que la matriz de entrada, debido a que en el proceso de convolución se reduce el tamaño. En caso de mantener el mismo tamaño se añaden filas y columnas auxiliares de valor cero. Esto no siempre es adecuado, ya que se evita reducir el número de parámetros, pero a la vez se obtiene más datos en los que intentar buscar patrones [1].

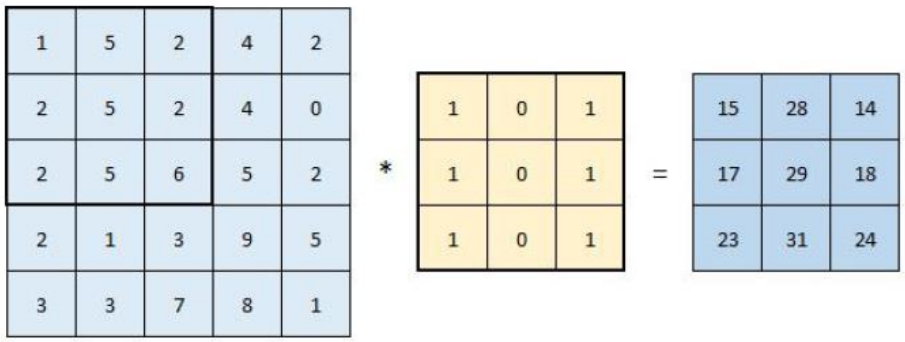


Figura 3.11. Proceso de convolución con padding.

Fuente: [1]

Al final del proceso de aprendizaje, los parámetros o pesos del filtro se modifican con el fin de minimizar la función de coste o error obtenido, por lo que estos parámetros no son datos que se puedan establecer a prioridad [1].

3.3.5.2. Capa pooling

Se reduce el número de datos/tamaño de las matrices, se suele colocar después de la capa convolucional. Estas capas son de gran utilidad ya que permiten reducir las dimensiones de la matriz bidimensional de la imagen que se estudia sin afectar al número de capas de dicha imagen. Esta operación también se llama reducción de muestreo debido a que la reducción de tamaño implica pérdida de información. En la capa de pooling se puede elegir entre varias opciones [1]:

Average Pooling: se calcula el valor medio de cada subconjunto de la matriz.

Max Pooling: se busca el valor máximo dentro de cada subconjunto de la matriz.

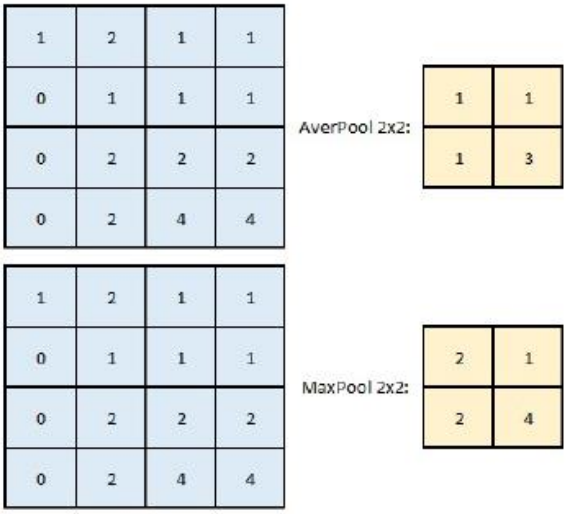


Figura 3.12. Averpooling y Maxpooling 2x2.

Fuente: [1]

3.3.5.3. Capa totalmente conectada

Este tipo de capa suele encontrarse al final de la estructura de la red neuronal convolucional. Es esta capa todas las neuronas artificiales estas conectadas entre ellas, al igual que en las redes neuronales tradicionales como el perceptrón multicapa, con el fin de clasificar usando los datos o características obtenidos mediante las convoluciones y reducciones de parámetros. Antes de esta capa, debe incluirse el proceso de flattened, que consiste en convertir la matriz bidimensional en un único vector de parámetros [1].

3.3.5.4. Capa dropout

Es uno de los posibles métodos de regularización que ayudan a evitar el overfitting, es decir, evita que la red aprenda con gran precisión a identificar las imágenes del conjunto de entrenamiento a costa de perder generalidad [1].

Esta capa suele usarse después de las capas totalmente conectadas, pero se pueden usar también después de las capas convolucionales o recurrentes.

La capa dropout elimina, de forma aleatoria y solo durante el proceso de entrenamiento de la red, algunas conexiones ocultas en la capa previa con una probabilidad variable configurada mediante el código del programa, siendo la probabilidad del 0.25-0.5% de forma general [1]

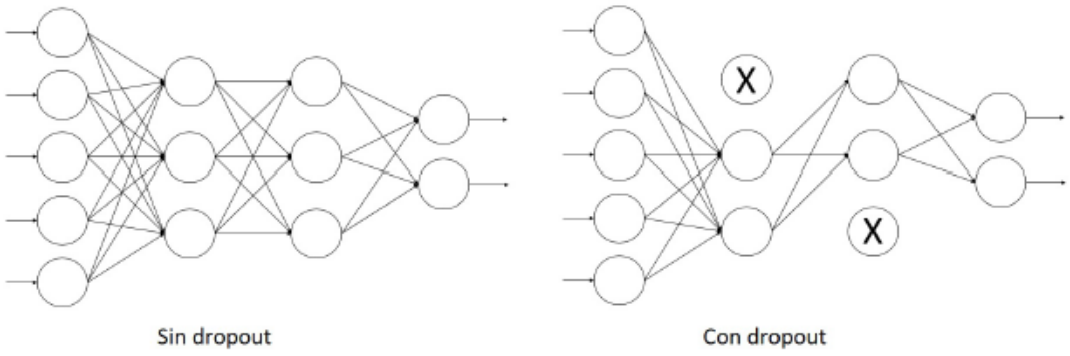


Figura 3.13. métodos de regularización

Fuente: [1]

3.4. Hardware libre

Se basa en la filosofía del software libre y amplía el espectro de su interés a los dispositivos físicos, entre los cuales se encuentran sensores, actuadores y elementos de control como microcontroladores y tarjetas de desarrollo [14] .

Básicamente consiste en un hardware que establece la libertad para que cualquier persona lo pueda estudiar, modificar, distribuir, materializar y vender, tanto el original como otros objetos basados en ese diseño [15].

A partir del desarrollo de este concepto se empiezan a generar comunidades denominadas Mackers, que se fundamentan en el desarrollo de proyectos de tecnología a partir de conocimiento libre [16] .

La información de estos periféricos que se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

Un ejemplo de esto es la tarjeta de desarrollo Arduino que fue un proyecto que surgió de la necesidad de tener un microcontrolador más barato para desarrollar proyectos, Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso [17].

Esta serie de dispositivos, presenta una gran variedad de tarjetas de desarrollo para la creación de proyectos de innovación, basados en electrónica digital, que vienen acompañadas de un entorno de desarrollo agradable, un lenguaje de programación sencillo y una comunidad internacional de todas las áreas del conocimiento que está constantemente nutriendo el conocimiento sobre esta tecnología. Adicionalmente, existen dispositivos como la Raspberry Pi, que es básicamente un mini computador de placa reducida y de bajo costo, que ha sido diseñada para facilitar la enseñanza y el aprendizaje de las ciencias de la computación, creado por la Raspberry Pi Foundation, este dispositivo permite la instalación de un sistema operativo para su funcionamiento, ampliando así las posibilidades de desarrollo [18].

3.4.1. Sistemas Embebidos

Este tipo de sistemas son una combinación de hardware y software de computadoras y en algunas ocasiones piezas mecánicas u otras adicionales diseñados para realizar alguna función específica. Los componentes principales de un sistema embebido son: un microprocesador; memoria primaria y secundaria; entradas y salidas periféricas; unidades de conexión a red; y unidades de entrada y salida (I/O) de propósito general. La característica principal que diferencia a los sistemas embebidos de los demás sistemas electrónicos es que al estar insertados dentro de un dispositivo que controlan están sujetos a cumplir requisitos de tamaño, fiabilidad, consumo y costo. La gran aplicabilidad de estos sistemas en cualquier ámbito ha generado gran interés en la industria para su desarrollo. En la actualidad existen diversas tarjetas de desarrollo como la BeagleBone, VIA APC, Android y Raspberry-Pi, las cuales presentan gran capacidad de cómputo y precios accesibles para el desarrollo de estos sistemas [19].

3.4.2. Tarjeta de desarrollo Raspberry-Pi

La Raspberry Pi se comenzó a comercializar en agosto de 2011 con el objetivo de ser utilizada para incentivar el conocimiento en el uso de la electrónica y la programación en edades tempranas [20].

Raspberry-Pi es un ordenador de placa reducida y bajo costo, el cual opera bajo sistema operativo Linux. El diseño de la tarjeta incorpora un System-on-a-chip Broadcom o en español (sistema en un chip) BCM2835, que contiene un procesador central (CPU) ARM1176JZF- S a 700 MHz, un procesador gráfico (GPU) Video Core IV, y 512 MB de memoria RAM. El diseño no incluye un disco duro ni unidad de estado sólido, sin embargo, permite el almacenamiento permanente a través de una tarjeta SD [20].

De igual forma la Raspberry nos proporciona diversos componentes de comunicación como lo son puertos UART/USB, Audio, salida de video HDMI y RCA, puerto Ethernet y un arreglo pines de entrada/salida de propósito general (GPIO por sus siglas en inglés) mediante los cuales es posible activar estados lógicos o establecer comunicación con microcontroladores u otros dispositivos por protocolos como I2C, SPI, UART entre otros [19].

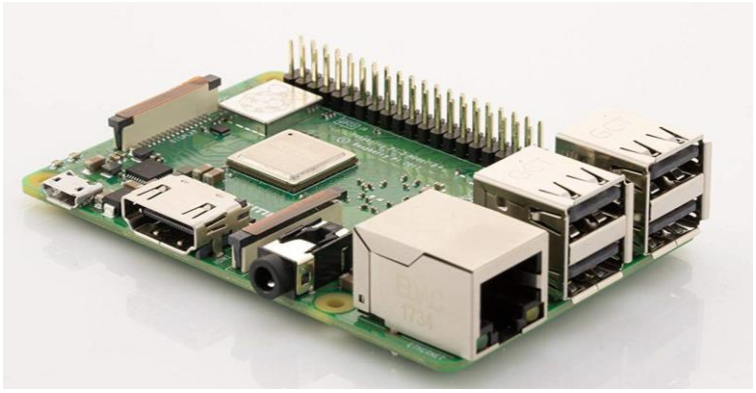


Figura 3.14. Tarjeta de desarrollo Raspberry-Pi.

Fuente: [21]

3.4.3. Hardware Raspberry Pi

El hardware de Raspberry Pi se asienta sobre una placa de 8,5 centímetros por 5,3 centímetros donde están integrados todos los circuitos, puertos y conectores. Cuenta con puerto Ethernet, conector Wi-Fi y bluetooth en los dispositivos modernos, una ranura para tarjetas SD o microSD donde irán instalados el sistema operativo, programas y archivos. También dispone de puertos USB para poder conectarle periféricos tales como teclado, ratón o discos duros externos o pendrives [22].

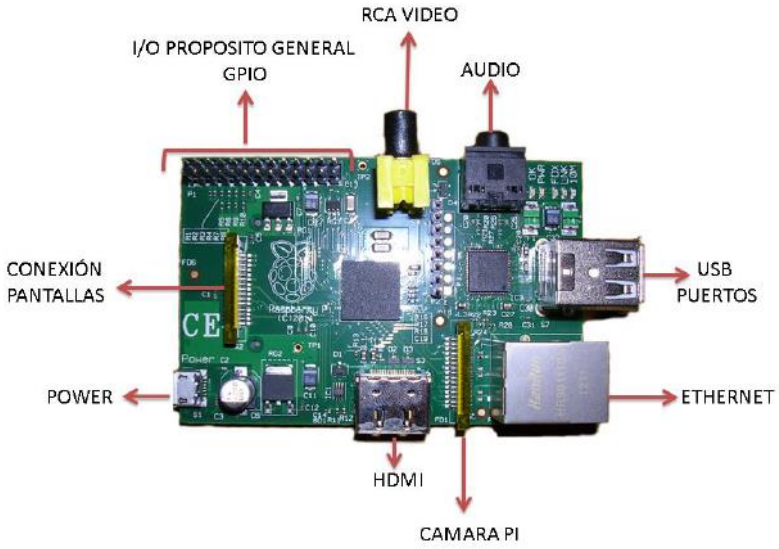


Figura 3.15. Hardware Raspberry Pi.

Fuente: [23]

3.4.4. Especificaciones Técnicas

Tabla 3.3. Especificaciones Técnicas Raspberry Pi.

MODELO	RASPBERRY PI MODELO B
Procesador con grafica integrada	Broadcom BCM2835, contiene ARM1176JFS, con unidad de coma flotante a 700 MHz y un videocore de 4 GPU.
Memoria	512 MB
Características técnicas de la GPU	La GPU es capaz de mover contenidos con calidad Bluray, usando H.264 hasta 40 MBits/s. Dispone un Core 3D con soporte para las librerías OpenVG. Es capaz de decodificar 1080p30 H.264 high-profile.
Dispositivo de Arranque	Memoria SD Card. Tras el arranque inicial de la SD se puede terminar el arranque desde un dispositivo USB.
Conectores	2x conectores USB 2.0
	Conector Ethernet RJ-45 10/100

	Salida de video digital HDMI (audio y video)
	Salida de video Analógico (S-video)
	Audio Analógico (conector 3.5mm)
	Conector GPIO
	Conector de alimentación Micro SD
	Lector de memorias SD (utilizado para arrancar el dispositivo)
Alimentación	Vía micro USB 5 voltios, casi cualquier dispositivo con alimentación USB puede servir como fuente de alimentación
Sistemas operativos soportados	RASPBIAN "Wheezy" (Debian)
	ArchLinux
	Fedora
	QtopenPi (QT SDK)
	OpenElec
	Raspbcm
	Android (en desarrollo por usuarios)
Rangos de temperatura	LAN9512 de 0°C a 70°C
	AP de -40°C a 85°C
Dimensiones	85.60mm x 53.98mm x 17mm

Fuente: [23]

3.4.5. Cámara compatible con Raspberry.

La cámara Raspberry Pi de alta definición se conecta a cualquier Raspberry Pi o Compute Module para crear fotografías y vídeos HD [24]. Utiliza el sensor de imagen IMX219PQ de Sony que ofrece imágenes de vídeo de alta velocidad y alta sensibilidad. También ofrece contaminación de imagen reducida como ruido de patrón fijo y borrones. Dispone también de funciones de control automático como el control de exposición, el balance de blancos y la detección de luminancia [24].

Aplicaciones:

El módulo de cámara Raspberry Pi es ideal para vídeos HD y fotografía. También puede probar la función de cámara lenta y time-lapse. Otras aplicaciones incluyen botánica, observación de vida salvaje y seguridad doméstica [24].

Características:

- Imágenes de alta calidad
- Alta capacidad de datos
- Enfoque fijo de 8 megapíxeles
- Compatible con 1080p, 720p60 y VGA90
- Sensor de imagen CMOS Sony IMX219PQ
- Cable plano de 15 contactos

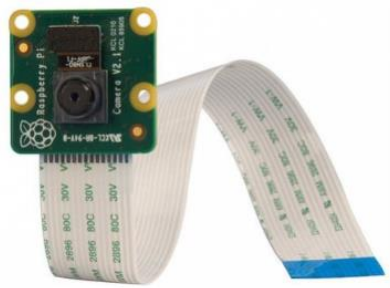


Figura 3.16. Cámara Raspberry-Pi.

Fuente: [24]

3.5. Software Libre

El hecho de que, en inglés, el idioma en el que se acuñó y difundió el término software libre (free software), una misma palabra (free) signifique tanto "libre" como "gratuito" y que gran parte del software libre sea efectivamente gratuito, ha favorecido a las malas interpretaciones: muchas personas consideran equivalente los términos software libre y software gratuito. Sin embargo, el rasgo esencial que define el software libre es la libertad, mas no el precio. Cuando se habla de software libre debemos pensar en "libertad de expresión" (free speech). El propietario de los derechos sobre el software libre garantiza a los usuarios, mediante una licencia, una serie de libertades que no otorga el propietario del software privativo, que se reserva numerosos derechos en base a la legislación sobre propiedad intelectual por ejemplo, no permite el acceso al código fuente o no permite ninguna modificación y su subsecuente distribución. El usuario de software privativo en realidad paga por el derecho a usar, con numerosas limitaciones, el software [25].

3.5.1. Definición Free Software Foundation (FSF); El Software libre

La definición del software libre es la libertad de la comunidad de usuarios para poder ejecutar, copiar, estudiar, mejorar y redistribuir el software. La palabra clave aquí es libertad. Libertad de usar el programa para cualquier propósito, de estudiar cómo funciona y adaptarlo a las diferentes necesidades, de distribuir copias, de poder mejorarlo y de hacer públicas las mejoras. La única restricción es que, si se redistribuye el programa, se tiene que hacer reconociendo los mismos derechos en los usuarios de nuestras modificaciones. Como vemos, el acceso al código fuente es un requisito previo y necesario para ejercer la mayoría de estas libertades. En concreto, la FSF se refiere a cuatro libertades que deben tener los usuarios del software para que pueda ser calificado como libre [25].

- **Libertad 0.** Libertad de usar el programa para cualquier propósito.
- **Libertad 1.** Libertad de estudiar cómo funciona el programa y adaptarlo a las propias necesidades. Una condición previa para que se dé esta libertad es el acceso al código fuente.
- **Libertad 2.** Libertad de redistribuir copias.
- **Libertad 3.** Libertad de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad pueda beneficiarse. Esta libertad también requiere el acceso al código fuente. Por tanto, un programa puede definirse como software libre solamente si los usuarios tienen todas estas libertades [25].

3.5.2. Debian

Debian es una organización formada por voluntarios dedicada a desarrollar software libre y promocionar los ideales de la comunidad del software libre. El Proyecto Debian comenzó en 1993, cuando Ian Murdock hizo una invitación a todos los desarrolladores de software a contribuir a una distribución completamente coherente basada en él, entonces relativamente nuevo, núcleo Linux. Ese grupo relativamente pequeño de entusiastas, ha crecido a lo largo de los años hasta convertirse en una organización de alrededor de 1026 desarrolladores Debian [26].

Los desarrolladores Debian están involucrados en una gran variedad de tareas, incluyendo la administración del Web y FTP, diseño gráfico, análisis legal de licencias de software, escribir documentación y, por supuesto, mantener paquetes de software.

Los desarrolladores de Debian también están involucrados en otros proyectos; algunos específicos de Debian, otros en los que está involucrado parte o toda la comunidad Linux [26]. Algunos ejemplos incluyen:

El Linux Standard Base (LSB). El LSB es un proyecto que pretende estandarizar el sistema básico de GNU/Linux, lo que permitiría a terceros desarrolladores de software y hardware desarrollar fácilmente programas y controladores de dispositivos para Linux en general, más que para una distribución de GNU/Linux en particular [26].

El Estándar para la jerarquía del sistema de ficheros (FHS) es un esfuerzo para estandarizar la distribución del sistema de ficheros de Linux. El FHS permitirá a desarrolladores de software concentrar sus esfuerzos en diseñar programas, sin tener que preocuparse sobre cómo se instalará su paquete en diferentes distribuciones de GNU/Linux [26].

3.5.3. Software Raspberry Pi

En la página oficial de Raspberry Pi podemos descargar los diferentes sistemas operativos que esta compañía viene actualizando continuamente, ellos recomiendan iniciar a los principiantes con [23]:

- NOOBS
- NOOBS LITE

Existen también sistemas operativos un poco más avanzados como son:

- RASPBIAN
- PIDORA
- OPENELEC
- RASPMC
- RISC OS

Al ser un ordenador, la Raspberry Pi necesita un software para su funcionamiento. Principalmente, está diseñada para usar sistemas operativos de entorno Linux. Raspbian es una distro basada en Debian y es el sistema operativo que recomienda la Fundación para empezar a usarla. También se pueden emplear otros sistemas operativos como Pidora, o entornos más específicos como OpenELEC o Kodi creados para funciones multimedia [22].

3.5.4. Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos [27].

3.5.4.1. Características de Python:

- **Lenguaje interpretado o de script;** Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo, los lenguajes interpretados son más flexibles y más portables [27].

  No obstante, Python tiene muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudo código máquina intermedio llamado bytecode la primera vez que se ejecuta, generando archivos pyc o pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones [27].
- **Tipado dinámico;** La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo [27].
- **Fuertemente tipado;** No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o string) no podremos tratarla como un número (sumar la cadena "9" y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores [27].
- **Multiplataforma;** El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios [27].
- **Orientado a objetos;** La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos. Python también permite la programación imperativa, programación funcional y programación orientada a aspectos [27].
- **Interactivo;** Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que

puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente [28].

- **Sintaxis clara;** Por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave begin y end. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar [28].

4. METODOLOGÍA

En la elaboración de la plataforma programada, así como el diseño y desarrollo del dispositivo de lectura de números y letras se emplea la metodología descriptiva, esto debido al análisis de los procesos que se llevan a cabo a fin de lograr el correcto funcionamiento del dispositivo. El método descriptivo también permite detallar todos los parámetros estudiados en el desarrollo de la plataforma a fin de comprender de manera correcta los pasos de la programación y líneas de código que se implementen, las mismas que se representaran en diagrama de bloques.

4.1. Operacionalización de variables

4.1.1. Variables de entrada

Tabla 4.1. Operacionalización de variables de entrada.

Variables	Indicadores	Ítem	Técnicas o instrumentos
Identificación de números.	Identificación de números mediante el uso de redes neuronales convolucionales	Porcentaje de exactitud para la identificación	Redes neuronales. Funciones de activación
Identificación de letras.	Identificación de letras mayúsculas y minúsculas mediante el uso de redes neuronales convolucionales	Porcentaje de exactitud para la identificación	Redes neuronales. Funciones de activación.

Fuente: Autores.

4.1.2. Variable de salida

Tabla 4.2. Operacionalización de la variable de salida.

Variables	Categoría	Indicadores	Ítems	Técnicas o instrumentos

Dispositivo de identificación de números y letras orientado a la enseñanza de niños de Educación Básica	Continuidad de la identificación de los números y letras	Cantidad de números y letras que se identifica.	Tipo de letra (mayúsculas/ minúsculas)	Programación Pesos sinápticos
		Margen de exactitud en la identificación.	Porcentaje de Exactitud	Matriz comparativa de datos identificados

Fuente: Autores.

4.2. Diseño y Selección de componentes

Los criterios dentro del diseño y desarrollo de la plataforma, así como del dispositivo se basan específicamente en los componentes y necesidades dentro del país. Los materiales deben ser de fácil acceso a fin de que, si existiere alguna falla o daño del dispositivo, los componentes deben constar en el mercado nacional.

Dentro de las necesidades del prototipo del dispositivo, este deberá cumplir con las normas que rijan en el país entorno a la educación básica.

4.2.1. Características preliminares

4.2.1.1. Requisitos funcionales

La finalidad del dispositivo de detección de letras y números a través de software libre con desarrollo en redes neuronales y reconocimiento a través de tarjetas, es fomentar el uso de la tecnología en el aprendizaje de letras y números enfocado en la educación básica, a fin de que sea más interactivo el aprendizaje.

A continuación, se enlistan ciertas funcionalidades que debe cumplir el dispositivo con el fin de alcanzar buenos resultados al momento de enfocarse en la enseñanza de letras y números:

- Mediante interfaz gráfica se visualizará la lera o número que se proyecte a través de la tarjeta en la cámara digital.
- Mediante botones de interacción se visualizará la letra o número equivalente a lo que se ingresó por cámara digital a través de una interfaz gráfica.
- Durante el reconocimiento de dicha tarjeta la interfaz implementada será capaz de emitir sonidos que estén acorde a lo que se muestra en la figura.
- Tendrá botones de navegación dentro de la interfaz aplicada, de forma que al momento de cambiar de interfaz entre números y letras este cambio se realice de forma interactiva.

- Contará con una pantalla interactiva donde se indicará como es el funcionamiento del dispositivo.

4.2.1.2. Requisitos hardware y software

La plataforma o programación que se empleara dentro del dispositivo de detección de letras y números tendrá que adaptarse a las características de un computador básico de costo medio, para que el dispositivo pueda trabajar de forma efectiva y eficaz sin importar el entorno donde se lo pruebe.

A continuación, se detallan ciertas características y requisitos que debe cumplir tanto la plataforma, como el dispositivo para su correcto funcionamiento al momento de enfocarse en la enseñanza de letras y números:

- Procesador ARM a 700 MHz o menor capaz de soportar desarrollo en entorno gráfico.
- Memoria de procesamiento RAM o SDRAM de al menos 2 Gb.
- Periféricos de salida entrada para teclado y mouse, indistintamente de las características que posean estos periféricos.
- Salidas de video o pantalla de visualización en el rango de puertos HDMI con un procesamiento de datos de al menos 60 Hz.
- Almacenamiento masivo interno de 16 Gb como mínimo, para alojar el sistema operativo y la plataforma de desarrollo.
- Salida de audio o puerto de salida para audio.
- Sistema operativo desarrollado o en desarrollo indistintamente si es propietario o libre.
- Debe poseer instalado software Python debido a que el Desarrollo de la plataforma se encuentra en este lenguaje de programación
- Contará con las librerías necesarias para que el procesamiento de imágenes en programación Python se pueda ejecutar sin ningún problema.

4.2.2. Tarjeta de control

La tarjeta de control o hardware cumple un papel fundamental en el dispositivo de detección de letras y números debido a que será la encargada de almacenar e interactuar con los softwares de la plataforma creada, para que esta sea capaz de interactuar con la red neuronal y el sistema de Visión Artificial implementados en el dispositivo. Además, debe ser de fácil acceso y obtención dentro del mercado ecuatoriano, y su uso debe ser de carácter sencillo.

4.2.2.1. Criterios de selección para la tarjeta de control

Dentro de los criterios más importantes que se necesita para que la tarjeta de control cumpla con todos los parámetros expuestos en requisitos de hardware y software se obtiene los siguientes parámetros:

- Costo
- Disponibilidad dentro del mercado ecuatoriano
- Compatibilidad
- Control de periféricos
- Control y acceso simple a cámara
- Programación

4.2.2.2. Selección de la tarjeta de control

En base a todo lo expuesto en: tarjeta de control, requisitos hardware y software y criterios para la selección de tarjeta de control. Dentro del mercado ecuatoriano existen algunas tarjetas de control de software libre, de las cuales una de las que cumplen con los requisitos del dispositivo es la tarjeta de desarrollo Raspberry Pi.

Dentro de la familia de tarjetas de desarrollo Raspberry tenemos muchos tipos de tarjetas de desarrollo y la que se ajusta a los requisitos de hardware y software es la tarjeta de desarrollo Raspberry Pi4 modelo B de las siguientes características:

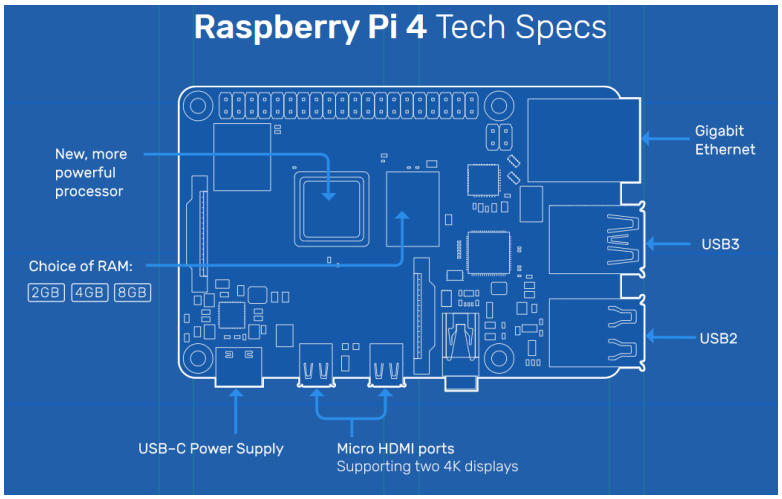


Figura 4.1. Tarjeta de desarrollo Raspberry Pi4.

Fuente: [20]

Tabla 4.3. Especificaciones Raspberry Pi4 modelo B.

Raspberry Pi 4 Modelo B	
Procesador:	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit

Memoria RAM:	4GB LPDDR4-3200 SDRAM
GPU:	VideoCore VI (con soporte para OpenGL ES 3.x)
Frecuencia de Reloj:	1,5 GHz
Almacenamiento:	Memoria SD 32 Gb clase 10
Conectividad	Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet
Puertos Periféricos:	GPIO 40 pines, USB 2.0, 2 x USB 3.0, Jack de 3.5 Audio
Puertos video:	2 x micro HDMI 2
Alimentación:	Adaptador de voltaje de 110V a 5V/3A

Fuente: Autores

Tabla 4.4. Criterio de selección de la tarjeta de desarrollo.

Detalle	**Raspberry pi ZERO**	**Raspberry Pi 2**	**Raspberry Pi 4**
Procesador	Broadcom BCM 2835@ 1 Ghz ARM 11	Quad core Cortex A7 a 900 MHZ	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit
Memoria RAM	512 MB	1 GB	4GB LPDDR4-3200 SDRAM
GPU	VideoCore IV	VideoCore IV	VideoCore VI (con soporte para OpenGL ES 3.x)
Frecuencia de Reloj	1 GHz	1 GHz	1,5 GHz
Almacenamiento	Ranura para tarjeta SD	Tarjeta micro SD	Memoria SD 32 Gb clase 10
Conectividad	Bluetooth 4.1	Bluetooth 4.1 10/100 Base Ethernet	Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet
Puertos periféricos	USB (On The Go) Cabezal de 40 pines compatible con HAT	GPIO 40 pines 4 USB 2.0 1 ethernet Combo audio/mic	GPIO 40 pines, 2 USB 2.0. 2 USB 3.0, Jack de 3.5 Audio
Puertos de video	Mini HDMI a 1080p	1 micro HDMI	2 micro HDMI

Compatibilidad con cámara	Conector para cámara (CSI)	Interfaz de cámara (CSI)	Cámara Raspberry Pi (CSI) más Pantalla táctil (DSI)
Alimentación	Micro USB	Micro USB	Micro USB más Adaptador de voltaje de 110V a 5V/3A

Fuente: Autores.

4.2.2.3. Implementación sistema operativo en tarjeta de control

La tarjeta de desarrollo Raspberry Pi. Tiene su propio sistema operativo basado en software libre ya sea en las plataformas: Linux y Ubuntu.

Los desarrolladores del sistema operativo de Raspberry Pi, han denominado al sistema operativo como RASPBIAN o RASPBERRY PI OS, el mismo que se cargara en el dispositivo para empezar con el desarrollo de la plataforma de lectura de números y letras.

Una de las ventajas de cargar el sistema operativo desarrollado por los mismos creadores de la tarjeta Raspberry Pi son las librerías de Python, Scratch, Sonic Pi, Java, las mismas que vienen incluidas y abiertas a fin de evitar la reinstalación del software mencionado.

Para instalar el sistema operativo Raspbian se debe ingresar a la página oficial de Raspberry a la zona de descargas y descargar la versión de Raspbian requerida.

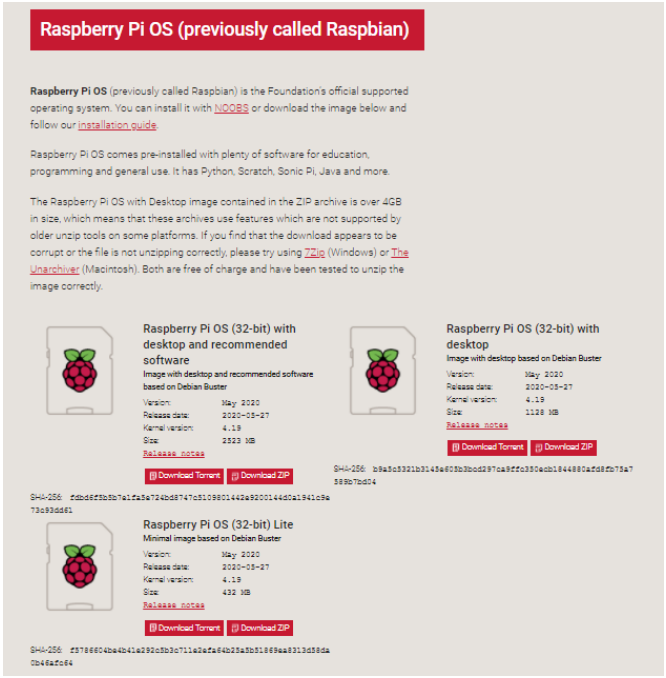


Figura 4.2. Proceso de descarga del sistema operativo.

Fuente: Autores

Para la finalidad del proyecto se descargará e instalará la opción: Raspberry PI OS (32-bits) with desktop and recommended software, en español (con escritorio y software recomendado). Para la instalación del sistema operativo se usará el método de flasheo, que consiste en usar un software libre en el caso del dispositivo será: BALENA ETCHER**,** este software sirve para el empaquetamiento de datos para la generación de una imagen ISO dentro la tarjeta SD. Con este método el sistema operativo se crea automáticamente dentro la tarjeta SD.

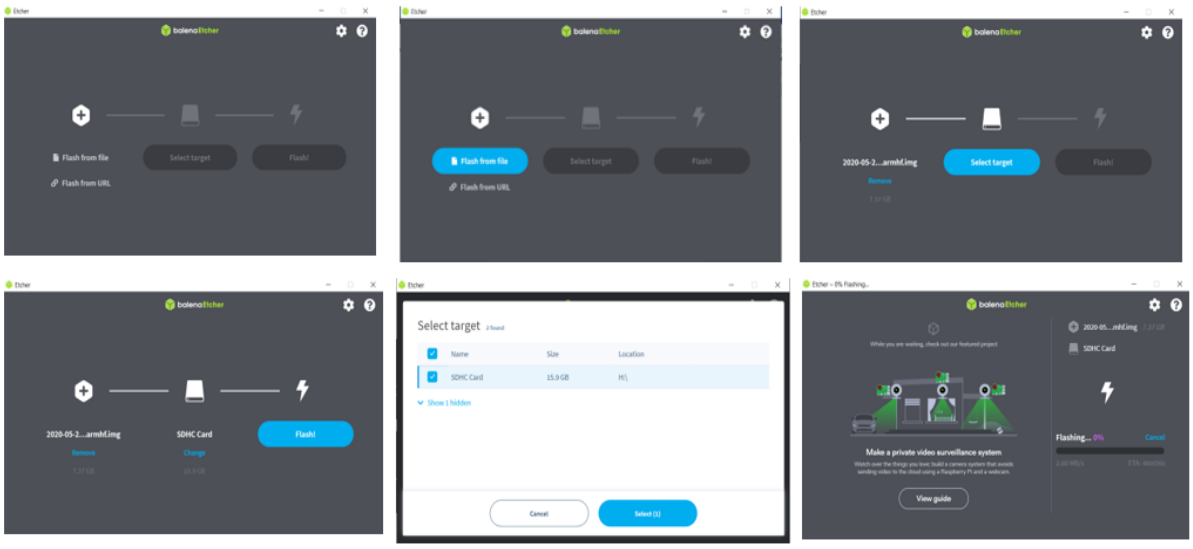


Figura 4.3. Proceso de instalación del sistema operativo.

Fuente: Autores

Una vez que se encuentre la micro SD con el sistema operativo cargado se procede a configurar las opciones del sistema operativo dentro de la tarjeta de desarrollo Raspberry Pi. Para ello se inserta la tarjeta micro SD en la tarjeta de desarrollo.

Figura 4.4. Inserción de la tarjeta micro SD en la tarjeta de desarrollo.

Fuente: Autores.

Una vez la tarjeta ingresada dentro de la tarjeta de desarrollo se conecta los periféricos y se configura la tarjeta de desarrollo Raspberry Pi.

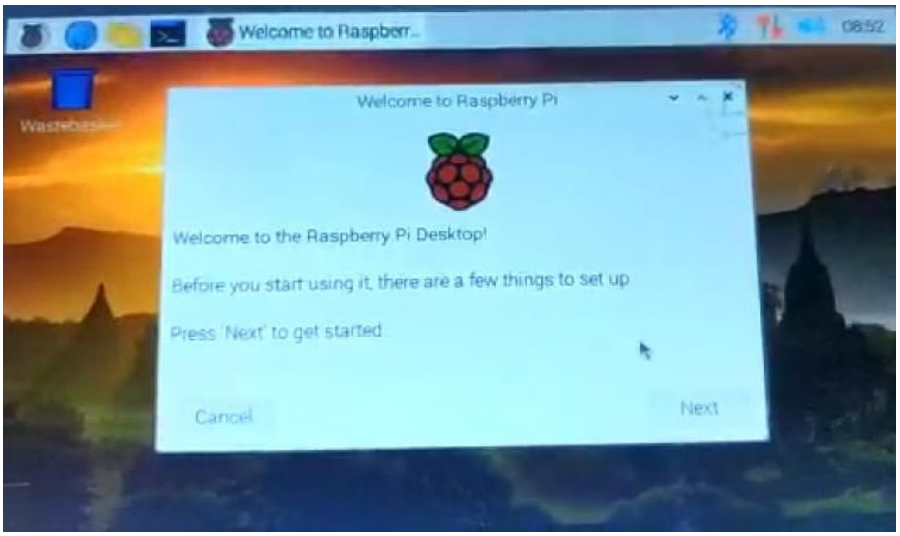


Figura 4.5. Configuración de la tarjeta de desarrollo.

Fuente: Autores

Las primeras modificaciones que se realizarán en la tarjeta de desarrollo Raspberry Pi serán: fecha, hora, ubicación.

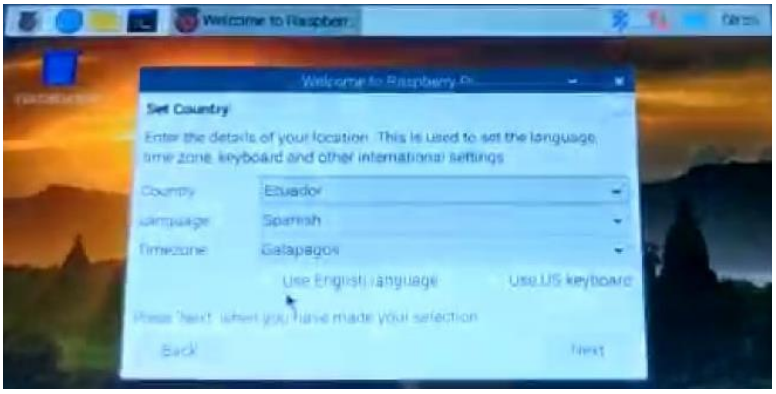


Figura 4.6. Paso 1 para la configuración de la tarjeta de desarrollo.

Fuente: Autores

La segunda configuración importante es el ingreso y confirmación de una clave de acceso, esto debido a que la tarjeta de desarrollo exige que para los atributos de usuario o super usuario se acceda mediante una contraseña.

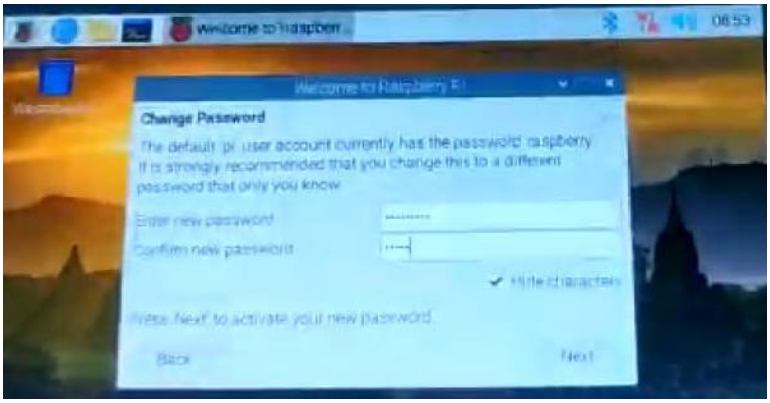


Figura 4.7. Paso 2 para la configuración de la tarjeta de desarrollo.

Fuente: Autores

La tercera configuración realizada a la tarjeta de desarrollo Raspberry Pi, será la configuración de la pantalla y la resolución de la misma.

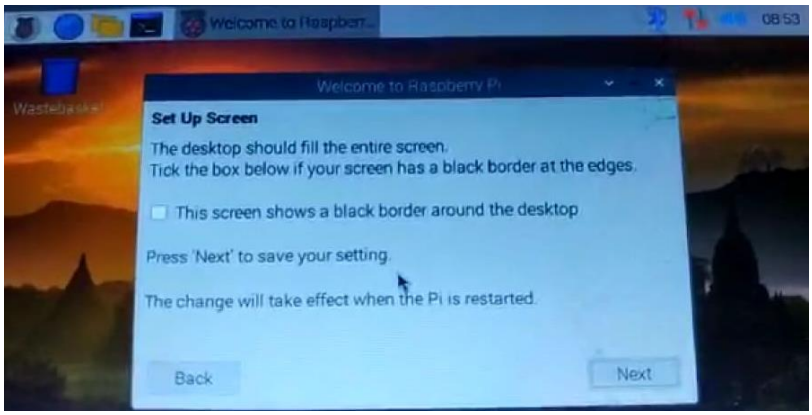


Figura 4.8. Paso 3 para la configuración de la tarjeta de desarrollo.

Fuente: Autores

Al final de la configuración de la tarjeta de desarrollo Raspberry Pi, indica que se es necesario reiniciar, con esto se terminará el proceso de instalación del sistema operativo dentro de la tarjeta de desarrollo Raspberry Pi.

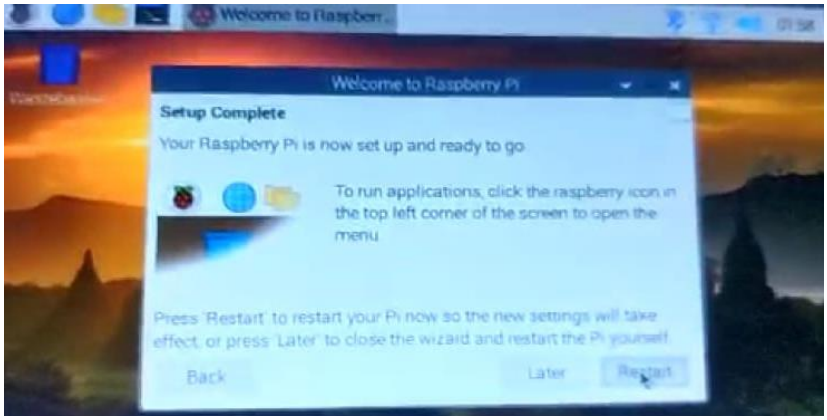


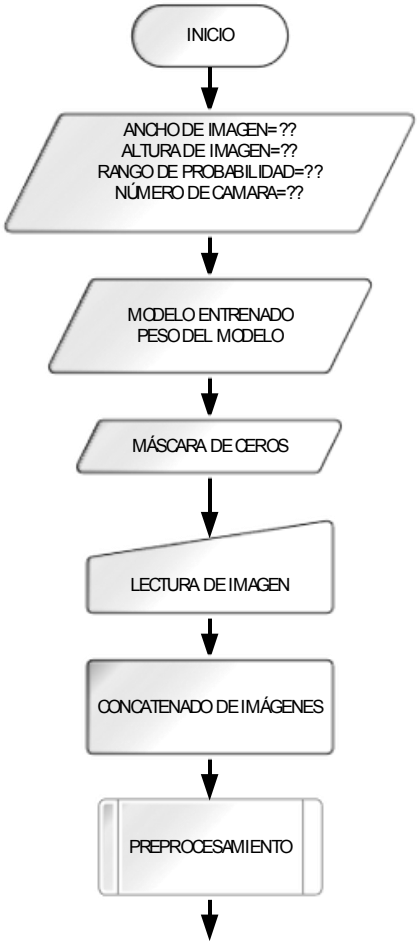
Figura 4.9. Paso final de la configuración de la tarjeta de desarrollo.

Fuente: Autores

4.2.3. Diseño de la plataforma del dispositivo

La plataforma a nivel de software que se implementa dentro del dispositivo esta realizado en Python, en programación secuencial y por bloques.

Para la detección de números y letras se usa redes neuronales, así como el procesamiento de imágenes a través de adquisición por cámara como se presenta en el siguiente diagrama.



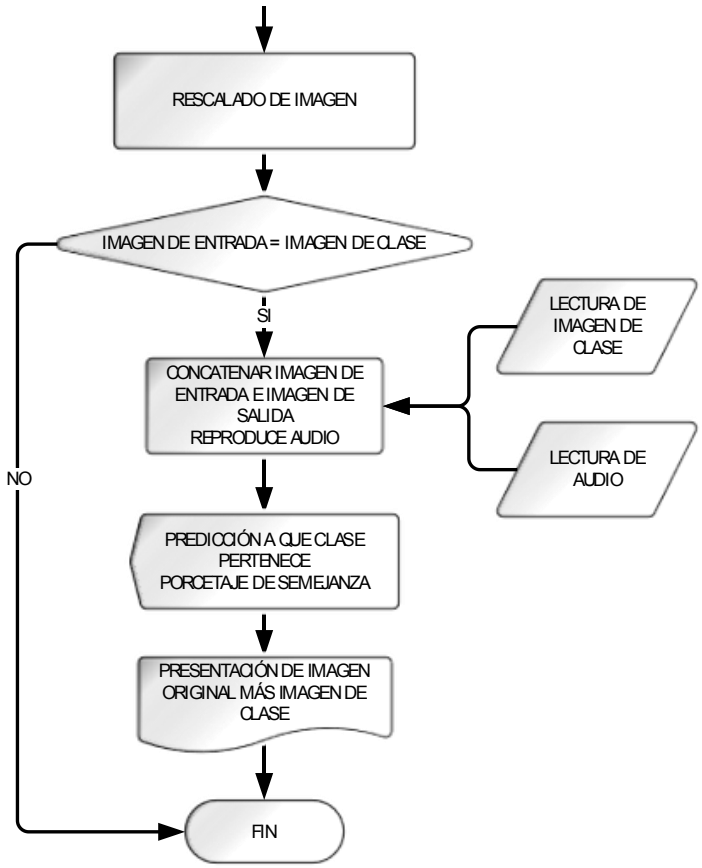


Figura 4.10. Proceso total de programación de la plataforma

Fuente: Autores

4.3. Desarrollo de la programación en Python

Dentro del sistema operativo Raspbian. Python viene pre instalado en el mismo, en su versión Python3 y Thonny con énfasis en la legibilidad y uso de palabras clave estándar en inglés.

Para el desarrollo de la plataforma se empelará la versión Thonny Python3. Adicional se utiliza varias librerías preinstalas o cargadas tales como:

- NUMPY
- CV2
- OS
- SKLEARN MODEL
- MATPLOTLIB
- PICKLE
- TKINDER

Mientras que a las demás se las tiene que instalar.

- OPENCV
- TENSROFLOW

- PYGAMES
- KERAS
  - KERAS PREPOCESSING
  - KERAS UTILS
  - KERAS MODELS
  - KERAS LAYERS
  - KERAS OPTIMIZERS
  - KERAS LAYERS

4.3.1. Instalación librerías Python

4.3.1.1. Opencv (CV2)

Librería software de código abierto de visión artificial, posee alrededor de 2500 algoritmos pre cargados que permiten identificar: objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios, etc.

En la plataforma del dispositivo creado, el uso de esta librería se enfoca en el reconocimiento de letras y números ingresadas a través de la cámara, para posteriormente procesar dicha imagen.

Figura 4.11. Imagen ingresada a través de la cámara de Raspberry Pi4.

Fuente: Autores

Los siguientes comandos sirven para la instalación de OpenCV dentro de Raspberry Pi los mismos que se deben ingresar dentro del terminal de comandos de dicha tarjeta de desarrollo.

Tabla 4.5. Comandos para la instalación de OpenCV

Comandos	
1	python3
2	quit()
3	sudo apt-get install libhdf5-dev libhdf5-serial-dev libatlas-base-dev libjasper-dev libqtgui4 libqt4-test
4	pip3 install opencv-contrib-python==4.1.0.25
5	import cv2 cv2.__version__

Fuente: Autores.

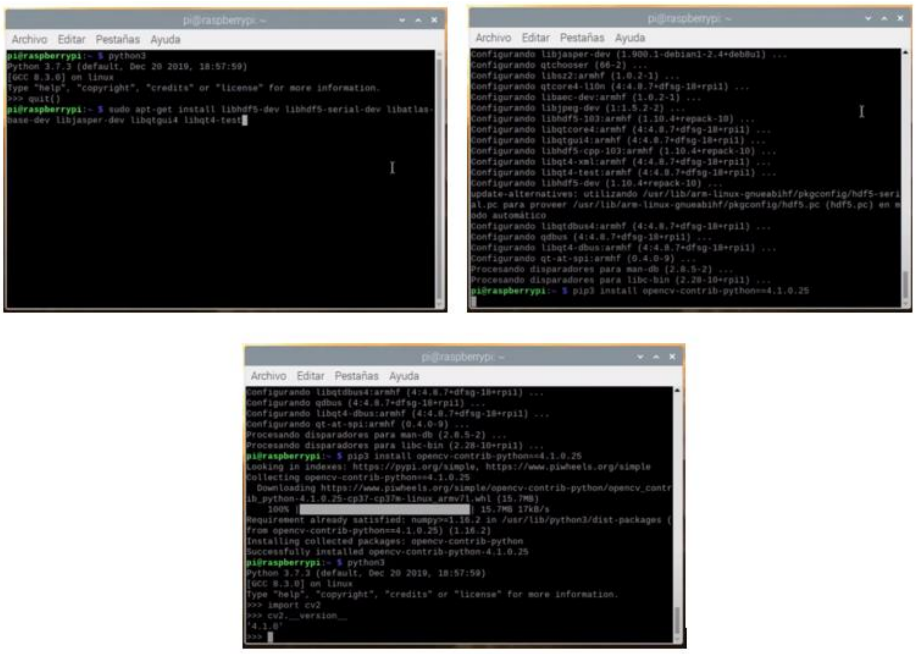


Figura 4.12. Instalación de OpenCV.

Fuente: Autores

4.3.1.2. TensorFlow

Biblioteca software de código abierto para aprendizaje automático a través de un rango de tareas, sirve principalmente para el entrenamiento de redes neuronales en el desarrollo de detectar y descifrar patrones y correlaciones. Estos procesos por lo general son de carácter análogo al aprendizaje y razonamiento usados por los humanos.

Dentro de la plataforma del dispositivo creado, el uso de esta biblioteca se enfoca en el entrenamiento de la red neuronal aplicada dentro del reconocimiento de letras y números a fin de entregar valores acordes a lo ingresado a través de la librería Opencv por la cámara.

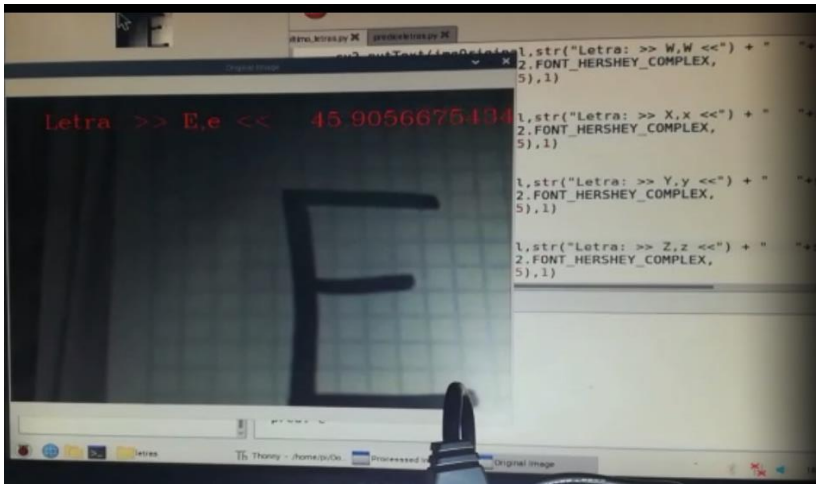


Figura 4.13. Aplicación de TensorFlow dentro de la plataforma del dispositivo.

Fuente: Autores

La instalación de la biblioteca TensorFlow dentro de Raspberry Pi se debe seguir los siguientes comandos, ingresados dentro del terminal de comandos de dicha tarjeta de desarrollo.

Tabla 4.6. Comandos para la instalación de TensorFlow.

	Comandos
1	sudo apt-get update
2	sudo apt-get upgrade
3	https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
4	mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi tflite1
5	cd tflite1
6	sudo pip3 install virtualenv
7	pythoon3 -m venv tflite1-env
8	source tflite1-env/bin/activate
9	bash get_pi_requirements.sh
10	Restart

Fuente: Autores.

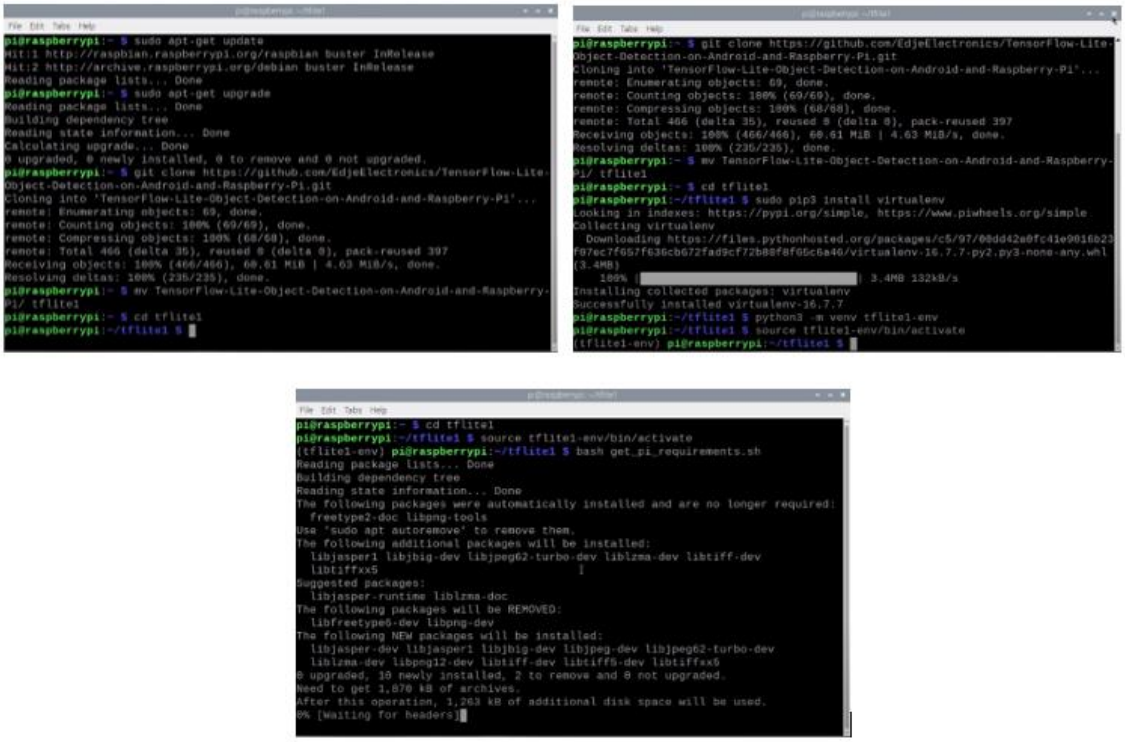


Figura 4.14. Instalación de TensorFlow.

Fuente: Autores.

4.3.1.3. Keras

Es una aplicación diseñada y escrita en Python que se puede ejecutar junto a TensorFlow para ofrecer la implementación de redes neuronales de alto nivel. También es muy usada para implementar redes neuronales de una manera rápida, modular y sencilla.

Para el dispositivo creado junto con la plataforma se usa la librería Keras junto a TensorFlow para el entrenamiento de las redes neuronales de números, letras mayúsculas y letras minúsculas.

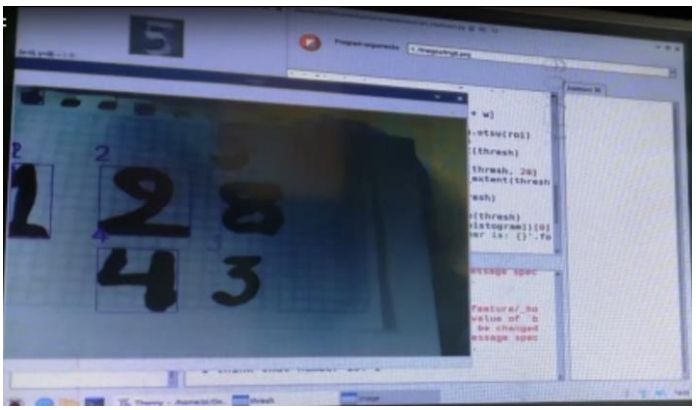


Figura 4.15. Keras junto a TensorFlow en entrenamiento de redes neuronales.

Fuente: Autores.

Para la instalación de la aplicación Keras dentro de Raspberry Pi, se debe tener pre instalado la biblioteca TensorFlow debido a que en la misma está incluida la aplicación Keras, para esto se ingresa el siguiente comando dentro del terminal de comandos de dicha tarjeta de desarrollo.

Tabla 4.7. Comando de instalación de Keras.

	Comando
1	sudo pip3 install keras

Fuente: Autores.

4.3.1.4. Pygame

Es una librería de código abierto encargada del desarrollo en el uso de los periféricos en Python aplicados en Raspberry Pi, de manera que se pueda manipular sonidos, teclados, mouse, etc. Dentro de la plataforma de programación.

La librería expuesta dentro de la plataforma del dispositivo creado, es la encargada de reproducir los sonidos de cada letra y numero conforme estos se ingresen en el reconocimiento por tarjeta.

La instalación de la librería Pygame, se la realiza dentro del terminal de comandos de la tarjeta de desarrollo, sin necesidad de descargar o ingresar a una dirección web, debido a que las librerías se encuentran pre cargadas dentro de la tarjeta de desarrollo Raspberry Pi.

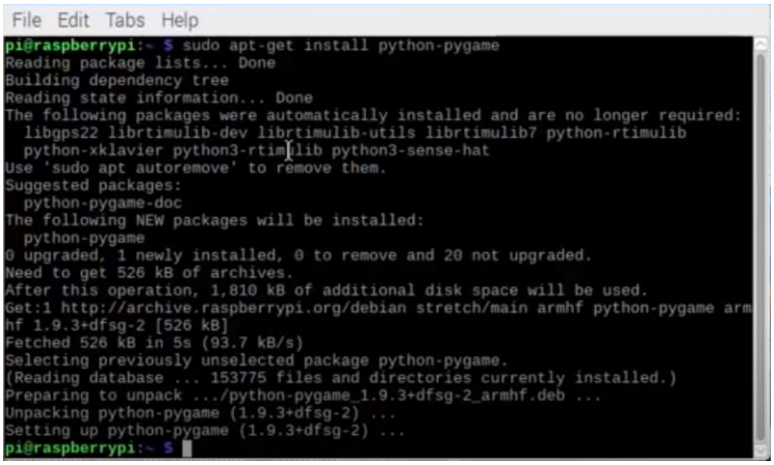


Figura 4.16. Instalación por comandos de la librería Pygame.

Fuente: Autores.

4.3.2. Visión artificial

En la plataforma que se implementa dentro del dispositivo, el desarrollo y manejo de visión artificial es muy importante debido a que con esta herramienta se puede procesar las imágenes adquiridas por cámara. Dicho proceso en las imágenes adquiridas se enfoca en la identificación, limpieza y tratamiento de letras y números, a fin de entregar una imagen limpia y binarizada para la posterior comparación.

Para el procesamiento o tratamiento de las variables se emplea conversiones en base a la imagen original para de esta forma obtener una imagen binarizada a comparar con la base de datos implementada dentro de la plataforma.

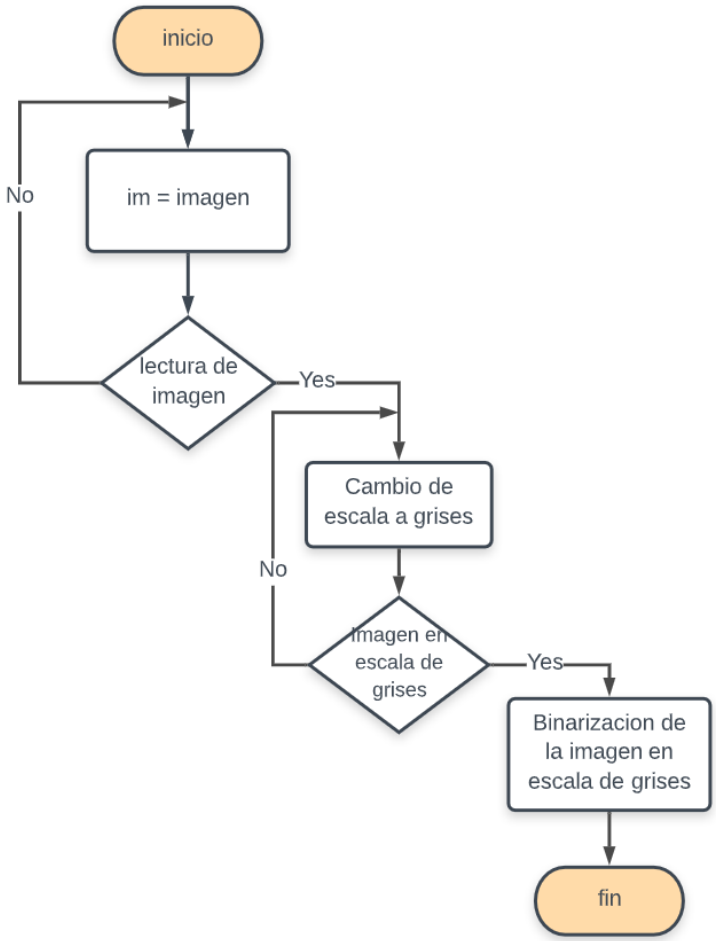


Figura 4.17. Diagrama del proceso de binarizacion de la imagen.

Fuente: Autores.

La imagen real al pasar por su primera conversión toma una escala de grises ya sea en la plataforma del dispositivo o en el entramiento de las redes neuronales, esto a fin de obtener un valor unitario y una ponderación la cual queda establecida en escala entre 0 y 1.

Figura 4.18. Transformación de una imagen a escala de grises.

Fuente: [29].

La imagen que se transformó en escala de grises pasa por un proceso de limpieza para eliminar las perturbaciones creadas al momento de ser capturada por la cámara, de esta forma la imagen en escala de grises aumenta mejor su calidad, para posteriormente pasar por la binarización.

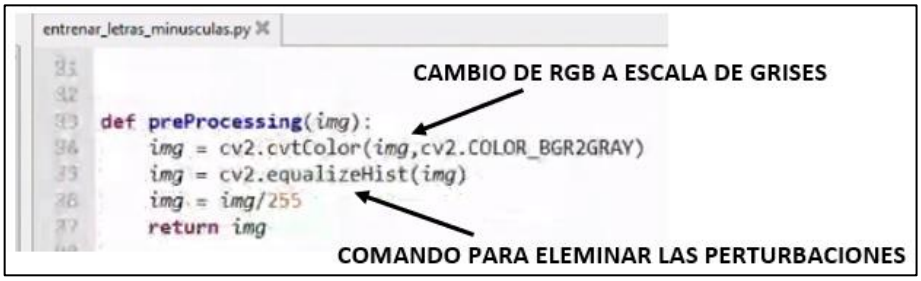


Figura 4.19. Comando de Binarización.

Fuente: Autores.

En el proceso de binarización, la imagen que se transforma a escala de grises y purificada de perturbaciones, el fondo de esta, toma un color blanco donde su valor binario corresponde a un 0 o cero lógicos, en cambio, que los valores que se tomaron en cuenta a partir de un determinado valor en la escala de grises, estos tomarán los valores de 1 o de uno lógico, dando como resultado una imagen clara y concisa del número o letra ingresado por la cámara.

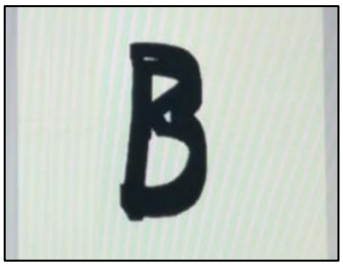


Figura 4.20. Imagen Binarizada.

Fuente: Autores.

4.3.3. Diseño redes neuronales

Dentro del desarrollo y entrenamiento en redes neuronales, existen diversos tipos y clasificaciones de redes neuronales, mismas que se detallan en marco teórico, para el diseño de la plataforma del dispositivo la aplicación de redes neuronales es la identificación tanto de números y letras ingresadas por la cámara, indistintamente como estén realizadas y diseñas las letras y números en las tarjetas de aplicación.

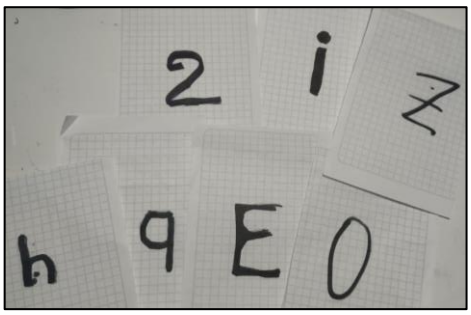


Figura 4.21. Tarjetas para el entrenamiento de las redes neuronales.

Fuente: Autores.

Para el diseño de la plataforma que identificara números y letras se toman varios parámetros en el desarrollo y entrenamiento estos son:

- Facilidad de implementación y uso dentro de la plataforma diseñada.
- Entrenamiento de la red, debe ser guiado o aplicado en el área de visión artificial.
- Las épocas de entrenamiento no deben superar al error esperado, debido al número de imágenes que se procesaran.
- Los pesos del entrenamiento serán porcentuales a la imagen que se ingresa en la comparación con el entrenamiento.
- La programación estará definida por el llamado simple a la red entrenada, sea esta de números o leras.

La red empleada para el desarrollo de la plataforma será RED NEURONAL CONVOLUCIONAL, que es una variación de un perceptrón multicapa la misma que presenta características particulares para el manejo de visión artificial en el desarrollo y aprendizaje en la clasificación y segmentación de imágenes.

Características:

Entrada: las imágenes ingresadas por la cámara se normarán a una escala de 32 pixeles tanto en ancho como largo, a fin de estandarizar la lectura de las imágenes ingresadas para posteriormente ser evaluadas. Los valores de evaluación serán alto, ancho y profundidad dando como resultado un valor de 1, cuando se trabaja en escala de grises.

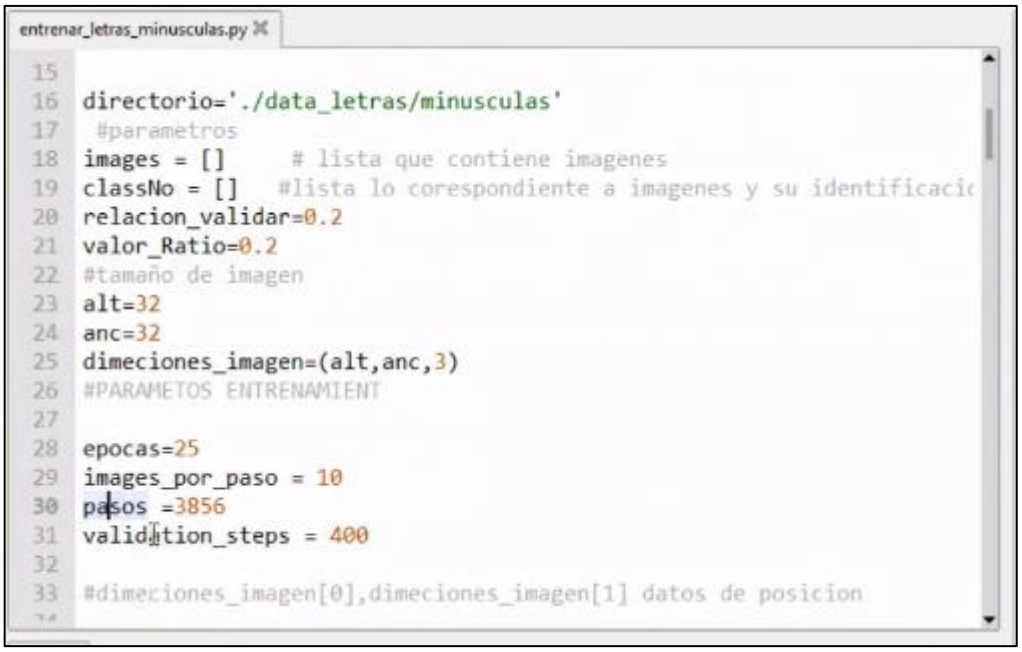


Figura 4.22. Líneas de código de lectura y estandarización.

Fuente: Autores.

Capa de convolución: procesará la salida de neuronas que están conectadas en regiones locales de entrada (es decir pixeles cercanos), calculando el producto escalar entre sus pesos (valor de pixel) y una pequeña región a la que están conectados en el volumen de entrada.

Funciones de transferencia: aplicará la función de activación en los elementos de la matriz en el caso de la plataforma se usará las funciones ReLU y Softmax.

Muestreo o Subsampling: Hará una reducción en las dimensiones alto y ancho, pero se mantiene la profundidad, esto a razón de que se entregue un valor entre 0 y 1 a la imagen evaluada.

Capa Tradicional: red de neuronas feedforward que conectará con la última capa de subsampling y finalizará con la cantidad de neuronas que se quiere clasificar. De esta forma se obtiene el valor y pesos de la imagen que se está evaluando.

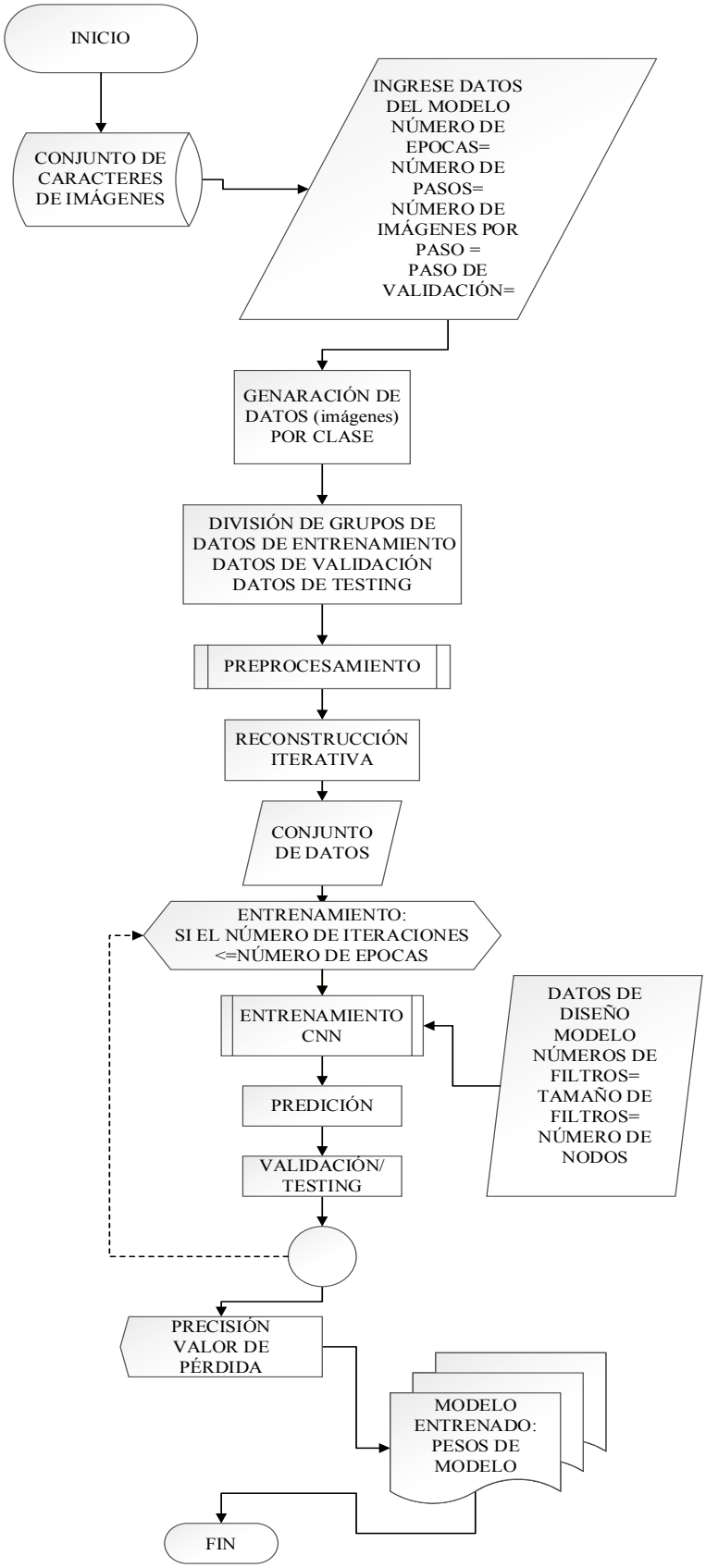


Figura 4.23. Diagrama del proceso de entrenamiento.

Fuente: Autores.

4.3.3.1. Funciones de activación

Una función de activación es la encargada de entregar un valor de salida a través de los valores de entrada esto incluye a la función de pre disposición conocida como Bias (es una función que controla que tan dispuesta esta la neurona a identificar un uno o cero dentro del entrenamiento), normalmente se define entre los valores de (1 o 0) y también se predispone en los valores negativos y positivos de (-1 o 1).

Dentro de redes neuronales existen diversos tipos de funciones de activación, de las cuales se aplicarán dos funciones de activación que cumplen con los requisitos de la plataforma creada en el dispositivo.

Función de activación ReLU - Rectified Lineal Unit (Unidad Lineal Rectificada)

Una de las más usadas dentro de redes neuronales, debido a que permite el aprendizaje muy rápido de los parámetros propuestos por el entrenador. Si a esta función se le da valores de entrada muy negativos el resultado esperado es un cero, pero si se le da valores positivos queda igual o toma el valor alto entrenado hasta la función, además el gradiente de esta función será cero en el segundo cuadrante y uno en el primer cuadrante. [30]

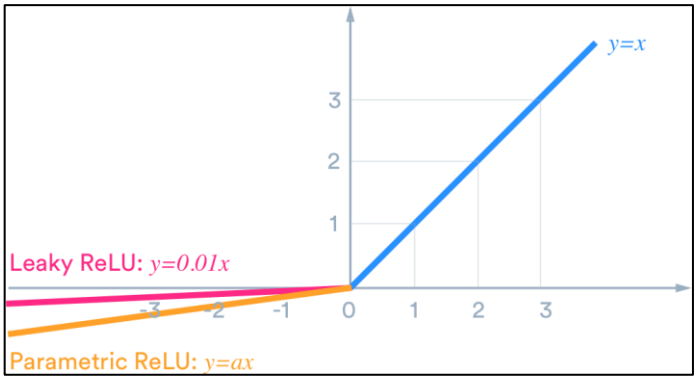


Figura 4.24. Función de activación ReLU.

Fuente: [30]

Se podría decir que la función ReLU es una función escalón modificada donde los valores de entramiento tienen una ponderación comprendida entre los valores de salida, los mismos que dependen de la vía por donde se ubicaran los datos correctos.

$$f(x) = \max(0, x) = \begin{cases} 0\ para\ x < 0 \\ x\ for\ x \geq 0 \end{cases} \quad (4.1)$$

Ecuacion representativa de la funcion ReLU

Dentro del desarrollo de la plataforma del dispositivo la función de transferencia ReLU, se usa para el primer entrenamiento de las imágenes almacenadas, a fin de proporcionar los mejores valores al entrenamiento de la red neuronal aplicada.

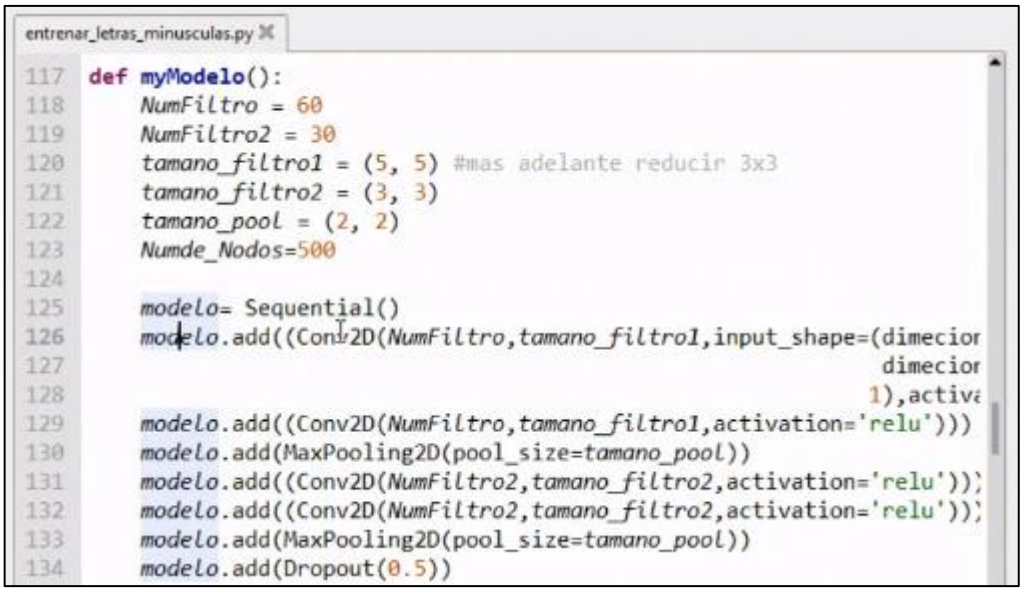


Figura 4.25. Líneas de código para la función de activación ReLU.

Fuente: Autores.

Función de activación Softmax

Conocida también como regresión logística multinomial es una generalización de la regresión logística cuando se trata de manejar múltiples clases (múltiples instrucciones de entramiento dentro de una sola salida). Se caracteriza por emplear en redes neuronales una clasificación binaria en base al desarrollo de su entrenamiento en la misma. [31]

$$f(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zj}} \tag{4.2}$$

Donde:

Z es un vector de las entradas a la capa de salida.

J es donde se indexa las unidades de salida

Dentro de la plataforma diseñada, la regresión logística simple no es suficiente. Se necesitó una distribución de probabilidad en todas las etiquetas, que es lo que nos brinda la función de transferencia softmax.

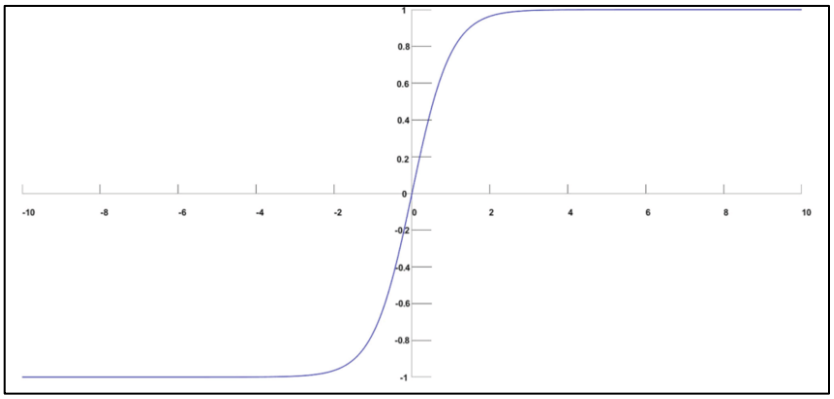


Figura 4.26. Función de activación Softmax.

Fuente: [30]

Esta función interactúa en base a la probabilidad del entrenamiento de letras o números de la siguiente forma: la imagen de una letra o número es ingresada por la cámara y se solicita saber el tipo de letra o número a que pertenece, aplicando la función de transferencia softmax la red nos dará la probabilidad de que puede ser 0.3 o 30% letra o número, etc.

Por lo que el resultado será el que tenga mayor probabilidad y cabe recalcar que la suma de estas probabilidades será igual a 1 o 100%. En otras palabras, Softmax se usa para clases múltiples y cuando se va a asignar probabilidades a cada clase que de un solo conjunto en este caso números o letras. [30]

entrenar_letras_minusculas.py

```
125     modelo= Sequential()
126     modelo.add((Conv2D(NumFiltro,tamano_filtro1,input_shape=(dimecior
127                                                              dimecior
128                                                             1),activa
129     modelo.add((Conv2D(NumFiltro,tamano_filtro1,activation='relu')))
130     modelo.add(MaxPooling2D(pool_size=tamano_pool))
131     modelo.add((Conv2D(NumFiltro2,tamano_filtro2,activation='relu')));
132     modelo.add((Conv2D(NumFiltro2,tamano_filtro2,activation='relu')));
133     modelo.add(MaxPooling2D(pool_size=tamano_pool))
134     modelo.add(Dropout(0.5))
135
136     modelo.add(Flatten())
137     modelo.add(Dense(Numde_Nodos,activation='relu'))
138     modelo.add(Dropout(0.5))
139     modelo.add(Dense(Num_deClases,activation='softmax'))
140     modelo.compile(Adam(lr=0.001),loss='categorical_crossentropy',
141                    metrics=['accuracy'])
142     return modelo
```

Figura 4.27. Conversión de la función ReLU a la función Softmax.

Fuente: Autores.

La función de transferencia Softmax se la puede observar en sus valores arrojados, también conocidos como pesos, donde en la imagen tratada indicará en porcentaje a que letra pertenece,

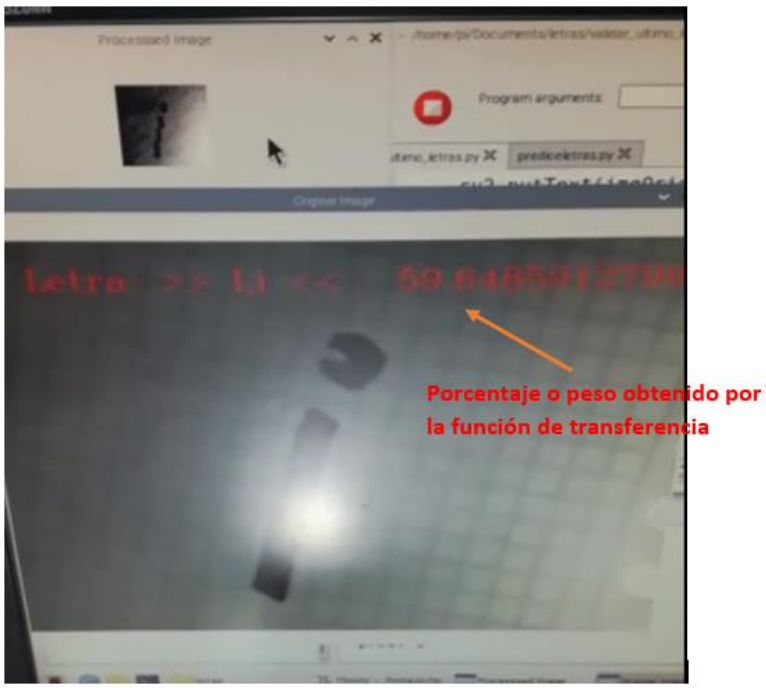


Figura 4.28. Ponderación del entrenamiento Softmax %.

Fuente: Autores.

4.4. Cálculo del tamaño de muestra

El tamaño de la muestra permite conocer cuántos ensayos es necesario estudiar para poder determinar el grado de confianza deseado, o el número necesario para poder identificar una determinada diferencia, en este caso, entre números y letras siendo estos los grupos de estudio. El cálculo del tamaño de la muestra se realiza mediante una función matemática que expresa la relación entre las variables, cantidad de participantes y poder estadístico.

Los factores estadísticos que determinan el tamaño de la muestra son: hipótesis, error alfa, error beta, poder estadístico, variabilidad, pérdidas en el estudio y el tamaño del efecto. También se revisan aspectos importantes como: tamaño de la muestra para estudios piloto, estrategias para disminuir el número necesario de sujetos, y software para el cálculo del tamaño de muestra.

La siguiente ecuación sirve para el cálculo del tamaño de la muestra. [32]

$$\boldsymbol{n = \frac{z^2 * p * q * N}{e^2(N-1) + z^2 * p * q}} \tag{4.3}$$

Donde:

n = muestra.

N = población o datos.

p = probabilidad a favor

q = probabilidad en contra

z = nivel de confianza

e = error de muestra

Tabla 4.8. Datos del nivel de confianza**.**

Nivel de confianza	Z α
99.7%	3
99%	2.58
98%	2.33
96%	2.05
95%	1.96

Fuente: [32]

Conociendo que el dispositivo de identificación de números y letras posee tres grupos de datos los cuales son:

Tabla 4.9. Datos de detección.

Datos	Cantidad
Letras Mayúsculas	26
Letras Minúsculas	26
Números	10
Total	62

Fuente: Autores

Esto es tomando en cuenta que para el aprendizaje del abecedario en niños de educación básica se lo hace diferenciando letras mayúsculas de letras minúsculas. Aplicando la ecuación 4.3 se obtiene:

$$\boldsymbol{n = \frac{1.96^2 * 0.5 * 0.5 * 62}{0.05^2(62-1) + 1.96^2 * 0.5 * 0.5} = 53.50} \tag{4.4}$$

En la ecuación anterior se considera un nivel de confianza del dispositivo del 95%, un error del 5%, una probabilidad en contra o a favor de que detecte la letra o número del 50% y tomando en cuenta el número total de datos igual a 62, se obtiene un resultado aproximado de 54 repeticiones.

4.5. Diseño Mecánico

Se toma en cuenta el diseño y construcción de la carcasa que forma parte del dispositivo, la misma que se construye en impresión 3D, con filamento de fibra de carbono en base a PLA, (ácido poliláctico) o polímero biodegradable, esto debido a la dureza y resistencia que muestra este material a la deformación ya sea por calor o por tensión.

Tabla 4.10. Características técnicas del material utilizado.

Especificación del PLA con Fibra de Carbono	
Resistencia a la flexión:	57 MPa
Temperatura de fusión:	190 ° C – 230 ° C
Resistencia a la tracción:	45.5MPa
Elongación en el descanso:	(73 ° F) 320%
Tolerancia estándar:	± 0.05mm
Espesor de la capa:	3 mm
Dureza Shore:	45D
Densidad:	1.3 g / cm3 (1300 kg / m3)
Desviación del calor:	21% a 85 ° C
Contracción:	Muy bajo, cuando se enfría a una temperatura ambiente elevada

Fuente: [32]

El software para el desarrollo mecánico que se emplea en el dispositivo será SolidWorks, a fin de obtener en el mismo la resolución de los parámetros mecánicos necesarios para obtener resultados favorables.

Parámetros de diseño:

Modular: la carcasa a crear debe ser de fácil acceso hacia sus distintos componentes, a fin de que si existiese algún defecto mecánico estos elementos se podrían remplazar, no debe existir aberturas por donde puedan ingresar objetos que puedan dañar el dispositivo.

Robustez: dentro del diseño mecánico debe existir rigidez en la carcasa de forma que esta sea compacta y resistencia a golpes moderados.

Acabado: la carcasa debe contar con un acabado fino y semi pulido a fin de que esta no sea resbalosa y se sienta firme en la superficie donde se va a asentar.

Pantalla: la pantalla debe ser visible y debe estar apoyado en una carcasa robusta y esta debe separarse de la estructura base,

Puertos: los puertos deben estar en taladros bien definidos, para que sean fáciles de colocar los periféricos de entrada en la tarjeta programadora base.

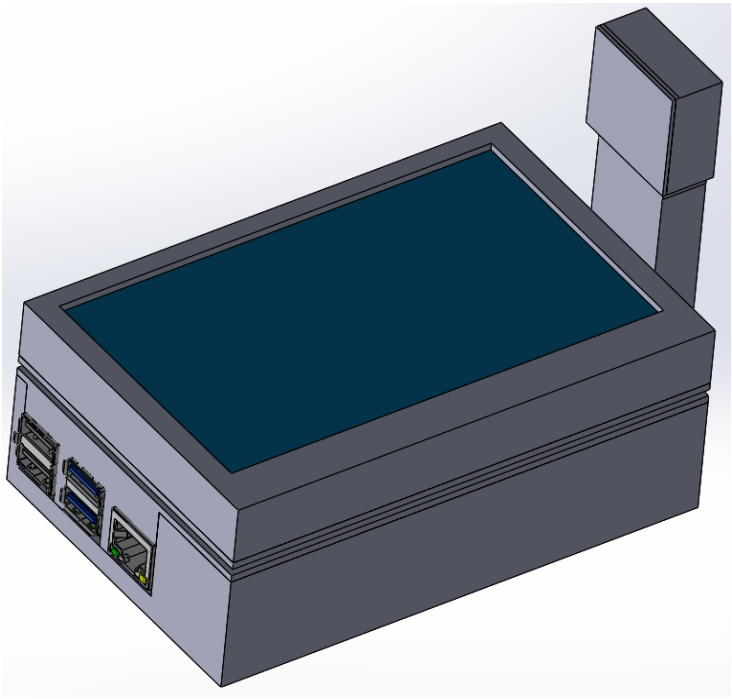


Figura 4.29. Diseño final del dispositivo para detección de números y letras.

Fuente: Autores.

El diseño se compone de varias etapas la primera es la construcción de la carcasa de la pantalla HMI, la misma que se realiza en opción multi sólidos, y se genera dos diseños, tapa y carcaza.

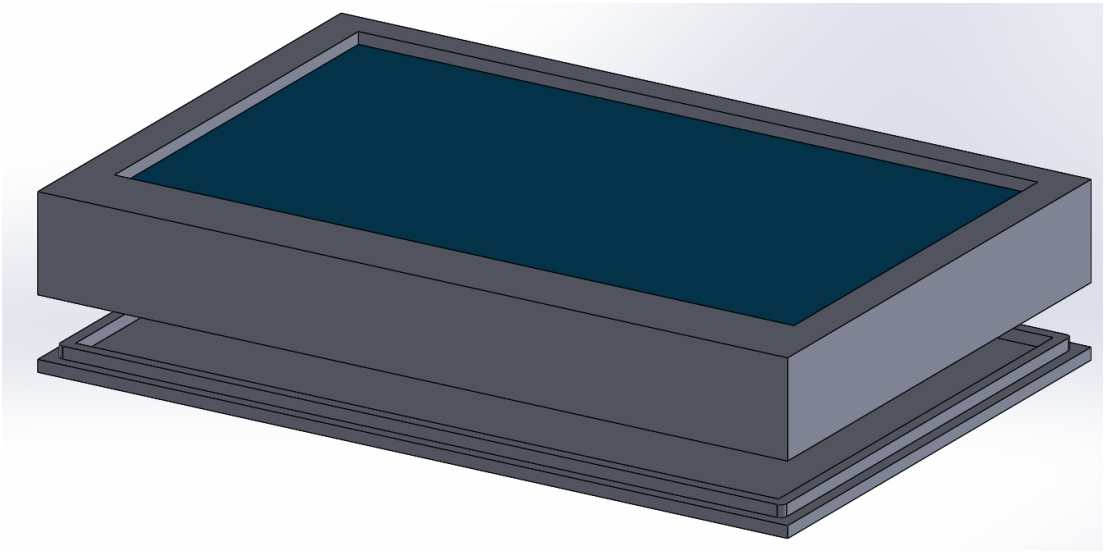


Figura 4.30. Diseño de carcasa y tapa de la pantalla HMI del dispositivo.

Fuente: Autores.

El segundo diseño a realizar, es la estructura de la cámara, así como su tapa, en donde se aloja el Flex y la cámara de la tarjera programadora.

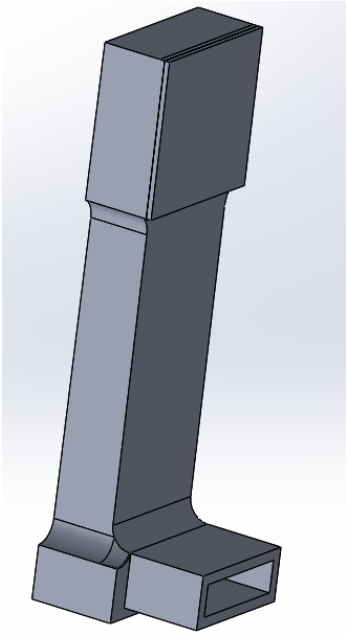


Figura 4.31. Diseño de la carcasa y tapa de la cámara Pi.

Fuente: Autores.

El ultimo diseño en realizar se compone de la tapa y base donde se alojará la tarjeta programadora, esta deberá tener taladros por donde se pueda ingresar los periféricos de control, así como su alimentación.

Figura 4.32. Diseño de la carcasa donde se aloja la tarjeta Raspberry.

Fuente: Autores.

Después de realizado el diseño mecánico, este se exporta en formatos STL. Para realizar la impresión 3D, los parámetros de impresión se basan en los contemplados en la tabla de este anexo.

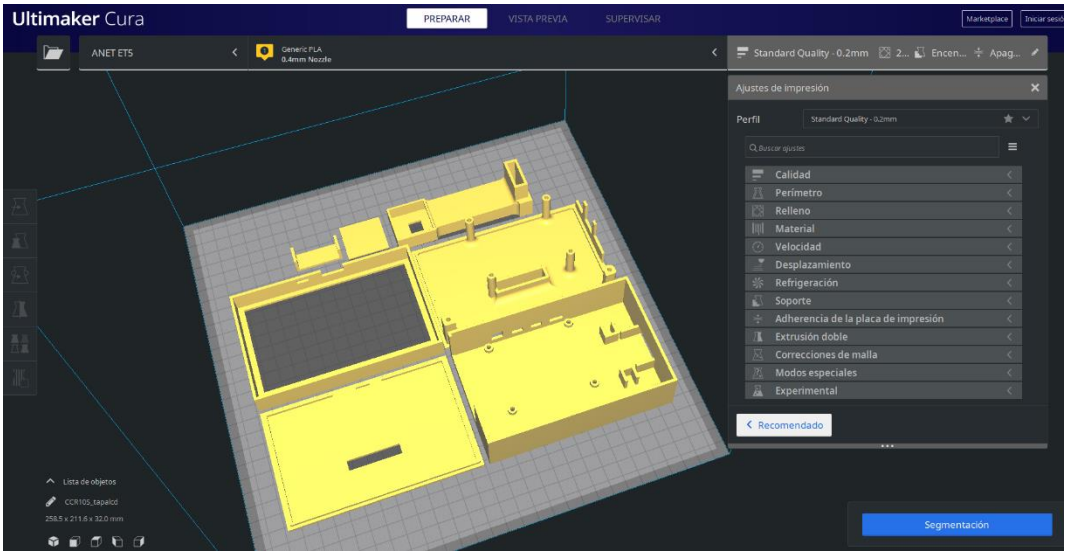


Figura 4.33. Plataforma para generación del código de impresión

Fuente: Autores.

5. ANÁLISIS Y DISCUSIÓN DE RESULTADOS.

Los diferentes procesos de diseño generan parámetros, que se pueden analizar si cumplen o no con sus respectivas funciones establecidas.

5.1. Procesamiento de imágenes

Dentro del procesamiento de imágenes se puede verificar si efectivamente se cumple con la transformación de la misma en los parámetros establecidos en la metodología, para posteriormente verificar si al momento de aplicar la red neuronal esta es capaz de identificar qué tipo de imagen.

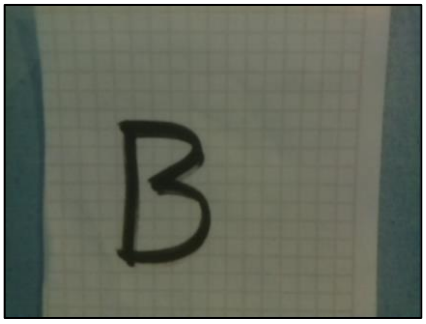


Figura 5.1. Imagen de la letra B obtenida por la cámara.

Fuente: Autores

El primer proceso será la transformación de la imagen original a escala de grises.

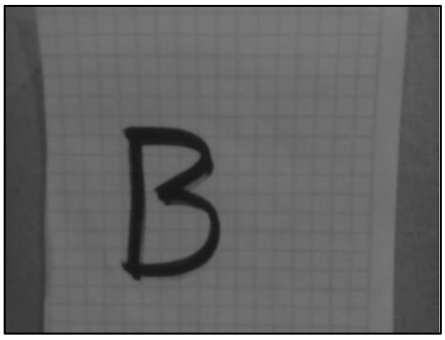


Figura 5.2. Imagen en escala de grises letra B.

Fuente: Autores

Después de la transformación de la imagen real a escala de grises esta pasa por el proceso de eliminación de ruidos y perturbaciones, para posteriormente invertir la coloración de las imágenes y así dejar a la imagen en dos rangos de colores que serán blanco y negro.

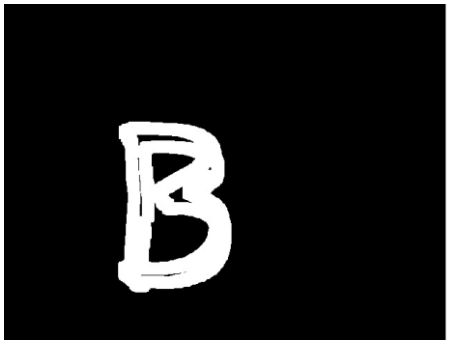


Figura 5.3. Imagen libre de perturbaciones obtenida por la cámara.

Fuente: Autores

Se aplica por último la binarización de la imagen en la cual se invierten de nuevo los colores base, dando como resultado una imagen clara y concisa de lo que vamos a analizar. Estos datos posteriormente pasaran al proceso de entrenamiento o proceso de lectura dentro de la plataforma.

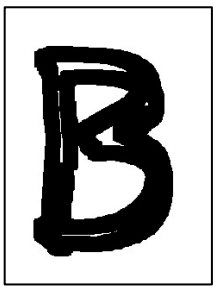


Figura 5.4. Imagen binarizada obtenida por la cámara.

Fuente: Autores

5.2. Entrenamiento red neuronal

El entramiento de la red neuronal se realiza con un promedio de 620 a 680 imágenes por cada letra o número que se va a entrenar, de esta ponderación al momento de establecer y ejecutar el entrenamiento de las redes neuronales estas nos arrojan un promedio en base a un histograma del desarrollo que se aplicó a cada imagen.

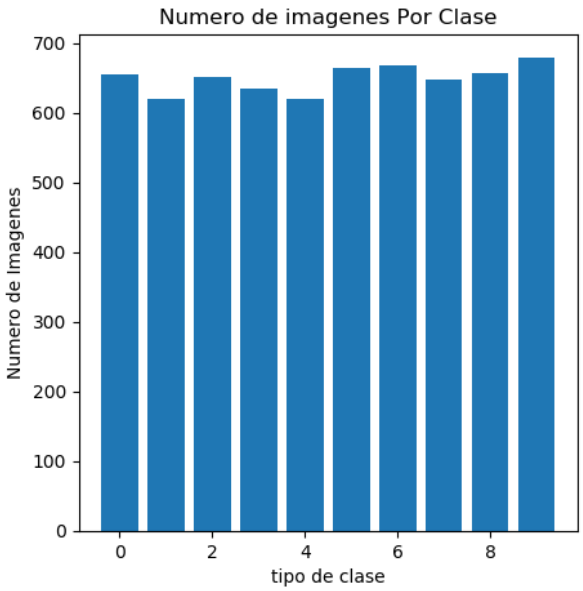


Figura 5.5. Ponderación de número de imágenes por clase.

Fuente: Autores

Podemos aplicar lo mismo para el entrenamiento de la red neuronal de letras y el siguiente histograma lo verifica.

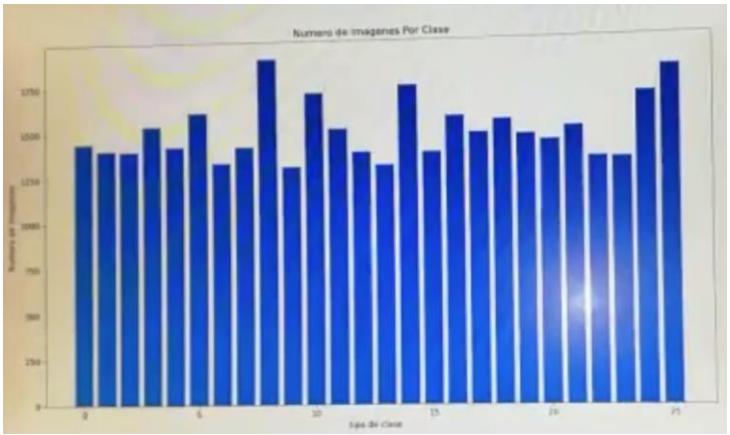


Figura 5.6. Ponderación de letras de imágenes por clase.

Fuente: Autores

Después de arrojar el histograma, se procede con el entrenamiento de la red neuronal de números y leras, el cual dura un promedio de 11 a 12 horas por cada entrenamiento, dependiendo del tipo de imagen y la capacidad de la tarjeta programadora.

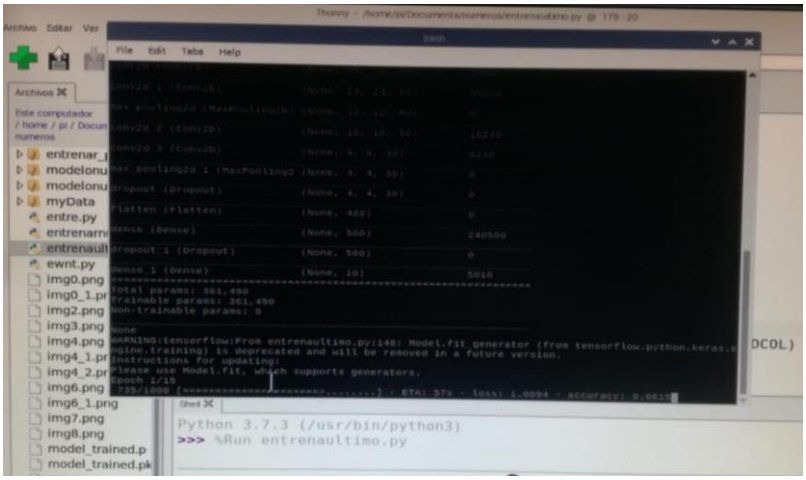


Figura 5.7. Desarrollo del entrenamiento de la red neuronal.

Fuente: Autores

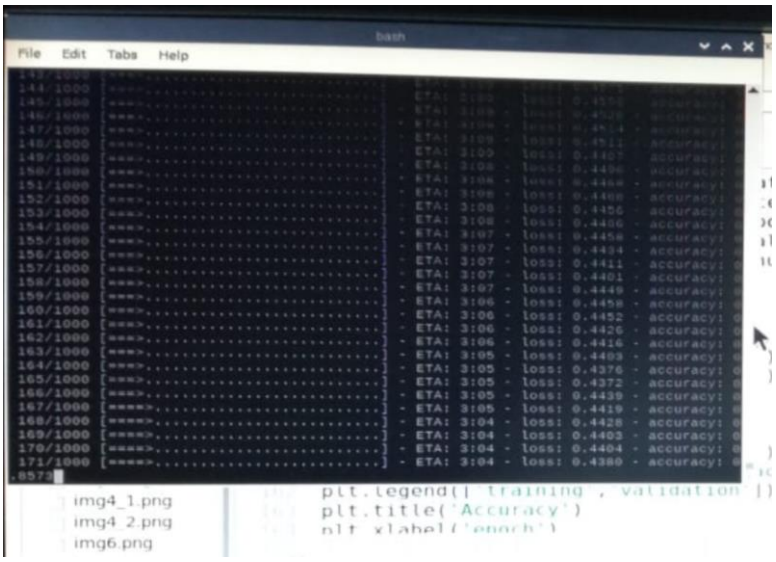


Figura 5.8. Lectura de imágenes y asignación dentro de la red neuronal.

Fuente: Autores

Al momento de culminar el entrenamiento de la red neuronal de números y letras el programa arroja el valor de la exactitud de validación y de entrenamiento de la red neuronal. En este proceso en la imagen 5.9 se muestra el recorrido que tomo la validación de los datos conforme avanzó en el proceso de entrenamiento. Se puede observar que ambos valores casi van a la par, por lo cual se puede decir que el programa tiene un 95 a 98% de exactitud para la detección de números y letras.

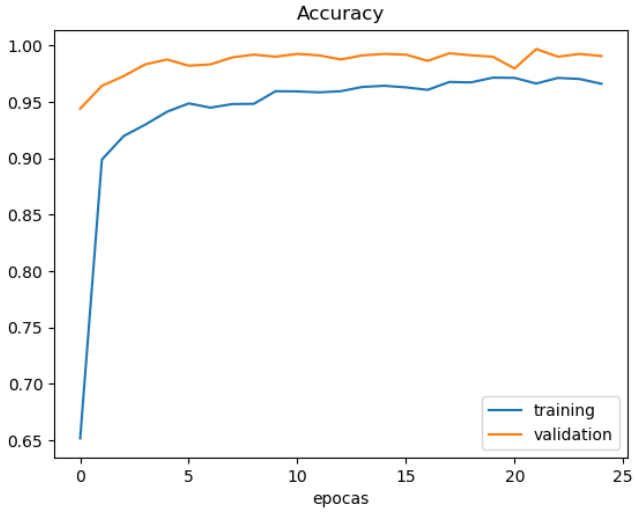


Figura 5.9. Porcentajes de exactitud por épocas.

Fuente: Autores.

También arroja en contraste el número de épocas perdidas dentro de la validación de los datos que tienden a un valor de cero.

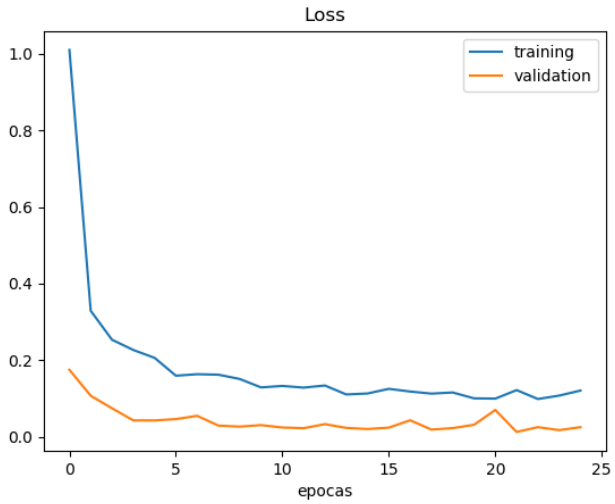


Figura 5.10. Valores de perdida por épocas.

Fuente: Autores.

Al momento de aplicar o entrenar la segunda capa o red neuronal arroja valores de su entrenamiento en porcentaje, de estos valores se obtiene los pesos enteros como se muestra dentro de la siguiente tabla.

Tabla 5.1. Valores del entrenamiento mediante Softmax.

Valores porcentuales del entrenamiento softmax										
a	89	**g**	92	**m**	93	**s**	90	**y**	94	**4**	89
b	94	**h**	98	**n**	95	**t**	91	**z**	91	**5**	93
c	92	**i**	88	**o**	90	**u**	93	**0**	97	**6**	90

d	93	**j**	89	**p**	95	**v**	88	**1**	89	**7**	88
e	91	**k**	90	**q**	94	**w**	95	**2**	90	**8**	95
f	90	**l**	90	**r**	96	**x**	96	**3**	95	**9**	91

Fuente: Autores.

5.3. Caracteres especiales

Dentro del desarrollo de la plataforma del dispositivo existen letras o números que podrían interpretarse en otras capas o bloques de programación, a estos se les considera como caracteres especiales.

Cuando se analiza letras minúsculas dentro del bloque de programación de números, algunas de ellas si pueden ser detectadas en su programación, estas podrán ser o, l, i. esto se debe al parecer que presentan estas letras con los números 0 y 1 respectivamente.

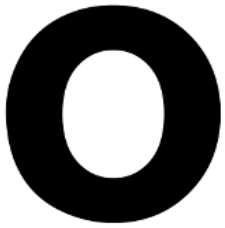


Figura 5.11. Letra “o” analizada en caracteres de números.

Fuente: Autores.

Por consecuente lo mismo puede suceder cuando se analiza letras mayúsculas en el bloque de programación de números, el efecto de estas también corresponden a las letras O, L, I, por su peculiar forma.

Cuando ocurre el efecto contrario es decir al analizar números dentro del bloque de programación de letras mayúsculas y minúsculas se presenta el mismo efecto.

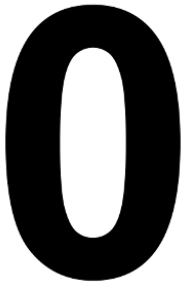


Figura 5.12. Número cero, evaluado en mayúsculas y minúsculas.

Fuente: Autores.

5.4. Evaluación diseño mecánico

Para la evaluación del diseño mecánico se tomará en cuenta solo la base el dispositivo ya que la misma al ser la que va a soportar todo el peso y la fricción contra la superficie de prueba va hacer la más crítica.

Para ello primero se declara el material con el que se va a trabajar o simular.

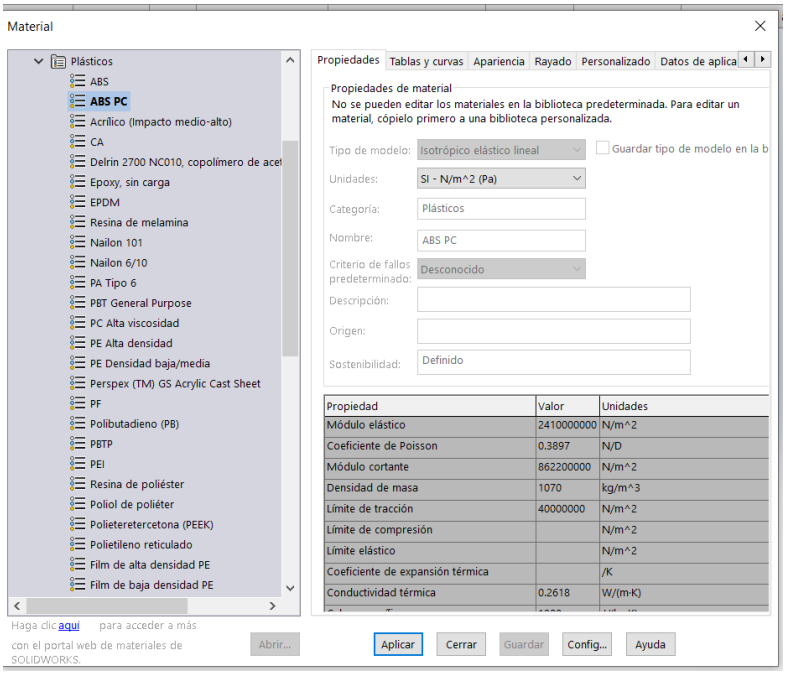


Figura 5.13. Selección del material para la simulación.

Fuente: Autores.

Cuando se aplique el material se cargan las características del mismo, para ser analizadas, según los parámetros de comportamiento térmico a ver si efectivamente le material impreso va a soportar las temperaturas de trabajo.

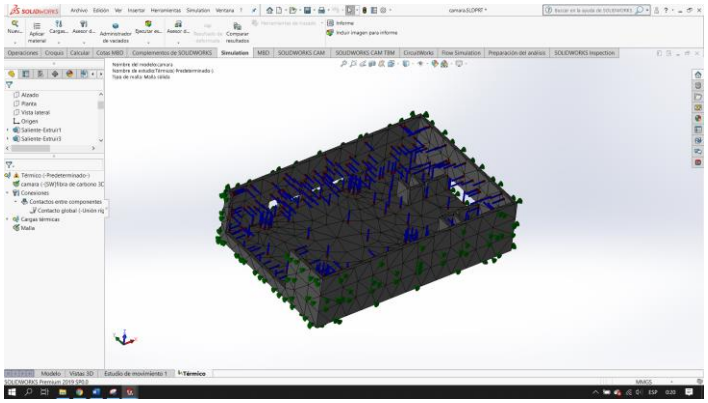


Figura 5.14. Parámetros de mallado para simulación de temperatura.

Fuente: Autores.

En la siguiente imagen se puede observar que los elementos que conforman el dispositivo no se van a deformar por calor ya que su temperatura máxima para llegar a la deformación es de 150 grados.

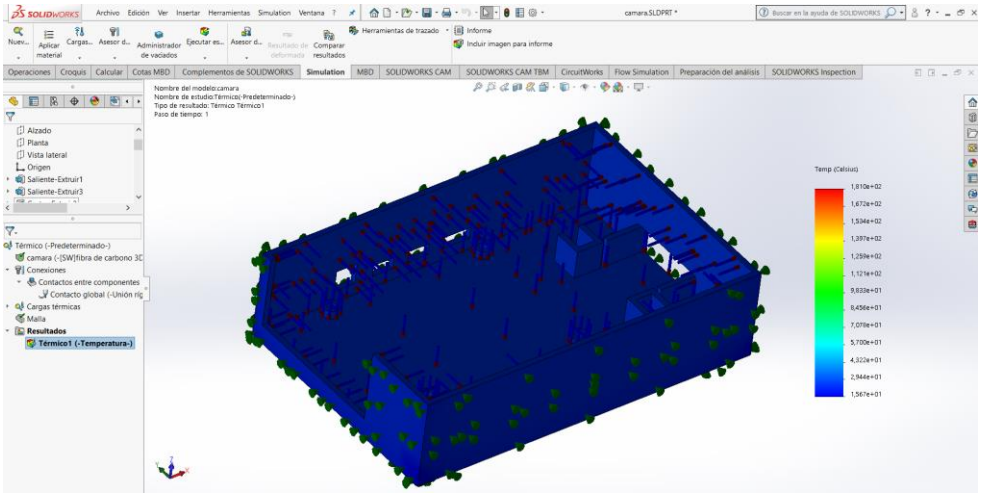


Figura 5.15. Análisis térmico aplicada a la caja del dispositivo.

Fuente: Autores.

Este análisis también nos indica que el mismo efecto pasara para las demás piezas esto debido a que todas estas están bajo las mismas condiciones de trabajo y temperatura que la pieza analizada.

Por último, se evalúa la misma pieza a los efectos de análisis estáticos a las fuerzas que se van a aplicar dentro del proyecto que sumado dan una fuerza de 1.8 lb lo cual es un valor bajo para lo que soporta el material impreso.

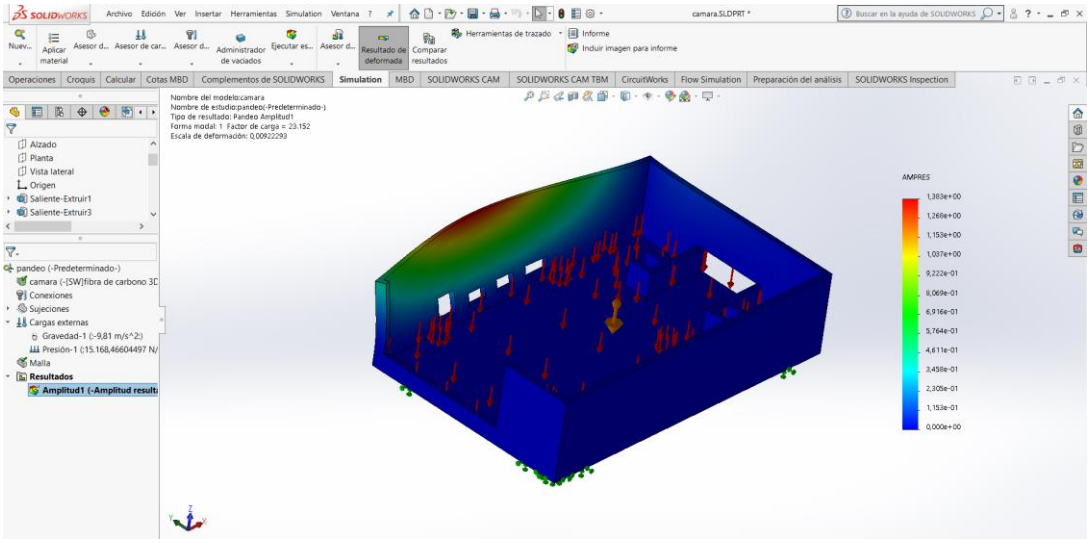


Figura 5.16. Material impreso analizado en fuerzas estáticas.

Fuente: Autores.

5.5. Comparativa entre imágenes

Se analiza el resultado mostrado entre la imagen adquirida por cámara y la almacenada para el llamado a la comparación. El resultado es prácticamente efectivo al momento de dar lectura a una imagen ingresada.

Figura 5.17. Obtención de una comparativa entre imágenes.

Fuente: Autores.

5.6. Análisis del tamaño de muestra.

De acuerdo a la ecuación (4.4), se determina que el número de pruebas de funcionamiento a realizarse en el dispositivo son 54.

Para la el desarrollo de pruebas se procede a tomar datos escogidos de números y letras de manera aleatoria.

Tabla 5.2. Validación de funcionamiento del dispositivo.

Número de pruebas	Dato mostrado Número / Letra	Dato identificado Número/Letra	Detecta Si/No
1	A	A	Si
2	b	b	Si
3	C	C	Si
4	1	1	Si
5	d	d	Si
6	E	E	Si
7	2	2	Si
8	f	f	Si
9	G	G	Si
10	H	H	Si

11	3	3	Si
12	I	I	Si
13	J	J	Si
14	K	K	Si
15	L	L	Si
16	4	4	Si
17	m	m	Si
18	N	N	Si
19	O	O	Si
20	P	P	Si
21	Q	Q	Si
22	r	r	Si
23	s	s	Si
24	t	t	Si
25	5	5	Si
26	u	u	Si
27	V	V	Si
28	W	W	Si
29	6	6	Si
30	7	7	Si
31	X	X	Si
32	8	8	Si
34	Y	Y	Si
35	0	O	No
36	Z	Z	Si
37	9	g	No
38	g	g	Si
39	2	2	Si
40	1	1	Si
41	r	r	Si
42	7	7	Si
43	5	5	Si

44	K	K	Si
45	n	n	Si
46	C	C	Si
47	3	3	Si
48	q	q	Si
49	l	l	Si
50	x	x	Si
51	T	T	Si
52	z	z	Si
53	D	D	Si
54	9	9	Si

Fuente: Autores.

Mediante el cálculo de tamaño de la muestra se pudo determinar que, de 54 pruebas realizadas al dispositivo, 52 datos fueron identificados correctamente y 2 fueron erróneos.

Cabe mencionar que existen parámetros especiales en algunos datos que dificulta la identificación como lo es la letra "O" y el cero, ya que ambos datos tienen rasgos de escritura similares.

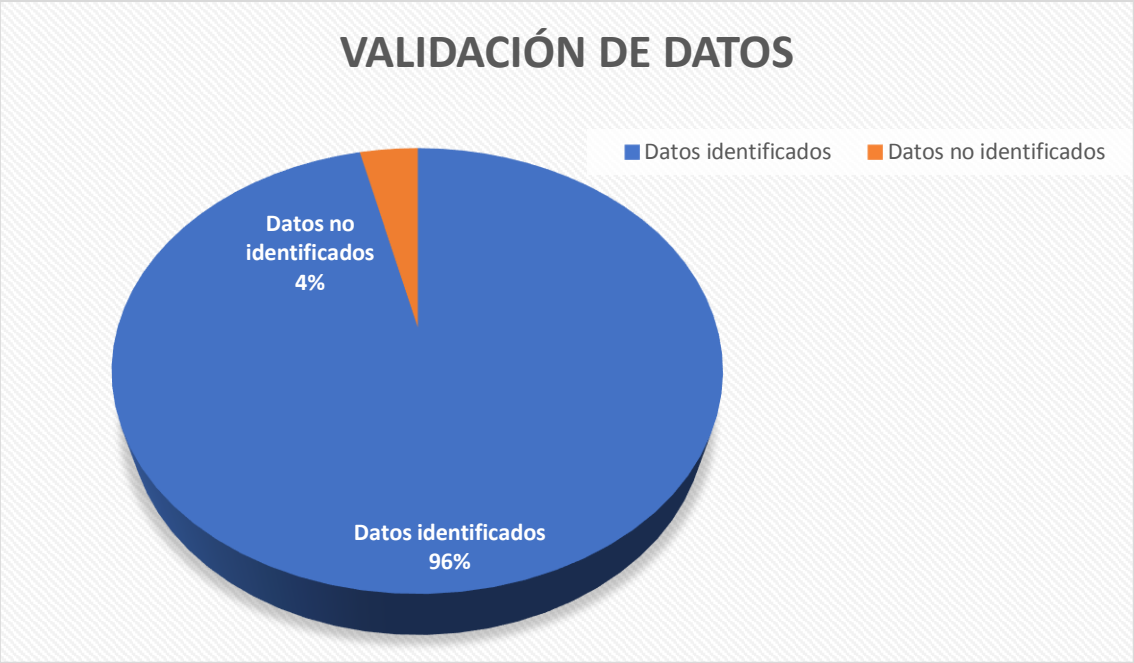


Figura 5.18. Porcentaje de validación de funcionamiento.

Fuente: Autores.

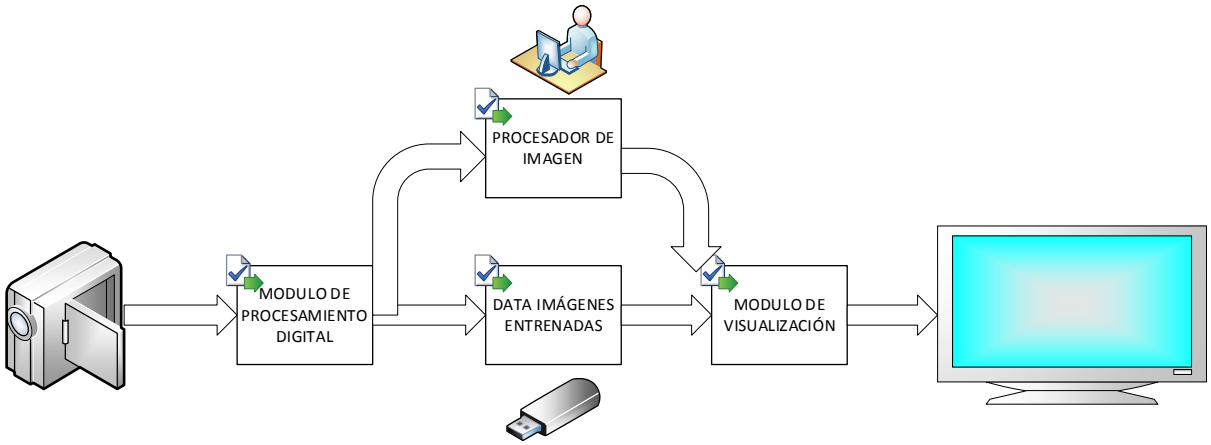


Figura 5.19. Diagrama de funcionamiento

Fuente: Autores.

6. PRESUPUESTO Y ANÁLISIS DE IMPACTOS

6.1. Presupuesto.

En las tablas que se ilustran a continuación se puede apreciar el costo de los materiales y equipos a utilizar para el proyecto.

6.1.1. Costos Directos

Los costos directos para el desarrollo del dispositivo de identificación de números y letras se detallan en la tabla 6.1.

Tabla 6.1. Costos Directos.

Cantidad	Detalle	Valor $
1	Kit Raspberry Pi 4 modelo B incluye: (cargador, case acrílico, cable USB, cable HDMI, 3 disipadores, 1 ventilador y una tarjeta SD de 32 GB.	140
1	Cámara 5 MPX	30
1	Pantalla programable TFT 5 pulgadas HD	60
1	Carcasa e impresión.	100
1	Mouse (ratón) para PC	12
1	Parlantes para PC	15
1	Cableado	20
TOTAL:		**377 $**

Fuente: Autores.

6.1.2. Costos Indirectos

Los costos indirectos para el desarrollo del dispositivo de identificación de números y letras se detallan en la tabla 6.2.

Tabla 6.2. Costos Indirectos.

Número	Detalle	Valor $
1	Asesoría externa	75
2	Servicios solicitados a terceros	80
3	Mano de obra	150
4	Transporte al interior de la Provincia	50
5	Costos de envío	20
6	Imprevistos	100
TOTAL:		**475 $**

Fuente: Autores.

6.1.3. Inversión total

El gasto total para el desarrollo del dispositivo de detección de números y letras se deriva de los costos directos e indirectos y se detalla en la tabla 6.3.

Tabla 6.3. Inversión total del proyecto.

Número	Rubro	Valor$
1	Costos directos	377$
2	Costos Indirectos	475$
TOTAL		**852$**

Fuente: Autores.

6.2. Análisis de Impactos.

Impacto Práctico: En la actualidad, las tecnologías que están influyendo en la sociedad, son las relacionadas con la informática y las comunicaciones, en las que destacan el Internet y las comunicaciones móviles. Sin embargo, El dispositivo de identificación de números y letras, no necesita de una conexión permanente a internet para su funcionamiento, permitiendo facilitar a docentes, a que impartan sus conocimientos en el área de educación básica de una manera tecnológica conllevando a que los estudiantes se interesen con mayor énfasis en aprender de mediante la innovación, sin la necesidad usar equipos de tamaño extravagante.

Impacto Tecnológico:

El desarrollo de la tecnología se ha convertido en uno de los productos fundamentales del consumo de la modernidad. Esto trae consigo cambios que repercuten en los procesos y fenómenos sociales, y más específicamente en la mente de los niños, en su forma de vivir, pensar y aprender. El dispositivo hace posible el reconocimiento de números y letras mediante la programación de software y hardware libre, Estas tecnologías de uso libre de licencias, en su mayor parte, han facilitado el desarrollo de la sociedad y han dotado de herramientas para afrontar problemas que talvez antes no tenían solución en su momento. El desarrollo del presente proyecto permite usar el dispositivo de manera didáctica y con una interacción amigable con el usuario, sin la necesidad de conexión a internet y con gran efectividad para la enseñanza en el área de la educación estudiantil.

7. CONCLUSIONES Y RECOMENDACIONES.

7.1. Conclusiones.

- El dispositivo cuenta con un sistema de visión artificial elaborado con elementos adquiridos en el mercado ecuatoriano, el cual permite identificar letras y números en tiempo real.
- La utilización de hardware y software libre, permite un amplio campo para desarrollar diversos tipos de proyectos, sin la necesidad de adquirir una licencia para poder ejecutar una idea plasmada o una propuesta.
- El uso de este tipo de plataforma de hardware y software libre, ha resultado en una reducción de los costos del proyecto, así como ha permitido una mayor portabilidad y facilidad de uso.
- El dispositivo desarrollado permite trabajar en tiempo real con visión artificial, de esta manera el usuario puede visualizar la detección de números y letras e interpretar el resultado de dicha detección.
- Se desarrolló un dispositivo que permite ofrecer apoyo al área estudiantil, lo cual se logró usando plataformas de licencia libre, obteniendo como resultado, la detección de números y letras.

7.2. Recomendaciones.

- Para un mejor rendimiento se recomienda aislar el dispositivo evitando que la iluminación externa afecte el correcto funcionamiento a fin de que pueda trabajar sin inconvenientes bajo cualquier ambiente.

- Para evitar errores de detección, se recomienda usar las cartillas de números y letras a la distancia para la cual el dispositivo está diseñado.
- Es importante para ampliar la funcionabilidad del prototipo que se puede implementar un sistema que identifique cualquier tipo de letras y números en cualquier tamaño y formato de escritura.
- Tomar especial énfasis en la etapa de entrenamiento ya que de acuerdo a ella se obtendrán los resultados, es por ello que para mejorar la calidad de reconocimiento se puede entrenar una vez más las redes neuronales.
- Para una mejor detección de caracteres especiales y que tienden a confundir la red es necesario tener una buena etapa de binarización y filtrado de la imagen.

8. REFERENCIAS

[1] J. M. Gómez, «Sistema de reconocimiento facial basado en redes neuronales convolucionales sobre el dispositivo Raspberry Pi,» Sevilla España, 2019.

[2] M. B. V. Calderón, «Modulo didáctico de entrenamiento de redes neuronales para el reconocimiento de patrones de imágenes y voz con Raspberry Pi,» Ibarra Ecuador, 2018.

[3] S. V. y. C. Gómez, «MONITOREO DE VARIABLES ANALOGICAS USANDO RASPBERRY PI,» Puerto Ordaz Venezuela, 2013.

[4] I. T. G. García, «Diseño e implementacion de sistema interactivo de información de docentes, con Raspberry Pi,» Guayaquil Ecuador, 2015.

[5] M. J. R. Gonzáles, «Reconocimiento de imágenes y datos financieros en streaming.,» Sevilla España, 2018.

[6] G. V. L. J. P. V. L. A. D. R. Miguel Quiroz Martínez, «Detección de Personal no autorizado en el departamento de TI, utilizando redes neuronales convolucionales en tiempo real con Raspberry Pi 3 B+,» *Revista Ciencia e Investigación,* vol. 5, nº 3, pp. 49-60, 2020.

[7] G. Benabides, «Visión artificial, la innovación disruptiva en la educación.,» Bogotá, Colombia.

[8] COGNEX, «WWW.COGNEX.COM,» 2016. [En línea]. Available: http://www.ikusmen.com/documentos/descargas/3cbb38_Introduction%20to%20Machine%20Vision.pdf. [Último acceso: 03 06 2020].

[9] S. Ingenieril, «Solucion Ingenieril,» 2017. [En línea]. Available: http://solucioningenieril.com/vision_artificial/etapas_de_un_sistema_de_vision. [Último acceso: 28 07 2020].

[10] Pixelvision, «Pixelvision.com,» 4 mayo 2019. [En línea]. Available: pixelvision.com.mx/como-reconocer-una-mano-con-vision-artificial/. [Último acceso: 28 07 2020].

[11] F. A. E. y. o. Ana Gonzales Marcos, «Técnicas y algoritmos básicos de vsión artificial.,» 2006. [En línea]. Available: https://www.researchgate.net/publication/231521316_Tecnicas_y_algoritmos_basicos_de_vision_artificial_Recurso_electronico_-_En_linea. [Último acceso: 04 06 2020].

[12] C. A. O. Clavel, «Modelado y simulación de un sistema de detección de intrusos utilizando redes neuronales recurrentes,» Cholula Puebla Mexico, 2007.

[13] V. J. M. H. José Ramón Hilera González, Redes Neuronales Artificiales, Madrid España, 2000.

[14] J. G. F. G. Iván González, «Hardware libre: Clasificación y desarrollo de hardware reconfigurable en entornos GNU/Linux,» 06 09 2003. [En línea]. [Último acceso: 28 07 2020].

[15] T. J. Lazalde A, «Hardware: Ecosistemas de innovacion y producción basados en hardware libre,» Quito Ecuador, 2015.

[16] M. Community, «Hacedores.com,» 2014. [En línea]. Available: https://hacedores.com/movimientomaker/. [Último acceso: 28 07 2020].

[17] X. Basics, «Que es arduio,» 03 08 2018. [En línea]. Available: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno. [Último acceso: 28 07 2020].

[18] L. D. C. Alexei Ochoa Duarte, «Alternativa Opensource en la implementación de un sistema IoT para la medición de la calidad del aire,» *Revista Cubana de ciencias Informáticas,* vol. 12, nº 1, pp. 189-200, 2018.

[19] R. G. Q. C. Fernando Fernández Castillo, «Sistema domótico de control de luces mediante una Targeta Raspberry PI.,» México, 2018.

[20] Raspberry, «Raspberry Pi.org,» [En línea]. Available: https://www.raspberrypi.org/. [Último acceso: 15 06 2020].

[21] Burundi, «Burundi Desertcart,» [En línea]. Available: https://burundi.desertcart.com/products/59401529-raspberry-pi-3-model-b. [Último acceso: 14 06 2020].

[22] J. Lucas, «OPEN WEBINARS,» 02 septiembre 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-raspberry-pi/. [Último acceso: 08 06 2020].

[23] J. R. Lacerna, «SISTEMA DE OPERACIÓN Y MONITOREO PARA UN VEHÍCULO MEDIANTE RASPBERRY PI,» PEREIRA RISARALDA, 2015.

[24] E. P. S. D. I. SL, «BRICOGEEK,» E PULSE, [En línea]. Available: bricogeek.com/accesorios-raspberry-pi/822-camara-raspberry-pi-v2-8-megapixels.html. [Último acceso: 08 06 2020].

[25] I. B. Jordi Adell, Software libre en educación, España.

[26] debian, «debian.org,» [En línea]. Available: https://www.debian.org/releases/jessie/mips/ch01s01.html.es. [Último acceso: 10 06 2020].

[27] R. G. Duque, Python para todos, España: Creative Commons.

[28] M. A. Alvarez, «Desarrolloweb.com,» [En línea]. Available: https://desarrolloweb.com/articulos/1325.php. [Último acceso: 04 07 2020].

[29] S. H. Azouzi, Editor de imágenes basado en regiones. Aplicacion en entorno Matlab, Terrassa: I.T. Telecomunicaciones, esp., 2005.

[30] E. F. S. Silva, «Medium,» 14 11 2019. [En línea]. Available: https://medium.com/@bootcampai/redes-neuronales-13349dd1a5bb#:~:text=Funciones%20de%20activaci%C3%B3n,que%20permitir%C3%A1%20reconstruir%20o%20predecir.. [Último acceso: 2020 08 18].

[31] Numerentur.org, «Numerentur.org,» 11 04 2018. [En línea]. Available: http://numerentur.org/funcion-de-activacion-softmax/. [Último acceso: 18 08 2020].

[32] Impresoras3D.com, «Impresoras3D.com,» 03 01 2018. [En línea]. Available: https://www.impresoras3d.com/filamento-pla-consejos-caracteristicas-y-mucho-mas/. [Último acceso: 19 08 2020].

…….

Hugo Livardo Alomía Landázuri

CC. 1500890932

Proponente 1

Email: hugo.alomia2@utc.edu.ec

Telf: 0960963522

……

David John Cedeño Arcos

CC: 0503455644

Proponente 2

Email: john.cedeno5644@utc.edu.ec

Telf: 0983802932

…….

Ing. Byron Paúl Corrales Bastidas M.Sc.

CC:0502347768

ANEXOS

Anexo I

Tabla I.1. Ensayos de letras Mayúsculas.

N°	LETRA	LETRA VISUALIZADA	DETECTA	N°	LETRA	LETRA VISUALIZADA	DETECTA
1	A	A	SI	**26**	Z	Z	SI
2	B	B	SI	**27**	A	A	SI
3	C	C	SI	**28**	B	B	SI
4	D	D	SI	**29**	C	C	SI
5	E	E	SI	**30**	D	D	SI
6	F	F	SI	**31**	E	E	SI
7	G	G	SI	**32**	F	F	SI
8	H	H	SI	**33**	G	G	SI
9	I	I	SI	**34**	H	H	SI
10	J	J	SI	**35**	I	I	SI
11	K	K	NO	**36**	J	J	SI
12	L	L	SI	**37**	K	K	SI
13	M	M	SI	**38**	L	L	SI
14	N	N	SI	**39**	M	M	SI
15	O	O	SI	**40**	N	N	SI
16	P	P	SI	**41**	O	O	SI
17	Q	Q	SI	**42**	P	P	SI
18	R	R	SI	**43**	Q	Q	SI
19	S	S	SI	**44**	R	R	NO
20	T	T	SI	**45**	S	S	SI
21	U	U	SI	**46**	T	T	SI
22	V	V	SI	**47**	U	U	SI
23	W	W	NO	**48**	V	V	SI
24	X	X	SI	**49**	W	W	SI
25	Y	Y	SI	**50**	X	X	SI

Tabla I.2. Ensayos de letras Minúsculas.

N°	LETRA	LETRA VISUALIZADA	DETECTA	N°	LETRA	LETRA VISUALIZADA	DETECTA
1	a	a	SI	**26**	z	z	SI
2	b	b	SI	**27**	a	a	SI
3	c	c	SI	**28**	b	b	SI
4	d	d	SI	**29**	c	c	SI
5	e	e	SI	**30**	d	d	SI
6	f	f	SI	**31**	e	e	SI
7	g	g	SI	**32**	f	f	SI
8	h	h	SI	**33**	g	g	SI
9	i	i	SI	**34**	h	h	SI
10	j	j	NO	**35**	i	i	SI
11	k	k	SI	**36**	j	j	SI
12	l	l	SI	**37**	k	k	SI
13	m	m	SI	**38**	l	l	SI
14	n	n	SI	**39**	m	m	SI
15	o	o	SI	**40**	n	n	SI
16	p	p	SI	**41**	o	o	SI
17	q	q	SI	**42**	p	p	SI
18	r	r	NO	**43**	q	q	NO
19	s	s	SI	**44**	r	r	SI
20	t	t	SI	**45**	s	s	SI
21	u	u	SI	**46**	t	t	SI
22	v	v	SI	**47**	u	u	SI
23	w	w	SI	**48**	v	v	SI
24	x	x	SI	**49**	w	w	SI
25	y	y	SI	**50**	x	x	SI

Tabla I.3. Ensayos de Números.

N°	NÚMERO	NÚMERO VISUALIZADO	DETECTA	N°	NÚMERO	NÚMERO VISUALIZADO	DETECTA
1	0	0	SI	**26**	5	5	NO
2	1	1	SI	**27**	6	6	SI
3	2	2	SI	**28**	7	7	SI
4	3	3	SI	**29**	8	8	SI
5	4	4	SI	**30**	9	9	SI
6	5	5	SI	**31**	0	0	SI
7	6	6	SI	**32**	1	1	SI
8	7	7	NO	**33**	2	2	SI
9	8	8	SI	**34**	3	3	SI
10	9	9	SI	**35**	4	4	SI
11	0	0	SI	**36**	5	5	SI
12	1	1	SI	**37**	6	6	SI
13	2	2	SI	**38**	7	7	SI
14	3	3	NO	**39**	8	8	SI
15	4	4	SI	**40**	9	9	NO
16	5	5	SI	**41**	0	0	SI
17	6	6	SI	**42**	1	1	SI
18	7	7	SI	**43**	2	2	SI
19	8	8	SI	**44**	3	3	NO
20	9	9	SI	**45**	4	4	SI
21	0	0	SI	**46**	5	5	SI
22	1	1	SI	**47**	6	6	SI
23	2	2	SI	**48**	7	7	SI
24	3	3	SI	**49**	8	8	SI
25	4	4	SI	**50**	9	9	SI

Anexo II	1 de 1
Código para calcular la magnitud del HOG o histograma de gradientes orientados de cada imagen.	Códigos utilizados para el entrenamiento de las redes neuronales.

```
from skimage import feature

class HOG(object):
    def __init__(self, orientations=9,
pixels_per_cell=(8, 8),
cells_per_block=(3, 3), transform=True):
        self.orientations = orientations
        self.pixels_per_cell =
pixels_per_cell
        self.cells_per_block =
cells_per_block
        self.transform = transform

    def describe(self, image):
        histogram = feature.hog(image,
orientations=self.orientations,
pixels_per_cell=self.pixels_per_cell,

cells_per_block=self.cells_per_block,
transform_sqrt=self.transform)

        return histogram
```

```
import argparse

from sklearn.externals import joblib
from sklearn.svm import LinearSVC

import dataset
from hog import HOG

argument_parser =
argparse.ArgumentParser()
argument_parser.add_argument('-d', '--
dataset', required=True, help='Path to the
dataset file.')
argument_parser.add_argument('-m', '--
model', required=True, help='Path to
where the model will be stored.')
arguments =
vars(argument_parser.parse_args())

(digits, target) =
dataset.load_digits(arguments['dataset'])
data = []

hog = HOG(orientations=18,
pixels_per_cell=(10, 10),
cells_per_block=(1, 1), transform=True)

for image in digits:
image = dataset.deskew(image, 20)
```

	image = dataset.center_extent(image, (20, 20)) histogram = hog.describe(image) data.append(histogram) model = LinearSVC(random_state=42) model.fit(data, target) joblib.dump(model, arguments['model'])

Anexo III. Manual de funcionamiento

UNIVERSIDAD TÉCNICA DE COTOPAXI

INGENIERÍA ELECTROMECÁNICA

MANUAL DE FUNCIONAMIENTO

"DISPOSITIVO DE IDENTIFICACIÓN DE NÚMEROS Y LETRAS EMPLEANDO HARDWARE Y SOFTWARE LIBRE"

1. INTRODUCCIÓN

El presente manual tiene la finalidad de guiar al usuario, con pasos detallados de cómo poner en funcionamiento el dispositivo.

2. OBJETIVOS:

- Describir el funcionamiento del dispositivo de identificación de números y letras.
- Describir los parámetros relevantes para el correcto funcionamiento.
- Presentar posibles fallas o errores del funcionamiento.

3. DESCRIPCIÓN DEL FUNCIONAMIENTO

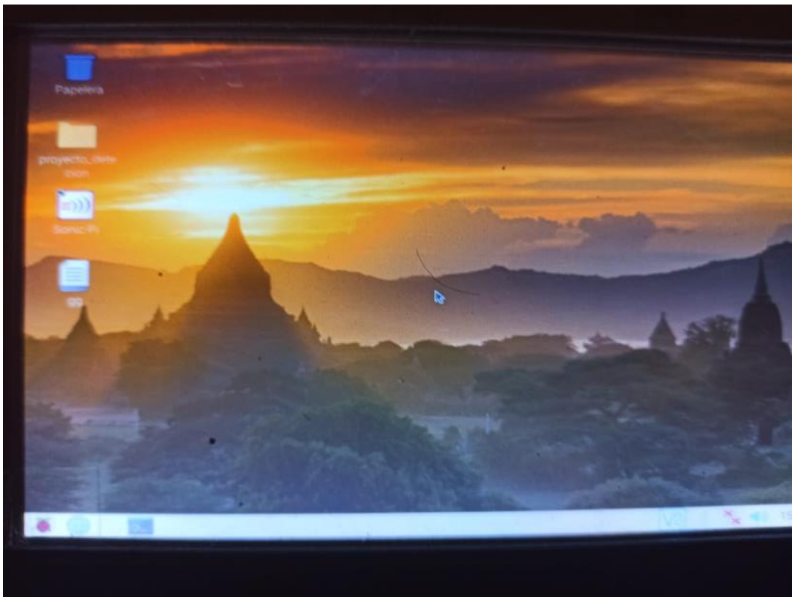
El dispositivo de identificación de números y letras cuenta con una pantalla táctil de 5 pulgadas, una cámara compatible con Raspberry Pi de 3 megapíxeles, puertos de entrada para fuente de poder y accesorios del dispositivo (mouse y parlantes). Dentro del ámbito del funcionamiento, el dispositivo cuenta con una interfaz amigable, permitiendo utilizarlo de una manera didáctica. Cabe mencionar que, para la correcta identificación de números y letras, el dispositivo debe ser usado en un ambiente iluminado.

4. DESCRIPCIÓN DE USO

Paso 1. Ubicar el cableado del dispositivo y accesorios en sus respectivos puertos de conexión.

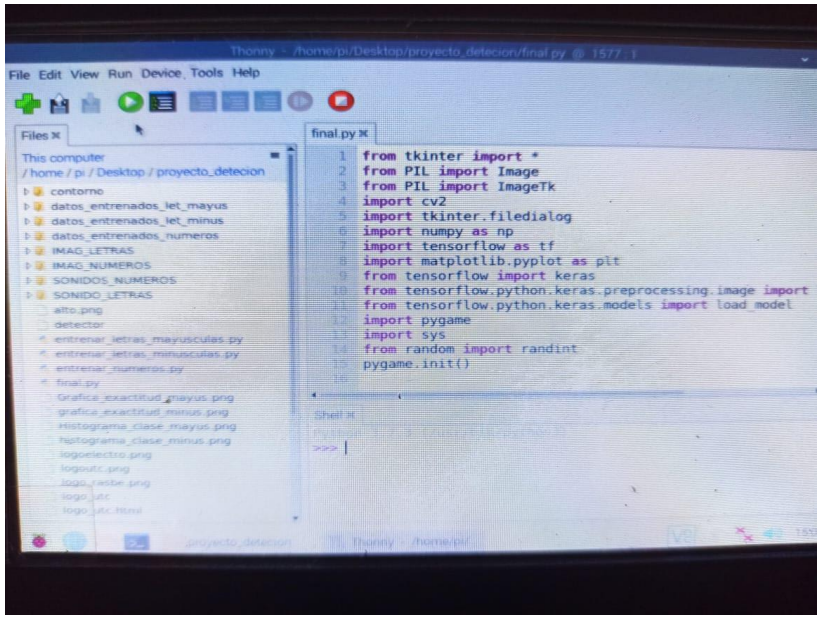
NOTA: Conectar la fuente de poder después de conectar todos los accesorios, ya que este enciende el dispositivo inmediatamente a su conexión.

Paso 2: Una vez encendido el dispositivo se procede a abrir la carpeta **"Proyecto"** que se encuentra en el escritorio del mismo.

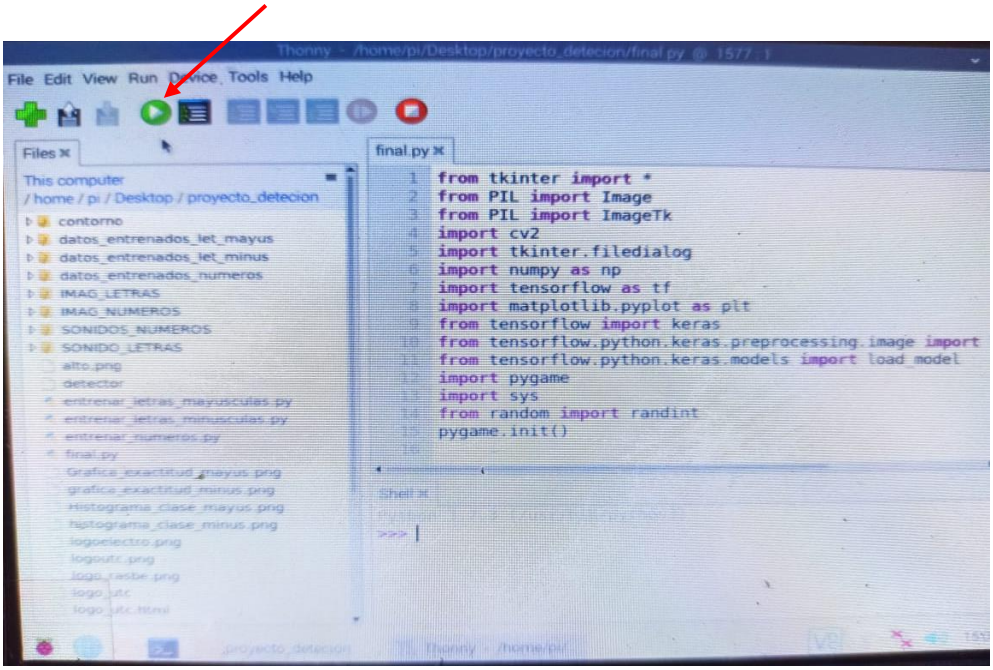


NOTA: La interfaz del dispositivo es muy amigable con el usuario ya que se asemeja a un computador normal.

Paso 3: Abrir la carpeta con el nombre **"Final Py",** lugar donde se encuentra ubicado el programa de identificación.

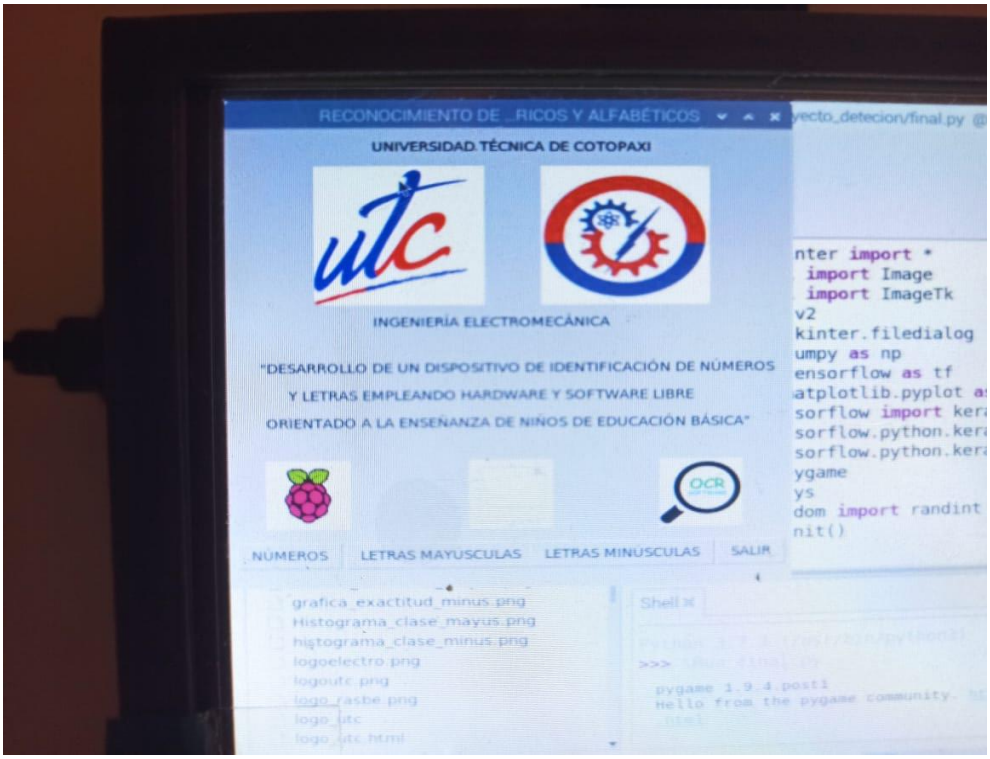


Paso 4: Abrir el ícono de color verde **"correr programa"**

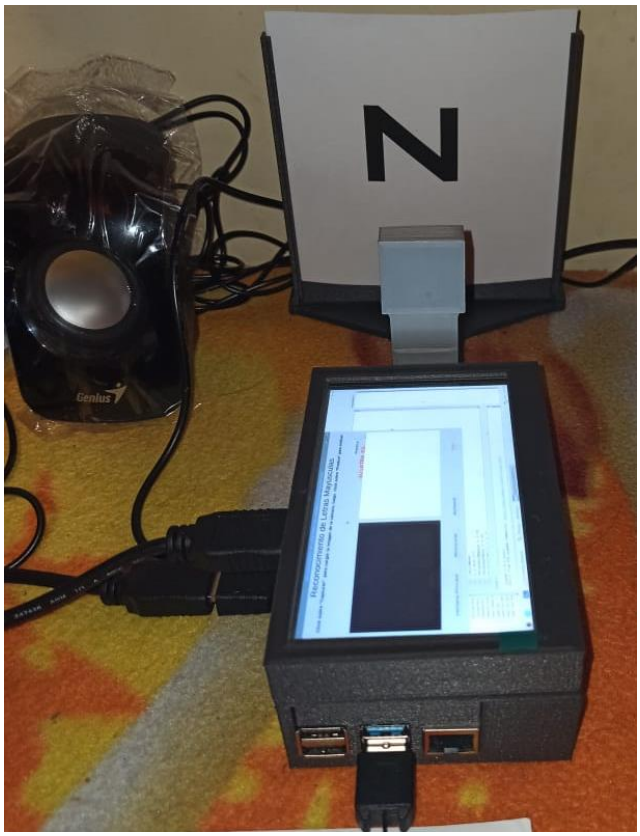


NOTA: Esperar unos segundos hasta que el programa cargue completamente.

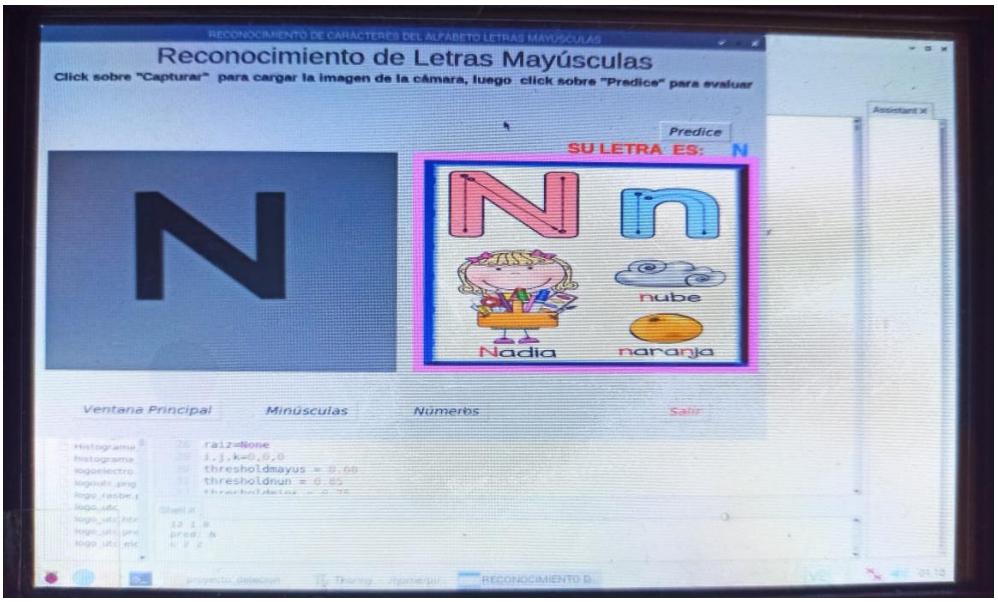
Paso 5: En la pantalla principal del programa, elegir la opción que se desee identificar **(Números, Letras Mayúsculas, Letras Minúsculas).**



Paso 6: Ubicar la tarjeta correspondiente de Número o Letra sobre su soporte.



NOTA: posterior a la ubicación de la tarjeta, oprimir en la pantalla la opción **"predice"**, lo que permitirá identificar el tipo de letra o número automáticamente.



NOTA: Para apagar el dispositivo desconectar la fuente de poder.