Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Telecomunicaciones

Diseño e implementación de un guante traductor de señas para personas con discapacidad auditiva y/o verbal.

Trabajo de Integración Curricular, previo a la obtención del título de Ingeniero en Telecomunicaciones.

AUTOR:

Josué Elian Betancourt Ochoa

DIRECTOR:

Ing. Marcelo Valdiviezo Mg. Sc.

Loja – Ecuador

2024

Certificación

Loja, 21 de noviembre de 2024

Ing. Marcelo Fernando Valdiviezo Condolo. Mg. Sc.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

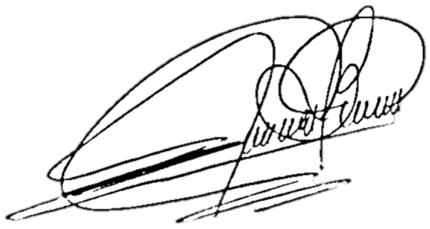
CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominado: **Diseño e implementación de un guante traductor de señas para personas con discapacidad auditiva y/o verbal**, previo a la obtención del título de **Ingeniero en Telecomunicaciones**, de la autoría del estudiante **Josué Elian Betancourt Ochoa**, con cédula de identidad Nro. **1150338885**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Ing. Marcelo Fernando Valdiviezo Condolo. Mg. Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Josue Elian Betancourt Ochoa**, declaro ser autor del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular, en el Repositorio Digital Institucional – Biblioteca Virtual



Autor: Josue Elian Betancourt Ochoa

Cedula: 1150338885

Fecha: 21 de noviembre del 2024

Correo electrónico: josue.betancourt@unl.edu.ec

Teléfono: 0993621659

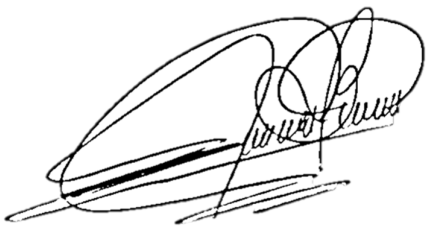
Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo del Trabajo de Integración Curricular.

Yo **Josue Elian Betancourt Ochoa**, declaro ser autor Trabajo de Integración Curricular denominado: **Diseño e implementación de un guante traductor de señas para personas con discapacidad auditiva y/o verbal**, como requisito para optar el título de Ingeniero en Telecomunicaciones, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular que realice un tercero.

Para constancia de esta autorización, lo suscribo en la ciudad de Loja, a los veintiún días del mes de noviembre del año 2024.



Autor: Josue Elian Betancourt Ochoa
Cédula:1150338885
Dirección: Loja
Correo electrónico: josue.betancourt@unl.edu.ec
Teléfono: 0993621659

DATOS COMPLEMENTARIOS:
Director del Trabajo de Titulación: Ing. Marcelo Valdiviezo Condolo. Mg. S

Dedicatoria

Esta investigación de integración curricular es dedicada a mi madre y mis queridas hermanas por su amor incondicional, su apoyo constante y su confianza en mis capacidades. Su guía y ejemplo han sido la base sobre la cual he podido desarrollar mi ética y carácter la cual será la base donde se construirá mi vida profesional. A mi novia y futura esposa, quien con su compañía y aliento ha hecho que este camino sea más llevadero y gratificante. A todos aquellos que, con su sabiduría y paciencia, han dejado una huella imborrable en mi formación.

Josué Elian Betancourt Ochoa

Agradecimiento

Quiero expresar mi más profundo agradecimiento a todos aquellos que hicieron posible la culminación de este trabajo de integración curricular. A mi asesor, por su orientación experta y por haberme brindado las herramientas necesarias para avanzar en esta investigación. A las diversas plataformas de inteligencia artificial que utilicé durante este proceso, su avanzada tecnología y versatilidad fueron fundamentales para el análisis y procesamiento de datos, permitiéndome explorar nuevos horizontes de conocimiento. Gracias a la comunidad académica y a mis compañeros de estudio, quienes enriquecieron mi perspectiva a través de debates y colaboraciones. Finalmente, agradezco a mi familia por su apoyo constante y su comprensión durante los momentos más desafiantes de esta etapa.

Josué Elian Betancourt Ochoa

Índice de Contenidos

Índice de Tablas:

Índice de Figuras:

Índice de Ecuaciones:

Índice de Anexos:

1 Título

Diseño e implementación de un guante traductor de señas para personas con discapacidad auditiva y/o verbal

2 Resumen

La comunicación basada en gestos es crucial para personas con discapacidades auditivas o del habla. Sin embargo, los sistemas actuales suelen ser costosos o difíciles de usar. Este proyecto se centró en el desarrollo de una aplicación móvil para la interpretación de gestos utilizando una red neuronal densa. El sistema incluye un circuito diseñado para capturar las señales de la mano del usuario mediante sensores flexibles y un MPU6050, enviando la información percibida a una base de datos en Firebase para ser procesados en una aplicación programada en Flutter. La aplicación móvil fue desarrollada con lenguaje Dart, la cual ha permitido incorporar funcionalidades principales: visualización de gestos en letras, pronunciación de los caracteres interpretados y enseñanza de gestos a través de imágenes y videos en plataformas como YouTube. Los resultados indican una precisión del 66% en la interpretación de gestos globales del modelo entrenado, la cual puede mejorar modificando ciertos parámetros del entrenamiento y ampliando la base de datos de cada clase de gestos.

Palabras clave: Guante traductor; Tecnologías Asistidas; Microcontrolador; Lenguaje de señas; Aprendizaje máquina; Flutter

Abstract

Gesture-based communication is crucial for people with hearing or speech disabilities. However, at the present time systems are often expensive or difficult to use. This project is focused on the development of a mobile application for gesture interpretation using a dense neural network. The system includes a circuit designed to capture the user's hand signals using flexible sensors and an MPU6050, sending the perceived information to a database in Firebase to be processed in an application programmed in Flutter. The mobile application was developed with Dart language, which has allowed the incorporation of main functionalities: visualization of gestures in letters, pronunciation of the interpreted characters and, teaching of gestures through images and videos on platforms such as YouTube. The results indicate a sixty six percent accuracy in the interpretation of global gestures of the trained model, which can be improved by modifying certain training parameters and expanding the database of each class of gestures.

Key words: Translator glove; Assisted Technologies; Microcontroller; Sign language; Machine learning; Flutter

3 Introducción

En el territorio ecuatoriano existen alrededor de 480 776 personas vulnerables con capacidades sensoriales reducidas, de las cuales el 12.93 % son sordas y el 1.15% tienen problemas con el habla (CONADIS,2023). Este último medio, el habla, es fundamental en nuestras vidas, por tanto, que una persona sin aptitudes de darse a entender, no pueda interactuar con su entorno; puede generar frustración y aislamiento. No obstante, aunque la comunidad que presenta estas limitaciones ha creado un mecanismo compensatorio para transmitir sus ideas, una gran parte de la sociedad lo desconoce. Acorde a la OMS (2011), "la interacción entre la condición de la salud, factores personales y ambientales puede producir una enorme variabilidad en la experiencia de la discapacidad". Es por ello que, al ser un tema de índole social, en el Plan Nacional de Desarrollo del Ecuador 2017-2021, se mencionan medidas para garantizar los derechos de las personas con discapacidad, lo que implica un enfoque inclusivo y de respeto a la diversidad resaltando la importancia abordar las necesidades de las personas con discapacidad auditiva y verbal para promover su inserción y mejorar su calidad de vida.

Gracias al esfuerzo de fundaciones y centros que promueven políticas que incentivan la inclusión, prevención y promoción, en ambientes estudiantiles y laborales se han implementado estrategias como textos ilustrados, carteles animados, textos en secuencia e inclusive por medio de prótesis auditivas proporcionadas por fundaciones como Fundación Hermano Miguel y Fundación DHEX vivir la sordera. Así mismo, han promovido actividades motoras que favorezcan la comunicación, lo cual ha permitido que este porcentaje del grupo afectado por la ausencia de algunos sentidos venga a compartir y relacionarse con personas que no se encuentran familiarizadas. Sin embargo, no es suficiente para poder asegurar un desarrollo sostenible en esta población afectada. En consecuencia, la manera en la que se pueden consolidar este tipo de estrategias debe ser por medio de sistemas de comunicación óptimos de la mano de tecnologías asistivas.

Este último, podría ser abordado mediante el diseño e implementación de un guante que incorpore tecnologías avanzadas como sensores de movimiento, inteligencia artificial, o tecnología háptica para proporcionar una experiencia de traducción de señas más precisa y práctica. A pesar de que existen algunas soluciones tecnológicas para la traducción de señas, estas presentan limitaciones notables que afectan su eficacia en situaciones cotidianas. La tecnología actualmente disponible para la traducción de señas para personas con discapacidad auditiva y/o verbal presenta limitaciones significativas en mínimo uno de los términos referentes a precisión, portabilidad o accesibilidad. Es por ello que el presente proyecto de tesis

plantea la posibilidad de desarrollar un guante traductor de señas que aproveche las últimas tecnologías disponibles para mejorar la comunicación efectiva, permitiendo una interacción más fluida y en tiempo real para este grupo de la población, superando las limitaciones actuales de dispositivos existentes, con capacidades de poder traducir la lengua de señas a texto y sonido, con la finalidad de que el usuario pueda comunicarse con personas que la desconozcan y así poder entablar mejores relaciones interpersonales con su entorno. La implementación de un guante traductor de señas aprovecha los avances tecnológicos actuales para abordar de manera más efectiva las necesidades comunicativas de este grupo de la población. Al incorporar sensores de movimiento, inteligencia artificial y otras tecnologías emergentes, se busca superar las limitaciones de las soluciones existentes, proporcionando una herramienta más precisa, portátil y accesible.

La relevancia de este proyecto se destaca en el contexto de la inclusión social y la igualdad de oportunidades. Un guante traductor de señas eficaz podría empoderar a las personas con discapacidad auditiva y/o verbal al facilitar su participación activa en diversos entornos, como educativos, laborales y sociales. Asimismo, se espera que este tipo de tecnología contribuya a reducir las brechas de comunicación y fomente la comprensión entre las personas con y sin discapacidad y de esta manera mejorar su calidad de vida y promover una sociedad más inclusiva y equitativa.

El uso de un guante traductor de señas para personas con discapacidades auditivas y verbales ofrece varios beneficios para el sector investigado ya que facilitará la interacción en diversos contextos, como en el hogar, el trabajo, la escuela y lugares públicos. Promoverá la autonomía de las personas con discapacidades, permitiéndoles comunicarse sin depender de intérpretes o familiares. Esto, en consecuencia, puede mejorar su autoestima y confianza en sí mismas (SunriseMedical, 2021).

Así mismo, al facilitar la comunicación, se promueve una mayor inclusión social y participación en la comunidad. Las personas pueden integrarse mejor en actividades sociales y culturales, lo que reduce su aislamiento. Además, mejorará el acceso a servicios esenciales como atención médica, servicios gubernamentales y atención al cliente, donde la comunicación clara y efectiva es crucial. En entornos educativos, el proyecto investigativo puede ayudar a los estudiantes con discapacidades auditivas y verbales a participar más plenamente en las clases y actividades escolares, mejorando su aprendizaje y rendimiento académico por ende asegurando un desarrollo económico dando una mayor probabilidad de tener ingresos monetarios.

La propuesta del presente proyecto incluye una fase de investigación y una de ejecución, donde se va a desarrollar tanto el prototipo embebido del guante junto con la aplicación por la cual se podrá interpretar el lenguaje de señas a caracteres con voz artificial para personas con discapacidades de lenguaje y audición. El guante estará conformado por sensores capaces de captar la mínima señal de flexión que el usuario haga con su muñeca, además de poder detectar movimientos inerciales del brazo, para pasar a ser procesado mediante un modelo de IA contando con microcontrolador que enviará la información obtenida y seguidamente interpretada a una aplicación que se desarrollará para poder visualizar los caracteres y ser pronunciados por una voz. La comunicación que se mantendrá entre el microcontrolador y la aplicación será de manera inalámbrica utilizando estándares como el 802.11 o el 802.15.1. La voz artificial será ejecutada por una app.

El proyecto de investigación se desarrollará en el período Abril –Septiembre 2024 cumpliendo con lo dispuesto en el Reglamento de Graduación para Obtener el Título Terminal de Tercer Nivel de la Universidad Nacional de Loja.

➢ Objetivo General

Diseñar y construir un guante electrónico para la traducción del lenguaje de señas a caracteres con voz artificial y conexión inalámbrica a dispositivos móviles para personas con discapacidad auditiva y de lenguaje.

➢ Objetivos específicos

- Estudiar el lenguaje de señas y técnicas de traducción para las personas con discapacidad auditiva, como complementario para el desarrollo de un prototipo tecnológico.
- Diseñar y construir el prototipo embebido además de efectuar pruebas de funcionamiento de precisión y portabilidad.
- Generar una aplicación que permita interpretar de manera inmediata los caracteres que se generan mediante el guante para traducirlos a texto y audio.

4 Marco Teórico

En el presente apartado se presentará el sustento teórico y conceptual para delimitar el conocimiento que será utilizado. Conceptos relacionados a tecnologías asistidas, discapacidades, entornos útiles para desarrollar la programación adecuada, y sin olvidar el uso de herramientas de aprendizaje automático, los cuales serán enmarcados para comprender el contexto de la presente investigación.

4.1 Antecedentes Investigativos

Las Tecnologías Asistidas (TA) constituyen herramientas electrónicas diseñadas para permitir la participación plena de individuos con discapacidades o necesidades especiales en actividades cotidianas, equiparándolos a sus pares sin discapacidad.

Las personas con discapacidad auditiva experimentan distintas necesidades según el grado de hipoacusia que presenten. Aquellos con leve a moderada pérdida auditiva pueden beneficiarse de audífonos, los cuales amplifican los sonidos perceptibles por el usuario. Por otro lado, los implantes cocleares, más complejos, son utilizados por individuos con sordera profunda, ya que estimulan directamente el nervio auditivo. La instalación de un implante coclear implica la intervención quirúrgica para implantar un dispositivo magnético y una bobina en la cóclea del oído interno, siendo especialmente diseñado para personas con pérdida auditiva neurosensorial.

Para aquellos con sordera profunda, sea adquirida o congénita, resulta crucial contar con herramientas que les permitan una comunicación efectiva con su entorno. Por ende, se han desarrollado diversos dispositivos y tecnologías orientadas a facilitar su interacción social, los cuales son detallados en los antecedentes de este proyecto.

4.1.1 Proyectos de grado similares

A continuación, se describen algunos prototipos de traducción de lenguaje de señas y de medición de flexión y orientación de los dedos, que han sido desarrollados y que se relacionan con el presente proyecto.

La investigación llevada a cabo por (Abhishek, Qubeley, & D. Ho, 2016) se centra en un sistema de traducción del lenguaje de señas que emplea sensores capacitivos ubicados en la parte superior de cada dedo. Además, utiliza un módulo PIC-116 para los sensores capacitivos y una tarjeta Raspberry Pi para determinar las configuraciones manuales correspondientes a las letras del alfabeto dactilológico americano. Sin embargo, una desventaja identificada en este

sistema es la ocurrencia de errores en ciertas configuraciones manuales debido a que los sensores capacitivos ofrecen únicamente una respuesta binaria.

A su vez en el proyecto desarrollado por (Meriño Guzman & Garizabalo Pedrozo, 2020) donde, diseñaron un sistema de guane electrónico para la interpretación y traducción del lenguaje de señas utilizando la tecnología provista por Arduino y a su vez desarrollando una app en Android para su posterior visualización de los caracteres, por medio del acondicionamiento de sensores resistivos para el reconocimiento de los grados de flexión y además un sensor MPU6050 para poder percibir la inclinación, rotación y aceleración del guante. El prototipo es capaz de reconocer los caracteres gracias a que el código tiene programado los rangos a los cuales pertenecen cada letra del abecedario. Sin embargo, presenta deficiencias para algunas letras que tiene ciertos rangos iguales a otros caracteres.

Otro gran avance se puede observar en la investigación implementada (Hernández Samacá & Calderón Quintero, 2022) , la cual aborda un desarrollo de un guante que es capaz de traducir el lenguaje de señas colombiano a un lenguaje natural utilizando un diseño basado en la obtención de las señales de las manos por medio de un sensores MPU6050 acoplados a cada dedo de la mano, que a su vez están conectados a un multiplexor TCA9548a para seguidamente pasar al filtrado de la señal utilizando un filtro ENMA el cual va a eliminar el ruido presente en el entorno para a su vez ser normalizado el valor, el cual pasaría a un microprocesador Raspberry Pi en el cual se encuentra un modelo de Red Neuronal entrenado por medio de la aplicación de *Neural Net Pattern Recognition* de Matlab. Los resultados globales en la traducción de ciertas palabras previamente seleccionadas, obteniendo en promedio de clasificación del 88.97%. Cabe recalcar que el prototipo presenta la desventaja del tamaño que ocupa.

4.1.2 Proyectos similares en el mercado

En el ámbito comercial, se han desarrollado sistemas de asistencia para personas con discapacidad auditiva que se basan en la traducción del lenguaje de señas. Por ejemplo, la empresa CyberGlove Systems LLC ha creado un catálogo de guantes electrónicos que permiten capturar el movimiento mediante 22 mediciones de ángulo de articulación de alta precisión. Este sistema utiliza tecnología resistiva curva de detección patentada para transformar con precisión los movimientos de manos y dedos en datos de ángulo de las articulaciones digitales en tiempo real. Además, cuenta con un software programable que puede ser empleado para el desarrollo de aplicaciones de traducción del lenguaje de señas (CyberGlove, s.f.).

Otro dispositivo de interés es el *Sign Language Ring* de traducción, que consiste en una pulsera y anillos desmontables capaces de detectar los movimientos de la lengua de señas y

traducirlos a voz o texto. Al utilizar el lenguaje de señas, el movimiento es reconocido por los anillos y enviado a la pulsera en forma de datos. Una vez interpretada la información, se traduce para reproducirse en forma de voz o texto (TECNONEO, 2013).

Adicionalmente, la Tableta *Motion Savy,* desarrollada en la Universidad de Washington, Estados Unidos, ofrece la capacidad de traducir el lenguaje de señas al inglés. Este dispositivo permite que una persona sorda o con graves problemas de audición se comunique con otra que pueda oír, pero no conozca el lenguaje de señas, sin necesidad de un intérprete. La tableta incorpora un dispositivo *Leap Motion* que reconoce los gestos y los traduce a voz. Del mismo modo, puede interpretar la voz de una persona que habla y traducirla al lenguaje de señas. La elección del dispositivo *Leap Motion* se debe a su capacidad para reconocer gestos y dar órdenes al ordenador, su tamaño y su alto nivel de reconocimiento, que supera a otras tecnologías como Kinect o las cámaras Go Pro, al menos en cuanto al reconocimiento de gestos con la mano. (Princesa, 2021)

Al contrario que otros sistemas de este tipo, *Google Gesture* no usa una cámara para reconocer los movimientos de las manos. La persona que se comunica a través del lenguaje de signos debe ponerse unas muñequeras electrónicas que recogen el movimiento de los músculos del brazo, y leen los impulsos nerviosos, mediante electromiografía. Ambos brazaletes se comunican de manera inalámbrica a un teléfono inteligente, que es el encargado de procesar los datos y reproducir su traducción usando una aplicación móvil. El software convierte estos movimientos de los músculos en símbolos del lenguaje de signos**,** y por tanto en palabras, y las lee de viva voz a través de los altavoces del smartphone. La figura 1 permite observar el funcionamiento de las muñequeras. (Estapé, 2019)

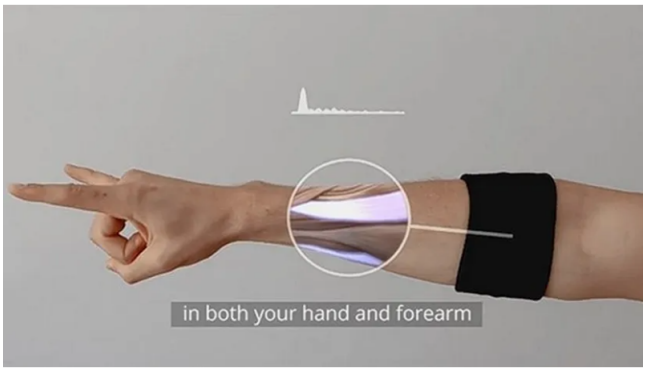


Figura 1. Muñequera desarrollo por Google. Fuente: PRNoticias

4.2 Discapacidad Auditiva

La discapacidad auditiva se define como la pérdida o anormalidad de la función anatómica y/o fisiológica del sistema auditivo, y tiene su consecuencia inmediata en una discapacidad para oír, lo que implica un déficit en el acceso al lenguaje oral. (García, 2019)

Partiendo de que la audición es la vía principal a través de la cual se desarrolla el lenguaje y el habla, debemos tener presente que cualquier trastorno en la percepción auditiva del niño y la niña, a edades tempranas, va a afectar a su desarrollo lingüístico y comunicativo, a sus procesos cognitivos y, consecuentemente, a su posterior integración escolar, social y laboral.

Hemos de tener en cuenta una diferenciación importante en la pérdida de audición: sordera e hipoacusia. La hipoacusia es la pérdida de audición que, con o sin ayuda técnica, permite acceder al lenguaje oral por vía auditiva. La sordera es la pérdida de audición que impide el acceso al lenguaje oral por vía auditiva, convirtiéndose la visión en el principal canal para llevar a cabo el proceso de comunicación.

Atendiendo al grado de pérdida

Una audición normal presenta un umbral auditivo que oscila entre 0-20 decibelios (dB). En la siguiente clasificación se destacan los diferentes grados de pérdida que pueden producirse atendiendo a los decibelios:

- Hipoacusia leve: 20-40 dB.
- Hipoacusia moderada: 40-70 dB.
- Hipoacusia severa: 70-90 dB.
- Hipoacusia profunda o sordera: más de 90 dB.
- También se destaca dentro de este apartado la cofosis o anacusia, siendo la pérdida total de audición.

La pérdida de audición puede existir ya en el momento del nacimiento (causas congénitas), y también puede suceder a cualquier edad (causas adquiridas). (Argentina.gob, s.f.)

Entre las causas congénitas se encuentran:

Factores hereditarios o no hereditarios.

- Complicaciones en el embarazo o en el parto. Por ejemplo, falta de oxígeno en el momento de nacer, ictericia grave durante el período neonatal, bajo peso en el nacimiento.
- Infecciones que haya sufrido la madre durante la gestación, como la rubeola o la sífilis.
- El uso incorrecto de determinados fármacos durante el embarazo (por ejemplo: aminoglucósidos, medicamentos citotóxicos, antipalúdicos y diuréticos).

Entre las "causas adquiridas", se destacan las siguientes:

- El envejecimiento.
- Padecer infecciones crónicas del oído. En los niños, la otitis media crónica - presencia de líquido en el oído- es una causa frecuente.
- Sufrir enfermedades infecciosas como la meningitis, el sarampión y la parotiditis.
- La obstrucción del conducto auditivo por cerumen o cuerpos extraños.
- Padecer traumatismos craneoencefálicos o de los oídos.
- El consumo de algunos medicamentos como los que se prescriben en el tratamiento de infecciones neonatales, el paludismo, algunos tipos de tuberculosis y algunos tipos de cáncer.
- Exponerse a un ruido excesivo las cuales puede producirse bajo circunstancias como en el puesto de trabajo, si está relacionado con maquinaria ruidosa o explosiones, o durante actividades y eventos recreativos en bares, discotecas o conciertos, donde se alcanzan a veces los 110 decibelios.

4.3 Lenguaje de señas

El lenguaje de señas es un sistema de comunicación natural y completo que utilizan principalmente las personas con discapacidad auditiva o del habla. Se basa en el uso de expresiones gestuales, movimientos corporales y la percepción visual, permitiéndoles establecer un canal efectivo de comunicación con su entorno social (Arellano, 2017).

Este idioma visual emplea movimientos de las manos, expresiones faciales y gestos corporales para representar diversos conceptos. A diferencia del lenguaje oral, la lengua de señas posee su propia estructura, gramática y sintaxis, según lo afirma Mauricio Méndez, director de la Asociación de Sordos de Guatemala. Además, es importante destacar que la lengua de signos no es un lenguaje universal; de hecho, existen tantas lenguas de señas como países en el mundo, y la Federación Mundial de Sordos estima que hay alrededor de 300 diferentes a nivel global. Asimismo, Martínez (2018) menciona que, dentro de cada país, pueden encontrarse variaciones regionales y modismos, aunque la mayoría de los signos se basan en principios lógicos. En este contexto, el Consejo Nacional de Igualdad de Discapacidades (CONADIS) tomó la iniciativa de desarrollar el diccionario de lengua de señas ecuatoriano "Gabriel Román" en formato web. Para llevar a cabo este proyecto, contó con el apoyo de la Federación Nacional de Sordos del Ecuador (FENASEC) y la Universidad Tecnológica Indoamérica (UTI).

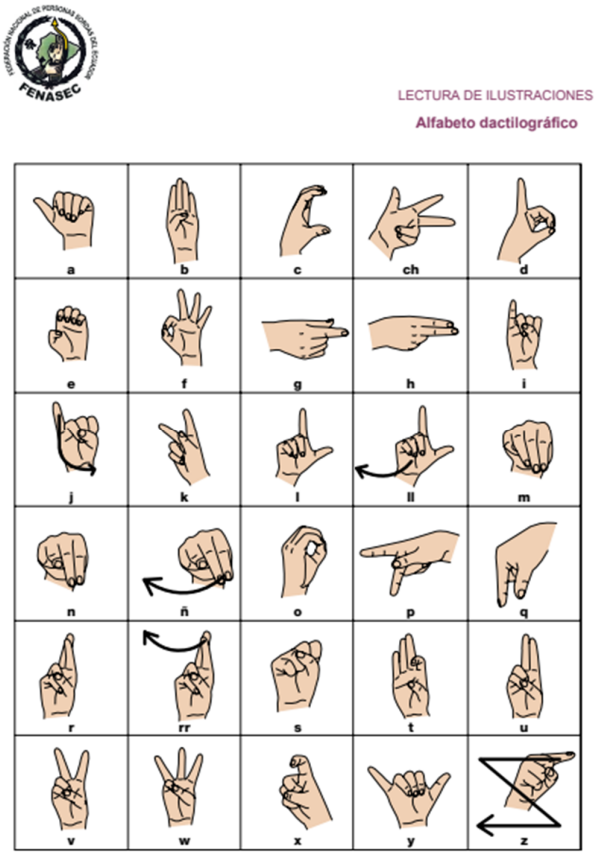


Figura 2. Alfabeto dactilográfico. Fuente: FENASEC

4.3.1 Parámetros formacionales del lenguaje de señas

De manera similar a lo que ocurre en el lenguaje oral, donde los fonemas son las unidades que permiten formar palabras, en el lenguaje de señas los queremas son las distintas partes que componen un signo. En otras palabras, estos queremas son los parámetros que facilitan la comunicación efectiva en la lengua de señas. Siendo así que en la estructura del signo gestual se pueden distinguir los 6 pilares articulatorios.

Configuración (Queirema): La forma que adoptan las manos al realizar un signo. Ejemplo mano abierta, cerrada, con los dedos replegados o no; índice o pulgares levantados

Localización (Toponema): El lugar del cuerpo o del espacio donde se realiza el signo. Ejemplo: ante el cuerpo o espacio neutro, ante la frente, las cejas, los labios, etc.

Movimiento (Kinema): La acción que realiza la mano al ejecutar el signo. Ejemplo: recto, circular, en arco, quebrado, etc., con sus componentes quinestésicos: movimiento simple o repetido, rotación del puño o del antebrazo, etc.

Direccion (Kineprosema): La dirección en la que apuntan los dedos de la mano al signar. Ejemplo: hacia arriba, hacia abajo, hacia la derecha, hacia la izquierda.

Orientacion (Queirotropema): El trayecto que sigue el movimiento de la mano al realizar el signo. Independientemente de qué configuración adquiera la mano, la palma siempre señalará una dirección u otra.

Expresión facial (Prosoponema): Los gestos faciales que acompañan y complementan el signo. Ejemplo: movimientos del cuerpo, de la boca, cejas, etc.

4.4 Cerebro del prototipo

A continuación, se describen algunos conceptos sobre los componentes internos de los dispositivos electrónicos;

4.4.1 Microprocesador

Un microcontrolador es un dispositivo con la capacidad de realizar operaciones de diversa índole de la Unidad Central de Procesamiento (*Central Processing Unit* -CPU) en un único circuito integrado. Su funcionamiento básico es convirtiendo las señales de entrada en un código binario con la finalidad de que la CPU pueda entender y ejecutar la tarea que se está pidiendo. (Sánchez, 2023)

4.4.2 Microcontrolador

Un microcontrolador tal como lo menciona (AWS, ¿Cuál es la diferencia entre microprocesadores y microcontroladores?, 2023) "Es la unidad básica dentro de los dispositivos electrónicos inteligentes, como las lavadoras y termostatos". Se podría decir que es un ordenador diminuto con sus propios sistemas de RAM, ROM y E/S, todos integrados en un solo chip con la capacidad de procesar señales digitales y responder a las entradas del usuario, pero con una capacidad computacional limitada, suficiente para realizar trabajos de sistemas que cuentan con funciones de control y automatización.

4.4.3 Comparativa entre microprocesador y microcontrolador

Los microprocesadores y microcontroladores son chips esenciales que aportan inteligencia tanto a computadoras personales como a dispositivos electrónicos. Estos componentes, fabricados con circuitos integrados de semiconductores, comparten ciertas características internas, pero se distinguen por sus arquitecturas.

Los microprocesadores están basados en la arquitectura Von Neumann, donde el programa y los datos comparten el mismo módulo de memoria *Ramdom Aleatory Memory* (RAM). Por otro lado, los microcontroladores utilizan la arquitectura Harvard, que separa la memoria en dos partes la memoria del programa y la memoria de datos. (Izquierdo, 2021)

En contraste con los microcontroladores, los microprocesadores contienen un mayor número de componentes de circuitos integrados con mayores capacidades, funciones y tamaño.

Estas diferencias arquitectónicas influyen en el diseño y las aplicaciones informáticas y sistemas integrados. Por lo tanto, resulta primordial conocer cuáles serán los alcances que se quiere lograr obtener, para poder elegir correctamente el cerebro del sistema a desarrollarse.

Microcontrolador ESP32

Desarrollado por Espressif Systems, el ESP32 es una serie de microcontroladores económicos y de bajo consumo con sistema en chip que incluye Wi-Fi y Bluetooth de modo dual integrados. Este avance es especialmente valioso para los sistemas de automatización que evitan las complejidades de la radiofrecuencia (RF) y el diseño inalámbrico. Como una solución combinada de Wi-Fi y Bluetooth asequible, la serie ESP32 ha generado popularidad tanto en los entornos que se requieren implementaciones de *Internet of things* (IoT). Su eficiencia energética, la disponibilidad de múltiples entornos de desarrollo de código abierto y una amplia gama de bibliotecas la hacen ideal para una variedad de escenarios (Beningo, 2020).

No obstante, la serie ESP32 se presenta en numerosos módulos, lo que puede dificultar la elección del modelo adecuado. Teniendo una amplia gama de equipos en su catálogo de placas de desarrollo, varios ESP32 diferentes de los cuales se puede seleccionar según las necesidades de la aplicación.

ESP-32S. NodeMCU Este microcontrolador es una placa de desarrollo NodeMCU basada en el ESP32, equipada con conectividad WiFi y Bluetooth, un chip CP2102 integrado y claves de acceso. Provisto con 39 pines de los cuales 32 son es de entrada/salida del módulo ESP32 tal como lo vemos en la Figura 3, los cuales son accesibles a través de los encabezados de extensión.

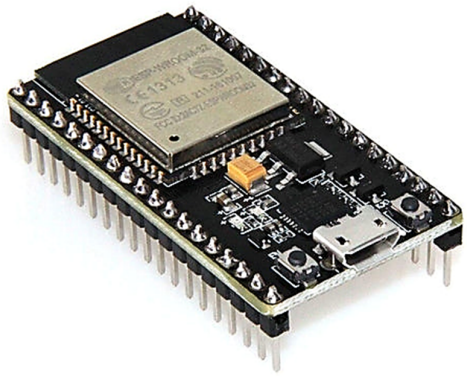


Figura 3. ESP32 NodeMCU 32S. Fuente: HiLetgo

Gracias a la abundancia de recursos de código abierto, admite diversas formas de lenguajes de programación, incluyendo Lua, AT, MicroPython, Arduino, lo que facilita la creación rápida de prototipos para aplicaciones IoT. (waveshare, 2023)

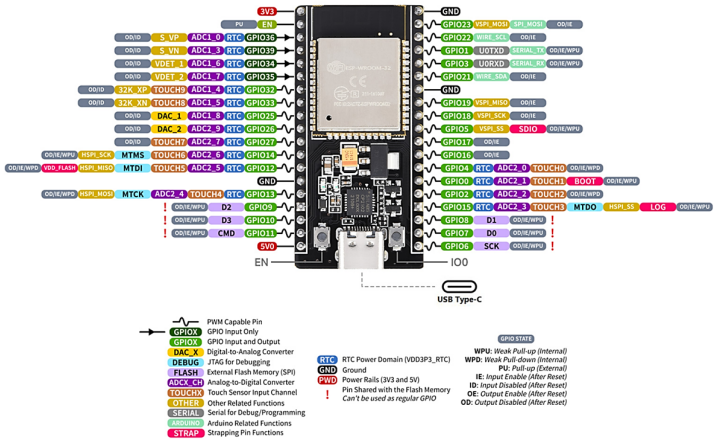


Figura 4. Pinout del ESP32 NodeMCU 32S. Fuente: Waveshare

Características

- Módulo ESP32 integrado
- CP2102 integrado, convertidor USB a UART
- Puerto USB para entrada de energía, programación de firmware o depuración de UART
- Conectores de extensión de 2x19 pines, separan todos los pines de E/S del módulo
- 2x teclas, utilizadas como reinicio o definidas por el usuario

Especificaciones

- Procesador: ESP32
- Flash incorporado: 32 Mbit
- Antena: antena PCB integrada
- Interfaz periférica: UART/GPIO/ADC/DAC/SDIO/PWM/I2C/I2S
- Protocolo WiFi: IEEE 802.11 b/g/n
- Bluetooth: Bluetooth 4.2
- Rango de frecuencia: 2,4G ~ 2,5G (2400M ~ 2483,5M)
- Modo WIFI: Estación / SoftAP / SoftAP+Estación
- Fuente de alimentación: 5V
- Nivel lógico: 3,3 V
- Dimensiones: 51,4 mm x 25,4 mm

4.5 Sistemas sensoriales

Los sistemas sensoriales son el conjunto de sensores que permiten la captura de una amplia gama de señales que proviene del entorno, con la finalidad de ser procesadas en el

cerebro que puede ser un microcontrolador o un microprocesador, dependiendo las necesidades de la aplicación.

4.5.1 Sensores de flexión

Constituyen el componente principal del guante electrónico comercial CyberGlove. Son sensores flexibles que, a medida que son deformados, varían la resistencia en los terminales del sensor. En la ilustración 3 se puede ver la imagen frontal de un sensor de flexión.

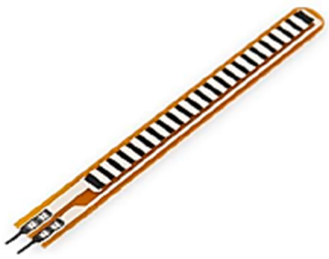


Figura 5. Sensor Flex. Fuente: Avelectronics

Características principales de los sensores Flex

Entre las principales características de los sensores flex se encuentran:

- Dimensiones: 2.2” y 4.5”
- Ciclo de vida de deformaciones >1 millón
- Altura alrededor de 0,43mm
- Rango de temperatura que soporta: -35°C a 80°C
- Resistencia sin deformación: alrededor de 10kΩ
- Tolerancia de la Resistencia:
- Rango de variación al deformarse: de 60K a 110K Ohms
- Potencia nominal: 0,5W continuo y hasta 1W pico

4.6 Unidades de medición inercial (IMUS)

Actualmente, el uso de las IMUs ha sido popularizado en infinidad de aplicaciones, tales como teléfonos celulares o tablets. En el campo de la investigación han sido utilizadas para la medición de movimiento corporal. Se han desarrollado aplicaciones que van desde asistencia a personas discapacitadas hasta teleoperación de diversos tipos de robots. Además, son el componente principal para el control y estabilidad de UAVs.

4.6.1 Sensor MPU-6050

Es un sensor de medida inercial, el cual integra en un mismo circuito un acelerómetro y un giroscopio de tres ejes cada uno. Además, tiene integrado un procesador digital de movimiento, el cual es capaz de utilizar algoritmos que permiten fusión sensorial.

El MPU-6050 es un sensor que tiene 6 grados de libertad, es decir, puede entregar 3 valores de medición correspondientes al acelerómetro y 3 correspondientes al giroscopio. Adicionalmente, tiene un bus para comunicación que permite envío de información del sensor a un dispositivo maestro, o, en su defecto, la comunicación con otros dispositivos externos como magnetómetros.

El MPU-6050 es una solución de bajo costo en comparación con otras unidades de movimiento inercial. Además, posee librerías que permiten implementar algoritmos que mejoran el resultado de la obtención de cada uno de los parámetros de medición angular.

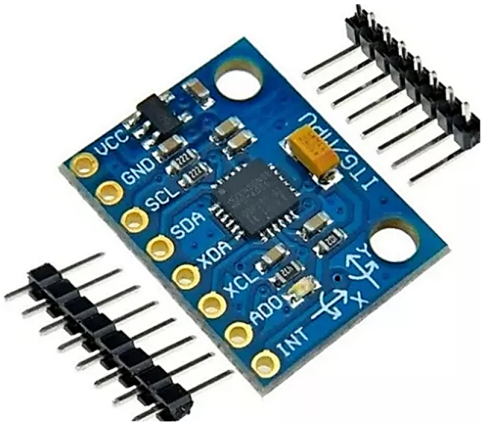


Figura 6. Sensor MPU6050. Fuente: Components101

4.7 Alimentación del prototipo

Para suministrar la energía suficiente para el funcionamiento adecuado se necesita de un módulo de carga el cual es un componente que se utiliza para gestionar la carga de una batería o acumulador, generalmente desde una fuente de energía externa, como paneles solares o un adaptador de corriente. Estos dispositivos tienen la función de regular la corriente y el voltaje de carga para evitar sobrecargas, sobrevoltajes o sobrecorrientes, protegiendo así tanto la fuente de energía como la batería.

4.7.1 Modulo SM5308

El SM5308 es un circuito integrado (IC) utilizado principalmente como un controlador de carga para baterías de iones de litio (Li-ion), polímero de litio (Li-Po) en dispositivos electrónicos portátiles. Este tipo de módulo gestiona de manera eficiente el proceso de carga de

la batería, protegiéndola de sobrecargas y asegurando que la carga se realice de forma controlada. (HeTPro, 2023)

Figura 7. Modulo controlador de carga SM5308. Fuente: HeTPro Store

Parámetros básicos:

- Compatible 4.2V/4.35V baterías de litio
- Rango de tensión de entrada: 5-5.5V
- Tensión de corte de carga: (4,2V/4,35V) más o menos 0,5 por ciento
- Corriente de carga: 2,4A más o menos 5%.
- Tensión de salida Boost: 5V-5,15V (con compensación de pérdida de línea)
- Ondulación de la tensión de salida Boost: 100 mA
- Corriente de salida Boost: 2A
- Eficiencia de conversión Boost: 92,5% (entrada 3,6V, salida 5V2A)
- Corriente de reposo del lado de la batería: menos de 30uA
- Tamaño del producto: 20*25*4,5 mm

4.7.2 Batería 18650

Una batería 18650 es un almacenador de iones de litio, cuyo nombre proviene de sus dimensiones: 18 mm de diámetro y 65 mm de largo, siendo más grande que una pila AA. Su voltaje es de 3,6 V y su capacidad varía entre 2600 mAh y 7500 mAh. Estas baterías son comunes en linternas, computadoras, dispositivos electrónicos y sistemas electrónicos portátiles, gracias a su fiabilidad, largos tiempos de funcionamiento y la capacidad de recargarse muchas veces. Se consideran baterías "de alto consumo", lo que significa que están diseñadas para ofrecer un alto voltaje y corriente, satisfaciendo las exigencias de dispositivos que requieren mucha energía. Esto las hace ideales para equipos más complejos y demandantes, que necesitan un suministro constante y elevado de energía. Además, tienen una gran profundidad de descarga, lo que les permite agotarse completamente y recargarse sin perder su capacidad original, ya que tienen un ciclo de vida de alrededor de 300 a 500 ciclos de carga. (FenixLighting, 2021)

Figura 8. Batería 18650. Fuente: Electrostore

4.8 Machine Learning

El machine learning o también conocido como aprendizaje automático, es la ciencia de desarrollo de algoritmos y modelos estadísticos que utilizan los sistemas de computación con el fin de llevar a cabo tareas sin instrucciones explícitas, en vez de basarse en patrones e inferencias.” (AWS, 2023). Los sistemas computacionales son los encargados de utilizar los algoritmos de machine learning para procesar grandes cantidades de información e identificar patrones de datos. Esto permite generar resultados con mayor precisión a partir de un conjunto de datos de entrada.

4.8.1 Tipos de machine learning

Existen varios tipos de aprendizaje automático entre los cuales se destacan:

Aprendizaje supervisado (Supervised Learning): El algoritmo aprende a partir de datos etiquetados *(labeled data),* es decir, se le proporcionan ejemplos de entradas *(input data)* y salidas esperadas. El objetivo es predecir la salida para nuevas entradas.

Aprendizaje no supervisado (Unsupervised Learning): El conjunto de procesos aprende a partir de datos no etiquetados, es decir, se le proporcionan solo entradas sin salidas esperadas. Su meta es encontrar patrones o estructuras en los datos.

Aprendizaje por refuerzo (Reinforcement Learning): El algoritmo aprende a partir de la interacción con un entorno, recibiendo recompensas o castigos por sus acciones. El objetivo es maximizar la recompensa total. (Santos, 2021)

Estos son los principales tipos de Machine Learning, pero existen otros subtipos y variantes dependiendo del contexto y la aplicación tales como:

Aprendizaje semi-supervisado (Semi-supervised Learning): El algoritmo aprende a partir de una combinación de datos etiquetados y no etiquetados.

Aprendizaje auto-supervisado (Self-supervised Learning): El algoritmo aprende a partir de datos no etiquetados, pero utiliza técnicas para generar etiquetas artificiales.

Aprendizaje profundo (Deep Learning): Utiliza redes neuronales profundas para aprender patrones complejos en los datos.

Aprendizaje en línea (Online Learning): El algoritmo aprende a partir de datos que llegan en tiempo real, actualizando sus parámetros continuamente.

Aprendizaje en lotes (Batch Learning): El algoritmo aprende a partir de un conjunto de datos completo, actualizando sus parámetros solo después de procesar todo el conjunto.

Aprendizaje transferido (Transfer Learning): El algoritmo aprende a partir de un dominio y lo aplica a otro dominio relacionado.

Aprendizaje federado (Federated Learning): El algoritmo aprende a partir de datos distribuidos en múltiples dispositivos o servidores, sin necesidad de centralizar los datos.

4.9 Redes Neuronales Artificiales

Una Red Neuronal Artificial (RNA) es un modelo matemático de computación inspirado en el comportamiento biológico y en cómo se organizan las neuronas formando la estructura de los sesos. Amazon Web Services menciona que se trata de un tipo de proceso de *machine learning*, un método de la inteligencia artificial que enseña a los dispositivos a procesar datos, donde utiliza las neuronas interconectadas en una estructura de capas que simula un cerebro humano. De este modo, crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente con asistencia humana limitada. Esto se debe a que pueden aprender y modelar las relaciones entre los datos de entrada y salida que no son lineales y que son complejos.

4.9.1 Estructura de una RNA

En 1943 el psiquiatra y neuroanatomista Warren McCulloch y el matemático Walter Pitts desarrollaron el primer modelo matemático de una neurona artificial, diseñado para realizar tareas simples. La neurona artificial que propusieron (Ver Figura 7) consta de los siguientes elementos:

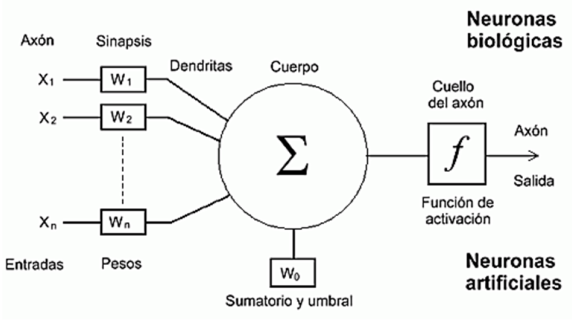


Figura 9. Modelo de McCulloch-Pitts. Fuente: Caparrini

- Un conjunto de entradas $x_1,\ldots, x_n$
- Los pesos sinápticos $w_1,\ldots, w_n$, correspondientes a cada entrada.
- Un peso adicional, w_0
- Una función de agregación, $\sum$
- Una función de activación, f
- Una salida, Y.

Caparrini (s.f) afirmo lo siguiente:

Las entradas son el estímulo que la neurona artificial recibe del entorno que le rodea, y la salida es la respuesta a tal estímulo. La neurona puede adaptarse al medio circundante y aprender de él modificando el valor de sus pesos sinápticos, y por ello son conocidos como los parámetros libres del modelo, ya que pueden ser modificados y adaptados para realizar una tarea determinada.

En este modelo, la salida neuronal Y está dada por:

Ecuación 1. Salida de la red neuronal. Fuente: McCulloch-Pitts.

$$Y = f(w_0 + \sum_{i=1}^{n} w_i x_i)$$

La función de activación se elige de acuerdo a la tarea realizada por la neurona.

4.9.2 Comparativa entre el Machine learning y red neuronal

El machine learning (aprendizaje automático) y las redes neuronales son conceptos relacionados, pero no son exactamente lo mismo. El aprendizaje automático es un subcampo de la inteligencia artificial que se enfoca en desarrollar algoritmos y modelos que permitan a las máquinas aprender de los datos y mejorar su desempeño en tareas específicas sin ser explícitamente programadas (AWS, What is Machine Learning, 2024).

Las redes neuronales, por otro lado, son un tipo de modelo de machine learning inspirado en la estructura y el funcionamiento del cerebro humano. Están compuestas por capas de nodos o "neuronas" artificiales que procesan y transmiten información. En otras palabras, el machine learning es el campo más amplio que abarca varias técnicas y algoritmos para aprender de los datos, mientras que las redes neuronales son una de las técnicas específicas dentro de ese campo.

Algunas otras técnicas de machine learning incluyen:

- Regresión lineal y logística
- Árboles de decisión y bosques aleatorios
- Máquinas de soporte vectorial

- Algoritmos genéticos

Y dentro de las redes neuronales, hay varias arquitecturas y variantes, como:

- Redes neuronales feedforward
- Redes neuronales recurrentes (RNN)
- Redes neuronales convolucionales (CNN)
- Redes neuronales profundas (Deep Learning)

4.10 Entorno de Desarrollo Integral

Un IDE (Integrated Development Environment) es una aplicación de software que reúne en un solo lugar todas las herramientas necesarias para llevar a cabo un proyecto de desarrollo de software (dataScientest, 2022). En otras palabras, un IDE es una plataforma que integra diversas funcionalidades para facilitar el proceso de programación. Proporciona una interfaz unificada que permite a los desarrolladores escribir código, organizar grupos de texto y automatizar tareas repetitivas de programación. Por ende, más que un simple editor de código, combina las funcionalidades de varios procesos de programación en un mismo entorno. Los IDE suelen incluir al menos un editor, un compilador, un depurador y funcionalidades de autocompletado de código o gestión de código genérico. Sin embargo, los IDE más avanzados van más allá y ofrecen funcionalidades adicionales, como visualización de datos, seguimiento de ejecución o referencias cruzadas.

4.10.1 Ejemplos de IDE más populares

Durante el proceso de redacción, creación y prueba de un programa, los programadores emplean una amplia gama de herramientas. Entre las más comunes se encuentran los editores de texto, las bibliotecas de código, los programas de detección de errores, los compiladores y otras plataformas de prueba. En contraste, si un desarrollador opta por no utilizar un IDE, deberá seleccionar, implementar, integrar y gestionar estas herramientas de manera individual y manual. Por esta razón, es más práctico utilizar un entorno de desarrollo integrado, ya que permite combinar todas estas tecnologías dentro de un único marco de trabajo. (Luna, 2019)

Algunos IDE más populares entre los programadores son:

- Visual Studio
- IntelliJ Idea
- PyCharm
- Xcode
- Eclipse
- PhpStorm

- WebStorm
- Android Studio
- NetBeans
- AWS Cloud 9.

4.11 Visual Studio Code

Tal como lo menciona su propio desarrollador Microsoft (2024):

"Visual Studio Code es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Incluye compatibilidad integrada con JavaScript, TypeScript y Node.js, y cuenta con un amplio ecosistema de extensiones para otros lenguajes y entornos de ejecución".

También conocido como VSCode, este es un editor de código diseñado para programadores de forma gratuita, de código abierto y multiplataforma. Aunque su denominación proviene de otra herramienta desarrollada por Microsoft, el IDE Visual Studio, VSCode es en realidad una aplicación independiente, construida sobre una base de código diferente y utilizando tecnologías completamente distintas (Zúñiga, 2024).

4.11.1 ¿Para qué sirve Visual Studio Code?

Visual Studio Code es una herramienta que permite editar el texto plano de los archivos de código para programación. Sin embargo, más allá de su objetivo base, los usuarios pueden configurar VSCode para realizar diversas tareas, gracias a la participación activa de la comunidad que conforma esta plataforma, se han generado extensiones que hacen que opere casi al nivel de un IDE. A continuación, se especifican algunas de sus funciones principales (Zúñiga, 2024).

Edición y desarrollo de código fuente

El propósito principal de Visual Studio Code es permitir a los desarrolladores escribir y editar el código fuente de sitios web y aplicaciones ofreciendo diversos métodos de ayudas a los programadores, como el resaltado de sintaxis o el autocompletado del código.

Depuración de aplicaciones y scripts

VSCode tiene integradas herramientas para depurar el código fuente. Instalando las extensiones adecuadas se puede cualquier lenguaje de programación, pudiendo ejecutar el código paso a paso o crear interrupciones para examinar las variables de los programas.

Gestión de proyectos y carpetas de trabajo

Permite organizar proyectos de una manera muy elemental integrando sencillas herramientas para trabajar con los archivos y carpetas que forman parte de las aplicaciones o los sitios web.

Integración con control de versiones (Git)

la integración nativa con Git, el cual es el sistema de control de versiones más popular que existe, permitiendo hacer uso de las herramientas sin salirse del software, facilitando el trabajo con repositorios y realizar operaciones de commit, pull, push o resolver conflictos.

Extensibilidad y uso de extensiones

Sus extensiones permiten adaptar VSCode para una enorme infinidad de tareas, tecnologías y lenguajes. La comunidad de desarrolladores ha creado una amplia gama de extensiones que se pueden encontrar e instalar sin salirse del propio editor.

Trabajo en colaboración con Live Share

Permite que varios programadores puedan ver y editar el código en tiempo real y de manera simultánea, incluso estando en localizaciones geográficas separadas.

Desarrollo web y front-end

Cualquier persona que necesite desarrollar para la web tiene VSCode, ya que su adaptación a lenguajes como HTML, CSS y Javascript viene de casa. Siendo asi, que la totalidad de frameworks frontend tienen extensiones para poder ampliar sus funcionalidades

Programación en diversos lenguajes

Permite el desarrollo backend, ya que se puede trabajar con la mayoría de los lenguajes de programación. Existen extensiones especializadas en caso de que el lenguaje no tenga soporte nativo. Por tanto, en la práctica los desarrolladores usan VS Code en múltiples lenguajes y frameworks como: PHP, Python, Go, Java, NodeJS, Ruby y muchos otros.

4.12 Flutter

Flutter es un marco de código abierto desarrollado y compatible con Google. Los desarrolladores de front-end y pila completa utilizan Flutter para crear una interfaz de usuario (IU) de aplicación para varias plataformas con un único código base.

Cuando Flutter se lanzó, en 2018, era compatible principalmente con el desarrollo de aplicaciones móviles. Ahora, Flutter es compatible con el desarrollo de aplicaciones en seis plataformas: iOS, Android, web, Windows, MacOS y Linux.

La codificación de una aplicación para una plataforma específica, como iOS, se denomina desarrollo de aplicaciones nativas. En cambio, el desarrollo de aplicaciones multiplataforma consiste en crear una aplicación para varias plataformas con un único código base.

Estas son algunas de las formas en las que Flutter destaca como marco de desarrollo multiplataforma:

- ***Rendimiento casi nativo.*** Flutter utiliza el lenguaje de programación Dart y se compila en código máquina. Los dispositivos host entienden este código, lo que garantiza un rendimiento rápido y eficaz.
- ***Rendimiento rápido, consistente y personalizable***. En lugar de depender de herramientas de renderización específicas de la plataforma, Flutter utiliza la biblioteca gráfica de código abierto Skia de Google para renderizar la interfaz de usuario. Esto proporciona a los usuarios visuales consistentes sin importar la plataforma que utilicen para acceder a una aplicación.
- ***Herramientas para desarrolladores***. El objetivo de Google es que Flutter sea fácil de usar. Con herramientas como la recarga en caliente, los desarrolladores pueden previsualizar el aspecto de los cambios de código sin perder el estado. Otras herramientas, como el inspector de *widgets*, facilitan la visualización y la resolución de problemas con los diseños de la interfaz de usuario.

Flutter utiliza el lenguaje de programación de código abierto Dart, que también desarrolló Google. Dart está optimizado para la creación de interfaces de usuario, y muchos de los puntos fuertes de Dart se utilizan en Flutter.

Por ejemplo, una característica de Dart que se utiliza en Flutter es la seguridad de los nulos. La seguridad de nulos de Dart facilita la detección de los errores más comunes, llamados errores de nulos. Esta característica reduce el tiempo que los desarrolladores dedican al mantenimiento del código y les da más tiempo para centrarse en la creación de sus aplicaciones.

Flutter viene con un amplio catálogo de *widgets* desde el momento en que lo descargas. El catálogo tiene 14 categorías, que incluyen estilos, Cupertino (*widgets* estilo iOS) y Material Components (*widgets* que siguen las directrices de *Material Design* de Google). Flutter también viene con diseños y temas incluidos, lo que ayuda a los desarrolladores a crear de inmediato.

4.13 Divisor de tensión

Un divisor de tensión o también conocido como divisor de voltaje es un circuito simple que se utiliza en electrónica para dividir la tensión de una fuente de energía entre los

componentes de un circuito eléctrico. Es un arreglo de resistencias en serie que permite establecer un valor de voltaje en la salida del circuito dependiendo de los valores de las resistencias que se han repartido en el circuito (Electronica, 2021).

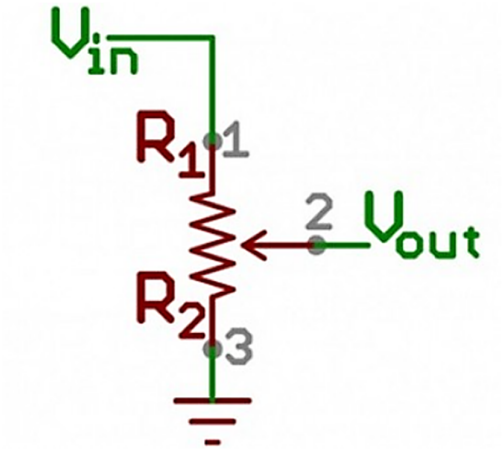


Figura 10. Esquemático de un divisor de tensión. Fuente: 5Hertz.

4.13.1 Ecuación

La ecuación del divisor de voltaje necesita de tres valores: el voltaje de entrada (Vin), y ambos valores de resistencia (R1 y R2). Teniendo en cuenta esos valores, podemos usar esta ecuación para encontrar el voltaje de salida (Vout):

Ecuación 2. Ecuación básica de un divisor de tensión.

$$Vout = Vin \frac{R_2}{R_1 + R_2}$$

4.13.2 Lectura de sensores resistivos

La mayoría de los sensores que existen en el mercado son dispositivos sensibles de constitución electrónica simple. Una fotocelda es una resistencia variable, que produce una resistencia proporcional a la cantidad de luz que detecta. Otros dispositivos como los sensores de flexión, resistencias sensibles a la fuerza (galgas) y termistores, también son resistencias variables que dependen de los agentes externos. Para la implementación de estos sensores se hace uso de microcontroladores, los cuales al contar con pines de entrada que poseen conversores de analógico a digital (ADC *Analogical Digital Converter*) es más sencillo medir el voltaje que la resistencia (5Hertz, 2023).

5 Metodología

En este capítulo se abordarán todo el conjunto de procedimientos y técnicas que se aplicaron de manera ordenada y sistemática para la ejecución de la investigación; permitiendo de esta manera determinar la forma en cómo se van a ordenan y analizan los datos obtenidos.

5.1 Área de Estudio

La presente investigación se inclinó a una población de estudio que cuenta con ciertas capacidades limitadas. Con la ayuda de información proporcionada por el Ministerio de Salud Pública, concretamente por el consejo Nacional para la igualdad de las discapacidades, quien, mediante su sistema de registro, actualizando constantemente la cantidad de personas con discapacidades tanto a nivel local como nacional. Ha permitido de esta manera, tener un contexto más amplio sobre el público objetivo al que se desea llegar con las nuevas tecnologías asistivas.

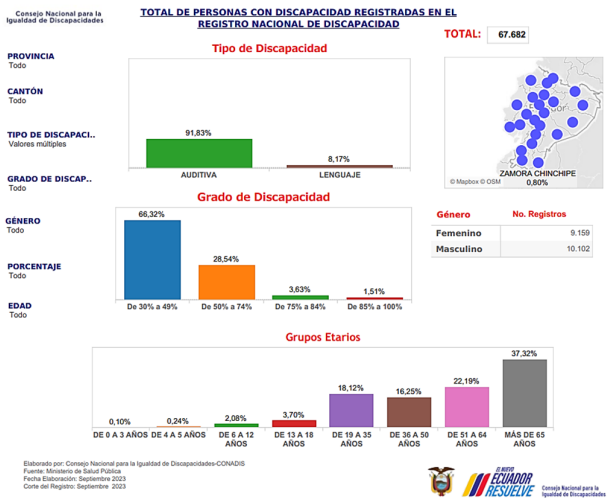


Figura 11. Registro de personas con discapacidad auditiva y verbal a nivel Nacional. Fuente: Ministerio de Salud Publica

5.2 Procedimiento

La metodología que se ha implementado en el transcurso del proyecto, aplicó varios bloques entre los cuales se destacan por etapas.

5.2.1 Investigación de los equipos:

En esta etapa, se dedicó tiempo a investigar a fondo los componentes disponibles en el mercado, que podrían cumplir con los requisitos de la investigación, tales como: sensores flexibles, sensores inerciales, microcontroladores, sintetizadores de voz, pantallas lcd y cualquier otro material electrónico necesario para la investigación.

5.2.2 Comparación de los equipos:

Después de investigar, se procedió a comparar cuidadosamente los componentes electrónicos, analizando sus características técnicas, funcionalidades, costo y disponibilidad para tomar una decisión informada sobre cuáles serían los más adecuados para el proyecto. Tal como podemos observarlo en las Tablas de la sección 5.2.

5.2.3 Selección de los equipos:

Basándose en los resultados de la búsqueda y comparación, se continuó con la selección de los componentes que consideraron más apropiados para el proyecto. Los equipos tienen que presentar las mejores funcionalidades en relación calidad/precio, puesto que, este trabajo está orientado a ser rentable en función del costo sin dejar de lado la calidad del producto final, es por ello, que la elección de los materiales fue minuciosa, para poder cumplir con las expectativas que se tiene en este trabajo.

A continuación, se va a revisar con detenimiento las razones principales por la cual se eligió cada componente para desarrollar el prototipo, empezando por los sensores encargados de capturar los gestos de la mano del usuario, lo cual se puede ver plasmado en la Tabla 1 y la Tabla 2.

Tabla 1. Comparación de sensores

Comparación de sensores				
Característica	**Sensor flexible**	**Sensor de Fuerza FSR 402**	**Sensor de Gestos APDS-9960**	**Sensor de fuerza flexiforce A201**
Imagen				
Voltaje de operación	5 -12 V	5 V	3.3 V	0.25 V – 1.25 V
Tiempo de respuesta	< 100 ms	< 3 us	< 65 us	< 3us
Tipo de respuesta	Variable en función de su resistencia	Variable en función de su resistencia	Caracteres binarios	Variable en función de su resistencia
Vida útil	>1 M de deformaciones	10 M pulsaciones	*	10 M pulsaciones
Temperatura de trabajo	-35 °C a 80 °C	-40 °C a 85 °C	-40 °C a 85 °C	-40 °C a 60 °C
Dimensiones	L 2.2” y 4.5”	L 56.49 mm	L 3.94 x W 2.36 x H 1.35 cm	L 190 mm
Costo	$ 5.54	$ 15.43	$ 8.45	$ 42.15

Nota: Elaborada por el autor.

Gracias a la revisión de los componentes existentes que mejor se adaptan al proyecto, se pudo optar por el sensor flexible como el encargado de poder capturar los valores de flexión de los dedos. La razón por la que se decantó por este componente se debe al tipo de respuesta que posee ya que, por medio de un divisor de voltaje, se puede obtener el resultado de la resistencia que tiene el circuito la cual va a variar de acuerdo al grado de flexión que tenga el sensor ya que mientras más flexión, la resistencia al paso de corriente aumentará por ende el valor del voltaje a la salida será menor. A pesar de que los dos sensores de fuerza tanto el FSR 402 y el flexiforce A21 nos permite tener valores de entrada por medio de un divisor de voltaje, la diferencia radica en la manera de obtener el voltaje de salida, ya que estos componentes funcionan de acuerdo a la fuerza aplicada sobre el mismo. Otra razón por optar por el sensor flex es la de su precio en relación a la calidad, puesto que, ofrece una vida útil de más de un millón de deformaciones lo que permite usarlo por tiempo considerable.

Otro sensor fundamental para poder capturar los gestos de la mano tiene que ver con su posición espacial, es decir un sensor inercial que permita capturar la información relevante del movimiento de toda la muñeca, en la Tabla 2 se puede mostrar los sensores a comparar y el elegido.

Tabla 2. Sensores inerciales

Comparación de sensores inerciales				
Característica	**Acelerómetro ADXL345 CJMCU-105**	**Acelerómetro MMA7361**	**Acelerómetro ADXL335**	**Giroscopio/Acelerómetro MPU-6050**
Voltaje de operación	3.6 V	2 – 3.6 V	3.3 - 5 V	3V/3.3V~5V DC. Regulador de voltaje en placa.
Grados de libertad (DoF)	3	3	3	6
Rango Acelerómetro:	2g/4g/8g/16g	1.5g - 6g	3 g	2g/4g/8g/16g
Rango Giroscopio	No tiene incluido	No tiene incluido	No tiene incluido	250, 500, 1000, 2000 Grad/Seg.
Temperatura de trabajo	-35 °C a 80 °C	-40 °C a 85 °C	-40 °C a 85 °C	-40 °C a 60 °C
Sensibilidad	256 LSB/g	800mV/g a 1.5G	300 mV/g	131 LSBs/dps
Interfaz	SPI e I2C	ADC	ADC	I2C
Salida	Digital	Analógica	Analógica	Digital
Dimensiones	30mm x 50mm x 10mm	30mm x 50mm x 10mm	L 1.94 x W 1.36 x H 0.35 cm	L 2 x W 1.6 x H 0.3 cm
Costo	$ 5.50	$ 6.50	$ 5,70	$ 8,50

Nota: Elaborada por el autor.

Podemos observar que existen una gran variedad de acelerómetros con la capacidad de poder medir la aceleración que sufre cuando está en movimiento, sin embargo, al observar la Tabla 2, se puede apreciar que existe un acelerómetro con giróscopo incluido, el cual ha sido elegido por tener seis grados de libertad, es decir 3 ejes que perciben la aceleración y 3 para los ejes que perciben los grados de inclinación. A pesar de que posee un costo mayor al resto, su diferencia no es mucha, siendo su calidad en relación a precio mucho mayor a los contrastados, sin olvidar que su salida es de valor digital, que se comunica por medio del protocolo de comunión serial I2C. Otro punto importante es la de su tamaño, ya que el prototipo final tiene que ser cómodo y no abarcar tanto espacio de la muñeca.

Un punto muy importante es el cerebro de todo el sistema el cual es dado por un microcontrolador, en la Tabla 3 se contrapuso modelos de diferentes marcas con características similares.

Tabla 3. Microcontroladores a contrastar

Comparación del Microcontrolador				
Aspecto	**Arduino nano**	**ESP32 NodeMCU 32S**	**Raspherry Pi Pico W**	**Arduino R3**
Microcontrolador	ATmega328	Xtensa LX6	RP2040	ATMega328P
Fabricante	Arduino Ltd	Espressif Systems	Raspherry Pi Ltd	Arduino Ltd
Voltaje de operación	5 V	3.3 V	3.3 V	5 V
Alimentación	7 – 12 V Conexión micro USB B Suministros externos o baterías	2.7 - 5.5 V Conexión micro USB B Suministros externos o baterías	1.8 – 5 V Conexión micro USB B Suministros externos o baterías	7 – 12 V Conexión USB B Conexión Tipo Jack Suministros externos o baterías
I/O	22 E/S 8 analógicos 6 PWM	38 E/S 32 GPIO 16 ADC	40 E/S 26 GPIO 4 ADC 16 PWM	14 E/S 6 analógicos 6 PWM
Módulos inalámbricos	No	WiFi, Bluetooth	WiFi, Bluetooth	No
Dimensiones	W 18 x L 45 mm	L 51,4 mm x W 25,4 mm	W 21 mm × L 51 mm	L 68,6 x W 53,4 mm
Precio	$ 24,90	$ 12,5	$ 15,21	$ 27,60

Nota: Elaborada por el autor.

El microcontrolador a elegir debe cumplir con ciertas características entre las cuales son un módulo de conexión inalámbrica y varias entradas analógicas para los sensores flex, sin olvidar que debe tener pines dedicados a una comunicación serial I2C. El módulo que cumple con todos los requisitos es el ESP32, ya que cuenta con un módulo integrado tanto de wifi y

bluetooth que permiten una conexión inalámbrica con otros dispositivos, permitiendo enviar los paquetes de datos que se capturan con los sensores hasta una base de datos. Las entradas analógicas que comprende el ESP32 vienen con un conversor ADC, lo que da como resultado una mejor interpretación del voltaje de salida. Tal como se muestra en la Tabla 3, otra posible elección pudo haber sido la Raspherry Pi Pico W, cuenta con un núcleo más potente y una RAM con mayor capacidad, sin embargo, su desventaja es la falta de canales ADC ya que posee solamente 4, siendo este su principal característica para no haber sido escogida, ya que, en cuestión de precio en relación con la calidad la Raspherry Pi Pico W tiene mejores prestaciones desde un punto electrónico.

Para concluir con la elección de los dispositivos, se tenía que elegir un circuito integrado con la capacidad de alimentar todo el sistema, para ello se dispuso de varios módulos de carga, tal como lo indica la Tabla 4, donde se cotejaron para encontrar al que tenga las características apropiadas para que el desempeño de todo el circuito sea el adecuado.

Tabla 4. Módulos de carga

Comparación de módulos de carga				
Característica	**TP4056**	**TP5400**	**BMS 2S**	**SM5308**
Voltaje de entrada	5 V	5 V	9 – 12 V	5V
Voltaje de salida completa	4.3 V	4.8 V	4.29 V	5 V
Conector	• Micro USB/ Tipo C • Almohadillas de soldadura	• Puerto de Carga Micro USB • Puerto de Descarga USB	• Almohadillas de soldadura	• Puerto de Carga Micro USB C • Puerto de Descarga USB • Almohadillas de soldadura
Corriente de salida	1 A	1.2 A	4 A	2 A
Celdas de carga	1 s	1 s	2 s	1s
Temperatura de trabajo	-10 °C a 80 °C	-10 °C a 80 °C	-40 °C a 80 °C	-40 °C a 50 °C
Tipo de baterías	LiPo Li-ion 18650	LiPo Li-ion 18650	LiPo Li-ion 18650	Li-ion LiPo 18650
Dimensiones	25 x 19 x 10 mm	18 x 22 x 11 mm	35 x17 mm	25 x 20 x 4.5mm
Costo	$ 1.90	$ 2.5	$ 2.29	$ 4.5

Nota: Elaborada por el autor.

Aparte de tomar en cuenta el voltaje de salida con el que se debe trabajar para alimentar todo el circuito, se debe optar por un módulo capaz de tener integrado una función de protección

de sobrecarga y descarga ideal para alargar la vida útil de la batería junto con una corriente de salida adecuada que puede consumir el prototipo, teniendo en cuenta estos datos proporcionados se puede hacer una elección más optima. En este caso, se seleccionó el SM5308, ya que, puede suministrar con una corriente de salida de 2A y además es compatible con baterías tipo 18650, la cual es una ventaja por su tamaño frente a su capacidad de almacenamiento de la energía. No se optó por un integrado de la familia BMS por la razón de la falta de espacio dentro del prototipo, ya que se debe tener en cuenta que el guante tiene que ser cómodo y ocupar el menor espacio para así facilitar el movimiento del prototipo. Además, no se optó por un módulo TP5400 por la falta de presencia de funciones requeridas para la protección de carga, descarga y protección contra cortocircuitos. Estas funciones las tiene presentes el módulo TP4056, sin embargo, la desventaja de este módulo es la salida de tensión, la cual no alcanzará alimentar el circuito, haciendo la necesidad de tener un elevador de tensión como el MT3608. Cabe recalcar que la razón que termino por convencer para optar por el módulo SM5308 es que soporta un botón externo, conectando un push button al punto K y salida negativa, se podrá encender la pantalla de corriente y por ende la salida de 5V, dos pulsaciones cortas consecutivas apagarán la pantalla de corriente y apagarán la salida de 5V.

5.2.4 Diseño del prototipo a implementarse con los equipos elegidos:

Con los equipos seleccionados, se procedió a delimitar detalladamente el prototipo. Para su bosquejo se tiene que tener en cuenta los requisitos específicos del proyecto con el fin de crear un diseño que fuera efectivo y práctico, es decir, desarrollar un diseño preciso del guante, considerando la ergonomía, los sensores necesarios, la duración de la batería y la selección de tecnologías tanto de comunicación como la de procesamiento como machine learning.

En la Figura 12 se muestra el diseño del circuito, para ello se utilizó el software Fritzing para modelarlo. Todo el diseño abarca los sensores escogidos previamente, el Esp32 con sus conexiones hacia cada módulo e inclusive las conexiones de alimentación.

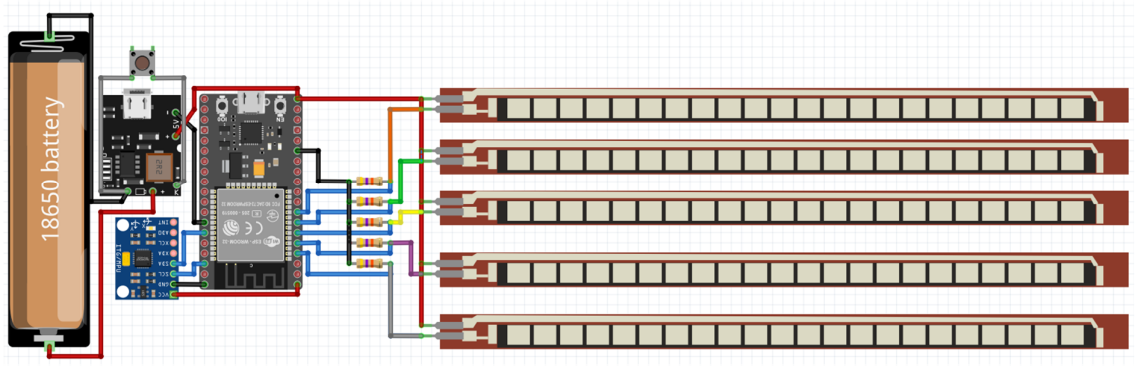


Figura 12. Diseño del circuito del prototipo. Fuente: Elaborado por el autor.

En la Figura 13 se detalló el PCB para su posterior impresión sobre una lámina de cobre sin perforar y de esta manera montar todos los elementos necesarios para el funcionamiento adecuado. Sin olvidar que una de los requerimientos a tomar en cuenta es el tamaño, el diseño cumple con esa condición ya que tiene una medida de 65 x 50 mm

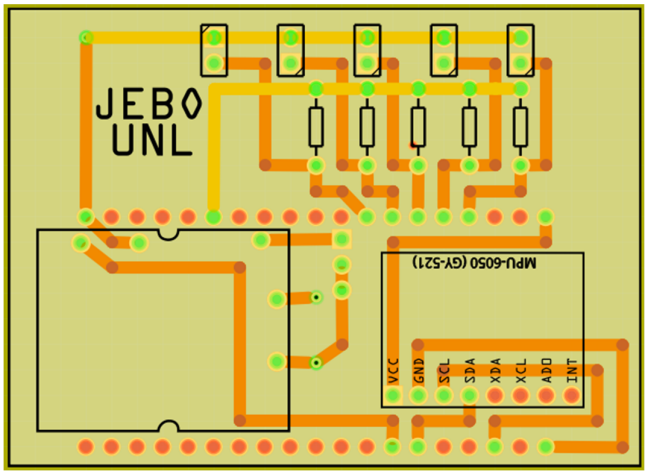


Figura 13. Diseño del PCB. Fuente: Elaborado por el autor.

5.2.5 Construcción:

Esta etapa implicó el ensamblaje y la configuración de los equipos adquiridos para crear el prototipo según el diseño que había desarrollado previamente. Se ha desarrolló el código para la recolección de datos y posteriormente enviarlos a la base de Firebase en el IDE de Arduino (Ver Anexo 2). Sin embargo, antes de ensamblar todo, se imprimió a laser el diseño del PCB sobre una hoja de papel fotográfico para seguidamente hacer uso de la técnica del planchado tal como se indica en la Figura 14 sobre una baquelita de cobre virgen y a continuación proceder con la quemada de está haciendo uso de ácido férrico y de esta manera eliminar las zonas de cobre que no sea parte del diseño del circuito (Ver Figura 15). Una vez concluida la quemada de la baquelita se la limpia con alcohol y se procede a lijarla para limpiar la tinta y observar los caminos de cobre para después perforarla con una broca 0.8 (Ver Figura 16) y así tener listos los orificios por donde se introdujeron los diferentes espadines para posteriormente haberlos soldado y de esta manera ensamblar todos los circuitos integrados necesarios para que funcione el prototipo (Ver Figura 17).

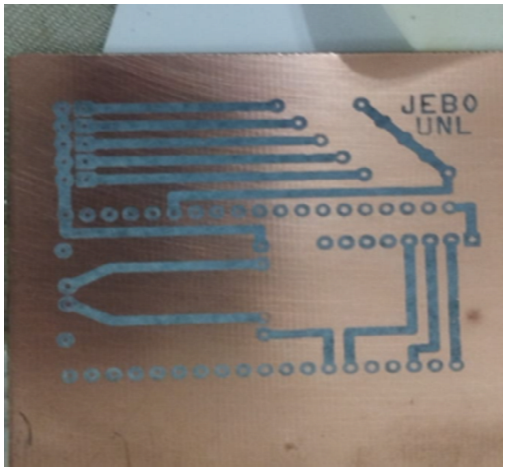


Figura 14. Circuito impreso sobre una baquelita de cobre sin perforar. Fuente: Elaborado por el autor.

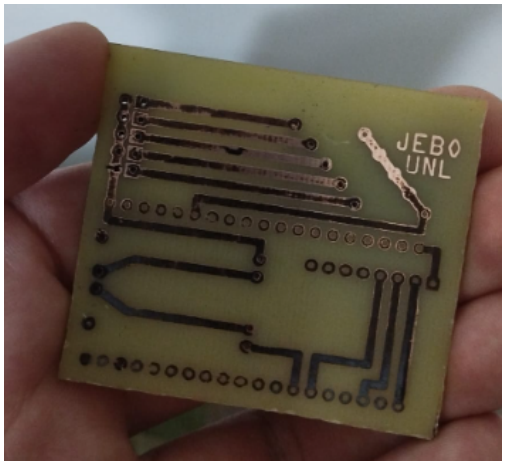


Figura 15. Circuito después de ser eliminadas todas las zonas de *cobre de* la placa virgen con ácido Férrico. Fuente: Elaborado por el autor.

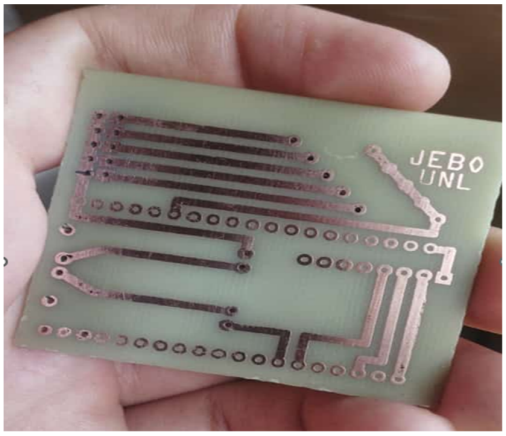


Figura 16. Placa PCB con sus agujeros. Fuente: Elaborado por el autor.

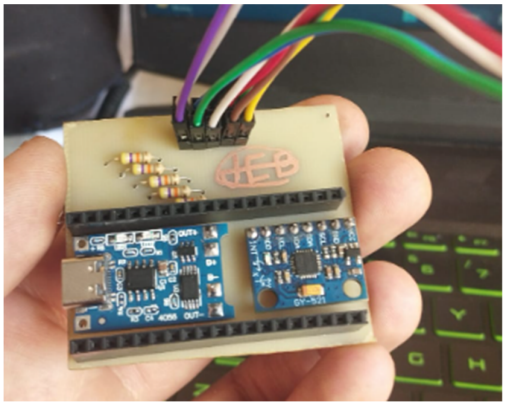


Figura 17. Equipos soldados en el PCB. Fuente: Elaborado por el autor

5.2.6 Diseño de la aplicación móvil:

Como parte de la investigación, también se diseñó una interfaz de usuario y la funcionalidad de una aplicación móvil que complemente al arquetipo. Esta aplicación es crucial para controlar y aprovechar al máximo el prototipo desde dispositivos móviles, además de que permite la conexión entre el teléfono móvil con el guante. La interfaz de usuario cuenta con ciertas configuraciones, entre ellas la de botones que permita activar o desactivar funcionalidades como la de usar un sintetizador de voz o simplemente mostrar los caracteres de las palabras que se traducen.

5.2.7 Implementación de la app

Una vez completado el diseño de la aplicación móvil, se procedió a desarrollarla en un entorno que previamente se eligió en la etapa 7, con un lenguaje de programación claro y fácil, con la finalidad de evitar que su mantenimiento se vuelva complejo. El marco que cumple con estos requisitos es Flutter, que no es nada más que, un software development kit (SDK) para interfaces de usuario de código abierto creado por Google. Se usa para desarrollar aplicaciones cross plataform desde una sola base de código. El IDE que se adaptó a este SDK es Visual Studio Code, el cual es una plataforma que cuenta con soporte y además es de licencia abierta compatible con el lenguaje de programación Dart, que permite programar desde la parte visual hasta la lógica de una app. Otro posible IDE fue Android Studio, el cual presentaba las características similares que el ya antes mencionado, contando con compatibilidad con el lenguaje Dart y los paquetes necesarios para poner en marcha la app, sin embargo, el uso de VScode es más intuitivo y además no demanda muchos recursos del dispositivo donde se programaba.

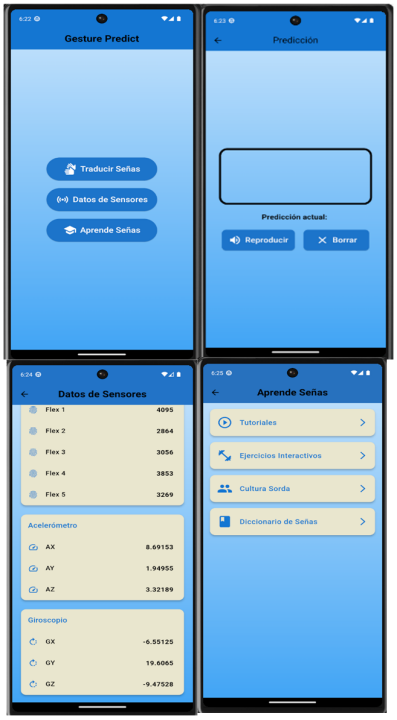


Figura 18. Implementación de la APP desarrollado en Flutter. Fuente: Autor

5.2.8 Pruebas y evaluación:

Para garantizar la eficacia y el rendimiento óptimo del proyecto, se realizó pruebas de validación de funcionamiento a nivel de hardware y de software. Está etapa fue fundamental para identificar y posteriormente corregir los problemas tanto a nivel de hardware como de software. En la figura 19 se observa la pantalla de las pruebas de funcionamiento del interpretador, junto a la pantalla donde llegan los valores brutos de los sensores.

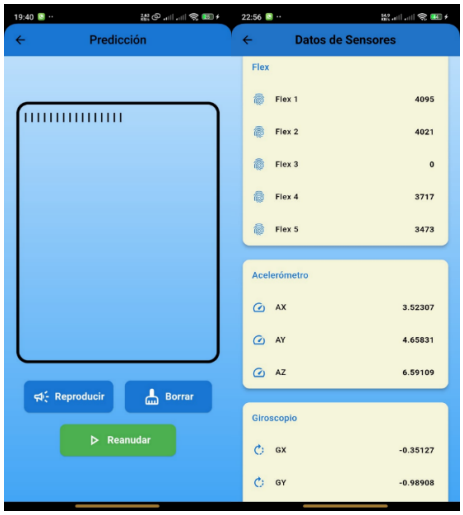


Figura 19. Pruebas de funcionamiento. Fuente: elaborado por el autor.

5.2.9 Prueba de portabilidad

Para certificar la comodidad del prototipo se procedió a efectuar pruebas de porte mediante el uso del mismo en escenarios típicos de la vida cotidiana de cualquier persona, que no incluyan acciones que tenga que manipular algún líquido para evitar estropear el dispositivo. Para ello se hizo una encuesta a un grupo de 10 personas que previamente hicieron uso del guante, tal como lo indica la Tabla 5, los resultados del uso y portabilidad son alentadores en cuanto a su portabilidad.

Tabla 5. Encuesta para determinar el grado de portabilidad del prototipo en casos de uso cotidiano.

Encuesta para verificar la portabilidad del prototipo		
Preguntas	**SI**	**NO**
¿Tuvo incomodidad al usar el guante en un período prolongado?	3	7
¿Tuvo alguna incomodidad en las manos o muñecas después de usar el guante?	0	10
¿El tamaño y peso del guante afectaron su portabilidad?	2	8
¿Pudo desarrollar sus actividades diarias con normalidad mientras usaba el guante?	3	7

5.3 Base de datos

La base de datos que se ha utilizado para entrenar el modelo de aprendizaje automático fue provista mediante el ingreso de valores por parte del autor. Para tomar estos valores se desarrolló un código que permite leer los valores de los sensores sin la necesidad de enviarlos a Firebase, el cual se compilo dentro del ESP32(Ver en Anexos). La programación del código permite obtener los valores de la salida del voltaje de cada sensor flex y además los valores de aceleración en $\frac{m}{s^2}$ y la del giroscopio en grados de inclinación.

Para optimizar la captura de los datos y agruparlos a todos en un dataseet, se ha desarrollado un código en Python (Ver en Anexos) el cual permite etiquetar los valores de entrada que pertenecerían a cada gesto, los cuales son obtenidos por una función que da acceso al monitor serial que se ejecuta una vez que el prototipo está conectado al ordenador, copia los valores recibidos y los almacena en un archivo con extensión .csv. Para robustecer aún más el dataseet, se incluyó una función con la capacidad de añadir más valores que se asemejan a los datos ya obtenidos utilizando interpolación cubica y añadiendo cierto nivel de ruido gaussiano que para simular variabilidad entre cada dato.

5.4 Entrenamiento del modelo

Para entrenar el modelo se ha tomado como punto de partida una arquitectura de red neuronal completamente conectado o también conocida como red neuronal densa. Las diferentes capas tal como se observa en la Figura 20, indican la distribución de cada capa, con sus neuronas correspondientes y la función de activación.

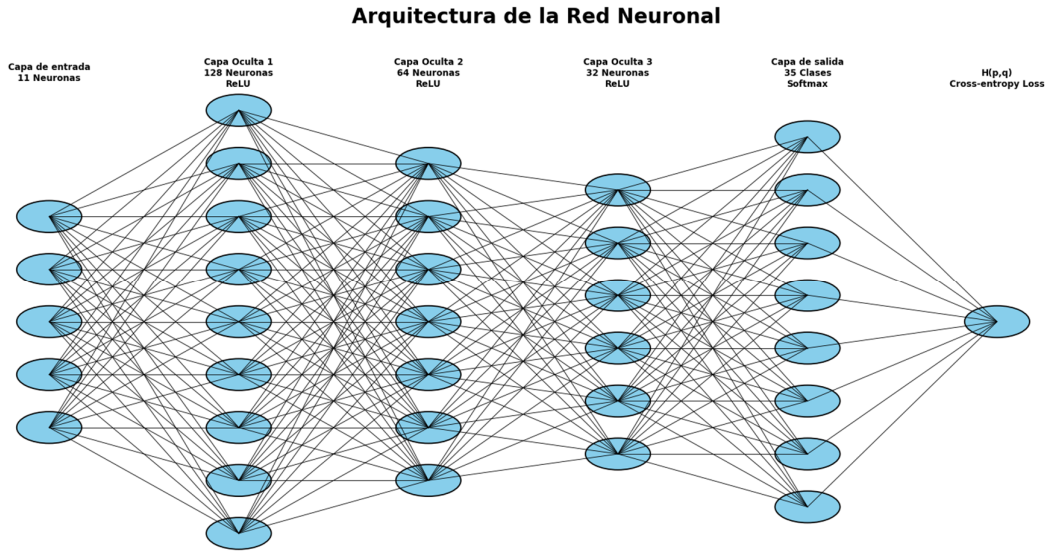


Figura 20. Arquitectura de la red neuronal. Fuente: Elaborado por el autor

Para la primera capa contamos con 11 neuronas, las cuales corresponden a las características de entrada que recibe por parte de los valores de cada sensor; los cinco sensores flex y las 6 medidas del sensor MPU6050. En las capas ocultas tenemos varias que la conforman, empezando con 128 neuronas con una función de activación de unidad lineal rectificada (Relu) la cual permite que las entradas de las escalas positivas no vengan a presentar cambios y que el gradiente pase sin modificaciones a la retropropagación de la red evitando el desvanecimiento de gradiente (Moez, 2024). Así mismo, cuenta con una capa de dropout para evitar un sobreajuste en la red. Seguidamente el número de neuronas va disminuyendo con las mismas funciones de activación y con una capa de dropout entre cada capa oculta hasta llegara a la capa de salida. La última capa contiene el número de neuronas correspondientes a la cantidad de clases que se tienen etiquetadas, se utiliza una función softmax ya que es la que permite clasificar mediante una distribución de probabilidades a cada clase permitiendo que cada salida pertenezca a una clase determinada.

En la parte de la compilación, se utilizó un compilador Adam, ya que, es un algoritmo de optimización eficiente y muy utilizado. Para evitar problemas de perdida se agregó la función sparse_categorical_crossentropy, un algoritmo que permite minimizar los problemas de

clasificación multiclase con etiquetado entero con la finalidad de que al pasar a la sección de la preparación los datos tengan el ínfimo decremento.

Para abordar el entrenamiento de la red se utilizó una configuración de las épocas que sea igual a 100 como el número de veces que el modelo verá el conjunto de entrenamiento completo, además un batch size de 32, el cual comprende la cantidad de muestras que se utilizarán para actualizar los pesos en cada paso, para posteriormente el modelo ser guardado en un formato .keras y luego, convertirlo a un formato TensorFlow Lite (.tflite) para su uso en dispositivos móviles. También se guardan el encoder de etiquetas y el scaler de características para su uso subsiguiente en el procesamiento que se da en la app móvil.

6 Resultados

Como resultado del proyecto se obtuvo el desarrollo de una APK para la interpretación de gestos previamente elegidos y entrenados por una arquitectura de red neuronal densa con la finalidad de traducir señas dactilológicas de personas con discapacidad auditiva. Además, se presenta el diseño del circuito y del PCB donde se acoplaron los sensores y circuitos electrónicos correspondientes para capturar las señales de la muñeca del usuario. La realización se ejecutó a través del IDE Arduino el cual permitió el correcto procedimiento de recolección de información que transmite el sistema a una base de datos en Firebase para su descarga hacia la app. En la investigación mediante el SDK de Flutter se programó la app, la cual está dividida por tres secciones, a las que se ejecutan por medio de botones, cada sección con sus propias funciones, visualizar los gestos en letras y una función de pronunciar con voz artificial los caracteres interpretado, observar los valores que pertenecen a cada sensor y la última está relacionada a la enseñanza de gestos por medio de videos demostrativos en YouTube para mejorar la experiencia de usuario y conocimientos sobre la dactilología. Todo el procedimiento que existe detrás del sistema interpretador de las señas, que comprende desde el acondicionamiento de los equipos para la recolección de datos hasta el interpretador se puede ver en la Figura 21.

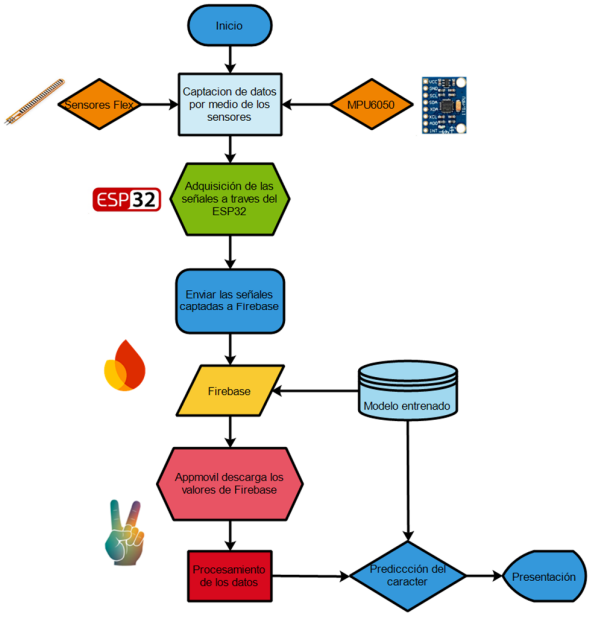


Figura 21. Diagrama de flujo del funcionamiento del sistema. Fuente: Autor

6.1 Precisión del sistema

La precisión con la que el sistema interactuaba en un inicio se puede ver referenciada a lo largo de toda esta sección, quien, por medio de un algoritmo de entrenamiento de validación cruzada, fue obteniendo una estimación más robusta del rendimiento del modelo. El modelo original entrenado fue en formato .keras el cual obtuvo un porcentaje de exactitud del 44%, sin embargo, la aplicación móvil desarrollada solo es compatible con extensión .tflite es por ello que se procedió hacer la conversión. En la Tabla 6 se puede evidenciar que al momento de transformar a un formato .tflite no existió un perdida significativa.

Tabla 6. Reporte de clasificación del 1er modelo TFLite entrenado.

Clase	Precisión	Recuperación	f1-score	Muestras
a	0.48	1.00	0.65	200
b	0.81	0.43	0.56	200
c	0.00	0.00	0.00	200
cómo estás?	0.25	0.22	0.24	150
d	0.00	0.00	0.00	200
e	0.35	0.42	0.38	200
f	0.49	0.93	0.64	200
g	0.39	0.70	0.51	200
gracias	0.00	0.00	0.00	250
h	0.42	0.72	0.53	250
hola	0.59	0.77	0.67	150
i	0.39	0.68	0.50	200
j	0.25	0.76	0.38	200
k	0.13	0.04	0.06	200
l	0.44	0.81	0.57	250
lo siento	0.00	0.00	0.00	150
m	0.80	0.90	0.85	200
n	0.92	0.85	0.88	200
no	0.00	0.00	0.00	150
o	0.68	0.88	0.76	250
p	0.88	0.60	0.72	200
por favor	0.00	0.00	0.00	150
q	0.69	0.99	0.81	200
r	0.34	0.81	0.48	200
s	0.40	0.40	0.40	200
si	0.00	0.00	0.00	200
t	0.03	0.02	0.02	200
u	0.13	0.10	0.11	200
v	0.00	0.00	0.00	200
w	0.00	0.00	0.00	100
x	0.02	0.01	0.01	200
y	0.31	0.50	0.38	200
z	0.92	0.41	0.56	200
Exactitud		**0.44**		**6450**

Nota: Elaborado por el autor.

Sin embargo, como se puede ver en la Tabla 6, la precisión es muy baja. Es por ello que se optó por métodos para mejorar la precisión, entre las cuales estuvieron; aumentar la cantidad de neuronas, evitar el sobreajuste entre las capas densas, optimizar el compilador de entrenamiento y por último aumentar la cantidad de muestra de la base de datos. Esta última fue la que presento mejoras significativas en relación al grado de rendimiento global que posee el modelo, el cual se evidencia en la Tabla 7, la cual en resumen nos demuestra que la base de datos con 10 699 muestras debidamente etiquetados a las clases correspondientes tiene una precisión del 58% de probabilidad a nivel global.

Tabla 7. Reporte de clasificación del 2do modelo entrenado y convertido a formato TFLite.

Clase	Precisión	Recuperación	f1-score	Muestras
a	0.66	1.00	0.80	350
b	0.36	0.93	0.52	350
c	0.85	0.62	0.72	350
cómo estás?	1.00	0.13	0.24	150
d	0.62	0.08	0.14	350
e	0.55	0.62	0.59	350
f	0.40	0.94	0.56	350
g	0.80	0.91	0.85	350
gracias	0.67	0.01	0.03	150
h	0.75	0.82	0.79	450
hola	0.45	0.82	0.58	300
i	0.34	0.81	0.48	350
j	0.84	0.53	0.65	386
k	0.71	0.55	0.62	350
l	0.66	0.84	0.74	400
lo siento	0.87	0.97	0.92	150
m	0.87	0.92	0.89	350
n	0.85	0.91	0.88	350
no	0.80	0.78	0.79	300
o	0.75	0.97	0.85	413
p	0.68	0.80	0.74	350
Por favor	0.58	0.77	0.66	150
q	0.77	0.85	0.81	350
r	0.28	0.90	0.43	350
s	0.49	0.54	0.51	350
si	0.65	0.56	0.60	200
t	0.00	0.00	0.00	400
u	0.00	0.00	0.00	350
v	0.00	0.00	0.00	350
w	0.00	0.00	0.00	250
x	0.00	0.00	0.00	350
y	0.00	0.00	0.00	350
z	0.00	0.00	0.00	350
Exactitud		**0.58**		**10699**

Nota: Elaborado por el autor.

A pesar de haber tenido una mejora en cuanto a la precisión del segundo modelo entrenado, se optó por incrementar la base de datos a una cantidad de 17 350 muestras etiquetadas para las clases descritas en la Tabla 8, obteniendo una exactitud del 61% a nivel global.

Tabla 8. Reporte de clasificación del 3er modelo TFLite entrenado.

Clase	Precisión	Recuperación	f1-score	Muestras
a	0.87	0.99	0.93	550
b	0.53	0.95	0.68	550
c	0.69	0.62	0.65	550
chao	0.38	0.65	0.48	500
cómo estás?	0.91	0.54	0.67	450
d	0.00	0.00	0.00	550
e	0.41	0.83	0.55	550
f	0.34	0.97	0.51	550
g	0.85	0.91	0.88	550
gracias	0.64	0.90	0.75	450
h	0.88	0.87	0.88	650
hola	0.46	0.97	0.63	300
i	0.67	0.91	0.77	550
j	0.70	0.87	0.77	550
k	0.66	0.82	0.73	550
l	0.75	0.87	0.81	600
lo siento	0.87	0.95	0.91	350
m	0.91	0.95	0.93	550
n	0.72	0.94	0.81	550
no	0.91	0.51	0.65	300
o	0.96	0.49	0.65	600
p	0.95	0.85	0.89	550
Por favor	0.86	0.75	0.80	350
q	0.71	0.99	0.82	550
r	0.29	0.85	0.43	550
s	0.53	0.70	0.60	550
si	0.93	0.26	0.40	200
t	0.00	0.00	0.00	600
u	0.00	0.00	0.00	550
v	0.00	0.00	0.00	550
w	0.00	0.00	0.00	450
x	0.00	0.00	0.00	550
y	0.00	0.00	0.00	550
z	0.00	0.00	0.00	550
Exactitud		**0.61**		**17350**

Nota: Elaborado por el autor.

Con la información recabada de las tres tablas presentadas, podemos hacer una inferencia por medio de una ecuación de regresión lineal sobre la cantidad posible de muestras para obtener un resultado de precisión igual al 80% el cual es de 104 200 aproximadamente. Sin embargo, al aumentar clases al modelo entrenado, se puede evidenciar que su rendimiento alrededor de la exactitud o *accuracy* global incrementa. Junto a ello también se propuso

aumentar la cantidad de muestras de las letras que cumplen con requisitos como; tener una mayor probabilidad de ser detectadas por el intérprete y además de pertenecer al conjunto de letras más usados en el idioma español. Estos resultados se analizan gracias a la Tabla 9, donde se refleja el rendimiento del sistema por clase, frente a su cantidad de muestras captadas y entrenadas para cada salida. Este último entrenamiento siguió los mismos parámetros preestablecidos, utilizando la misma arquitectura descrita y la misma configuración de validación cruzada, la única diferencia del resto de modelos, es la cantidad de clases (35) y la cantidad de muestras globales para el entrenamiento (20 040).

Tabla 9. Reporte de clasificación del 4to modelo entrenado y convertido a fomrato TFLite.

Clase	Precisión	Recuperación	f1-score	Muestras
a	0.87	0.99	0.92	750
b	0.55	0.95	0.69	550
c	0.68	0.82	0.74	750
chao	0.76	0.72	0.74	700
cómo estás?	0.84	0.72	0.78	450
d	0.37	0.83	0.51	750
e	0.48	0.85	0.61	750
f	0.83	0.03	0.07	550
g	0.95	0.91	0.93	550
gracias	0.46	0.95	0.62	650
h	0.80	0.87	0.83	650
hola	0.39	0.77	0.52	300
i	0.80	0.90	0.84	750
j	0.81	0.77	0.79	550
k	0.75	0.71	0.73	550
l	0.81	0.89	0.85	800
lo siento	0.93	0.95	0.94	350
m	0.95	0.95	0.95	550
n	0.68	0.95	0.79	750
no	0.86	0.76	0.81	295
no hay problema	0.90	1.00	0.95	300
o	0.81	0.95	0.88	800
p	0.98	0.85	0.91	550
por favor	0.75	0.74	0.74	350
q	0.75	1.00	0.86	550
r	0.37	0.89	0.52	745
s	0.87	0.54	0.66	750
si	0.00	0.00	0.00	200
t	0.00	0.00	0.00	600
u	0.00	0.00	0.00	550
v	0.00	0.00	0.00	550
w	0.00	0.00	0.00	450
x	0.00	0.00	0.00	550
y	0.00	0.00	0.00	550
z	0.00	0.00	0.00	550
Exactitud		**0.66**		**20040**

Nota: Elaborado por el autor.

7 Discusión

Los resultados obtenidos en este proyecto son concordantes con estudios relacionados de aprendizaje autónomo supervisado utilizando técnicas como la aplicación de redes neuronales para la clasificación multiclase. Utilizando como base parte de códigos y anotaciones se pudo desarrollar el procesamiento que este detrás de la app. Investigaciones anteriores han demostrado la eficacia de la interpretación de gestos utilizando sensores flexibles y sensores mpu6050, sin embargo, su propuesta incluye algoritmos previamente seleccionados y calibrados para funcionar con un lote de muestras menor a las planteadas en este proyecto. Cabe recalcar que las redes neuronales densas para el reconocimiento de patrones de movimiento, son un método confiable para predecir e interpretar valores, no obstante, el proyecto proporciona estos hallazgos con una precisión del 66% en la interpretación de gestos. Aunque, estudios que han empleado algoritmos más avanzados como el Deep Learning han reportado precisiones superiores, sugiriendo una posible mejora en futuras iteraciones del sistema mediante la implementación de estas técnicas más sofisticadas.

El método utilizado en este proyecto, que incluye la recolección de datos con el IDE Arduino y su procesamiento en una red neuronal densa, ha mostrado ser efectivo para los objetivos planteados. No obstante, se identificaron limitaciones, como la necesidad de evitar movimientos bruscos para mantener la conectividad de los cables, además de una fuente de alimentación con mayores capacidades de retención de carga, lo que podría afectar la robustez y comodidad del sistema en aplicaciones del mundo real. Esta observación subraya la importancia de un diseño de hardware más integrado, resistente y portable.

La hipótesis principal del proyecto, que planteaba que un sistema basado en sensores y una red neuronal densa podría interpretar gestos con una precisión aceptable, se ha verificado. La precisión del 66% alcanzada por el modelo entrenado para poder interpretar respalda esta hipótesis, indicando que los antecedentes teóricos fueron correctos en predecir el desempeño del sistema. Sin embargo, esta exactitud alcanzada se debe al reconocimiento de ciertas letras y gestos que pertenecen a palabras, las razones por las que ciertas clases tienen un mayor grado de precisión e interpretación son: Gestos que tienen menor relación en cuestiones de sus queremos con respecto a otros gestos. Es decir que los gestos con movimiento, tienen mayor probabilidad de ser reconocidos, así mismo, letras con gesto en la cual los valores de entrada son muy diferentes a otras clases, también tienen probabilidades más altas de ser reconocidas. Para potenciar el rendimiento del modelo, se ajustó el número de muestras por clase dependiendo del grado de usabilidad en el idioma español, es decir letras como la E, A, O, S,

R, N, I, D, L, C, tienen una mayor cantidad de datos para mejorar su probabilidad de ser interpretada.

Al trabajar en este proyecto se pudo dar cuenta que la viabilidad de desarrollar un sistema portátil para la interpretación de gestos utilizando tecnología de sensores y redes neuronales, puede ser rentable dependiendo el caso de uso. Si bien, el prototipo desarrollado no solo es funcional, sino que también ofrece una precisión significativa en la interpretación de gestos, se identificaron áreas de mejora, como la expansión de la base de datos de gestos y la incorporación de algoritmos más avanzados para mejorar la precisión del sistema.

Una de las limitaciones identificadas es la generalización de los resultados. Aunque el sistema demostró ser preciso en un entorno controlado, su desempeño en escenarios más diversos y con una muestra de usuarios más amplia aún debe ser evaluado. Además, la dependencia de la conectividad de los cables sugiere la necesidad de un diseño de hardware más robusto. Futuras investigaciones podrían beneficiarse de explorar el uso de algoritmos de Deep Learning o diferentes arquitecturas de redes neuronales, como redes neuronales convolucionales para la identificación de gestos mediante un modelo pre-entrenado con imágenes de los gestos a reconocer.

En resumen, esta sección no solo ofrece una comparación entre los resultados obtenidos en esta investigación validando los objetivos planteados, sino que, además, también ofrece un análisis crítico de las metodologías utilizadas y las proyecciones para estudios futuros, destacando la importancia de continuar mejorando el sistema para lograr una mayor precisión y robustez en aplicaciones del mundo real.

8 Conclusiones

- Se concluye el desarrollo de los objetivos propuestos de investigación los cuales abarcan el diseñar e implementar un prototipo capaz de poder recolectar la información provista por sensores que seguidamente se envían a firebase para ser procesados una vez que se descarguen por la aplicación móvil desarrollada para poder visualizar y pronunciar los caracteres interpretados.

- El sistema tiene un promedio de interpretación del 66% para el modelo entrenado, sin embargo, si se quiere hacer uso de más gestos se tiene que tener una base de datos de los valores de los nuevos gestos, previamente obtenidos para su posterior entrenamiento y conversión a extensión ftlite.

- La capacidad de interpretación del modelo entrenado, puede mejorar con mayores muestras etiquetadas de cada clase, sin embargo, este aumento de mejora incluye un mayor procesamiento por parte del dispositivo móvil, lo que ocasiona que la app se vuelva lenta, para ello se tendría que optimizar de una mejor manera los hiperpárametros para alcanzar un equilibrio entre el procesamiento y la precisión.

- El desempeño del sistema viene acompañado de la integración del diseño del circuito que incluye el microcontrolador, los sensores flex de 2.2”, el Mpu6050, junto con los valores del divisor de tensión que se usen, ya que el uso de diferentes componentes puede venir afectar los resultados, puesto que, los valores recolectados para la interpretación serian diferentes a los entrenados y no existiría una predicción correcta por parte de la app.

- El sistema tiene un mejor rendimiento para los gestos que incorporen movimientos, a su vez, el modelo final tiene una mayor probabilidad d de interpretar los caracteres mas usados en el idioma español, sin embargo, algunas letras como la E u la O, son difíciles ya que cuenta con entradas similares por parte de los sensores flex y el MPU6050, donde no puede diferenciar con exactitud una clase de otra.

9 Recomendaciones

- Se recomienda evitar el uso del prototipo con movimientos bruscos para evitar la desconexión de cualquier cable. Además, antes de verter algún líquido para la limpieza del guante, se debe desarmar el sistema.
- Se recomienda que, si se desea hacer uso de más clases, se sigan los pasos descritos en la metodología para aumentar la base de datos con las nuevas clases y de esa manera reentrenar el modelo para que conozca los nuevos gestos.
- Se recomienda que, para seguir potenciando el aprendizaje del modelo, se centre en el aumento de muestras de las letras con mayor usabilidad en el idioma español, además de los gestos más usuales.
- Se recomienda el uso de algoritmos más completos como Deep learning o alguna arquitectura diferente de red neuronal descrita en este proyecto para así contrastar los resultados y evidenciar cual es la más rentable y posiblemente mejorar la precisión de la traducción.
- Se recomienda que, para mejorar este sistema, se opte por desarrollar un modelo de red neuronal utilizando una técnica de redes neuronales convolucionales, evitando así crear el prototipo y centrándose en la creación de una app con las capacidades de interpretar mediante el uso de su cámara, los gestos y poder plasmarlos tanto en texto como en audio.

10 Referencias bibliográficas

5Hertz. (2023). *Divisor de Tensión*. Obtenido de 5Hertz.com: https://www.5hertz.com/index.php?route=tutoriales/tutorial&tutorial_id=11

Abhishek, K. S., Qubeley, L. C., & D. Ho. (2016). *Glove-based hand gesture recognition sign language translator using capacitive touch sensor*. Recuperado el Febrero de 10, de https://ieeexplore.ieee.org/document/7785276

Argentina.gob. (s.f.). *Hipoacusia*. (Argentina.gob.ar) Recuperado el 13 de Diciembre de 2023, de https://www.argentina.gob.ar/salud/glosario/hipoacusia-sordera#:~:text=La%20hipoacusia%2C%20sordera%20o%20deficiencia,cuando%20ambos%20o%C3%ADdos%20est%C3%A1n%20afectados.

AWS. (2023). *¿Cuál es la diferencia entre microprocesadores y microcontroladores?* Obtenido de Amazon Web Servicies: https://aws.amazon.com/es/compare/the-difference-between-microprocessors-microcontrollers/#:~:text=Los%20microprocesadores%20permiten%20operaciones%20de,las%20entradas%20en%20tiempo%20real.

AWS. (2024). *What is Machine Learning*. Obtenido de aws.amazon.com: https://aws.amazon.com/es/what-is/machine-learning/#:~:text=El%20machine%20learning%20es%20la,basarse%20en%20patrones%20e%20inferencias.

aws. (2024). *What is neural network*. Obtenido de aws.amazon.com: https://aws.amazon.com/es/what-is/neural-network/

Beningo, J. (21 de Enero de 2020). *How to select and use the right esp32 module*. Obtenido de digikey,.com: https://www.digikey.com/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module

Caldas, E. (19 de Abril de 2024). *Sensores de Fuerza*. Obtenido de Electronicos Caldas: https://www.electronicoscaldas.com/es/sensores-de-fuerza-peso-estres/255-sensor-de-fuerza-fsr-402.html

Caparrini, F. S. (s.f.). *Redes Neuronales*. Obtenido de cs.us.es: https://www.cs.us.es/~fsancho/Blog/posts/Redes_Neuronales.md

CONADIS. (Septiembre de 2023). *Estadísticas de Discapacidad*. Obtenido de Consejo de Discapacidades: https://www.consejodiscapacidades.gob.ec/wp-content/uploads/downloads/2015/09/estadistica_conadis.pdf

CyberGlove. (s.f.). *CyberGlove II*. (CyberGlove Systems LLC) Recuperado el Febrero de 2024, de https://www.cyberglovesystems.com/cyberglove-ii/

dataScientest. (2 de Septiembre de 2022). *IDE que es* . Obtenido de datascientest.com: https://datascientest.com/es/ide-que-es

DHEX. (s.f.). *Fundacion DHEX vivir la sordera*. (Facebook.com) Recuperado el 31 de Enero de 2024, de https://www.facebook.com/FundacionDhexVivirLaSorderaEcuador/?locale=es_LA

Electronica. (22 de Octubre de 2021). *DIVISOR DE VOLTAJE*. Obtenido de Electronica.com.gt: https://laelectronica.com.gt/extras/divisor-de-voltaje

Estapé, J. A. (21 de Junio de 2019). *Google Gesture para personas sordomudas*. (computerhoy.com) Recuperado el 20 de Diciembre de 2023, de https://computerhoy.com/noticias/hardware/google-gesture-voz-personas-sordomudas-14631

FenixLighting. (8 de Febrero de 2021). *La guia definitiva de la bateria 18650*. Obtenido de fenixlighting.com: https://www.fenixlighting.com/blogs/news/the-ultimate-guide-to-the-18650-battery

García, J. C. (Abril de 2019). *LA DISCAPACIDAD AUDITIVA. PRINCIPALES MODELOS Y AYUDAS TÉCNICAS PARA LA INTERVENCIÓN.* (Redalyc.org) Recuperado el 13 de Diciembre de 2023, de https://www.redalyc.org/pdf/5746/574661395002.pdf

Hernández Samacá, S. F., & Calderón Quintero, O. D. (10 de Junio de 2022). *Desarrollo de guantes traductores de lengua de señas* . Obtenido de unab.edu.co: https://repository.unab.edu.co/handle/20.500.12749/16875

HeTPro. (2023). *Cargador para baterias SM5308*. Obtenido de hetpro-store.com: https://hetpro-store.com/cargador-para-baterias-18650-2a/

Izquierdo, J. (22 de Noviembre de 2021). *Arquitectura Von Neumann y arquitectura Harvard*. Obtenido de weblinus.com: https://weblinus.com/arquitectura-von-neumann-y-arquitectura-harvard/

Luna, E. (2019). *Que es un Editor de Texto*. Obtenido de Platzi.com: https://platzi.com/blog/que-es-ide-editor-de-texto/

Martínez, B. (20 de Febrero de 2018). *Aprendamos un poco del lenguaje de señas*. Obtenido de prensalibre.com: https://www.prensalibre.com/vida/salud-y-familia/aprendamos-un-poco-del-lenguaje-de-seas/

Meriño Guzman, J. A., & Garizabalo Pedrozo, D. (6 de Julio de 2020). *Diseño de un guante electrónico para la interpretación y traducción del lenguaje de señas en personas con discapacidad auditiva mediante tecnología arduino e interfaz de visualización por medio de una aplicación en android*. Obtenido de UTS.edu.co: http://repositorio.uts.edu.co:8080/xmlui/handle/123456789/3302

Microsoft. (2024). *Visual Studio*. Obtenido de microsoft.com: https://visualstudio.microsoft.com/es/

Miguel, F. H. (s.f.). *Acerca de*. (Fundacinhermanomiguel.wordpress.com) Obtenido de https://fundacinhermanomiguel.wordpress.com/

Moez, A. (Abril de 2024). *Introduction to activation functions in neural networks*. Obtenido de Datacamp.com: https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks

OMS. (2011). *Informe mundial de la discapacidad*. (OAS.org) Recuperado el 2024 de Enero de 30, de https://www.oas.org/es/sedi/ddse/paginas/documentos/discapacidad/DESTACADOS/ResumenInformeMundial.pdf

Princesa, P. (21 de Julio de 2021). *Nuevas tecnologías para pacientes con problemas auditivos* . (Psicólogos Madrid | Centro Psicólogos Princesa) Recuperado el Febrero de 2024, de https://psicologosprincesa81.com/blog/nuevas-tecnologias-para-pacientes-con-problemas-auditivos/

Sánchez, M. (2023). *Que es un micrprocesador y cual es su funcion*. Obtenido de PCcomponentes.com: https://www.pccomponentes.com/que-es-un-microprocesador-cual-es-su-funcion

Santos, P. R. (2 de Diciembre de 2021). *Que algoritmo elegir en m aprendizaje*. Obtenido de Telefonicatech.com: https://telefonicatech.com/blog/que-algoritmo-elegir-en-ml-aprendizaje

SunriseMedical. (08 de Marzo de 2021). *Autoestima y autoimagen en personas con discapacidad*. Obtenido de Sunrisemedical.es: https://www.sunrisemedical.es/blog/autoestima-y-autoimagen-en-personas-con-discapacidad

TECNONEO. (2013). *Anillos traductores de lenguaje*. (TECNONEO) Recuperado el Febrero de 2024, de http://www.tecnoneo.com/2013/11/anillos-traductores-del-lenguaje-de.html

waveshare. (2023). *NodeMCU-32S, ESP32 WiFi+placa de desarrollo Bluetooth*. Obtenido de waveshare.com: https://www.waveshare.com/nodemcu-32s.htm

Zúñiga, F. G. (1 de Junio de 2024). *Que es visual studio code y cuales son sus ventajas*. Obtenido de arsys.es: https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas

11 Anexos

Anexo 1. Código para enviar los datos obtenidos desde los sensores, y son enviados a Firebase Database Realtime.

```
#include <MPU6050_tockn.h>
#include <FirebaseESP32.h>
#include <WiFi.h>
#include <Wire.h>

const char* ssid = "JosuNet";
const char* password = "152000Josu";
#define FIREBASE_HOST "appmovil-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "Fl2aTehE72BGUs4mm1e5QGbjOII9ac1xDo"
const int LED_BUILTIN = 2;

FirebaseData firebaseData;
FirebaseConfig firebaseConfig;
FirebaseAuth firebaseAuth;

const int Pulgar = 32;
const int Indice = 34;
const int Medio = 33;
const int Anular = 35;
const int Menique = 39;

MPU6050 mpu6050(Wire);
void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  // Iniciar WiFi
  WiFi.begin(ssid, password);
  Serial.print("Conectando a WiFi......");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
  }
  Serial.println("\nConectado a WiFi");
  // Configurar Firebase
  firebaseConfig.host = FIREBASE_HOST;
  firebaseConfig.signer.tokens.legacy_token = FIREBASE_AUTH;
  Firebase.begin(&firebaseConfig, &firebaseAuth);
  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);
}
void loop() {
```

```
  delay(500);
  mpu6050.update();
  int flex1 = analogRead(Pulgar);
  int flex2 = analogRead(Indice);
  int flex3 = analogRead(Medio);
  int flex4 = analogRead(Anular);
  int flex5 = analogRead(Menique);
  float ax = mpu6050.getAccX() * 9.81; // Convertir a m/s^2
  float ay = mpu6050.getAccY() * 9.81;
  float az = mpu6050.getAccZ() * 9.81;
  float gx = mpu6050.getGyroX(); // grados/s
  float gy = mpu6050.getGyroY();
  float gz = mpu6050.getGyroZ();
    Serial.println("Valores de los sensores flex:");
// Imprimir valores en el monitor serial
  Serial.println("Pulgar: " + String(flex1));
  Serial.println("Indice: " + String(flex2));
  Serial.println("Medio: " + String(flex3));
  Serial.println("Anular: " + String(flex4));
  Serial.println("Meñique: " + String(flex5));
  Serial.println("Valores del MPU6050:");
  Serial.println("Aceleración X: " + String(ax) + " m/s^2");
  Serial.println("Aceleración Y: " + String(ay) + " m/s^2");
  Serial.println("Aceleración Z: " + String(az) + " m/s^2");
  Serial.println("Giroscopio X: " + String(gx) + " grados/s");
  Serial.println("Giroscopio Y: " + String(gy) + " grados/s");
  Serial.println("Giroscopio Z: " + String(gz) + " grados/s");
  if (Firebase.ready()) {   // Actualizar datos en Firebase
    Firebase.setInt(firebaseData, "/sensores/Flex1", flex1);
    Firebase.setInt(firebaseData, "/sensores/Flex2", flex2);
    Firebase.setInt(firebaseData, "/sensores/Flex3", flex3);
    Firebase.setInt(firebaseData, "/sensores/Flex4", flex4);
    Firebase.setInt(firebaseData, "/sensores/Flex5", flex5);
    Firebase.setFloat(firebaseData, "/sensores/AX", ax);
    Firebase.setFloat(firebaseData, "/sensores/AY", ay);
    Firebase.setFloat(firebaseData, "/sensores/AZ", az);
    Firebase.setFloat(firebaseData, "/sensores/GX", gx);
    Firebase.setFloat(firebaseData, "/sensores/GY", gy);
    Firebase.setFloat(firebaseData, "/sensores/GZ", gz);
    Serial.println("Datos actualizados en Firebase");
  }

  delay(500); // Ajusta este valor según la frecuencia de muestreo que se
necesite
}
```

Anexo 2. Código para capturar los datos y visualizarlos en el monitor serial.

```
#include <MPU6050_tockn.h>
#include <Wire.h>
const int Pulgar = 32;
const int Indice = 34;
const int Medio = 33;
const int Anular = 35;
const int Menique = 39;
MPU6050 mpu6050(Wire);
void setup() {
  Serial.begin(115200);
  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);
}

void loop() {
  mpu6050.update();
  int flex1 = analogRead(Pulgar);
  int flex2 = analogRead(Indice);
  int flex3 = analogRead(Medio);
  int flex4 = analogRead(Anular);
  int flex5 = analogRead(Menique);
  float ax = mpu6050.getAccX() * 9.81; // Convertir a m/s^2
  float ay = mpu6050.getAccY() * 9.81;
  float az = mpu6050.getAccZ() * 9.81;
  float gx = mpu6050.getGyroX(); // Ya está en grados/s
  float gy = mpu6050.getGyroY();
  float gz = mpu6050.getGyroZ();

  // Enviar datos por el puerto serial
  Serial.print(flex1);  Serial.print(",");
  Serial.print(flex2);  Serial.print(",");
  Serial.print(flex3);  Serial.print(",");
  Serial.print(flex4);  Serial.print(",");
  Serial.print(flex5);  Serial.print(",");
  Serial.print(ax);  Serial.print(",");
  Serial.print(ay);  Serial.print(",");
  Serial.print(az);  Serial.print(",");
  Serial.print(gx);  Serial.print(",");
  Serial.print(gy);  Serial.print(",");
  Serial.println(gz);

  delay(500); // Ajusta este valor según la frecuencia de muestreo que
necesites
}
```

Anexo 3. Código para automatizar la captura de datos y agregar muestras

```
import serial
import csv
import time
import numpy as np
from sklearn.preprocessing import StandardScaler
from scipy.interpolate import interp1d
import joblib

# Configuración del puerto serial
SERIAL_PORT = 'COM3'
BAUD_RATE = 115200
TIMEOUT = 0.5

# Configuración del archivo CSV
CSV_FILE = 'guante1_df.csv'
PROCESSED_CSV_FILE = 'guante1_processed_df.csv'
HEADER = ['Flex1', 'Flex2', 'Flex3', 'Flex4', 'Flex5', 'Ax', 'Ay', 'Az', 'Gx',
'Gy', 'Gz', 'Letra']
SAMPLES_PER_LETTER = 50
AUGMENTED_SAMPLES = 50  # Número de muestras adicionales a generar por letra

def initialize_serial():
    try:
        return serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=TIMEOUT)
    except serial.SerialException as e:
        print(f"Error al abrir el puerto serial: {e}")
        return None

def read_sensor_data(ser):
    data = ser.readline().decode('utf-8').strip()
    return data.split(',')

def augment_data(data):
    augmented_data = []
    for letter in np.unique(data[:, -1]):
        letter_data = data[data[:, -1] == letter][:, :-1]

        # Interpolación
        x = np.linspace(0, 1, len(letter_data))
        x_new = np.linspace(0, 1, len(letter_data) + AUGMENTED_SAMPLES)

        augmented_letter_data = []
        for i in range(letter_data.shape[1]):
            f = interp1d(x, letter_data[:, i], kind='cubic')
            augmented_letter_data.append(f(x_new))
```

```
        augmented_letter_data = np.array(augmented_letter_data).T

        # Añadir ruido gaussiano
        noise = np.random.normal(0, 0.01, augmented_letter_data.shape)
        augmented_letter_data += noise

        # Añadir etiquetas
        augmented_letter_data = np.column_stack((augmented_letter_data,
np.full(len(augmented_letter_data), letter)))
        augmented_data.append(augmented_letter_data)

    return np.vstack(augmented_data)

def main():
    ser = initialize_serial()
    if not ser:
        return

    all_data = []

    with open(CSV_FILE, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(HEADER)

        try:
            while True:
                letter = input("Ingrese la letra correspondiente (o '1' para
salir): ").lower()
                if letter == '1':
                    break

                print(f"Capturando datos para la letra '{letter}'. Presione
Ctrl+C para detener.")
                count = 0

                while count < SAMPLES_PER_LETTER:
                    try:
                        sensor_values = read_sensor_data(ser)
                        if len(sensor_values) == 11:  # Para asegurarse de que
se tiene todos los valores
                            writer.writerow(sensor_values + [letter])
                            all_data.append(sensor_values + [letter])
                            count += 1
                            print(f"Muestra {count}/{SAMPLES_PER_LETTER}
capturada", end='\r')
                            print(sensor_values)
                        else:
```

```
                            print("Datos incompletos recibidos, ignorando esta
muestra.")
                    except KeyboardInterrupt:
                        print("\nCaptura para esta letra interrumpida.")
                        break

                print(f"\nSe capturaron {count} muestras para la letra
'{letter}'")

        except KeyboardInterrupt:
            print("\nPrograma terminado por el usuario.")
        finally:
            ser.close()
            print("Puerto serial cerrado.")

    # Procesar y aumentar datos
    all_data = np.array(all_data)
    augmented_data = augment_data(all_data)

    # Guardar datos procesados
    with open(PROCESSED_CSV_FILE, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(HEADER)
        for row in augmented_data:
            writer.writerow(row)

    print(f"Datos procesados y aumentados guardados en {PROCESSED_CSV_FILE}")

if __name__ == "__main__":
    main()
```

Anexo 4. Código para entrenar el modelo

```
import os
#os.environ["TF_USE_LEGACY_KERAS"] = "1"
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
#from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
import joblib

# Leer el archivo CSV
data = pd.read_csv('guante_c.csv')

# Separar las características (X) y etiquetas (y)
X = data[['Flex1', 'Flex2', 'Flex3', 'Flex4', 'Flex5', 'Ax', 'Ay', 'Az', 'Gx',
'Gy', 'Gz']]
y = data['Letra']

# Codificación de las etiquetas (letras) como valores numéricos
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Normalización de las características
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# División de los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded,
test_size=0.2, random_state=42)

#Ddefinición de la arquitectura de la red neuronal
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu',
input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax')
])

# Compilación del modelo
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

```
# Callbacks
#early_stopping = EarlyStopping(patience=10, restore_best_weights=True)
#reduce_lr = ReduceLROnPlateau(factor=0.2, patience=5, min_lr=0.0001)

# Entrenamiento del modelo
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_split=0.2,
                    )

# Evaluación del modelo
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Precisión del modelo: {accuracy * 100:.2f}%')

# Guardar el modelo entrenado
model.save('Mi_modelo.keras')

# Conversión el modelo a TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()

# Guardado del modelo TensorFlow Lite en un archivo .tflite
with open('model1.tflite', 'wb') as f:
    f.write(tflite_model)

# Guardado del encoder y el scaler para uso futuro
joblib.dump(label_encoder, 'label_encoder.joblib')
joblib.dump(scaler, 'scaler.joblib')

print("Modelo, encoder y scaler guardados exitosamente.")
```

Anexo 5. Cuantización del modelo .keras

```
from typing import Counter
import joblib
import tensorflow as tf
import numpy as np

# Carga del modelo Keras
model = tf.keras.models.load_model('Mi_modelo.keras',compile=False)

# Configuración del conversor con opciones de cuantización
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.inference_input_type = tf.int8
converter.inference_output_type = tf.int8

# Generación de datos representativos para la calibración del modelo
def representative_dataset_gen():

    for _ in range(100):
        yield [np.random.randn(1, 11).astype(np.float32)]

converter.representative_dataset = representative_dataset_gen

# Conversión del modelo
tflite_model = converter.convert()

# Guardado del modelo cuantizado
with open('ModelFlutter.tflite', 'wb') as f:
    f.write(tflite_model)
```

Anexo 6. Código que genera los parámetros necesarios para normalizar los valores que llegan a la app.

```
import joblib
scaler = joblib.load('scaler.joblib')
means = scaler.mean_.tolist()
scales = scaler.scale_.tolist()

print("Means:", means)
print("Scales:", scales)
label_encoder = joblib.load('label_encoder.joblib')
classes = label_encoder.classes_.tolist()
print("Classes:", classes)
```

Anexo 7. Código para demostrar mediante validación cruzada la precisión del modelo entrenado.

```
import os
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report
import joblib

# Carga del archivo CSV
data = pd.read_csv('Mi_modelo.keras.csv')

# Separación de características (X) serian los valores de los sensores y
etiquetas (y) que serian las letras a las que correspondan
X = data[['Flex1', 'Flex2', 'Flex3', 'Flex4', 'Flex5', 'Ax', 'Ay', 'Az', 'Gx',
'Gy', 'Gz']]
y = data['Letra']

# Codificación de las etiquetas (letras) como valores numéricos
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Normalizacion de las características o valores de entrada
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Configuracion de la validación cruzada
n_splits = 5
kfold = KFold(n_splits=n_splits, shuffle=True, random_state=42)

# Listas para almacenar los resultados
cv_scores = []
histories = []

for fold, (train_index, val_index) in enumerate(kfold.split(X_scaled)):
    print(f'Fold {fold + 1}/{n_splits}')

    X_train, X_val = X_scaled[train_index], X_scaled[val_index]
    y_train, y_val = y_encoded[train_index], y_encoded[val_index]

    # Definición de la red neuronal
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(128, activation='relu',
input_shape=(X_train.shape[1],)),
        tf.keras.layers.Dropout(0.3),
```

```
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(len(label_encoder.classes_),
activation='softmax')
    ])

    # Compilcion  del modelo
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

    # Callbacks
    early_stopping = tf.keras.callbacks.EarlyStopping(patience=10,
restore_best_weights=True)
    reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(factor=0.2, patience=5,
min_lr=0.0001)

    # Entrenamiento del modelo
    history = model.fit(
        X_train, y_train,
        epochs=100,
        batch_size=32,
        validation_data=(X_val, y_val),
        callbacks=[early_stopping, reduce_lr]
    )

    # Evaluación del modelo
    scores = model.evaluate(X_val, y_val, verbose=0)
    cv_scores.append(scores[1])
    histories.append(history)

    print(f'Fold {fold + 1} - Accuracy: {scores[1]*100:.2f}%')

# Impresiones en la terminal de los resultados de la validación cruzada
print(f'Mean CV Accuracy: {np.mean(cv_scores)*100:.2f}% (+/-
{np.std(cv_scores)*100:.2f}%)')

# Entrenamiento del modelo final con todos los datos
final_model = tf.keras.Sequential([
        tf.keras.layers.Dense(128, activation='relu',
input_shape=(X_train.shape[1],)),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(len(label_encoder.classes_),
activation='softmax')
])
```

```
final_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

final_history = final_model.fit(
    X_scaled, y_encoded,
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping, reduce_lr]
)

# Formato en que se guarda el modelo final
final_model.save('Modelo_final_1.keras')

# Conversion del modelo a TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_keras_model(final_model)
tflite_model = converter.convert()

# Guardar el modelo TensorFlow Lite en un archivo .tflite
with open('Modelo_final_1.tflite', 'wb') as f:
    f.write(tflite_model)

# Guardamos el encoder y el scaler para uso futuro
joblib.dump(label_encoder, 'label_encoder_1.joblib')
joblib.dump(scaler, 'scaler_1.joblib')

print("Modelo final, encoder y scaler guardados exitosamente.")

# Verificar la precisión del modelo TFLite
interpreter = tf.lite.Interpreter(model_content=tflite_model)
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Hacer predicciones con el modelo TFLite de cada clase
X_scaled = X_scaled.astype(np.float32)

y_pred_tflite = []
for sample in X_scaled:
    interpreter.set_tensor(input_details[0]['index'], [sample])
    interpreter.invoke()
    output = interpreter.get_tensor(output_details[0]['index'])
    y_pred_tflite.append(np.argmax(output[0]))

# Comparar con las predicciones del modelo original
y_pred_original = np.argmax(final_model.predict(X_scaled), axis=1)
```

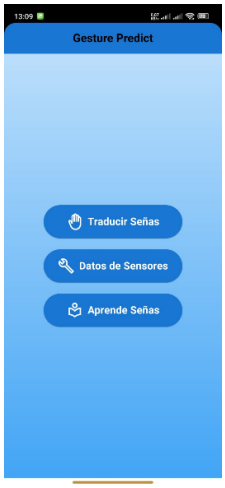
```
accuracy_tflite = np.mean(y_pred_tflite == y_encoded)
accuracy_original = np.mean(y_pred_original == y_encoded)

print(f"Precisión del modelo original: {accuracy_original*100:.2f}%")
print(f"Precisión del modelo TFLite: {accuracy_tflite*100:.2f}%")

# Impresión del reporte de clasificación
print("\nReporte de clasificación del modelo TFLite:")
print(classification_report(y_encoded, y_pred_tflite,
target_names=label_encoder.classes_))
```

Anexo 8. Manual de Usuario

Manual de Usuario del Guante Traductor de Señas



1. Preparativos Iniciales

Antes de comenzar, asegúrate de tener lo siguiente:

- Un guante traductor de señas.
- Un smartphone compatible con la aplicación.
- La aplicación descargada e instalada.
- Acceso a una red Wi-Fi o datos móviles en el smartphone.

2. Encendido del Guante

- Presionar el botón una vez para encender el guante.
- Esperar a que el indicador LED se ilumine, lo que indica que el guante está listo para funcionar.

3. Configuración del Smartphone

- Vaya a la configuración de su smartphone.
- Buscar la opción de Compartir Internet o Hotspot Móvil.
- Activa esta opción.
- Asegúrate de que el smartphone esté configurado con las siguientes credenciales:
  - SSID: JosuNet
  - Clave: 152000Josu
- Verificar la conexión con el led del guante, dejara de parpadear y pasara a encenderse constantemente la luz cuando se conecte el guante al teléfono.
- Verificar en el teléfono la conexión con el guante, ya que saldrá una notificación de dispositivo conectado.

4. Conexión del Guante a Firebase

- El guante se conectará automáticamente a la red compartida por el smartphone una vez que esté encendido y el smartphone esté configurado para compartir Internet. El guante se conectará a Firebase automáticamente para enviar y recibir datos relacionados con los gestos.

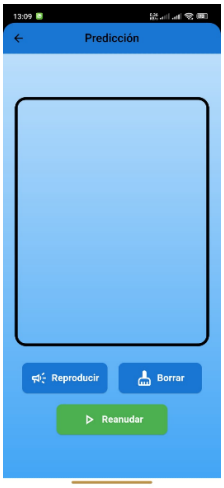
5. Instalación y Configuración de la Aplicación

En la pantalla principal de la aplicación, existen tres botones, los cuales dirigen a secciones diferentes:

Traducir señas

Datos de Sensores

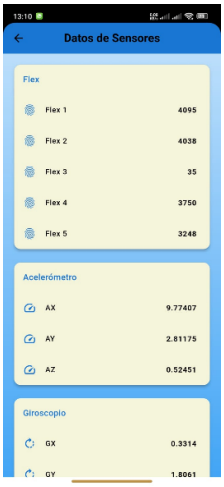
Aprende Señas



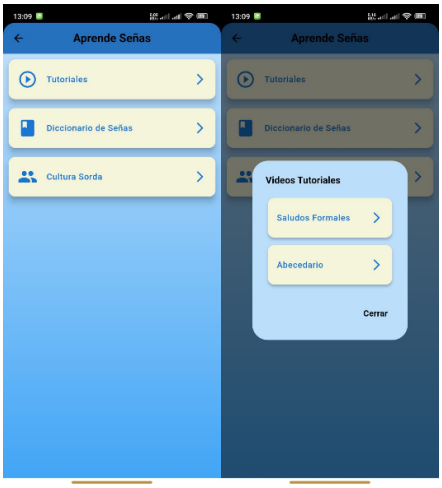
Para iniciar la captura de gestos hay q pulsar el botón *Traducir Señas*. A continuación, conducirá a una nueva pantalla donde se verán los gestos interpretados, existe tres botones en esta nueva sección.

- **Reproducir**: Reproduce mediante el altavoz del teléfono la letra o la palabra que se haya interpretado y se esté mostrando en el recuadro.
- **Borrar**: Borra todo el contenido del recuadro.
- **Reanudar**: Detiene y reanuda el intérprete.

En el botón de *Datos de Sensores* se muestran los valores que llegan directamente desde el guante. para iniciar la captura de gestos.



Al hacer click en el botón de *Aprende Señas* se conduce a la sección donde existen tres etiquetas, donde se hace click aparece un recuadro de dialogo donde hay botones que redirigen a diferentes urls en las cuales hay videos tutoriales de gestos del lenguaje de señas ecuatoriano, diccionario del LSEC fomentado por la Vicepresidencia de la Republica del Ecuador, además de la última etiqueta que redirige a la página oficial de la organización Cultura Sorda.



6. **Finalización del Uso**
    - Cuando se haya terminado de usar el guante, regresa a la aplicación y presiona el botón para detener la captura de gestos.
    - Apagar el guante presionando dos veces el botón y así conservar la batería.
7. **Solución de Problemas**
    - **Problemas de Conexión:** Si el guante no se conecta, verifica que el smartphone esté compartiendo Internet y que ambos dispositivos estén en la misma red.
    - **Fallo en la Aplicación**: Si la aplicación no responde, intenta reiniciarla o reiniciar el smartphone.

Anexo 9. Certificación de la traducción del resumen.

Loja, 10 de octubre del 2024

Yo, **Tania Carlina Betancourt Ochoa,** con cédula de ciudadanía **Nº1105945925 Licenciada en Ciencias de la Educación mención Idioma Inglés** con el registro **Senescyt 1008-2021-2371359**, certifico:

Que, la traducción del documento adjunto realizado por el Sr. **Josué Elián Betancourt Ochoa** con cédula de ciudadanía **Nº1150338885**, cuyo tema de investigación se titula: **Diseño e implementación de un guante traductor de señas para personas con discapacidad auditiva y/o verbal**, ha sido revisada y aprobada por mi persona.

El apartado del *Abstract* es una traducción textual del *Resumen* aprobado en español.

Particular que comunico en honor a la verdad para los fines académicos pertinentes, facultando al portador del presente documento hacer el uso que estime conveniente.

Lcda. Tania Carolina Betancourt Ochoa
LICENCIADA EN CIENCIAS DE
LA EDUCACIÓN MENCIÓN IDIOMA INGLÉS