Instituto Politécnico Nacional

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Trabajo Terminal II

"Dispositivo traductor de LSM a español escrito para la extremidad superior derecha"

Que para obtener el título de

"Ingeniero en Mecatrónica"

Presenta:

María del Rocío Casas Sánchez

Asesores:

Dra. Obdulia Pichardo Lagunas

Ing. Carlos Ríos Ramírez

upiita·ipn

Ciudad de México., a 11 de julio de 2022

Instituto Politécnico Nacional

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Trabajo Terminal II

"Dispositivo traductor de LSM a español escrito para la extremidad superior derecha"

Que para obtener el título de

"Ingeniero en Mecatrónica"

Presenta:

María del Rocío Casas Sánchez

Asesores:

Dra. Obdulia Pichardo Lagunas

Ing. Carlos Ríos Ramírez

Presidente del Jurado

M. en C. Bella C. Martínez Seis

Profesor titular

M. en C. Griselda Sánchez Otero

upiita·ipn

Autorización de uso de obra

Instituto Politécnico Nacional

P r e s e n t e

Bajo protesta de decir verdad el que suscribe María del Rocío Casas Sánchez, manifiesto ser autor (a) y titular de los derechos morales y patrimoniales de la obra titulada Dispositivo traductor de LSM a español escrito para la extremidad superior derecha, en adelante "La Tesis" y de la cual se adjunta copia, por lo que por medio del presente y con fundamento en el artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo a el Instituto Politécnico Nacional, en adelante El IPN, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales, Plataforma de la Dirección de Bibliotecas del IPN y/o consulta directa en la Coordinación de Biblioteca de la UPIITA "La Tesis" por un periodo de 5 años contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a "El IPN" de su terminación.

En virtud de lo anterior, "El IPN" deberá reconocer en todo momento mi calidad de autor de "La Tesis". Adicionalmente, y en mi calidad de autor y titular de los derechos morales y patrimoniales de "La Tesis", manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito respecto de "La Tesis", por lo que deslindo de toda responsabilidad a El IPN en caso de que el contenido de "La Tesis" o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México, a 11 de Julio de 2022

Atentamente

DEDICATORIA

A mis padres, gracias por todo su apoyo.

Rocío.

AGRADECIMIENTOS

A mi familia, por ser ese apoyo incondicional.

A Cinthya Rivera, por nunca dudar de mi capacidad para lograr las cosas y ser esa persona motivadora en mi vida.

Al IPN, maestros y asesores que me brindaron las oportunidades para poder concluir este proyecto.

A mi tutor, Agustín, por brindar siempre su apoyo y orientación académica.

A mis amigos de UPIITA, por aportar siempre un consejo, conocimiento y un bonito recuerdo a lo largo de estos años.

A todas las personas que hicieron posible que concluyera esta etapa de mi vida.

Rocío.

Contenido

Resumen

El presente trabajo muestra el diseño e implementación de un prototipo de dispositivo para la extremidad superior derecha, que proporciona la traducción del alfabeto dactilológico, así como una lista de señas básicas de la lengua de señas mexicana, usando una red neuronal artificial para su correcta clasificación.
El dispositivo cuenta con un conjunto de sensores que de forma inalámbrica permiten la identificación de la posición y movimiento de la mano y brazo. El dispositivo utiliza una interfaz gráfica de usuario donde se puede visualizar en español escrito la seña realizada una vez que es identificada.

Palabras Clave: Lengua de señas, sordo, traducción, red neuronal, extremidad superior.

Abstract

The present work describes the design and implementation of a prototype device for the upper right limb, which provides the translation of the fingerprint alphabet, as well as a list of basic signs of the Mexican Sign Language, using an artificial neural network for its correct classification.

The device has a set of sensors that wirelessly allow the identification of the position and movement of the hand and arm. The prototype uses a graphical user interface where the signal made once it has been identified can be viewed in written Spanish.

Keywords: Sign Language, deaf, translator, artificial neural network, upper limb.

Índice de figuras

Índice de Tablas

Nomenclatura

ABS Acrilonitrilo Butadieno Estireno

ADC Analog-to-Digital Converter (Convertidor analógico digital)

AHP Analytic Hierarchy Process (Proceso de análisis jerárquico)

API Application Programming Interface (Interfaz de programación de aplicaciones)

ASL American sign language (Lengua de señas americana)

BLE Bluetooth Low Energy (Bluetooth de baja energía)

FBS Functional Breakdown Structure (Estructura de desglose funcional)

I^2C Inter integrated circuits (Circuitos integrados)

IDE Integrated Development Environment (Entorno de desarrollo integrado)

IMU Inertial Measurement Unit (Unidades de medición inercial)

Li-Po Polímero de litio

LIS Lingua italiana dei segni (Lengua de señas italiana)

LSM Lengua de señas mexicana

MEMS Microelectromechanical Systems (Sistemas microelectromecánicos)

MUX Multiplexor

Ni-Mh Níquel-metal hidruro

PDS Product Design Specifications (Especificaciones de diseño del producto)

PLA Ácido poliláctico

PVC Policloruro de vinilo

RNA Red Neuronal Artificial

SDA Serial data (Datos en serie)

SLC Serial clock (Reloj serial)

WBS Work Breakdown Structure (Estructura de desglose de trabajo)

Simbología

a aceleración

E error

e error estimado

F fuerza

m masa

R resistencia

S sensibilidad

t tiempo

V velocidad

W peso de la red neuronal artificial

$\mathbf{Y}$ *input* de la red neuronal artificial

$\boldsymbol{\alpha}$ ángulo de inclinación

$\boldsymbol{\omega}$ velocidad angular

$\boldsymbol{\theta}$ ángulo de rotación

Introducción

Es la comunicación una característica primaria de los seres vivos; se hace necesaria la comunicación, dada la naturaleza del ser humano, así como para su desarrollo en las distintas etapas de la vida.

Para lograr la comunicación, es necesario contar con un mensaje, al menos un emisor y al menos un receptor; otros puntos a considerar son el código y el canal. Para los seres humanos, el código puede ser el idioma, y el canal más utilizado es la voz. He aquí un punto importante a considerar, la población que no utiliza la voz como canal y el canal que utiliza esta población.

En el año 2014, en los resultados de la encuesta del INEGI (Instituto Nacional de Estadística y Geografía), se presentó que había alrededor de 7.1 millones de personas con discapacidad en México, haciendo esto un 6 % de la población nacional. En la *Imagen 1* se muestra que de este porcentaje, el 33.5 % presentó discapacidad para escuchar, aún con aparato auditivo y 18 % con alguna discapacidad para hablar o comunicarse, éstas pueden ir ligadas una con otra, ya que una persona puede presentar más de una discapacidad [18].

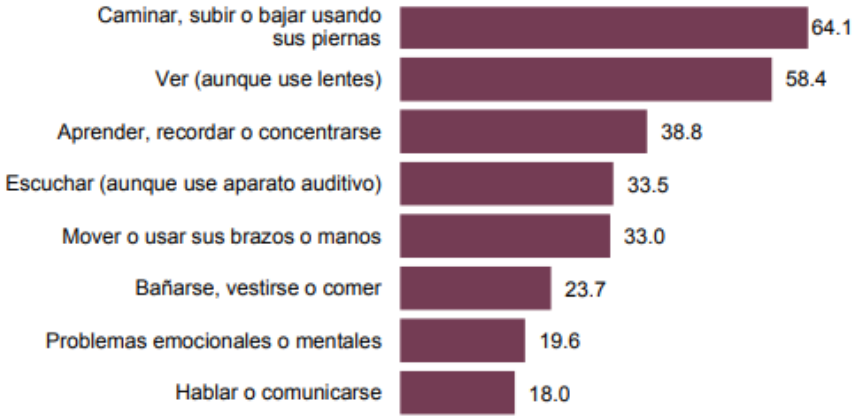


Imagen 1: Porcentaje de población con discapacidad, por tipo de discapacidad [18].

Dentro de la población de personas con sordera, existe una parte que no puede articular palabras o emitir algún sonido y sólo algunos pueden leer los labios. Fue por esto, que se creó un mecanismo de comunicación: la lengua de señas.

La lengua de señas mexicana (en adelante LSM, por su siglas) es una lengua que utiliza la comunidad sorda para comunicarse mediante señas manuales, corporales o gestuales. El camino que han llevado para el reconocimiento e inclusión de la lengua ha sido larga y difícil. No es hasta el 2005 que el Diario Oficial de la Federación, mediante la Ley General de las Personas con discapacidad en México, reconoce el uso de la lengua de señas mexicana [26]. También se han elaborado diccionarios para que la sociedad pueda acceder a ellos; el último diccionario publicado con la recopilación de varios diccionarios fue el Diccionario de la Lengua de Señas Mexicana de la Ciudad de México en el 2017.

Hoy en día la tecnología es un recurso a los que muchos seres humanos recurren para llevar a cabo tareas cotidianas o para solucionar problemas. Estos recursos también han sido usados para mejorar la vida de las personas que requieren la lengua de señas. Alrededor del mundo ya existen varios dispositivos, ya sean guantes de adquisición de datos o sistemas de procesamiento de imágenes para la traducción de la lengua de señas a la lengua de cada país. La mayoría se ha hecho para la lengua de señas americana (American Sign Language, en adelante ASL por sus siglas en inglés) a

inglés. Pero las señas varían incluso por idioma, cada país, cada región, cada ciudad, tiene sus propias señas o varían.

Desde el 2001, se ha incrementado el desarrollo de guantes de adquisición de datos en todo el mundo. Inicialmente se contaba con una estructura más pesada, menos precisa, sólo con la traducción del alfabeto dactilológico, con circuitos grandes y con materiales menos adecuados [46]. Actualmente los guantes son más ligeros, con una mayor eficiencia, con traducción de palabra por palabra, con circuitos compactos y materiales ergonómicos. La traducción por medio del guante es más fácil, ya que se puede acceder por un teléfono móvil; incluso es posible, mediante el uso del guante, controlar los sistemas de calentamiento, encendido y apagado de luces, televisión, entre otras cosas [19].

Enfoque mecatrónico

La importancia del diseño del sistema mecatrónico implica un diseño simultáneo de mecánica, electrónica, hardware, software y funciones de control integradas que permitan un resultado óptimo de todo el sistema. La sinergia de todos los elementos es lo que da resultado a un sistema mecatrónico con soluciones innovadoras. En la *Imagen 2* se muestra la representación de la sinergia de la mecatrónica con sus disciplinas [35].

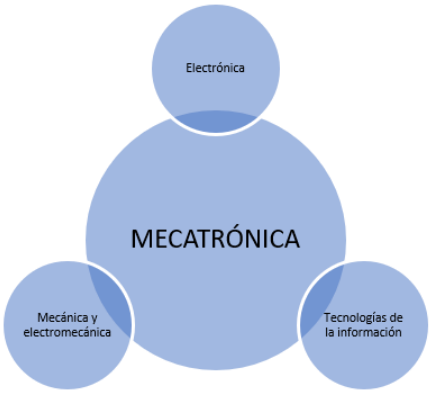


Imagen 2: Integración sinérgica de diferentes disciplinas en mecatrónica [35].

Para el desarrollo del dispositivo traductor, se identifican los retos a enfrentar por funciones:

- La captura de señas: Se busca obtener resultados precisos y un dispositivo ligero, usando distintos componentes:
  - Componentes piezoresistivos resistentes a la flexión de los movimientos del brazo.
  - Componentes mecánicos para la medición de la orientación y el cambio del movimiento del brazo y de precisión (giroscopios).
  - Sistemas micro-electromecánicos (Microelectromechanical Systems, en adelante MEMS por sus siglas en inglés).
- El acondicionamiento de señales: Se busca la conexión para la captura de señas y su clasificación. Se considera un microcontrolador por sus características: ligero, pequeño, delgado. Adicionalmente se busca una conexión inalámbrica y protocolos de comunicación para la correcta lectura de los componentes piezoresistivos, mecánicos y micro-electromecánicos.
- La clasificación de señales: Se busca mejorar los resultados con cada prueba de una forma automática e inteligente por lo cual se considera una arquitectura de microcontroladores, arquitectura de software, aprendizaje automático (*Deep Learning*), modelado y simulación.
- La comunicación con el usuario: Se busca la visualización de la seña clasificada correspondiente a la representada por el usuario, por lo que se considera una interfaz gráfica con un diseño de fácil interacción.

La integración se inicia por los componentes mecánicos y electromecánicos en la captura de señas, ya que, a través de estos se toman los ángulos y orientación que se desean medir. Los datos se transmiten mediante el acondicionamiento de señales para que el microcontrolador pueda procesar las señales a la computadora. El microcontrolador, permite la conexión con la red neuronal artificial (en adelante RNA, por sus

siglas) y el procesamiento de los datos. Una vez, procesados los datos por el microcontrolador, la RNA puede realizar su entrenamiento obteniendo los datos como sus entradas (*inputs*). Cuando la red finaliza su entrenamiento, el microcontrolador es el que trabaja con su modelo. Los datos de los componentes mecánicos se leen con el microcontrolador para que sinérgicamente trabajen los componentes de la mecatrónica juntos: electrónica, tecnologías de la información, mecánica y electromecánica.

Gracias a la intregración de dichos componentes mecatrónicos podemos tener dispositivos más complejos y efectivos, que se complementan entre sí.

Justificación

Hoy en día la tecnología nos da la oportunidad de cambiar la manera en que hacemos las cosas, de mejorar la comunicación entre nosotros. Comunicarnos con distintas personas de otros países se ha convertido en una forma más fácil con los traductores en el teléfono móvil, haciendo uso de alguna aplicación de idiomas. De la misma forma que lo hacemos para comunicarnos con otras lenguas, es necesario realizarlo con la LSM. En este documento se propone un dispositivo que ayude a las personas que desconocen la LSM para poder entenderla de manera básica.

La mayoría de las personas en México desconoce esta lengua y hay solamente 40 intérpretes especializados en el país [44], lo que refleja la gran escasez de las personas que pueden contar con un intérprete para comunicarse. Algo a considerar, para usar fluidamente la LSM, es necesario practicarlo durante años para volverse hábiles, requiere de mucha dedicación y esfuerzo, además de variar por regiones. Las personas que necesitan de los intérpretes requieren pasar juntos gran tiempo y suelen formar una fuerte dependencia. Además, es importante mencionar el gran esfuerzo económico que resulta de contar con un intérprete, mencionando ya los pocos que se encuentran en el país. Según testimonios de personas que utilizan la LSM, es muy común que se aíslen en su propio grupo para lograr sentirse incluidos [30].

El prototipo propuesto busca facilitar con palabras básicas, la comunicación de las personas con alguna discapacidad auditiva o del habla para que independientemente

de un intérprete puedan comunicarse.

A pesar de que actualmente existen dispositivos en el mercado a nivel mundial para la comunicación de la lengua de señas, los precios, la eficiencia y la certeza de la región donde se encuentra, influyen mucho para cada persona. La mayoría lo ha hecho para la ASL. Es por eso que los últimos dispositivos desarrollados internacionalmente tienen la opción de incluir más vocabulario propuesto por el usuario a la aplicación donde se esté trabajando. Casi todos los prototipos creados hasta el momento, han sido diseñados para una sola medida de persona (adultos generalmente) y con un vocabulario limitado de las distintas lenguas de señas del mundo.

En México, son pocos los proyectos relacionados con el desarrollo de guantes para la traducción de la LSM, considerando que la mayoría solo traduce letra por letra. Es por esto que se diseña un prototipo que no solo cuente con la traducción letra a letra, sino también con una lista de señas básicas, incluyendo en ésta los días de la semana y los meses de año. Esta lista debe adaptarse a la extremidad superior derecha con sus respectivos sensores en la mano y el antebrazo, lo anterior para poder tener una mayor certeza de las diferentes posiciones de la seña. También se busca que sea un prototipo ligero y no estorbe para realizar actividades como manipular objetos, lo que no cumplen la mayoría de los guantes hasta ahora diseñados.

Definición del problema

Las personas que requieren usar la LSM, enfrentan problemáticas para comunicarse en sus relaciones interpersonales, en su desarrollo como profesionistas y en sus tareas del día a día.

Las personas con alguna discapacidad auditiva o del habla que requieren aprender la LSM para comunicarse con su entorno, enfrentan una gran desventaja al percatarse que en México, hay un porcentaje muy bajo que conoce la LSM [28]. Este porcentaje, en su mayoría, son personas muy cercanas a la persona sorda.

El rezago en la educación se torna un problema cuando hay pocas o nulas escuelas

especiales para la LSM. Las universidades públicas de México no cuentan con personal de apoyo a las personas que se comunican con esta lengua, ya que hay muy pocas personas especializadas [20]. Lo que conlleva a que la mayoría no alcanza un nivel de educación mayor y por consiguiente no consigan un trabajo fijo.

Aunado a los puntos anteriormente mencionados, la comunidad sorda sufre de discriminación por parte de la sociedad y por sus propias familias; de acuerdo a testimonios de la comunidad sorda, en algunos casos, solo un miembro de su familia aprende la LSM. [20].

La problemática que conlleva la traducción de los días de la semana y los meses del año, es que son representadas con la misma posición de flexión de los dedos pero con distintos movimientos de rotación y traslación de la mano.

El problema que se busca resolver, es facilitar parte de la comunicación para las personas sordas, siendo éste el canal que podrían emplear para que las personas que ignoren la LSM comprendan algunas de las palabras básicas como lo son los días de la semana, los meses del año, así como la comunicación letra a letra, sin que las señas se confundan debido a similitud entre ellas.

Objetivos

Objetivo general

Diseñar y construir un dispositivo para el miembro superior derecho que obtenga mediante sensores, los datos de posicionamiento para su identificación mediante redes neuronales y transcripción mostrada en una interfaz gráfica de las señas manuales que caracterizan al alfabeto dactilológico, además de una lista de palabras básicas de la LSM.

Objetivos específicos

- Documentar una lista de señas manuales básicas de la LSM que tengan similitud entre sí de las variantes en toda la República Mexicana.

- Analizar e implementar los tipos de sensores que cumplan con los datos necesarios para la posición de la mano y brazo.
- Diseñar e implementar los sensores seleccionados en una estructura de guante.
- Diseñar e implementar la etapa de acondicionamiento de los sensores y la adquisición de las señales.
- Analizar y programar una red neuronal para la correcta identificación del alfabeto dactilológico y la lista de palabras documentadas.
- Diseñar y programar una interfaz gráfica, mostrando en la computadora la traducción al español escrito de México, mediante los resultados dados en la red neuronal entrenada.

Antecedentes

Sign Language Translator

En el año 2001, Randall R. Patterson, en Estados Unidos, desarrolló un guante de cuero que traduce la ASL con apoyo de 10 sensores colocados a lo largo de los dedos y en la muñeca. Este prototipo transmite las letras del abecedario de manera inalámbrica a una pantalla LCD, donde se van almacenando las letras para formar una palabra. Además, envía los datos al ordenador y muestra letra por letra [46]. En la *Imagen 3* se observa su prototipo.

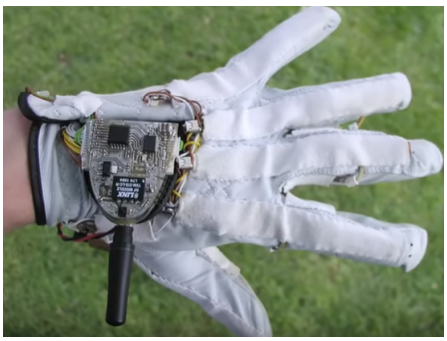


Imagen 3: Prototipo "Sign Language Translator" [16].

Method and Apparatus for Translating Hand Gestures

En el año 2003, José L. Hernández Rebollar, estudiante de doctorado de *The George Washington University*, en Estados Unidos, desarrolló un dispositivo que permite reconocer el alfabeto, algunas palabras y pocas frases de la ASL transmitiéndolos a una computadora donde se produce una voz sintetizada o se muestra el texto escrito. Para el desarrollo se utilizó un microcontrolador PIC, sensores en la mano (acelerómetros en los dedos y el pulgar, dos en la parte posterior de la mano, en la parte posterior de la muñeca), el brazo (un sensor de ángulo para detectar la flexión del codo, dos sensores en la parte superior del brazo para detectar la elevación y rotación) y el hombro [37]. En la *Imagen 4* se observa su esquema.

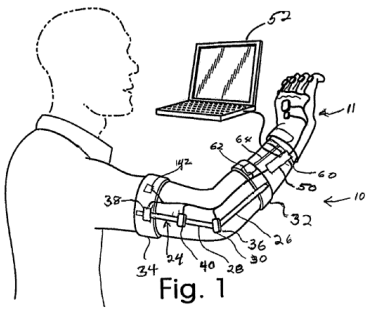


Imagen 4: Esquema "Method and apparatus for translating hand gestures" [37].

Enable Talk

En el año 2012 el equipo QuadSquad (Stepanov Anton, Valeriy Yasakov, Osika Maxim) de Ucrania ganó el Premio de Diseño de Software de Microsoft Imagine Cup al diseñar dos guantes capaces de transmitir la lengua de señas vía Bluetooth a una aplicación móvil. La aplicación tiene los datos almacenados y al coincidir reproduce el sonido de la seña utilizando la API de Microsoft Speech. Tarda aproximadamente dos segundos en hacer el proceso y tiene un 90 % de eficiencia. Los sensores utilizados son sensores de flexión, giroscopio y un acelerómetro, los cuales determinan la posición de los guantes en el espacio. Adicionalmente tiene un panel solar [6]. En la *Imagen 5* se observa el diagrama de posición de los sensores en los guantes.

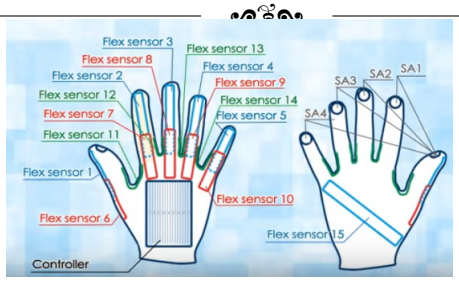


Imagen 5: Diagrama de los guantes "Enable Talk" [23].

Talking hands

En el año 2014, Francesco Pezzuoli, en Italia, diseñó un dispositivo tipo guante portátil que traduce la lengua de señas italiana (Lingua italiana dei segni, LIS por sus siglas en italiano) vía Bluetooth a una aplicación móvil que vocaliza el sonido desde la misma aplicación. El dispositivo es una impresión 3D que cuenta con un microcontrolador, sensores de flexión, acelerómetro, giroscopio y magnetómetro. Tras la victoria del prototipo en la presentación en el Startup Weekend of Ascoli Piceno 2014 se fundó LiMiX srl, una empresa dedicada actualmente a la mejora del guante y otras innovaciones. El dispositivo tipo guante promete ser más preciso al tener un vocabulario más amplio. Una de las innovaciones es que es personalizable, ya que el usuario puede agregar palabras al vocabulario desde la aplicación [19], [41]. En la *Imagen 6* se observa el dispositivo.

Imagen 6: Dispositivo "Talking Hands" [33].

Sistema para Traducción de Señas

En el año 2015, José Manuel Lira Ávalos, estudiante de la Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas (UPIIZ) del IPN, desarrolló un prototipo de guante para la traducción de la LSM letra por letra al español escrito en una pantalla LCD mientras lo reproduce en una bocina. El exoesqueleto fue diseñado en CAD/CAM e impreso en 3D. Cuenta con un modo aprendiz, dónde el usuario puede indicar la letra y guardar el promedio [55]. En la *Imagen 7* se observa el guante.

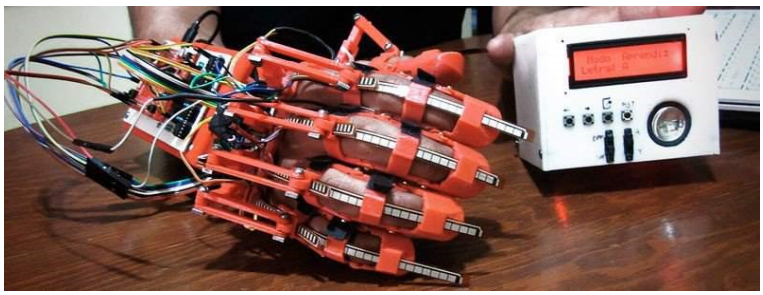


Imagen 7: Prototipo del "Sistema para traducción de señas" [31].

Tock Tock

En el año 2016, estudiantes del Tecnológico de Monterrey, desarrollaron un guante que traduce letra por letra la LSM a una aplicación móvil ("Tock Tock"). La aplicación cuenta con un modo de comunicación y con un modo de entrenamiento. En el primero se tiene las opciones de iniciar, regresar o salir y en caso de no identificar la letra te manda un mensaje de error [54]. En la *Imagen 8* se observa el guante.

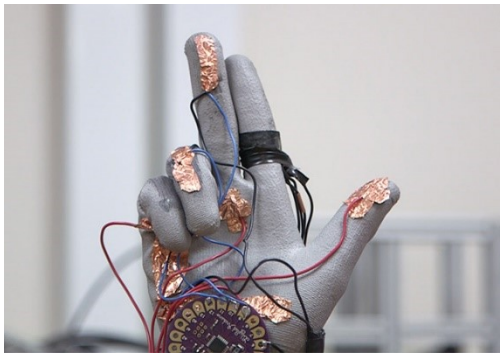


Imagen 8: Guante "Tock" [27].

Intelligent Talking Hand

En el año 2017, Bryan Howard Tarre Álvarez, estudiante de la Universidad Autónoma de Querétaro (UAQ), desarrolló un guante que identifica la letra o palabra de la LSM y la muestra en pantalla, así mismo se reproduce por voz en un dispositivo móvil. El guante cuenta con sensores de flexión y un microcontrolador; se conecta al ordenador, donde se utiliza una RNApara reconocer los patrones ya entrenados dentro de ésta [17]. En la *Imagen 9* se observa el guante.

Imagen 9: Guante "Intelligent Talking Hand" [17].

Prototipo de Guantes Traductores de la Lengua de Señas Mexicana para Personas con Discapacidad Auditiva y del Habla

En el año 2018, en el XLI Congreso Nacional de Ingeniería Biomédica, representando a la Universidad Autónoma de Chihuahua, se presentaron dos guantes traductores del alfabeto dactilológico y 18 palabras básicas de la LSM a texto y sonido. Cada guante cuenta con botones distribuidos, un sensor en cada dedo, acelerómetros en la muñeca y meñique. Para la comunicación se utiliza un módulo Bluetooth y para el procesamiento se utiliza un Arduino Nano. Como salida se tiene una pantalla LCD con bocina integradas en el antebrazo [47]. En la *Imagen 10* se observa su prototipo.

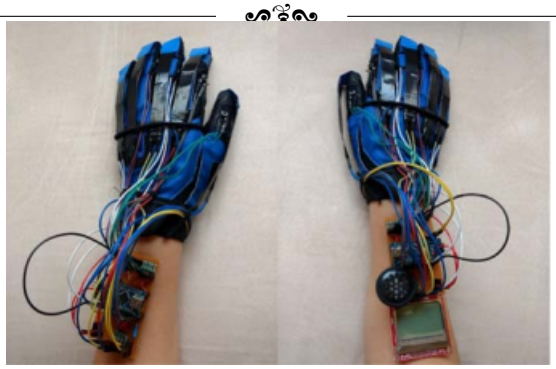


Imagen 10: Prototipo de "Guantes traductores de la Lengua de Señas"[47].

Prototipo de un sistema electrónico intérprete de la lengua de señas mexicana

En el año 2018, los alumnos Samuel Castro Solis y Samuel Santiago Sanjuan de la UPIITA, desarrollaron un prototipo electrónico para traducir la LSM, parcialmente al español (voz). Diseñaron un guante con sensores flexibles a lo largo de los dedos y unidades de medición inercial (Inertial Measurement Unit, en adelante IMU por sus siglas en inglés) de 6 ejes (combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes) en la punta de cada dedo y en la muñeca. Los datos de los sensores son transmitidos por Arduino para su identificación y posteriormente se envían a un dispositivo portátil [48]. En la *Imagen 11* se observa el prototipo.

Imagen 11: Prototipo de "Sistema electrónico intérprete de la LSM" [48].

Prototipo Traductor de Lenguaje de Señas Mexicano Mediante el Uso de Redes Neuronales

En el año 2018, el alumno José Ángel Avelar Barragán de la UPIITA, desarrolló un prototipo que traduce en pantalla y voz las letras del alfabeto de la LSM, además, algunas palabras usando redes neuronales. El prototipo cuenta con una estructura de PVC Sintra como soporte, hilo, tela elástica, velcro y cable plano. Los sensores con los que cuenta son: sensor flexible, acelerómetro y giroscopio [39]. En la *Imagen 12* se observa el prototipo.

Imagen 12: Prototipo "Traductor de lenguaje de señas mexicano mediante el uso de redes neuronales"[39].

Organización del documento

En el contenido de este documento se encuentran cuatro capítulos.
El capítulo 1 hace referencia al marco teórico y al procedimental. El marco teórico brinda bases de la lengua de señas mexicana, los guantes de datos, los sensores utilizados, las medidas ergonómicas para guantes y de la RNA. El marco procedimental brinda las bases de algunas metodologías utilizadas en el desarrollo del diseño del proyecto (el modelo en V, las especificaciones de diseño del producto, la estructura de desglose funcional y el proceso de análisis jerárquico).

El capítulo 2, se divide en el diseño conceptual y el diseño detallado del dispositivo. En el diseño conceptual se definen los requerimientos, la arquitectura funcional, las propuestas de solución, para así poder evaluar las combinaciones y seleccionar un diseño preliminar. En el diseño detallado se definen los 5 módulos del dispositivo, así como su validación de cada uno de ellos.

El capítulo 3 especifica la implementación del sistema por cada uno de los módulos, detallando las pruebas realizadas por componentes para la validación de cada uno de éstos. Al final de este capítulo se muestra la implementación de los módulos integrados para la validación del dispositivo.

El capítulo 4 brinda el análisis de resultados, dividido en tres secciones: ingeniería, costos y valor. En el análisis de ingeniería se realizan las observaciones de los resultados obtenidos y los errores presentados durante la implementación del dispositivo. En el análisis de costos se muestra la descomposición del costo total y en el análisis de valor se presenta el valor de uso del dispositivo.

Finalmente se presentan las conclusiones del diseño e implementación del dispositivo.

CAPÍTULO 1

Marco de referencia

1.1. Marco teórico

1.1.1. La lengua de señas mexicana

La lengua de señas es la lengua de la comunidad de sordos, está consiste en "una serie de signos gestuales articulados con las manos y acompañados de expresiones faciales, mirada intencional y movimiento corporal, dotados de una función lingüística", según el Diario Oficial de la Federación en el apartado de la Ley General de las Personas con Discapacidad [26].

Tipos de señas

Las señas se pueden clasificar en 4 tipos diferentes, según sea con el número de manos o de su movimiento simultáneo:

1. Seña manual (se articula con una mano).

2. Seña bimanual (se articula con dos manos).

3. Seña simétrica (se articula con dos manos a la vez a través de movimientos simétricos).

4. Seña compuesta (se articula con dos señas simples o tres configuraciones distintas).

Las señas icónicas son las señas que copian las características físicas del objeto. Un ejemplo es: Árbol.
En otros casos se señala alguna parte del cuerpo con las características del objeto. Un ejemplo es: Manzana, que se articula en la mejilla. Las señas arbitrarias son cuando no hay relación alguna entre el objeto y la seña. Un ejemplo es: Gracias [26].

Configuración de la palma de la mano

La orientación de la palma es la posición de la mano con respecto al cuerpo del señante al momento de hacer la configuración manual, en la *Imagen 1.1*, se observan las diferentes orientaciones, que se listan a continuación [26]:

1. Palma de la mano orientada hacia arriba y puntas de los dedos hacia la izquierda.

2. Palma de la mano orientada hacia arriba y puntas de los dedos hacia adelante.

3. Palma orientada hacia abajo y puntas de los dedos hacia la izquierda.

4. Palma orientada hacia abajo y puntas de los dedos hacia adelante.

5. Palma orientada hacia la izquierda y puntas de los dedos hacia arriba.

6. Palma orientada hacia la izquierda y puntas de los dedos adelante.

7. Palma orientada hacia el frente y puntas de los dedos hacia arriba.

8. Palma orientada frente al cuerpo con las puntas de los dedos hacia arriba.

9. Palma orientada frente al cuerpo con las puntas hacia la izquierda.

Imagen 1.1: Configuraciones de la palma de la mano [26].

Ubicación

La ubicación es el lugar sobre el espacio donde se ejecutan las señas [26]. Puede dar referencia a la ubicación del cuerpo mediante "altura":

- A la altura del cuello.
- A la altura del hombro.
- A la altura del pecho.
- A la altura del plexo.
- A la altura de la cintura.
- A la altura de la cadera.

Cuando incluye un desplazamiento en la ubicación:

- Del pecho a la cintura.
- Del hombro a la cadera.
- Del cuello a la cadera.

1.1.2. Guante de datos

Un guante de datos sirve para obtener información de los movimientos de la mano de la persona que lo porta, siendo este un modo de comunicación e interacción con diferentes dispositivos. El uso de estos guantes se da principalmente en la realidad virtual como videojuegos.

Actualmente existen diversos guantes de datos que brindan la facilidad de obtener los datos para el uso requerido. Un ejemplo de ellos es el guante AcceleGlove, el cual viene con un software que permite a los desarrolladores usar Java para programar cualquier aplicación que deseen. El objetivo inicial de este guante son funciones relacionadas con el control robótico, pero también puede ser usado para videojuegos, entrenamiento deportivo o rehabilitación física [34].

Los guantes de datos ocupan sensores específicos que permiten la lectura precisa del movimiento y flexión de los dedos, la orientación de la palma de la mano y la muñeca. Cada fabricante elige como desarrollarlo y que tipo de sensores utiliza. El precio de los guantes de datos es muy elevado debido a los sensores y materiales que se eligen. Para medir la flexión de los dedos se han utilizado tres tipos de tecnología de guante: sensores de fibra óptica, medidas mecánicas y galgas extensiométricas. Los guantes que emplean sensores de flexión son más cómodos comparados con los mecánicos pero estos últimos suelen ser más precisos.

Un ejemplo de guante que cuenta con sensores de fibra óptica es el "Dataglove"que se muestra en la *Imagen 1.2*, un guante de neopreno con dos lazos de fibra óptica en cada dedo. En el extremo del lazo se encuentra un fotosensor y en el otro el led, cuando el usuario dobla el dedo se escapa la luz del cable de fibra óptica a través de los pequeños cortes que tiene este cable. Con la cantidad de luz medida, se puede determinar la

flexión de cada nudillo. Este guante, presenta un problema con el tamaño de mano, ya que la posición de los nudillos siempre deben coincidir con las articulaciones.

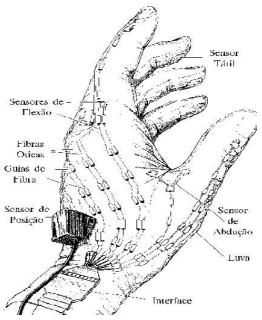


Imagen 1.2: Diagrama "Dataglove" [42].

Los guantes de datos que obtienen los datos por mediciones mecánicas suelen ser más pesados, robustos y por lo mismo, menos utilizados. Un ejemplo es el guante "Exos Dextrous Hand Master", éste es un exoesqueleto adaptable a la mano, el cual contiene cuatro sensores de posición por dedo de efecto Hall situados en las articulaciones de la estructura mecánica. En la *Imagen 1.3* se muestra el diagrama con las partes que componen a este guante.

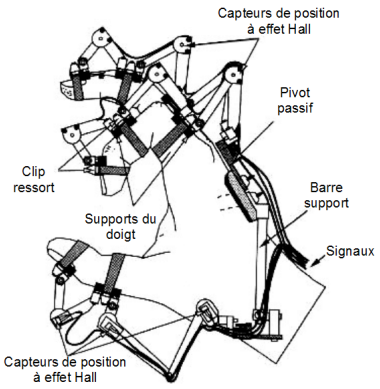


Imagen 1.3: Diagrama "Exos Dextrous Hand Master" [25].

El principio de la galga extensiométrica se da en varios casos de guantes de datos. Uno de ellos es el "Matel Power Glove", elaborado con tiras de poliéster cubiertas de una tinta especial que dependiendo de la flexión que se le de, varía la resistencia.

Otro ejemplo son los guantes con sensores flex, los cuales son sensores comerciales muy utilizados para este tipo de guantes, que consisten en tiras delgadas, que se ajustan a cada dedo, al variar su resistencia dependiendo de la flexión que el usuario le de. En la *Imagen 1.4* se muestra un ejemplo de guante con sensores flex.

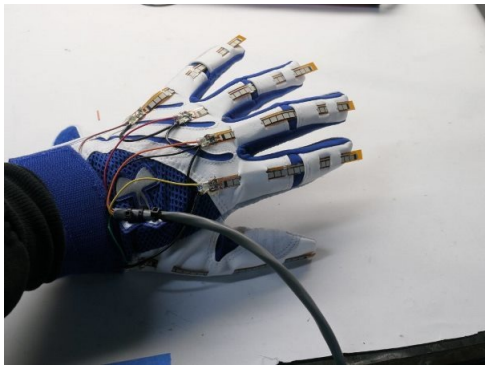


Imagen 1.4: Prototipo de "Guante de datos con sensores flex"[9].

Hay diferentes materiales con los cuales los guantes de datos son fabricados, siendo el más común el nylon, ya que, este material es transpirable y se estira para cualquier tamaño de mano [45].

1.1.3. Sensores

Los siguientes sensores son considerados hasta ahora las opciones mejor calificadas por los trabajos anteriores, es por ello que se hace una investigación más amplia de ellos.

Sensores flex

Los Sensores flex, son sensores que cambian la resistencia en función de la cantidad de curvatura que se emplea, al doblarse el sensor, la curvatura aumenta su resistencia eléctrica, es decir, mientras mayor sea la curva, mayor será el valor de la resistencia. Se emplean los guantes con sensores flex para detectar el movimiento del dedo, para controlar automóviles, aparatos de medición, tecnología de asistencia, instrumentos musicales, smartphones, entre otros [50]. En la *Imagen 1.5*, se muestra un sensor flex.

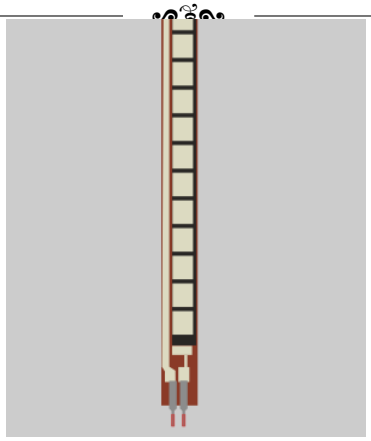


Imagen 1.5: Sensor flex [50].

Su objetivo inicial era ser implementados en las bolsas de aire para los carros, hoy en día se usan incluso en bocinas de los carros, juguetes que detectan diferentes grados de presión o de flexión, entre otros [50]. Los sensores flex son resistencias analógicas y están constituidos por elementos resistivos de carbono colocados en un arreglo lineal. Este arreglo se coloca sobre un estrato conductor que a su vez se coloca sobre un sustrato flexible y delgado, es decir, más carbono significa menos resistencia. Cuando se dobla el sustrato del sensor produce una salida de resistencia en relación con el radio de curvatura. Con un sensor típico flex, una flexión de 0° dará la resistencia de 10K será una flexión de 90° dará entre 30 a 40 K ohmios [38].

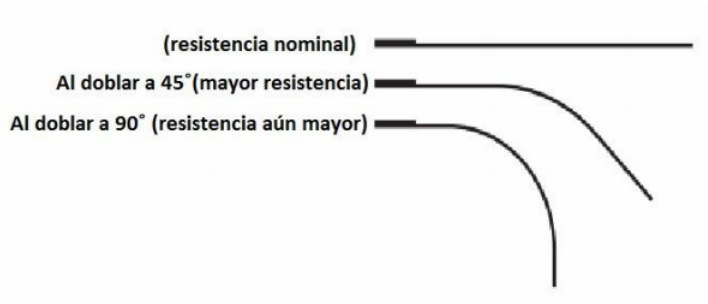


Imagen 1.6: Flexión sensor flex [38].

Acelerómetro y giroscopio MPU6050

Es una IMU de 6 grados de libertad pues combina un acelerómetro de 3 ejes y un

giroscopio de 3 ejes. Este sensor es muy utilizado en navegación, goniometría (ciencia y técnica de la medición de ángulos), estabilización, etc. La aceleración es la variación de la velocidad por unidad de tiempo es decir razón de cambio en la velocidad respecto al tiempo:

$$a = \frac{dV}{dt} \tag{1.1}$$

Así mismo la segunda ley de Newton indica que en un cuerpo con masa constante, la aceleración del cuerpo es proporcional a la fuerza que actúa sobre él mismo:

$$a = \frac{F}{m} \tag{1.2}$$

Este segundo concepto es utilizado por los acelerómetros para medir la aceleración. Los acelerómetros internamente tienen un MEMS que de forma similar a un sistema de masa resorte, permite medir la aceleración.

Con un acelerómetro podemos medir esta aceleración, teniendo en cuenta que a pesar que no exista movimiento, siempre el acelerómetro estará sensando la aceleración de la gravedad. Con el acelerómetro podemos hacer mediciones indirectas, por ejemplo, si integramos la aceleración en el tiempo, tenemos la velocidad y si la integramos nuevamente, tenemos el desplazamiento, necesitando en ambos casos la velocidad y la posición inicial respectivamente. La velocidad angular es la tasa de cambio del desplazamiento angular por unidad de tiempo, es decir que tan rápido gira un cuerpo alrededor de su eje:

$$\omega = \frac{d\theta}{dt} \tag{1.3}$$

Los giroscopios utilizan un MEMS para medir la velocidad angular usando el efecto Coriolis.

Con un giroscopio podemos medir la velocidad angular, y si se integra la velocidad angular con respecto al tiempo se obtiene el desplazamiento angular (posición angular si se sabe dónde se inició el giro) [53].

1.1.4. Medidas ergonómicas para guantes

Las medidas dispuestas para los guantes varían según el tipo de guante que se use, el material, el fabricante, etc.

Guía para determinar las tallas de guantes:

- El primer paso consiste en medir la circunferencia de la mano a la altura de los nudillos, excluyendo el pulgar. La mano debe estar abierta con los dedos juntos. La medición se puede hacer con una cinta métrica. La *Tabla 1.1* muestra la comparación de mediciones para determinar la talla [5].

5 in (12 cm) - 6 in (15 cm)	5/XXS
7 in (18 cm) - 7 1/2 in (19 cm)	6/XS
7 1/2 in (19 cm) - 8 in (20 cm)	7/S
8 in (20 cm) - 8 1/2 in (21.5 cm)	8/M
8 in (21.5 cm) - 9 in (23 cm)	9/L
9 in (23 cm) - 10 in (25 cm)	10/XL
10 in (25 cm) - 11 in (28 cm)	11/XXL
11 in (28 cm) - 12 in (30 cm)	12/3XL

Tabla 1.1: Tallas por mediciones de circunferencia [5].

- El segundo paso consiste en medir el ancho de la mano desde la base del primer dedo a través de los nudillos excluyendo el dedo pulgar, tal como se muestra en la *Imagen 1.7* [13].

Imagen 1.7: Talla de guantes por medida de ancho de mano [13].

La *Tabla 1.2* muestra la comparación de mediciones para determinar la talla.

5/2XS	50 mm/2 pulgadas
6/XS	63 mm/2.5 pulgadas
7/S	75 mm/3 pulgadas
8/M	88 mm/3.5 pulgadas
9/L	101 mm/4 pulgadas
10/XL	113 mm/4.5 pulgadas
11/2XL	126 mm/5 pulgadas
12/3XL	140 mm/5.5 pulgadas

Tabla 1.2: Tallas por mediciones de ancho de mano [13].

1.1.5. Red Neuronal Artificial

Las redes neuronales están, inspiradas en las redes neuronales biológicas, aunque poseen otras funcionalidades y estructuras de conexión distintas a las vistas desde la perspectiva biológica.
El elemento básico de computación (modelo de neurona) se llama habitualmente nodo o unidad. Recibe un *input* desde otras unidades o de una fuente externa de datos. Cada *input* tiene un peso asociado W, que se va modificando en el llamado proceso de aprendizaje. Cada unidad aplica una función dada f de la suma de los *inputs* ponderadas mediante los pesos, como se indica a continuación [7]:

$$Y_i = \sum_j W_{ij} Y_j \tag{1.4}$$

El resultado puede servir como *output* de otras unidades.
Se usan arquitecturas que comprenden elementos de procesado adaptativo paralelo, combinados con estructuras de interconexiones jerárquicas.
Hay dos fases en la modelización con redes neuronales [7]:

- Fase de entrenamiento: en esta fase se define el modelo de la RNA usando un conjunto de datos o patrones de entrenamiento para determinar los pesos (parámetros).

- Fase de operación: al finalizar la fase de entrenamiento, la red puede ser utilizada para realizar la tarea para la que fue entrenada. Una de las principales ventajas que posee este modelo es que la red aprende la relación existente entre los datos, adquiriendo la capacidad de generalizar conceptos. De esta manera, una RNA puede tratar con información que no le fue presentada durante la fase de entrenamiento [15].

1.1.6. Keras y TensorFlow

TensorFlow es una plataforma informática para la compilación de modelos de aprendizaje automático. Ofrece diversos kits de herramientas que permiten crear modelos con distintos niveles de abstracción. Se puede usar una interfaz de programación de aplicaciones (en adelante API, por sus siglas en inglés) de nivel inferior para compilar modelos definiendo una serie de operaciones matemáticas ó bien, se puede usar una API de nivel superior para especificar arquitecturas predefinidas, como regresores lineales o redes neuronales [12].

Keras es la API de alto nivel de TensorFlow que permite definir y entrenar modelos de redes neuronales en menos líneas de código. Se utiliza para la creación rápida de prototipos con tres ventajas clave [8]:

- Amigable al usuario: tiene una interfaz simple y optimizada para casos de ejemplos que se usan comúnmente, además que proporciona información clara y procesable sobre los errores del usuario.
- Modular y configurable: los modelos se fabrican conectando bloques de construcción configurables entre sí, con pocas restricciones.
- Fácil de extender: permite escribir bloques de construcción personalizados, así como crear nuevas capas, métricas, funciones de perdida y desarrollar modelos.

Los pasos para la creación de modelos de redes neuronales en Keras son [43]:

1. Cargar datos de la base de datos

2. Definir el modelo.

3. Compilar el modelo.

4. Entrenar el modelo.

5. Evaluar el modelo.

1.2. Marco procedimental

1.2.1. Sistema Mecatrónico

Un sistema mecatrónico es un sistema integrado por los componentes (hardware) y las funciones basadas en tecnologías de información (software).

1.2.2. Modelo en V

El modelo en V describe los principales pasos para definir el desarrollo de un sistema. En este modelo se relacionan el análisis y diseño con las pruebas del sistema, ya que se describen las etapas del proyecto, así como la implementación y verificación de estas mismas.
La representación gráfica es una V, como se muestra en la *Imagen 1.8*, en la que se forman cuatro niveles de cada fase de desarrollo del proyecto y donde existe para cada fase, otra correspondiente de verificación. La distancia entre las fases de desarrollo y su correspondiente, están separadas con respecto al tiempo, es decir, entre más arriba de la V se encuentren, más tiempo requiere para la verificación. La fase de desarrollo (parte izquierda) surge de las necesidades del cliente, para definir los requerimientos del proyecto. Una vez teniendo los requerimientos del proyecto, se puede realizar el análisis de diseño tanto conceptual como detallado. La fase correspondiente a cada fase de desarrollo (parte derecha) es para la verificación de cada una de las fases, hasta llegar de nuevo a la verificación de las necesidades del cliente. Las dos fases se unen en la implementación del proyecto, es por eso que se pueden hacer las pruebas en orden [36].

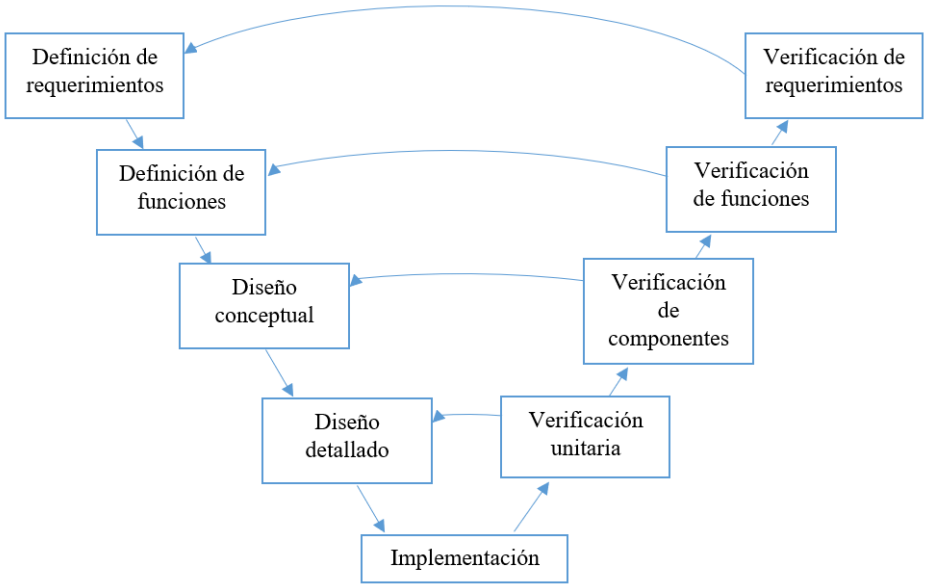


Imagen 1.8: Modelo en V [2].

1.2.3. Especificaciones de diseño del producto (PDS)

Como se muestra en la *Imagen 1.8*, el primer paso para comenzar a diseñar es definir los requerimientos del dispositivo. Las especificaciones de diseño del producto (Product Design Specifications, en adelante PDS por sus siglas en inglés) describen a detalle los requerimientos y necesidades con los que se planea diseñar y/o producir. El número de especificaciones depende de cada producto y su novedad en el mercado, ya que no se puede cuantificar los requisitos de un prototipo aún inexistente. Al final del diseño las especificaciones se tornan en las características del producto. Algunas de las consideraciones que se tienen son [4]:

- Entorno de funcionamiento.
- Seguridad.
- Funcionamiento.
- Materiales.
- Ergonomía.

- Legalidad y Normalización.
- Instalación y mantenimiento.
- Vida útil.
- Peso y tamaño.
- Transporte.
- Fiabilidad.
- Calidad.
- Periodo previsto de lanzamiento al mercado.
- Costo.
- Embalaje.

1.2.4. Estructura de desglose funcional (FBS)

La estructura de desglose funcional (Functional Breakdown Structure, en adelante FBS por sus siglas en inglés) es un desglose estructurado por módulos de una actividad principal. Es diferente a una estructura de desglose de trabajo (Work Breakdown Structure, WBS por sus siglas en inglés), la FBS es una estructura orientada a funciones y no al producto.
La FBS aborda las problemáticas de la ingeniería desde el punto de vista funcional, donde se puede evaluar fácilmente la integridad de todo un diseño mecatrónico. En la estructura es fácil encontrar si es que existen funciones faltantes para el diseño [29].

1.2.5. Proceso de Análisis Jerárquico (AHP)

El Proceso de Análisis Jerárquico (Analytic Hierarchy Process, en adelante AHP por sus siglas en inglés) es un método de elección de diseño que se basa en la evaluación de los diferentes criterios que permiten jerarquizar un proceso y su objetivo final consiste en optimizar la toma de decisiones. Se utiliza para resolver problemas en

los cuales existe la necesidad de priorizar distintas opciones y posteriormente decidir cual es la opción más conveniente. Para este método se pueden evaluar problemas tanto cualitativos como cuantitativos.

AHP ayuda a organizar por la toma de decisiones multi-criterio los aspectos críticos de un problema en una estructura jerárquica, similar a la estructura de un árbol familiar, reduciendo las decisiones complejas a una serie de comparaciones que permiten la jerarquización de los diferentes aspectos o criterios evaluados [21].

En la *Tabla 1.3*, se muestra la comparación de la escala numérica con la escala verbal.

Escala verbal	**Escala numérica**	**Explicación**
1	Igual de importancia	Los dos elementos contribuyen igualmente al criterio
3	Moderadamente un elemento más importante que otro	El juicio y la experiencia previa favorecen a un elemento frente a otro
5	Fuertemente un elemento más importante que en otro	El juicio y la experiencia previa favorecen fuertemente a un elemento frente a otro
7	Mucho más fuerte un elemento más importante que en otro	Un elemento domina fuertemente. Su dominación está probada en práctica
9	Importancia extrema de un elemento frente a otro	Un elemento domina al otro con el mayor orden de magnitud posible

Tabla 1.3: AHP escalas [24].

CAPÍTULO 2

Diseño del dispositivo

2.1. Diseño Preliminar

El diseño conceptual es un conjunto de tareas encaminadas a obtener una solución de una problemática planteada. Su proceso consiste en dar una solución a partir de los requerimientos y funciones. Como resultado, sintetiza la solución en forma de conceptos, obteniendo un diseño preliminar.
El proceso del diseño conceptual es precedido de una investigación que justifique la solución final. Su objetivo es generar y evaluar una serie de soluciones alternativas con objeto de identificar la más adecuada. Para una mejor visualización del proceso, en la *Imagen 2.1* se muestran los pasos a seguir [3].

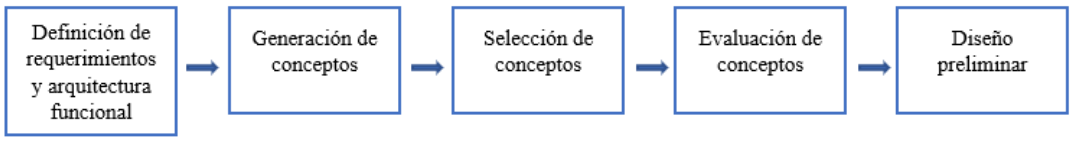


Imagen 2.1: Proceso genérico de diseño conceptual.

2.1.1. Necesidades - Requerimientos

Para definir los requerimientos, se comienza con las PDS y se hace un análisis de algunas consideraciones en el dispositivo.

Entorno de funcionamiento

El dispositivo debe ser portado por el usuario que representará las señas manuales de la LSM, colocando y ajustando este mismo en su brazo derecho y mano derecha. No se debe tener cubierta la mano ni el antebrazo del usuario. Puede ser usado siempre y cuando se cuente con una computadora portátil que permita visualizar la interfaz de resultados. La distancia máxima de alejamiento del usuario a la computadora es de 5 metros, al ser este un dispositivo inalámbrico. La alimentación del dispositivo debe ser colocada en el brazo derecho del usuario. Para la energía de la computadora portátil, se debe tener al alcance un contacto con tierra para recargar si es necesario. La temperatura en la que se encuentre el dispositivo debe estar en un rango de 0°C a 35°C para su correcto funcionamiento. El dispositivo no debe estar expuesto al agua en ningún momento, tampoco estar expuesto al sol por un tiempo prolongado o en ambientes húmedos. Para el correcto funcionamiento, la fuente de alimentación debe ser cargada cuando el indicador lo requiera.

Seguridad

El usuario debe ser responsable de no tener ningún contacto con el agua o algún líquido, ya que quedan expuestos varios componentes electrónicos como sensores y cables. La temperatura en el dispositivo puede aumentar en la alimentación, por lo que la alimentación no debe estar en contacto directo con la piel. El dispositivo no se puede dejar expuesto al sol, cerca de fuentes de calor o ambientes inflamables. No debe ser golpeado ni tirado. No se debe intentar abrir la fuente de alimentación ni dejar al alcance de niños menores. No existen riesgos para las personas que se encuentran cerca del usuario (zona potencial).

Materiales

El material del dispositivo debe ser ligero, para evitar el agotamiento al usuario al manejar el dispositivo. Debe ser rígido y termo deformable para su correcta implementación en la mano derecha. Debe ser material de fácil acceso en el mercado nacional y de precio accesible. El material para el ajuste del dispositivo se debe contar con tiras de velcro.

Ergonomía

La posición de la mano varía dependiendo de la seña indicada por el usuario, por lo que la parte del guante del dispositivo, debe tener un libre movimiento para los límites de ángulos máximos y mínimos de los dedos y muñeca. Para la parte de antebrazo y brazo también se debe permitir un libre movimiento que se encuentre dentro de los límites de rotación y traslación de la mano. El dispositivo debe ser ajustable a una medida de guante XS solamente. Los movimientos en la mano deben variar rápidamente, por lo que el dispositivo no debe alterar el movimiento del usuario ni el área restringida de operación. La altura a la que debe de encontrarse el dispositivo respecto al usuario es como viene representada en la *Imagen 1.1*, ya que son las diferentes alturas de las posiciones de la palma de la mano en la LSM.

Peso y tamaño

El peso máximo para el dispositivo debe ser de 1 kilogramo, ya que al poner más peso puede llegar a cansar al usuario y afectar en el movimiento de las señas. El usuario va a estar expuesto a una carga prolongada al hacer uso del dispositivo en la extremidad superior derecha, por lo que este mismo debe ser considerado el menor peso posible. La talla del dispositivo debe ser XS.

En la *Tabla 2.1* se muestra el despliegue de las necesidades y requerimientos del dispositivo.

Necesidades	Requerimientos
Estructura para la mano que permita el libre movimiento en cada articulación y espacio en los sensores.	Talla de la mano XS. ▪ Ángulo máx. de rotación en X para la mano: 125°. ▪ Ángulo máx. de rotación en Y para la mano: 35°. ▪ Ángulo mín. de rotación en X para la mano: -80°. ▪ Ángulo mín. de rotación en Y para la mano: -130°. ▪ Ángulo máx. de rotación en X para el antebrazo: 85°. ▪ Ángulo máx. de rotación en Y para el antebrazo: 85°. ▪ Ángulo mín. de rotación en X para el antebrazo: -10°. ▪ Ángulo mín. de rotación en Y para el antebrazo: -35°.
Estructura que sea ligera para ser portátil.	Plástico ligero y rígido.
Comodidad en la estructura siendo inalámbrica para su uso.	Pila recargable o batería como suministro de energía adaptada al dispositivo.

(pasa a la página siguiente)

Necesidades	Requerimientos
Dispositivo capaz de traducir una lista de señas manuales básicas de la LSM, incluyendo el alfabeto dactilológico.	Lista de palabras: ▪ 27 letras del alfabeto. ▪ Números del 1 al 10. ▪ 12 meses. ▪ 7 días de la semana. ▪ Hola. ▪ Bien. ▪ Mal. ▪ Perdón. ▪ No. ▪ Si.
Una cantidad de sensores suficientes para la correcta lectura y datos suficientes para la lista de palabras de la LSM.	Contar con: ▪ Sensores en cada uno de los dedos. ▪ Sensor de orientación en el dorso de la mano. ▪ Sensor para medir el ángulo formado por la muñeca. ▪ Sensor para medir el ángulo del codo ▪ Sensor de orientación en antebrazo.
Clasificador de aprendizaje automático.	Clasificador usando una RNA.
Agilización en la ejecución del programa con pocas líneas de código.	Librerías de redes neuronales que cuenten con algoritmos ya desarrollados.
Interfaz donde se pueda interactuar con funciones simples.	Interfaz mostrando opciones para iniciar, pausar y borrar contenido.

Tabla 2.1: Necesidades y requerimientos.

2.1.2. Arquitectura funcional

Para definir las funciones del dispositivo se aplica la FBS del dispositivo, que se muestra en la *Imagen 2.2*, donde se identifican 5 funciones principales.

Captar señas

Esta función es la que proporciona los valores caracterizados por el movimiento y posición de la mano y antebrazo mediante el uso de sensores. Para identificar claramente la lista de señas básicas manuales a interpretar, se deben de tener los siguientes datos:

- Ángulo de cada uno de los dedos de la mano.
- Ángulo formado en la muñeca.
- Ángulo formado en el codo
- Orientación del antebrazo.
- Orientación del dorso de la mano.
- Verificar si existe contacto entre los dedos índice y dedo medio.

Acondicionar señales

La función de acondicionamiento de señales depende del sensor con el que se esté trabajando. Las etapas son las siguientes [22]:

- Ajustar la tensión o voltaje: sirve para adaptar al rango de convertidor analógico digital (analog-to-digital converter, en adelante ADC por sus siglas en inglés) de la tarjeta de procesamiento.
- Linealizar: la linealización es necesaria cuando los sensores producen señales de tensión que no están linealmente relacionados con las medidas físicas.

- Comunicar con computadora: la comunicación inalámbrica de la tarjeta de procesamiento a la computadora, así como los protocolos de comunicación.

Clasificar señales

La clasificación de señales se realizará mediante una RNA previamente entrenada, de acuerdo a la lista de palabras ya definidas en la *Tabla 2.1*, donde se identifican cinco grupos:

- Seña letra por letra de las 27 letras del alfabeto.
- Seña de los números del 1 al 10.
- Seña de los 7 días de la semana.
- Seña de los 12 meses del año.
- Seña de palabras básicas: hola, bien, mal, perdón, no y si.

Comunicar en interfaz

La interfaz es el resultado de la función principal, donde se visualizará la interpretación de las señas. En esta se deben ubicar 3 botones para su manejo:

- Iniciar.
- Pausar.
- Borrar.

Proveer energía

La fuente de alimentación debe de estar dispuesta del tal forma que alimente a dos componentes electrónicos que se encuentran en el dispositivo:

- La placa de procesamiento.

- Los sensores, tanto de orientación como de medición de los ángulos.

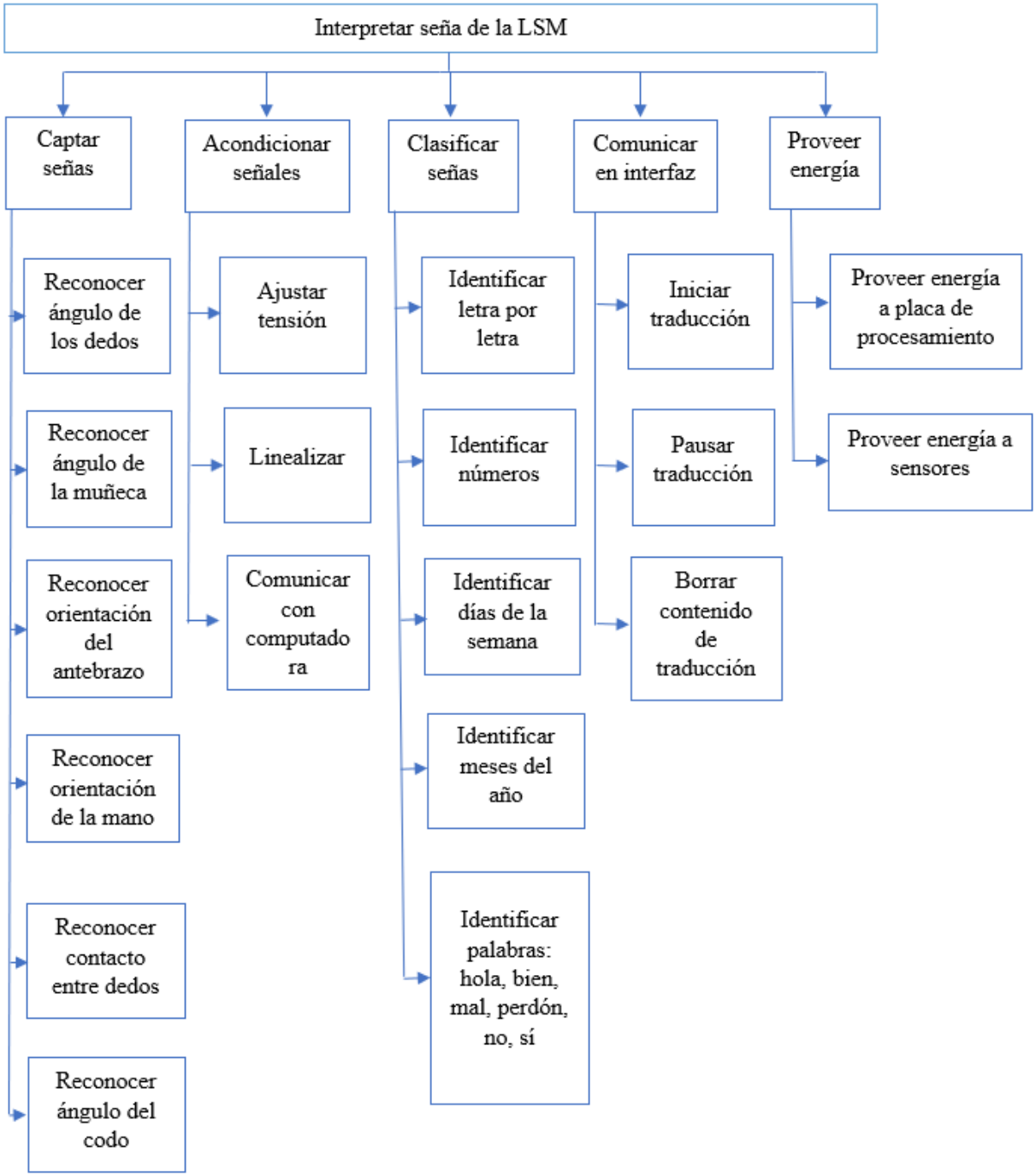


Imagen 2.2: FBS del dispositivo.

2.1.3. Arquitectura física

La arquitectura física del dispositivo se muestra en la *Imagen 2.3*, donde cada recuadro representa un componente tangible dentro del dispositivo, así como su distribución y la relación.

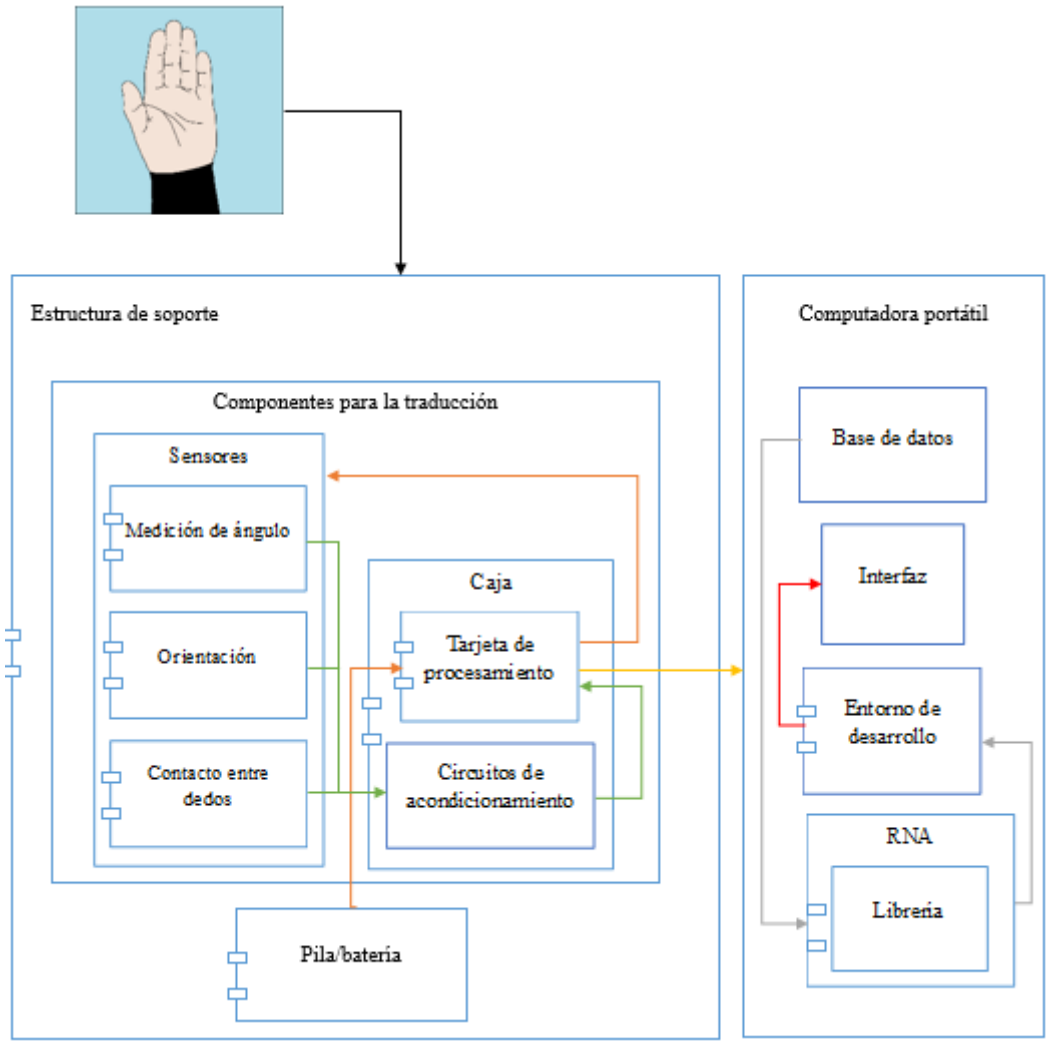


Imagen 2.3: Arquitectura física del dispositivo

El dispositivo inicia con la seña que representa el usuario, la entrada es la estructura sujeta a la extremidad superior derecha. Dentro de la estructura de soporte se tienen los componentes para la traducción y la batería. La batería puede alimentar

tanto a la tarjeta de procesamiento como a los sensores, como se indica en la *Imagen 2.2*, sin embargo, la tarjeta de procesamiento cuenta con salidas de alimentación, por lo que, se dispuso conectar la batería directamente a la tarjeta de procesamiento y alimentar los sensores. Los sensores que se utilizan son 3 diferentes y van directo al acondicionamiento para después pasar la conexión a la tarjeta de procesamiento. Tanto la tarjeta como los circuitos están dispuestos en una caja. Los datos son procesados de forma inalámbrica y almacenados en una base de datos para el correcto entrenamiento de la RNA. Para el procesamiento del modelo de la RNA se tiene un Entorno de Desarrollo Integrado (Integrated Development Environment, en adelante IDE por sus siglas en inglés). Como salida se tiene la interfaz, donde se muestra la traducción de la seña.

2.1.4. Propuestas de solución

Para las propuestas de solución (generación de conceptos) se crea una tabla, mostrada en la *Tabla 2.2*, de características y criterios a considerar en el dispositivo.

No.	**Criterio**	**Características**
1.	Dispositivo ligero.	Tipo de sensor para la medición de flexión.
2.	Dispositivo de precio accesible.	Tipo de sensor para la medición de orientación.
3.	Dispositivo que permita el libre movimiento al realizar las señas.	Tipo de sensor para la detección de contacto entre dedos.
4.	Precisión de medición de los sensores.	Forma del soporte para la mano.
5.	Sensores delgados.	Distribución de los sensores para la medición de flexión en la mano.
6.	Sensores pequeños.	Posición del sensor del codo.
7.	Pila/batería pequeña.	Posición del sensor de la muñeca.

(pasa a la página siguiente)

No.	Criterio	Características
8.	Duración de la pila/batería.	Posición del sensor de orientación para la mano.
9.	Consumo bajo de energía para la transmisión de datos.	Posición del sensor de orientación para el antebrazo.
10.	Distancia de transmisión de datos.	Posición del sensor para la detección del contacto entre dedos.
11.		Protocolo de comunicación para la transmisión inalámbrica de datos.
12.		Tipo de pila/batería.

Tabla 2.2: Características y Criterios del dispositivo.

En la *Tabla 2.3* se muestra la relación de las dos columnas de la *Tabla 2.2*, marcando los criterios a considerar por cada característica.

		Criterio									
Característica		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	1	*	*	*	*	*	*				
	2	*	*	*	*	*	*				
	3	*	*	*	*	*	*				
	4	*	*	*							
	5	*	*	*	*						
	6			*	*						
	7			*	*						
	8			*	*						
	9			*	*						
	10			*	*						
	11									*	*
	12	*	*	*				*	*		

Tabla 2.3: Relación Criterio-Característica.

El método AHP se aplica en la *Tabla 2.4* para comparar la importancia de los

criterios en el dispositivo.

		Criterio									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Criterio	**1**	1	5	1	3	3	3	5	5	7	7
	2	1/5	1	1/7	1/5	1/5	1/5	1/5	1/5	1/3	1/3
	3	1	7	1	3	5	5	7	7	7	7
	4	1/3	5	1/3	1	3	3	5	7	7	7
	5	1/3	5	1/5	1/3	1	3	5	7	5	7
	6	1/3	5	1/5	1/3	1/3	1	3	5	5	7
	7	1/5	5	1/7	1/5	1/5	1/3	1	5	5	5
	8	1/5	5	1/7	1/7	1/7	1/5	1/5	1	1/3	1
	9	1/7	3	1/7	1/7	1/5	1/5	1/5	3	1	3
	10	1/7	3	1/7	1/7	1/7	1/7	1/5	1	1/3	1
		3.89	44	3.45	8.5	13.2	16.1	26.8	41.2	38	45.3

Tabla 2.4: Importancia de Criterios.

En la *Tabla 2.5* se muestra la matriz normalizada.

		Matriz Normalizada									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Criterio	**1**	0.26	0.11	0.29	0.35	0.23	0.19	0.19	0.121	0.18	0.15
	2	0.05	0.02	0.04	0.02	0.02	0.01	0.01	0.005	0.01	0.01
	3	0.26	0.16	0.29	0.35	0.38	0.31	0.26	0.17	0.18	0.15
	4	0.09	0.11	0.1	0.12	0.23	0.19	0.19	0.17	0.18	0.15
	5	0.09	0.11	0.06	0.04	0.08	0.19	0.19	0.17	0.13	0.15
	6	0.09	0.11	0.06	0.04	0.03	0.06	0.11	0.121	0.13	0.15
	7	0.05	0.11	0.04	0.02	0.02	0.02	0.04	0.121	0.13	0.11
	8	0.05	0.11	0.04	0.02	0.01	0.01	0.01	0.024	0.01	0.02
	9	0.04	0.07	0.04	0.02	0.02	0.01	0.01	0.073	0.03	0.07
	10	0.04	0.07	0.04	0.02	0.01	0.01	0.01	0.024	0.01	0.02

Tabla 2.5: Matriz Normalizada.

En la *Tabla 2.6* se muestran los vectores promedio que se utilizan más adelante.

Vector promedio

1	2	3	4	5	6	7	8	9	10
0.21	0.02	0.25	0.15	0.12	0.09	0.07	0.031	0.04	0.02

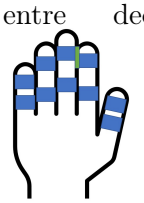
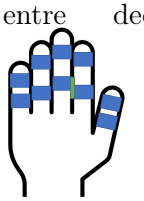
Tabla 2.6: Vector promedio.

Tabla de propuestas

La tabla de propuestas de solución mostrada en la *Tabla 2.7* es construida a partir de las características mostradas en la *Tabla 2.2*, donde se tienen de dos a tres propuestas para cada una de ellas.

No.	Característica	Propuesta A	Propuesta B	Propuesta C
1	**Tipo de sensor para la medición de flexión**	Sensor piezoresistivo.	Sensor óptico.	Sensor magnético.
2	**Tipo de sensor para la medición de orientación.**	Acelerómetro.	Sensor magnético + acelerómetro.	Giroscopio + acelerómetro.
3	**Tipo de sensor para la detección de contacto entre dedos.**	Sensor piezoeléctrico.	Sensor táctil capacitivo.	Sensor como interruptor (abrir – cerrar circuito).
4	**Forma del soporte para la mano.**	Plana sobre la palma de la mano.	Moldeada sobre la palma de mano.	Moldeada sobre la palma de la mano y dedos.

(pasa a la página siguiente)

No.	Característica	Propuesta A	Propuesta B	Propuesta C
5	**Distribución de los sensores para la medición de flexión en la mano.**	Un sensor cubriendo totalmente cada dedo. 	Dos sensores cubriendo cada dedo. 	Un sensor cubriendo totalmente cada dedo y un sensor entre cada dedo. 
6	**Posición del sensor del codo.**	Parte interna del codo.	Parte externa del codo	
7	**Posición del sensor de la muñeca.**	Parte posterior de la muñeca.	Parte frontal de la muñeca.	
8	**Posición del sensor de orientación para la mano.**	Dorso de la mano.	Palma de la mano.	
9	**Posición del sensor de orientación para el antebrazo.**	Parte interna del antebrazo.	Parte externa del antebrazo.	
10	**Posición del sensor para la detección del contacto entre dedos.**	Parte superior entre dedos. 	Parte inferior entre dedos. 	Parte media entre dedos. 
11	**Protocolo de comunicación para la transmisión inalámbrica de datos.**	Wifi.	Bluetooth serial.	BLE.

(pasa a la página siguiente)

No.	Característica	Propuesta A	Propuesta B	Propuesta C
12	**Tipo de pila/batería.**	Batería Li-Ion.	Batería Li-Po.	Batería Ni-MH.

Tabla 2.7: Tabla de propuestas de solución por característica.

2.1.5. Combinación y evaluación

Las distintas combinaciones del dispositivo se muestran en la *Tabla 2.8*, donde se evalúa cada combinación de las propuestas dadas en la *Tabla 2.7*. En este paso se analiza si algunas de las opciones no se pueden integrar en el dispositivo debido a que no son compatibles con las otras características. La tabla *Tabla 2.8* muestra 4 combinaciones diferentes.

No.	Característica	1	2	3	4
1	**Tipo de sensor para la medición de flexión**	Sensor piezoresistivo.	Sensor óptico.	Sensor magnético.	Sensor piezoresistivo.
2	**Tipo de sensor para la medición de orientación.**	Sensor magnético + acelerómetro.	Acelerómetro.	Giroscopio + acelerómetro.	Giroscopio + acelerómetro.
3	**Tipo de sensor para la detección de contacto entre dedos.**	Sensor piezoeléctrico.	Sensor táctil capacitivo.	Sensor táctil capacitivo.	Sensor como interruptor (abrir – cerrar circuito).

(pasa a la página siguiente)

No.	Característica	1	2	3	4
4	**Forma del soporte para la mano.**	Plana sobre el dorso de la mano.	Moldeada sobre el dorso de la mano y dedos.	Moldeada sobre el dorso de la mano y dedos.	Moldeada sobre el dorso de mano.
5	**Distribución de los sensores para la medición de flexión en la mano.**	Un sensor cubriendo totalmente cada dedo y un sensor entre cada dedo.	Un sensor cubriendo totalmente cada dedo.	Un sensor cubriendo totalmente cada dedo.	Dos sensores cubriendo cada dedo.
6	**Posición del sensor del codo.**	Parte externa del codo.	Parte externa del codo.	Parte interna del codo.	Parte interna del codo.
7	**Posición del sensor de la muñeca.**	Parte frontal de la muñeca.	Parte posterior de la muñeca.	Parte frontal de la muñeca.	Parte posterior de la muñeca.
8	**Posición del sensor de orientación para la mano.**	Dorso de la mano.	Dorso de la mano.	Palma de la mano.	Dorso de la mano.
9	**Posición del sensor de orientación para el antebrazo.**	Parte interna del antebrazo.	Parte externa del antebrazo.	Parte interna del antebrazo.	Parte externa del antebrazo.
10	**Posición del sensor para la detección del contacto entre dedos.**	Parte inferior entre dedos.	Parte media entre dedos.	Parte media entre dedos.	Parte superior entre dedos.

(pasa a la página siguiente)

No.	Característica	1	2	3	4
11	**Protocolo de comunicación para la transmisión inalámbrica de datos.**	BLE.	Wifi.	BLE.	Bluetooth serial.
12	**Tipo de pila/batería.**	Batería Li-Ion.	Batería Li-Po.	Batería Ni-MH.	Batería Li-Ion.

Tabla 2.8: Tabla de combinaciones.

El último paso es evaluar cada criterio con el método AHP comparando las combinaciones 1, 2, 3 y 4. Para cada criterio se tienen las características asociadas como se muestra en la *Tabla 2.3*, por lo que se facilita evaluar cada combinación por criterio. En la *Tabla 2.9* se muestra una asignación de evaluación de 0 a 5 para cada criterio en su respectiva combinación.

	Combinación				
Criterio		**1**	**2**	**3**	**4**
	1	3	3	1	4
	2	2	3	3	2
	3	2	3	0	5
	4	3	3	2	4
	5	3	4	2	5
	6	2	5	2	5
	7	3	5	3	3
	8	5	3	1	5
	9	5	1	5	5
	10	4	5	4	1

Tabla 2.9: Evaluación de combinaciones por criterio.

En la *Tabla 2.10*, se muestra el vector promedio obtenido en la *Tabla 2.6* asociado a cada criterio. Este valor se multiplica por la evaluación realizada por criterio en la *Tabla 2.9*. Finalmente se obtiene una suma para cada combinación, donde la

combinación con la suma más alta es la combinación elegida para el dispositivo. Los resultados indican que la mejor opción de combinación es la 4.

		Valor	**1**	**2**	**3**	**4**
Criterio	**1**	0.21	0.6	0.6	0.2	0.83
	2	0.02	0	0.1	0.1	0.04
	3	0.25	0.5	0.8	0	1.26
	4	0.15	0.5	0.5	0.3	0.61
	5	0.12	0.4	0.5	0.2	0.6
	6	0.09	0.2	0.5	0.2	0.45
	7	0.07	0.2	0.3	0.2	0.2
	8	0.03	0.2	0.1	0	0.15
	9	0.04	0.2	0	0.2	0.18
	10	0.02	0.1	0.1	0.1	0.02
	SUMA		2.8	3.4	1.5	4.35

Tabla 2.10: Resultados finales de evaluación.

2.1.6. Concepto ganador

El dispositivo cuenta con sensores piezoresistivos para la medición de flexión en los dedos, muñeca y codo. En cada dedo se cuenta con dos de sensores, con un total de 10 sensores para toda la mano. En la parte posterior de la muñeca y en la parte interna del codo, se cuenta con un sensor de medición. El soporte para los sensores de la mano esta moldeado de tal forma que cubre el dorso de la mano. Cuenta con dos sensores para la medición de orientación (giroscopio y acelerómetro) en el dorso de la mano y en la parte externa del antebrazo. El sensor del dorso está colocado dentro del soporte de la mano, mientras que el sensor del antebrazo es colocado en la parte media del antebrazo, donde se cuenta con una estructura de soporte de cinta elástica. Para la detección del contacto entre los dedos índice y dedo medio se cuenta con un detector como interruptor en la parte superior de los soportes de los dedos

para que al hacer contacto entre ellos se pueda cerrar un circuito. El protocolo de comunicación para la transmisión inalámbrica de datos es Bluetooth serial y para la fuente de alimentación del dispositivo, se dispone de una batería Li-Ion recargable, colocada en el brazo de su portador.

En la *Imagen 2.4* se muestra el dibujo del dispositivo completo para la extremidad superior derecha.

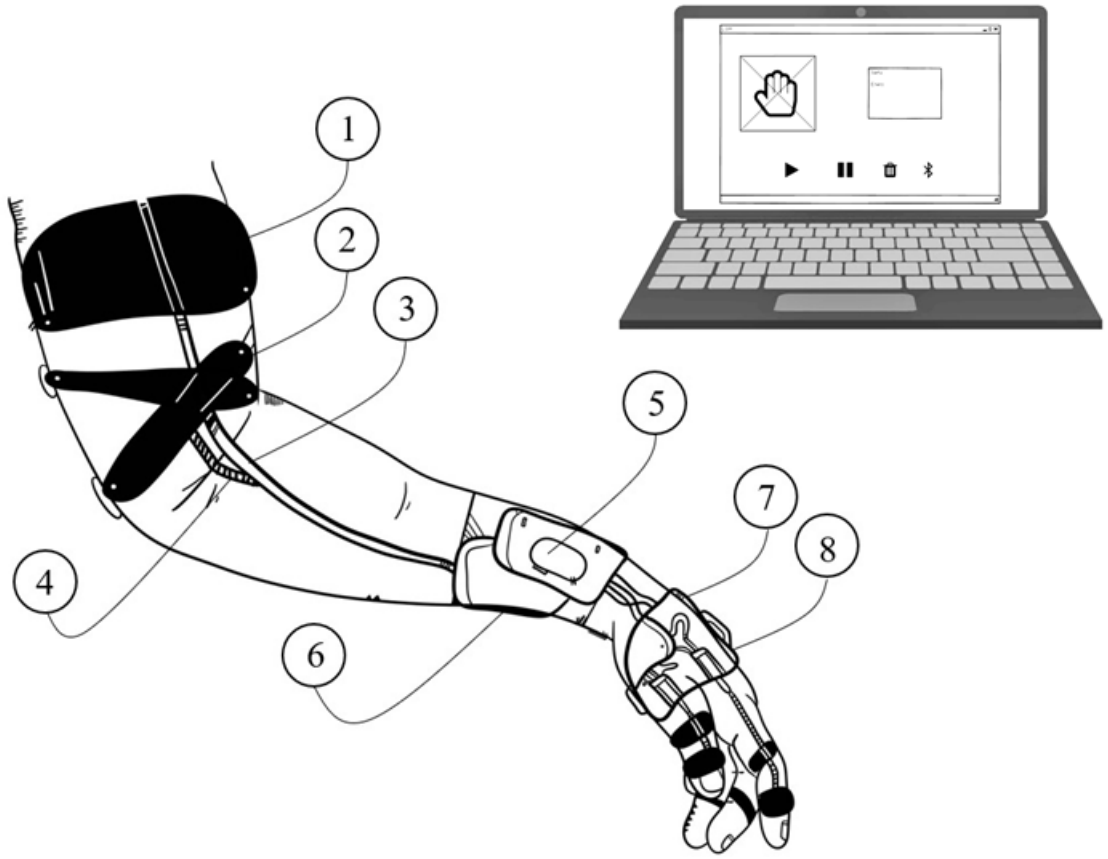


Imagen 2.4: Diseño preliminar del dispositivo traductor de LSM.

En la *Imagen 2.5* se muestra el dibujo de la estructura de la mano del dispositivo.

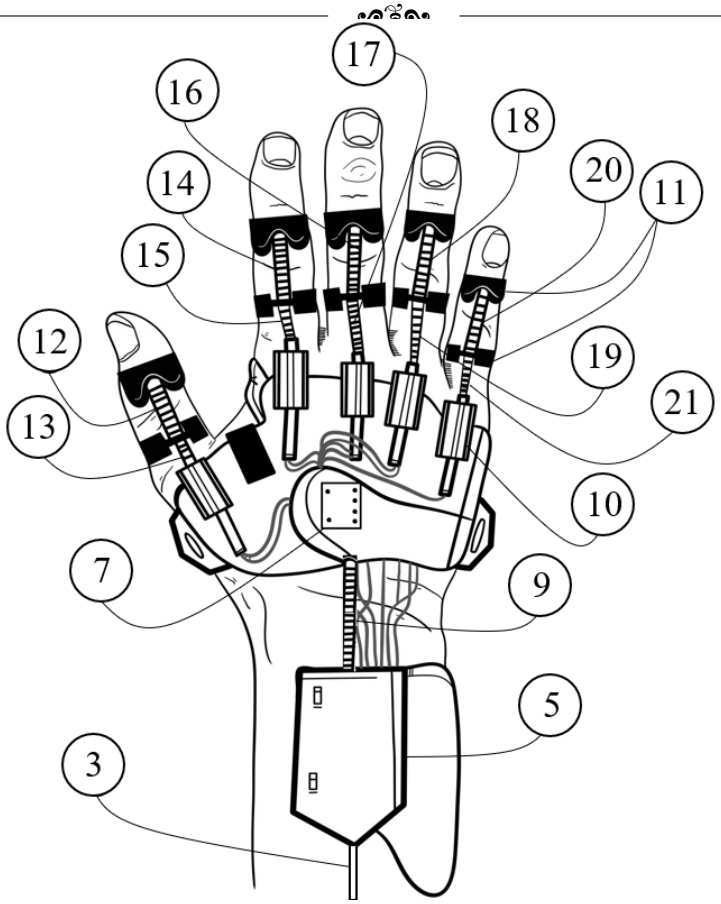


Imagen 2.5: Diseño preliminar de la estructura de la mano del dispositivo traductor de LSM.

La *Tabla 2.11* muestra la nomenclatura marcada en la *Imagen 2.4* e *Imagen 2.5*.

1. Soporte de pila
2. Soporte de agarre del codo
3. Cable de conexión a la pila
4. Sensor flex del codo (S12)
5. Tarjeta de procesamiento y circuitos de acondicionamiento
6. Acelerómetro y giroscopio del antebrazo (M1)
7. Acelerómetro y giroscopio del dorso de la mano (M2)
8. Soporte de la mano

(pasa a la página siguiente)

9. Sensor flex de la muñeca (S11)
10. Soporte de sensores de articulación superior al dorso
11. Soporte de sensores en los dedos
12. Sensor flex de articulación superior del dedo pulgar (S1)
13. Sensor flex de articulación inferior del dedo pulgar (S2)
14. Sensor flex de articulación superior del dedo índice (S3)
15. Sensor flex de articulación inferior del dedo índice (S4)
16. Sensor flex de articulación superior del dedo medio (S5)
17. Sensor flex de articulación inferior del dedo medio (S6)
18. Sensor flex de articulación superior del dedo anular (S7)
19. Sensor flex de articulación inferior del dedo anular (S8)
20. Sensor flex de articulación superior del dedo meñique (S9)
21. Sensor flex de articulación inferior del dedo meñique (S10)

Tabla 2.11: Nomenclatura del dispositivo traductor de la LSM.

2.2. Diseño Detallado

El diseño detallado es una estructura que proporciona las representaciones concretas de los módulos a un nivel más alto de abstracción y notación que el proporcionado por el diseño conceptual. El diseño detallado es el diseño sistemático de modelado y simulación de los componentes, para esto se requiere el uso de herramientas de software para representarlo, así como cálculos y diagramas.

2.2.1. Módulo de captura de señas

El sensor flex puede clasificarse como un sensor piezoresistivo; debido a sus características es el más usado en guantes de datos. Existen dos tamaños comerciales de estos sensores: 2.2 pulgadas (5.588 cm) y 4.5 pulgadas (11.43 cm). Se utilizan los sensores flex 2.2 por su tamaño para una talla XS.

El módulo MPU6050 se utiliza para medir la orientación del dispositivo, ya que, contiene un giroscopio de 3 ejes y un acelerómetro igualmente de 3 ejes. Para la detección del contacto entre los dedos índice y medio se coloca un hilo conductivo cosido en los soportes superiores de cada uno de estos dos dedos. El hilo conductivo es igual a un hilo normal, sólo que éste conduce electricidad.

Para la captura de señas es necesario colocar los sensores para que las lecturas de los sensores se tome de forma correcta. En las siguientes tablas se enlistan todas las señas a detectar, relacionadas con la medición de cada uno de los 15 sensores: 12 sensores flex, 2 MPU6050 y el hilo conductivo.

Las tablas se agrupan en 5 grupos distintos: el alfabeto, los números, los meses, los días de la semana y las palabras.

Sensores flex: En las tablas se considera un ángulo igual a 90° como el ángulo del dedo cuando se encuentra recto y en posición vertical, este valor es representado

con un 0. Cuando existe un doblez del dedo hacia la palma de la mano se representa con un 1 y para representarlo al lado opuesto se representa con un -1.

MPU6050: Para los sensores de posición y movimiento se considera:

- si la posición de la palma de la mano va hacia en frente (P), si es opuesta (D), hacia un lado (L), hacia abajo (A) y si la posición cambia en una misma seña (X).

- si es un seña de movimiento (1) y si no (0).

Hilo conductivo: Si existe contacto entre los dos dedos (1) y si no existe (0).

En la *Tabla 2.12* se muestran los valores anteriormente explicados para las letras del alfabeto dactilológico. Los sensores flex están representados desde S1 hasta S12, nomenclatura asignada de acuerdo a la *Tabla 2.11*, así como los MPU6050 (M1 Y M2).

		Sensores flex														
Alfabeto dactilológico		**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**M1**	**M2**	**H**
	A	0	0	1	1	1	1	1	1	1	1	0	1	P0	P0	1
	B	1	1	0	0	0	0	0	0	0	0	0	1	P0	P0	1
	C	1	1	1	1	1	1	1	1	1	1	0	1	L0	L0	1
	D	1	1	0	0	1	1	1	1	1	1	0	1	L0	L0	0
	E	1	1	1	-1	1	-1	1	-1	1	-1	0	1	P0	P0	1
	F	1	1	1	1	0	0	0	0	0	0	0	1	P0	P0	0
	G	0	0	0	0	1	1	1	1	1	1	1	1	D0	D0	0
	H	0	0	0	0	0	0	1	1	1	1	1	1	D0	D0	1
	I	1	1	1	1	1	1	1	1	0	0	0	1	P0	P0	1
	J	1	1	1	1	1	1	1	1	0	0	1	1	X1	X1	1
	K	0	1	0	1	0	1	1	1	1	1	1	1	X1	L1	0
	L	0	0	0	0	1	1	1	1	1	1	0	1	P0	P0	0
	M	1	1	1	1	1	1	1	1	1	1	1	1	A0	A0	1
	N	1	1	1	1	1	1	1	1	1	1	1	1	A0	A0	1
	Ñ	1	1	1	1	1	1	1	1	1	1	1	1	A1	A1	1
	O	1	1	1	1	1	1	1	1	1	1	1	1	L0	L0	1
	P	0	1	0	1	0	1	1	1	1	1	0	1	L0	L0	0
	Q	1	1	1	1	1	1	1	1	1	1	1	1	X1	X1	0
	R	0	1	0	1	0	0	1	1	1	1	0	1	P0	P0	0
	S	1	1	1	1	1	1	1	1	1	1	0	1	P0	P0	1
	T	0	1	1	1	1	1	1	1	1	1	0	1	P0	P0	0
	U	1	1	0	0	0	0	1	1	1	1	0	1	P0	P0	1
	V	1	1	0	0	0	0	1	1	1	1	0	1	P0	P0	0
	W	1	1	0	0	0	0	0	0	1	1	0	1	P0	P0	0
	X	1	1	1	1	1	1	1	1	1	1	1	1	L0	L0	0
	Y	0	0	1	1	1	1	1	1	0	0	1	1	D0	D0	1
	Z	1	1	0	0	1	1	1	1	1	1	1	1	A1	A1	0

Tabla 2.12: Relación alfabeto dactilológico con sensores.

En la *Tabla 2.13* se muestra la relación de los sensores con los números del 1 al 10.

		Sensores flex														
Números		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	M1	M2	H
	1	1	1	0	0	1	1	1	1	1	1	0	1	D0	D0	0
	2	1	1	0	0	0	0	1	1	1	1	0	1	D0	D0	0
	3	1	1	0	0	0	0	0	0	1	1	0	1	D0	D0	0
	4	1	1	0	0	0	0	0	0	0	0	0	1	D0	D0	0
	5	0	0	0	0	0	0	0	0	0	0	0	1	D0	D0	0
	6	0	0	1	1	1	1	1	1	1	1	0	1	D0	D0	1
	7	0	0	0	0	1	1	1	1	1	1	0	1	D0	D0	0
	8	0	0	0	0	0	0	1	1	1	1	0	1	D0	D0	0
	9	1	1	1	1	1	1	1	1	1	1	0	1	D1	D1	1
	10	0	0	0	0	0	0	0	0	0	0	1	1	X1	X1	0

Tabla 2.13: Relación de números del 1 al 10 con sensores.

En la *Tabla 2.14* se muestra la relación de los sensores con los meses del año correspondientes a su número del 1 al 12.

		Sensores flex														
Meses del año		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	M1	M2	H
	1	1	1	1	-1	1	-1	1	-1	1	-1	0	1	X1	X1	1
	2	1	1	1	1	0	0	0	0	0	0	0	1	X1	X1	0
	3	1	1	0	0	0	0	0	0	1	1	1	1	L1	A1	1
	4	0	0	1	1	1	1	1	1	1	1	0	1	P1	P1	1
	5	1	1	0	0	0	0	0	0	1	1	0	1	X1	X1	1
	6	1	1	1	1	1	1	1	1	0	0	0	1	X1	X1	1
	7	0	0	0	0	1	1	1	1	0	0	0	1	X1	X1	0
	8	0	0	1	1	1	1	1	1	1	1	0	1	X1	X1	1
	9	1	1	1	1	1	1	1	1	1	1	0	1	X1	X1	1
	10	1	1	1	1	1	1	1	1	1	1	0	1	X1	X1	1
	11	1	1	0	0	0	0	1	1	1	1	0	1	X1	X1	1
	12	1	1	0	0	1	1	1	1	1	1	0	1	X1	X1	0

Tabla 2.14: Relación de los meses del año con sensores.

En la *Tabla 2.15* se muestra la relación de los sensores con los días de la semana correspondientes a su número del 1 al 7.

		Sensores flex														
Días de la semana		**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**M1**	**M2**	**H**
	1	0	0	0	0	1	1	1	1	1	1	0	1	P1	P1	0
	2	0	0	0	0	0	0	1	1	1	1	0	1	P1	P1	1
	3	0	0	0	0	0	0	0	0	1	0	0	1	P1	P1	1
	4	1	1	1	1	1	1	1	1	0	0	0	1	P1	P1	1
	5	1	1	0	0	0	0	1	1	1	1	0	1	P1	P1	0
	6	1	1	1	1	1	1	1	1	1	1	0	1	P1	P1	1
	7	1	1	0	0	1	1	1	1	1	1	0	1	L1	L1	0

Tabla 2.15: Relación de los días de la semana con sensores.

En la *Tabla 2.16* se muestra la relación de los sensores con las 6 palabras:

1. hola.
2. bien.
3. mal.
4. perdón.
5. no.
6. si.

		Sensores flex														
Palabras		**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**M1**	**M2**	**H**
	1	0	0	0	0	0	0	1	1	1	1	0	1	X1	X1	1
	2	1	1	0	0	0	0	0	0	0	0	0	1	P1	P1	1
	3	1	1	0	0	0	0	0	0	0	0	1	1	L1	L1	1
	4	0	0	1	1	1	1	1	1	0	0	0	1	D0	D0	0
	5	0	1	0	1	0	1	1	1	1	1	0	1	P1	P1	1
	6	1	1	1	1	1	1	1	1	1	-1	0	1	P1	P1	1

Tabla 2.16: Relación de las 6 palabras con sensores.

En las tablas anteriores se demuestra que los sensores son suficientes para la correcta lectura de todas las señas, ya que una misma combinación de todos los

sensores no se repite, excepto para las letras m y n y para los meses de septiembre y octubre. Estas son excepciones irrelevantes, ya que el ángulo de los dedos es el que diferencia realmente este par de señas entre sí.

Los sensores son colocados en dos partes: mano y antebrazo. En la mano son colocados diez sensores flex (S1-S10), un sensor MPU6050 (M2) y el hilo conductivo.

Para una correcta lectura de los sensores flex se debe colocar cada uno en el vértice del ángulo que se quiera medir, el punto medio del elemento resistivo del sensor. En la *Imagen 2.6* se muestran las medidas en cm. de los sensores flex que se utilizan, indicando en la medida del lado derecho el punto medio del elemento resistivo en el sensor.

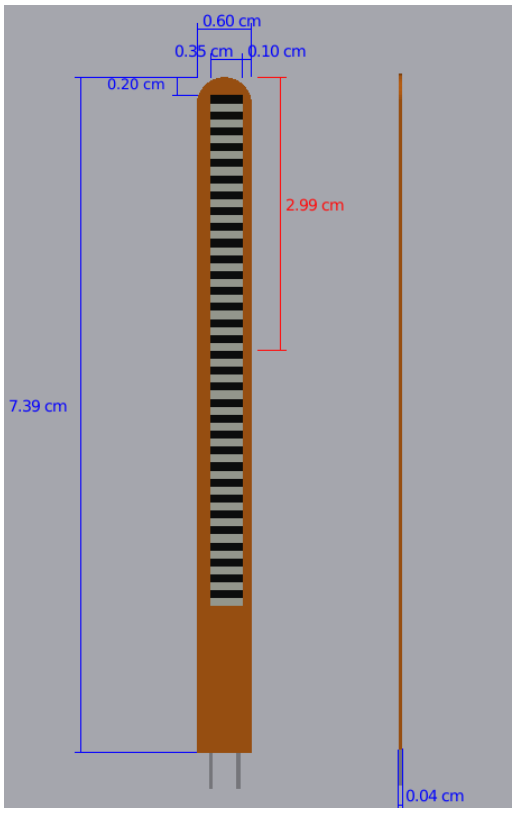


Imagen 2.6: Medidas y punto medio del sensor flex.

En cada dedo se miden dos ángulos: el ángulo que corresponde del dorso de la mano a cada dedo y el ángulo formado en la articulación inferior de cada dedo. El diseño para los sensores flex colocados en la mano se muestra en la *Imagen 2.7* con

las medidas correspondientes a una talla de mano XS.

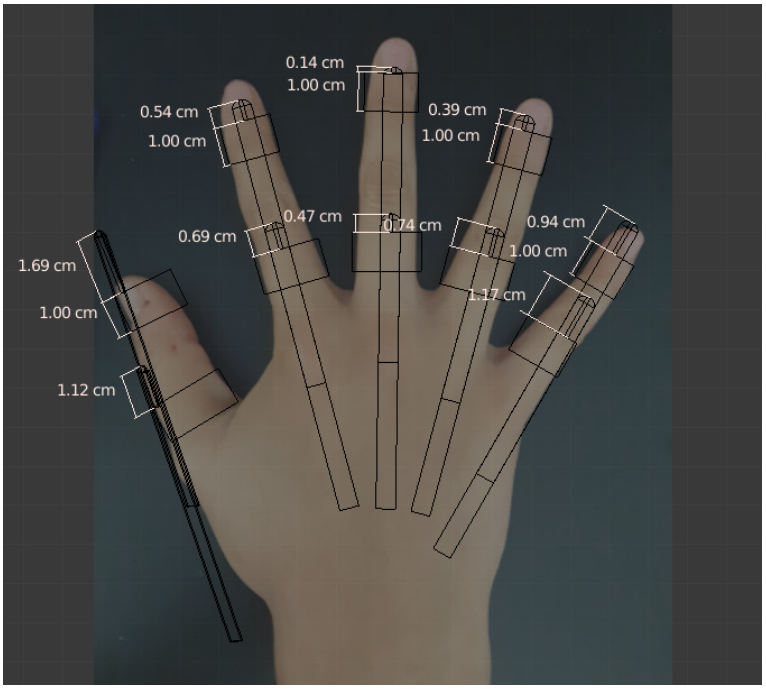


Imagen 2.7: Posición de los sensores flex en la mano.

Las bases que sostienen a los sensores flex mostrados, son de velcro para mejor comodidad y ajuste. Las medidas de estas cintas de velcro por dedo son tomadas de las medidas ajustadas para una talla XS. En la *Tabla 2.18* se muestran las medidas de cada cinta de velcro para los cinco dedos de la mano.

	Ancho	**Largo**
S1	1 cm	7.3 cm
S2	1 cm	7.6 cm
S3	1 cm	5.6 cm
S4	1 cm	7.3 cm
S5	1 cm	5.8 cm
S6	1 cm	7.3 cm
S7	1 cm	5.6 cm

(pasa a la página siguiente)

	Ancho	Largo
S8	1 cm	7.2 cm
S9	1 cm	5.1 cm
S10	1 cm	6.5 cm

Tabla 2.17: Medidas de cintas de velcro para ajuste en los dedos.

Para sostener los sensores colocados en la mano, se usa una estructura de Sintra PVC espumado. Este material es termo deformable, por lo que se utiliza calor para moldearla a una talla de mano XS. La estructura se muestra en la *Imagen 2.8*.

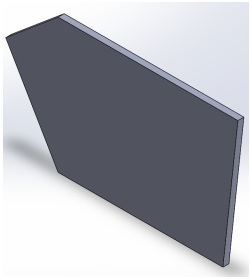


Imagen 2.8: Estructura de Sintra PVC espumado *[Apéndice 1]*.

Al contar con dos sensores flex en cada dedo, se presenta un roce entre estos dos, por lo que se colocan soportes de PLA para separar dichos sensores sobre la estructura de PVC espumado. El diseño del soporte para los dedos de la mano (no incluyendo el dedo pulgar) se muestra en la *Imagen 2.9*.

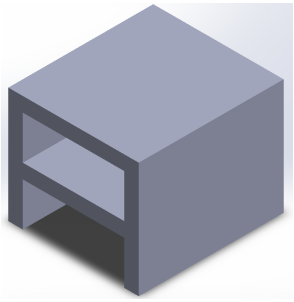


Imagen 2.9: Soporte para los sensores flex *[Apéndice 2]*.

2.2.2. Módulo de acondicionamiento de señales

La tarjeta de desarrollo ESP32 cuenta con 15 canales ADC, 2 canales I^2C y Bluetooth, todo en un mismo chip. Es una tarjeta con un peso de 7 gramos y con dimensiones de 51 mm x 27 mm. Adicionalmente se tiene que contar con la lectura de los 12 sensores flex como canales analógicos, la lectura de los 2 MPU6050 con protocolos de comunicación I^2C y con Bluetooth serial, por lo que, la tarjeta ESP32 cuenta con todos los requerimientos para el desarrollo del dispositivo usando ésta como microcontrolador. De los criterios más importantes para el diseño del dispositivo traductor es el peso y tamaño, mismos que se cumplen con el uso de la tarjeta ESP32. Para el acondicionamiento de señales se tienen 3 subfunciones: ajuste de la tensión, linealización del sensor y comunicación con la computadora.
En la *Tabla 2.18* se muestran las especificaciones de la tarjeta de desarrollo ESP32, para ajustar el voltaje de entrada a los pines analógicos de la tarjeta.

Especificaciones técnicas
Voltaje de Alimentación (USB): 5V DC
Voltaje de Entradas/Salidas: 3.3V DC
SoC: ESP32
CPU principal: Tensilica Xtensa 32-bit LX6
Frecuencia de Reloj: hasta 240Mhz
Desempeño: Hasta 600 DMIPS
Procesador secundario: Permite operaciones básica en modo de ultra bajo consumo.
Wifi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz hasta 150 Mbit/s)
Bluetooth:v4.2 BR/EDR and Bluetooth Low Energy (BLE)
Xtensa® Dual-Core 32-bit LX6 microprocessors, up to 600 DMIPS
Memoria: 448 KByte ROM, 520 KByte SRAM, 16 KByte SRAM in RTC, QSPI Flash/SRAM, 4 MBytes
Pines Digitales GPIO: 24 (Algunos pines solo como entrada)

(pasa a la página siguiente)

Especificaciones técnicas
Convesor Analogolico Digital: Dos ADC de 12 bits tipo SAR, soporta mediciones en hasta 18 canales, algunos pines soporta un amplificador con ganancia programable
UART: 2
Chip USB-Serial: CP2102
Antena en PCB
Seguridad: Estandares IEEE 802.11 incluyendo WFA, WPA/WPA2 and WAPI 1024-bit OTP, up to 768-bit for customers

Tabla 2.18: ESP32 Especificaciones técnicas [10].

El voltaje de entradas/salidas de la tarjeta es de 3.3V, por lo que se acondicionan los sensores a un rango de voltaje de 0V a 3.3V. Para el sensor MPU6050 no se necesita este ajuste de voltaje, debido a que se diseña de tal forma que su voltaje no exceda el voltaje de alimentación.

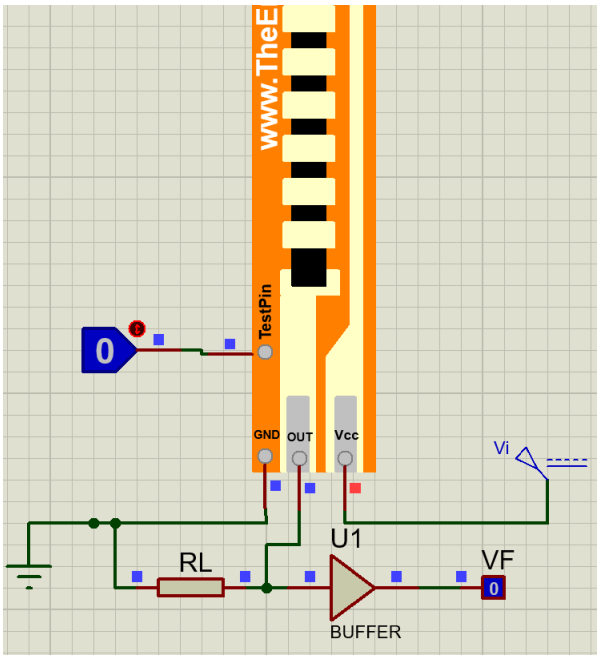


Imagen 2.10: Simulación sensor flex indicando que está en reposo.

En los sensores flex se diseña un divisor de voltaje, para que al variar la resistencia se obtenga un voltaje de salida (V_f) equivalente a una fracción del voltaje de entrada(V_i), que es 3.3V. En la *Imagen 2.10* e *Imagen 2.11* se muestra la simulación del circuito de divisor de voltaje para el sensor flex, con una resistencia variable $R(x)$ correspondiente al sensor flex y una resistencia R_L. El TestPin indica si el sensor está doblado, indicando solo valores 1 y 0, al igual que en el valor V_f, los cuales son los mismos valores al aplicar un 1 y un 0 respectivamente.

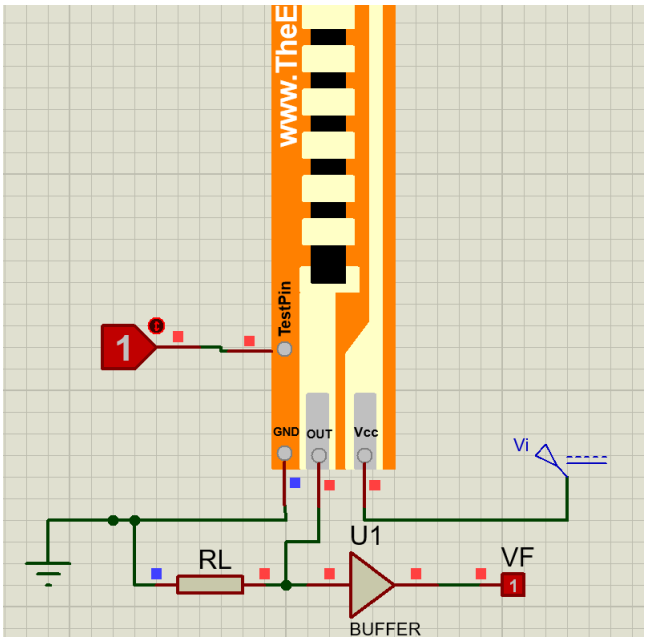


Imagen 2.11: Simulación sensor flex indicando que se encuentra doblado.

Para el cálculo de la resistencia R_L se linealiza el sensor flex de acuerdo a pruebas realizadas en cinco de estos sensores. En la *Tabla 2.19* se tiene una relación aproximada de los ángulos de un sensor flex con su resistencia variable $R(x)$ correspondiente, medido en $k\Omega$. Cuando el sensor flex no se encuentra doblado se considera un ángulo igual a 90°. Cuando se flexiona hacia adelante aumenta su resistencia y al doblar hacia atrás disminuye su resistencia.

Ángulo	$R(x)1$	$R(x)2$	$R(x)3$	$R(x)4$	$R(x)5$
10°	78.4	86.8	64.7	69	73.2
20°	63.3	77.2	63.4	65.4	61.3
30°	56.8	73.2	61.5	58.9	57.9
40°	54.4	64.5	57.9	54	48.2
50°	45.8	61.3	50.3	47.2	45.5
60°	43.7	52.7	45.8	42.9	43.6
70°	38.84	45.1	42	41.2	40.3
80°	36.85	41.3	38.29	38.12	35.45
90°	32.87	35.41	33.42	33.88	32.77
100°	32.92	32.32	32.51	31.89	31.42
110°	30.89	30.82	31.25	31.29	30.57
120°	30.7	31.66	31.09	30.89	31.12
130°	30.51	31.38	30.83	30.8	30.76
140°	30.33	30.87	30.41	30.63	30.5
150°	30.15	30.45	29.42	30.41	30.24
160°	30.07	30.37	29.39	29.87	30.09
170°	29.8	30.96	29.36	29.77	30.35

Tabla 2.19: Pruebas en sensores flex.

En la *Tabla 2.20* se muestra el promedio de resistencias de las cinco pruebas con sus respectivos ángulos medido en $k\Omega$.

x	**Ángulo**	$R(x)$
1	10°	74.42
2	20°	66.12
3	30°	61.66
4	40°	55.8
5	50°	50.02
6	60°	45.74
7	70°	41.488
8	80°	38.002

(pasa a la página siguiente)

x	Ángulo	$R(x)$
9	90°	33.67
10	100°	32.212
11	110°	30.964
12	120°	30.91
13	130°	30.856
14	140°	30.548
15	150°	30.134
16	160°	29.958
17	170°	30.048

Tabla 2.20: Resistencias por ángulo del sensor flex.

En la *Imagen 2.12* se puede visualizar la gráfica de la *Tabla 2.20*, donde se puede observar que la salida no es lineal.

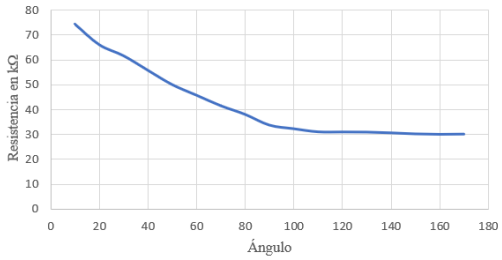


Imagen 2.12: Salida del sensor flex sin linealizar.

El método para linealizar la respuesta del sensor flex es encontrar una resistencia en paralelo a la resistencia variable $R(x)$, como se muestra en la *Imagen 2.13*.

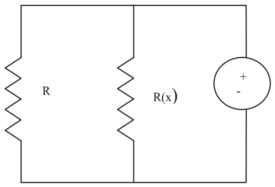


Imagen 2.13: Circuito en paralelo con una resistencia fija de linealización R [34].

El cálculo de la resistencia equivalente en paralelo se resuelve con la siguiente ecuación:

$$r_x = \frac{R_x R}{R_x + R} \quad \forall\ x = 1, 2...n-1 \tag{2.1}$$

Se requiere que r_x sea lineal, por lo que se toma la ecuación de la pendiente:

$$m = \frac{r_i - r_j}{x_i - x_j} \quad i \neq j \ \ \forall\ i, j = 1, 2...n-1 \tag{2.2}$$

Ahora, sustituyendo la ecuación (2.1) en (2.2) se tiene:

$$m = \frac{\frac{R_i R}{R_i + R} - \frac{R_j R}{R_j + R}}{x_i - x_j} \quad \forall\ i, j \tag{2.3}$$

Los valores de i pueden ser representados como los máximos y los valores de j como los mínimos. Despejando el denominador de la ecuación (2.3), sustituyendo las r_i por r_{max} y las r_j por r_{min} de la (2.2) y sustituyendo estas mismas por la (2.1) se tiene:

$$\frac{R_i R}{R_i + R} - \frac{R_j R}{R_j + R} = \left(\frac{x_i - x_j}{x_{max} - x_{min}}\right)\left(\frac{R_{max} R}{R_{max} + R} - \frac{R_{min} R}{R_{min} + R}\right) \quad \forall\ i, j \tag{2.4}$$

Factorizando a R y eliminándola de las dos partes de la ecuación se obtiene:

$$\frac{R_i}{R_i + R} - \frac{R_j}{R_j + R} = \left(\frac{x_i - x_j}{x_{max} - x_{min}}\right)\left(\frac{R_{max}}{R_{max} + R} - \frac{R_{min}}{R_{min} + R}\right) \quad \forall\ i, j \tag{2.5}$$

Se define la siguiente ecuación:

$$k = \frac{x_{max} - x_j}{x_{max} - x_{min}} \quad \forall\ j = 1, 2, 3...n-1 \tag{2.6}$$

Reemplazando la ecuación (2.6) en (2.5) y sustituyendo las R_i por R_{max}, la ecuación final queda de la siguiente forma:

$$\frac{R_{max}}{R_{max} + R} - \frac{R_j}{R_j + R} = k\left(\frac{R_{max}}{R_{max} + R} - \frac{R_{min}}{R_{min} + R}\right) \quad \forall\ i, j \tag{2.7}$$

Despejando a R de la ecuación, ya que es el valor que se busca:

$$R = \frac{R_{min}(R_{max} - R_j) - kR_j(R_{max} - R_{min})}{k(R_{max} - R_{min}) - R_{max} + R_j} \quad \forall\ j \tag{2.8}$$

Para sacar el valor R se obtiene primero el valor de k, para el cual se propone un valor de j=5. Para el valor máximo en el sensor se considera x = 11, debido a que la curva del sensor termina en este ángulo. Sustituyendo los valores en la ecuación (2.6):

$$k = \frac{110-50}{110-10} = 0.6$$

Se sustituyen valores en la ecuación (2.8):

$$R = \frac{74.42(30.964-50.02)-0.6(50.02)(30.964-74.42)}{0.6(30.964-74.42)-30.964+50.02} = 16.23k\Omega$$

Se puede comprobar la linealización de la curva, sustituyendo los valores de la ecuación (2.1), en la *Tabla 2.21* se muestran los resultados de las resistencias equivalentes resultantes con un valor de $R = 15.\text{k}\Omega$, debido a que es el valor comercial más cercano al resultado obtenido.

x	**Ángulo**	$R(x)(k\Omega)$	$r_x(k\Omega)$
1	10°	74.42	12.48
2	20°	66.12	12.23
3	30°	61.66	12.06
4	40°	55.8	11.82
5	50°	50.02	11.54
6	60°	45.74	11.3
7	70°	41.488	11.02
8	80°	38.002	10.75
9	90°	33.67	10.38
10	100°	32.212	10.23
11	110°	30.964	10.1

Tabla 2.21: Resistencia equivalente (linealizada).

En la *Imagen 2.14* se muestra la gráfica de la resistencia equivalente y sus ángulos, como se puede observar la respuesta del sensor flex ya está linealizada.

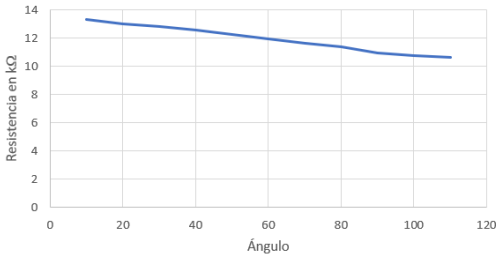


Imagen 2.14: Salida del sensor flex linealizada

Para obtener la ecuación se utiliza el método de mínimos cuadrados, el cual consiste en multiplicar las columnas (x_k, r_k) y elevar al cuadrado la primer columna. Estas operaciones se muestran en la *Tabla 2.22*

x_k	r_k	$(x_k)(r_k)$	$(x_k)(x_k)$
10	12483	124837	100
20	12226	244526	400
30	12064	361948	900
40	11822	472881	1600
50	11539	576976	2500
60	11295	677741	3600
70	11016	771179	4900
80	10754	860390	6400
90	10377	933932	8100
100	10234	1023426	10000
110	10104	1111535	12100
Σ 660	Σ 123920	Σ 7159376	Σ 50600

Tabla 2.22: Método de mínimos cuadrados.

La ecuación característica de la pendiente es $y = mx + b$, para obtener los valores de m y b se ocupan las ecuaciones (2.9) y (2.10) respectivamente, donde n es el número de x_k totales.

$$m = \frac{\Sigma(x_k, r_k) - \frac{\Sigma(x_k)\Sigma(r_k)}{n}}{\Sigma(x_k, x_k) - \frac{(\Sigma(x_k))^2}{n}} \tag{2.9}$$

$$b = \frac{\Sigma(r_k)}{n} - m\frac{\Sigma(x_k)}{n} \tag{2.10}$$

Resolviendo las ecuaciones se tienen los siguientes resultados de m y b:

m = -25.07, b = 12770

Ecuación del sensor flex linealizada: $r(x_k) = -25.07x_k + 12770$

Para el método de mínimos cuadrados existe un error estimado el cual se define mediante la ecuación (2.11).

$$e_k = r_k - r(x_k) \tag{2.11}$$

La sensibilidad del sensor mide la variación de la resistencia por grado del ángulo, lo que es el concepto de la pendiente. Sustituyendo los valores de la ecuación (2.2) se tiene el valor de la sensibilidad:

$$S = m = \frac{10651-13329}{110-10} = -26.78 \approx -25.07$$

El valor obtenido de la ecuación (2.9) no varía mucho al cálculo realizado.
Para el cálculo de la sensibilidad en cada punto puede ser calculada mediante la ecuación (2.12).

$$S(X_k) = \frac{r(x_{k+1}) - r(x_k)}{x_{k+1} - x_k} \tag{2.12}$$

El error en la medición se define como la razón entre el error de estimación y la sensibilidad, la ecuación (2.13) muestra la ecuación de este error.

$$E_k = \frac{e_k}{S(x_k)} \tag{2.13}$$

En la *Tabla 2.23* se muestran los resultados obtenidos de las ecuaciones (2.11),(2.12) y (2.13) para cada x_k

x_k	$R(x)(k\Omega)$	$r_k(\Omega)$	$S(x_k)$	$r(x_k)$	e_k	E_k
10	74.42	12.48	-25.74	12519	-35.49	1.37
20	66.12	12.23	-16.13	12268	-42.18	2.61
30	61.66	12.06	-24.29	12017	47.20	-1.94
40	55.8	11.82	-28.25	11766	55.03	-1.94
50	50.02	11.54	-24.38	11516	23.29	-0.95
60	45.74	11.3	-27.88	11265	30.21	-1.08
70	41.488	11.02	-26.19	11014	2.13	-0.08
80	38.002	10.75	-37.78	10763	-9.07	0.24
90	33.67	10.38	-14.27	10513	-136.16	9.53
100	32.212	10.23	-12.94	10262	-28.17	2.18
110	30.964	10.1		10011	93.19	

Tabla 2.23: Sensibilidades y errores.

El circuito de medición para el sensor flex queda como se muestra en la *Imagen 2.15*, donde el valor de R_L es el valor de la resistencia ya calculada $R = 15$ kΩ y

donde $R(x)$ es la resistencia del sensor flex variable.

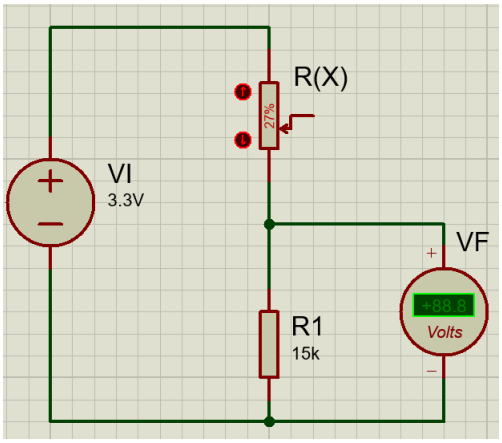


Imagen 2.15: Circuito de medición para el sensor flex.

La ecuación para determinar el V_f es un divisor de voltaje, esta ecuación se muestra en (2.14).

$$V_f = V_i \frac{R_L}{R(x) + R_L} \tag{2.14}$$

En la *Tabla 2.24* se muestran los resultados del V_f para cada ángulo x_k.

x_k	$V_f(V)$
10	0.55
20	0.61
30	0.65
40	0.7
50	0.76
60	0.81
70	0.88
80	0.93
90	1.02
100	1.05
110	1.08

Tabla 2.24: Salidas de voltaje por ángulo.

El comportamiento del voltaje se muestra en la *Imagen 2.16*, donde se puede observar un comportamiento lineal.

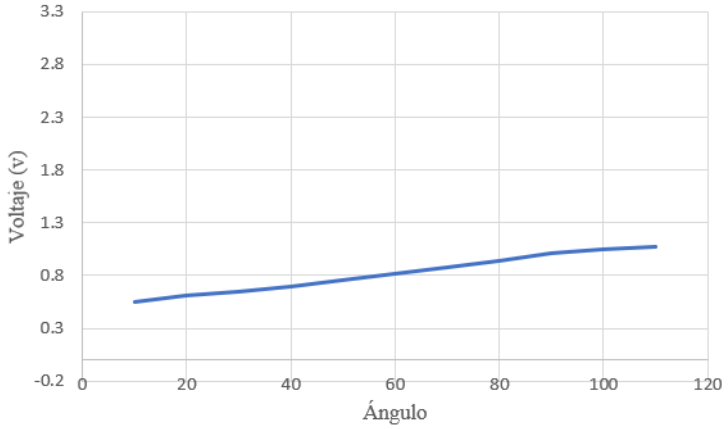


Imagen 2.16: Voltaje final del sensor flex.

Para la asignación de pines analógicos de la tarjeta, se muestra en la *Imagen 2.17* el diagrama de pines de la tarjeta ESP32. En este diagrama se muestran los 15 canales de conversión ADC que tiene la tarjeta utilizada (con 30 pines). De los cuales se usan únicamente los 6 canales de ADC1, ya que, al habilitar el WiFi y el Bluetooth de la tarjeta, todos los pines de ADC2 se deshabilitan. Se necesitan 12 canales ADC para la lectura de los sensores flex y 1 canal ADC para la lectura de la conexión del hilo conductivo. Al no contar con los canales disponibles se dispone de un MUX de 8 canales, el cual realiza la lectura de los sensores correspondientes a los dedos índice, medio, anular y meñique (2 sensores por cada dedo). Se asigna igualmente para la lectura en el MUX, 3 salidas digitales, correspondientes a los pines de salida GPIO25, GPIO26 y GPIO27. Las otras 5 entradas del canal ADC1, corresponden a la lectura de los dos sensores del dedo pulgar, al sensor de la muñeca, al sensor del codo y al hilo conductivo.

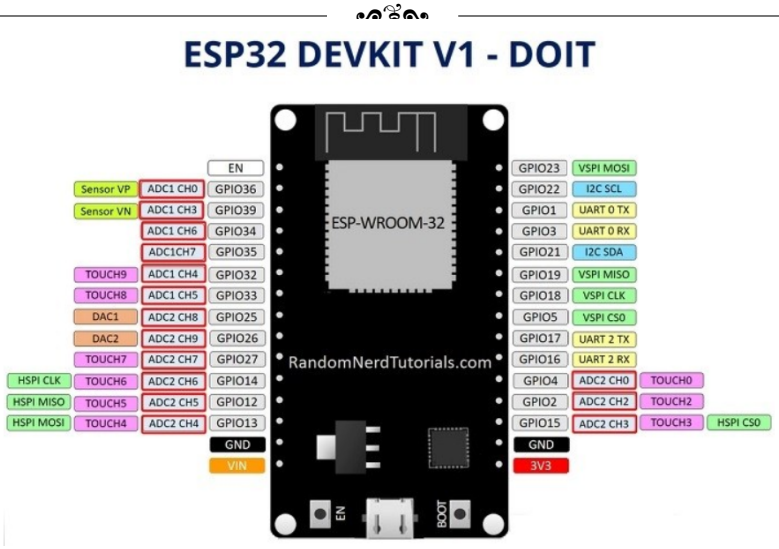


Imagen 2.17: Tarjeta ESP32 pines.

El MPU6050 utiliza un protocolo de comunicación I^2C por sus dos puertos para realizar esta comunicación:

- SDA: Puerto serial síncrono bidireccional, por donde se transmiten los datos de dispositivo a dispositivo [14].
- SCL: Puerto serial de sicronía de datos, este genera una señal de reloj con la velocidad de transferencia de datos adecuado al sensor [14].

La tarjeta ESP32 cuenta con dos canales I^2C, cuya dirección predeterminada es 0x68 cuando el pin ADD del MPU6050 se encuentra conectado a tierra, cuando este pin se encuentra conectado a positivo, se almacena en la dirección 0x69. En la *Tabla 2.25* se indican las conexiones de todos los pines del dispositivo a la tarjeta ESP32.

PIN	**Descripción**
VCC	3.3V
GND	Ground
SCL	I^2C Clock
SDA	I^2C Data

(pasa a la página siguiente)

PIN	Descripción
XDA	Auxiliary I^2C Data
XCL	Auxiliary I^2C Clock
ADD	I^2C address selection, either 0x68 or 0x69. Low = 0x68, high = 0x69.

Tabla 2.25: Conexión ESP32 con MPU6050 [40].

En la *Imagen 2.18* se muestra el esquema de conexiones de todos los sensores a la tarjeta ESP32.

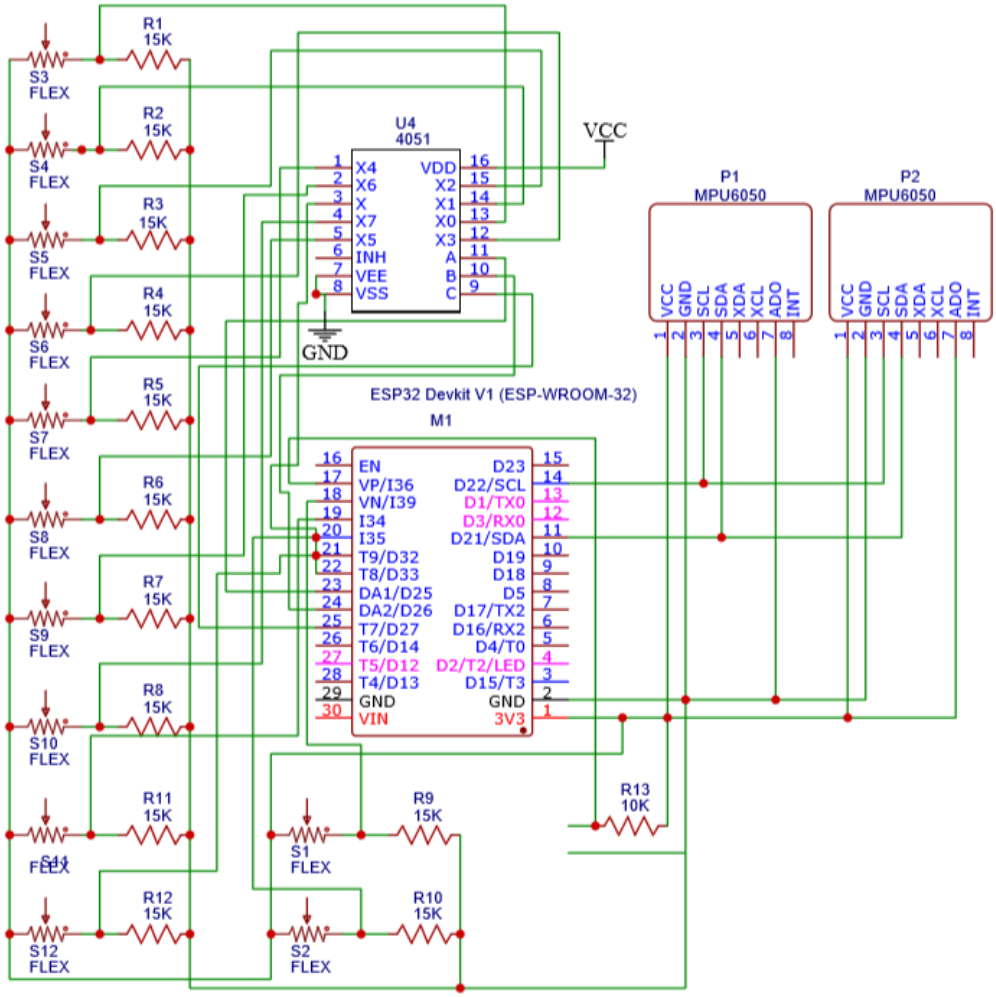


Imagen 2.18: Esquema de conexión de sensores a ESP32.

El IDE en el que se trabaja es Arduino. En Arduino IDE se puede programar la tarjeta ESP32 para la lectura de los sensores, también es posible trabajar con el modelo de la RNA mientras se mantiene la misma lectura de dichos sensores.
Para la lectura de los sensores flex en Arduino IDE, se asignan los pines analógicos y digitales mostrados en la *Tabla 2.26*.

PIN	**Conexión**	**pinMode**
GPIO39	S1	INPUT
GPIO34	S2	INPUT
GPIO35	MUX	INPUT
GPIO32	S11	INPUT
GPIO33	S12	INPUT
GPIO25	Selector 0	OUTPUT
GPIO26	Selector 1	OUTPUT
GPIO37	Selector 2	OUTPUT

Tabla 2.26: Asignación de pines en Arduino.

Los canales analógicos de la tarjeta ESP32 tienen una resolución de 12 bits, esto quiere decir que los resultados de la lectura varían entre 0 y 4095 ($2^{12} = 4096$), donde 0 es igual a 0V y 4095 es igual a 3.3V. Como resultado de los sensores flex, se tiene que entre mayor sea la flexión, menor es el voltaje y por lo tanto menor es la lectura analógica.
El hilo conductivo se coloca tanto en el dedo índice como en el dedo medio, para la detección de contacto entre ellos. En la *Imagen 2.19* se muestra el circuito para la conexión del hilo conductivo y los dedos.

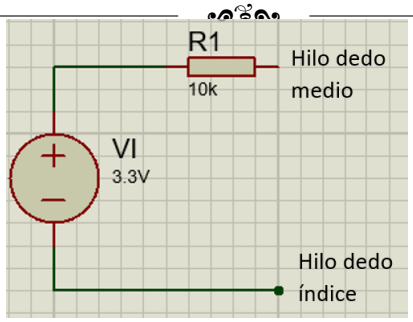


Imagen 2.19: Circuito de conexión del hilo conductivo.

Cuando los dedos no tienen contacto, el circuito se encuentra abierto, por lo que la medición de voltaje es de 3.3V como se muestra en la simulación de la *Imagen 2.20.*

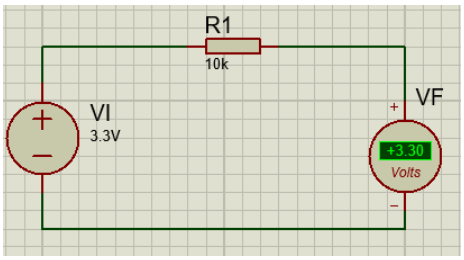


Imagen 2.20: Circuito abierto del hilo conductivo.

Cuando los dedos tienen contacto, se cierra el circuito, por lo que la medición del voltaje es de 0V, como se muestra en la simulación de la *Imagen 2.21.*

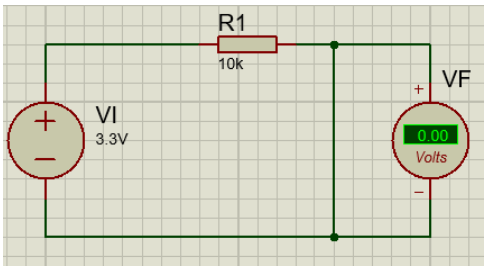


Imagen 2.21: Circuito cerrado del hilo conductivo.

Por lo tanto, si existe contacto entre los dedos, la lectura analógica del hilo conductivo corresponde a 0 y cuando no existe corresponde a 4095.

Para la lectura de los sensores MPU6050 en Arduino IDE, se utiliza la librería para protocolos de comunicación I^2C: wire.h. Dentro de esta librería es indispensable crear los dos objetos tipo MPU6050, asignando a cada uno la dirección 0x68 y 0x69 según corresponda. Para usar el protocolo I^2C se debe iniciar una transmisión del MPU6050, donde se envía un 0 al registro PWR_MGMT_1 (0x6B). La resolución de las lecturas es de 16 bits, por lo que el rango de lectura varía entre -32768 y 32767. Para la lectura de estos valores, se almacenan en un entero de 2 bytes (16 bits) que corresponde al tipo de dato int16_t. En la *Tabla 2.27* se muestran los registros de los *offsets* del MPU6050, donde se indica el número, nombre y descripción para cada registro. En esta tabla se visualiza que cada valor ocupa 2 registros (uno alto y otro bajo), por lo que para cada valor se piden 2 registros y se guardan en la variable asignada.

Registro	**Offset**	**Nombre**	**Descripción**
0x3B	0	ACCEL_XOUT_H	AccelX High
0x3C	1	ACCEL_XOUT_L	AccelX Low
0x3D	2	ACCEL_YOUT_H	AccelY High
0x3E	3	ACCEL_YOUT_L	AccelY Low
0x3F	4	ACCEL_ZOUT_H	AccelZ High
0x40	5	ACCEL_ZOUT_L	AccelZ Low
0x41	6	TEMP_OUT_H	Temp High
0x42	7	TEMP_OUT_L	Temp Low
0x43	8	GYRO_XOUT_H	GyroX High
0x44	9	GYRO_XOUT_L	GyroX Low
0x45	10	GYRO_YOUT_H	GyroY High
0x46	11	GYRO_YOUT_L	GyroY Low
0x47	12	GYRO_ZOUT_H	GyroZ High
0x48	13	GYRO_ZOUT_L	GyroZ Low

Tabla 2.27: Registros del MPU6050 [40].

Los valores que se reciben del acelerómetro y del giroscopio son valores brutos (valores sin procesar). El cálculo de los valores procesados se muestra en la ecuación (2.15), donde VB es el valor bruto o no procesado.

$$Valor = VB/S \tag{2.15}$$

Los límites de aceleración con los que trabaja el MPU6050 se muestran en la *Tabla 2.28*, se muestra también el factor de sensibilidad para cada límite.

Límte de aceleración	**Sensibilidad**
2g	16384
4g	8192
8g	4096
16g	2048

Tabla 2.28: Límites de aceleración del MPU6050 [40].

El rango por defecto del acelerómetro del MPU6050 es de -2g a 2g, por lo que su límite de aceleración es de 2g, lo que corresponde a un valor de sensibilidad igual a 16384. El cálculo de los valores de aceleración (por eje) se calcula mediante la ecuación (2.16).

$$a_{x,y,z} = VB_{x,y,z}/16384 \tag{2.16}$$

Lo límites de aceleración angular con los que trabaja el MPU6050 se muestran en la *Tabla 2.29*, mostrando, igualmente el factor de sensibilidad.

Límte de velocidad angular	**Sensibilidad**
250°/s	131
500°/s	65.5
1000°/s	32.8
2000°/s	16.4

Tabla 2.29: Límites de velocidad angular del MPU6050 [40].

El rango por defecto del giroscopio del MPU6050 varía entre -250°/s y 250°/s. Su límite de velocidad angular es de 250°/s, lo que corresponde a un factor de sensibilidad igual a 131. El cálculo de los valores de velocidad angular se calcula mediante la ecuación (2.17).

$$\omega_{x,y,z} = VB_{x,y,z}/131 \tag{2.17}$$

El sensor no siempre se encuentra en una posición horizontal al 100 %, por lo que se necesita una calibración para los 3 ejes. Se configura el módulo de los *offsets* para compensar los errores en la lectura, donde se intenta eliminar el valor del error para que el valor sea 0 en los valores del giroscopio y en los ejes x y y del acelerómetro. Para el eje z del acelerómetro se debe aproximar a 1. Durante el proceso de calibración, el sensor debe posicionarse horizontalmente y se mantenerse estático. Estos valores se quedan en los registros del MPU6050 para que, al realizar una nueva lectura, este valor sea el valor de referencia. Para escalar los valores, en la *Tabla 2.30* se muestran los valores mínimos, medios y máximos del giroscopio y acelerómetro, así como para las lecturas del MPU6050.

Variable	**Valor mínimo**	**Valor medio**	**Valor máximo**
Lectura MPU6050	-32768	0	32767
Aceleración	-2g	0g	2g
Velocidad angular	-250°/s	0°/s	250°/s

Tabla 2.30: Escala de valores del MPU6050 [40].

Si se considera que el MPU6050 se encuentra estático, los valores de la aceleración son la fuerza de gravedad en sus tres ejes. La gravedad siempre es vertical, por lo que los ángulos de la resultante son la inclinación del plano del sensor. El cálculo de estos ángulos, son entonces calculados en las ecuaciones (2.18) y (2.19).

$$\alpha_x = arctan(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}) \tag{2.18}$$

$$\alpha_y = arctan(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}) \tag{2.19}$$

Los ángulos se dan en radianes, por lo que en las ecuaciones (2.20) y (2.21) se obtiene el resultado del ángulo en grados.

$$\alpha_x = arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right)\left(\frac{180}{\pi}\right) \tag{2.20}$$

$$\alpha_y = arctan\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right)\left(\frac{180}{\pi}\right) \tag{2.21}$$

El valor de estos ángulos es eficiente sólo si la aceleración se mantiene en 9.8 m/s^2. Por lo que, al realizar algún movimiento en el MPU6050, se generan errores en la lectura.

Para el cálculo de los ángulos de la velocidad angular se necesita la integración de la velocidad y el valor inicial de lectura como se muestra en las ecuaciones (2.22) y (2.23).

$$\theta_x = \theta_{x0} + \omega_x \Delta t \tag{2.22}$$

$$\theta_y = \theta_{y0} + \omega_y \Delta t \tag{2.23}$$

Se define una velocidad de $100\mu s$ para una mayor velocidad de lectura con un delay igual a 0.1. El cálculo de los ángulos de rotación son los mostrados en las ecuaciones (2.24) y (2.25).

$$\theta_x = \theta_{x0} + .0001\omega_x \tag{2.24}$$

$$\theta_y = \theta_{y0} + .0001\omega_y \tag{2.25}$$

Al realizar las medidas de los ángulos se genera un error denominado *DRIFT*, este error es causado por una mala medición del tiempo o por el ruido de lectura del MPU6050. El error es pequeño pero con el tiempo se va acumulando y creciendo

en cada lectura, por lo que únicamente es útil en tiempo cortos. Para disminuir este error se crea un filtro complemento que integra los ángulos del acelerómetro y giroscopio. El filtro complemento consiste en aplicar un filtro pasa bajos al ángulo de inclinación para amortiguar las variaciones bruscas de aceleración y un filtro pasa altos para el ángulo de rotación para las rotaciones rápidas. Estos filtros se aplican en las ecuaciones (2.26) y (2.27), donde 0.98 y 0.02 son variables que pueden cambiar dependiendo de la respuesta del filtro. Estas variables siempre deben sumar 1.

$$\theta_x = 0.98(\theta_x + .0001\omega_x) + 0.02(\alpha_x) \tag{2.26}$$

$$\theta_y = 0.98(\theta_y + .0001\omega_y) + 0.02(\alpha_y) \tag{2.27}$$

Para la conexión de Bluetooth serial en Arduino IDE se trabaja con la librería BluetoothSerial.h. En la librería se crea un objeto tipo BluetoothSerial para que se inicie posteriormente la transmisión de datos al puerto serie de la computadora.

2.2.3. Módulo de clasificación

El proceso para la clasificación de los datos se muestra en la *Imagen 2.22.*

Imagen 2.22: Proceso de clasificación.

El proceso de clasificación comienza con la obtención de datos de las lecturas en Arduino IDE de todos los sensores para ser almacenadas en un archivo .csv.

Se obtienen las lecturas analógicas de los 12 sensores flex y del hilo conductivo. Los valores de estas lecturas son los *input* de la RNA, los cuales deben ser lo más cercanos a 0 para un mejor funcionamiento. Los valores de las lecturas analógicas varían en un rango de 0 a 4095, por lo que en la ecuación (2.28) se muestra el cálculo de los valores correspondientes a los sensores flex y en la ecuación (2.29) el valor correspondiente al hilo conductivo.

$$S[n+1] = analogRead[n]/4095 \quad \forall\ n = 0, 1, .., 10 \tag{2.28}$$

$$H = analogRead[11]/4095 \tag{2.29}$$

Donde *analogRead* es un vector de las 12 lecturas analógicas.

Para las lecturas del MPU6050, se obtienen los ángulos x y y de rotación con el filtro complementario. Los ángulos varían generalmente en un rango de -100° a 100°, puesto que la RNA no trabaja con números negativos, en la ecuación (2.30) se muestra el cálculo de los valores correspondientes a los ángulos de rotación.

$$M[n]_{x,y} = (angulo_{x,y} + 100)/200 \quad \forall\ n = 1, 2 \tag{2.30}$$

Con la lectura de los MPU6050, se obtiene otro dato que corresponde a la identificación del cambio de posición de los sensores. Este valor se obtiene en la ecuación (2.31) de la resta de una lectura inicial y una lectura realizada pasados 300 ms definidos con un delay igual a 300. Si el valor absoluto de la resta excede 15°, el valor obtenido es 1, indicando un cambio de posición en alguno de los ejes. Si no hay cambio de ángulo de rotación se obtiene 0.

$$Mov = \begin{cases} 0 & ,|M[n]_{x,y}t(0) - M[n]_{x,y}t(300)| < 15 \\ 1 & ,|M[n]_{x,y}t(0) - M[n]_{x,y}t(300)| > 15 \end{cases} \tag{2.31}$$

La suma de la lectura de los 12 sensores flex, la lectura de los ángulos x y y de los dos MPU650, la lectura del hilo conductivo y la lectura de la detección del

movimiento da entonces, un total de 18 datos. Para la transferencia de los datos de Arduino IDE a Excel (creación del archivo .csv), se utiliza una interfaz en Java. La interfaz muestra los datos leídos del puerto serie de Arduino en una tabla, donde se puede exportar a un archivo de Excel. Para la transferencia de datos se utiliza la librería PanamaHitek_Arduino, versión 3.1.0, diseñada especialmente para facilitar la comunicación de Arduino y Java a través del puerto serie. La librería incluye 3 clases principales [32]:

- PanamaHitek_Arduino: encargada de manejar todas las conexiones y la comunicación con Arduino.
- PanamaHitek_MultiMessage: incluye las herramientas necesarias para recibir múltiples mensajes de forma simultánea en Java.
- PanamaHitek_DataBuffer: almacena datos de forma ordenada, permite la visualización en una tabla y la exportación de datos a Excel.

Se utiliza Netbeans IDE para la programación de la interfaz. Se crea un objeto para cada una de las clases principales y en el objeto de almacenamiento de datos (buffer) se agregan el nombre de las columnas para el almacenamiento de esos datos. Se tiene un total de 21 columnas, donde 18 corresponden a los datos de los valores de las lecturas de los sensores, otra columna corresponde al tiempo (hora en la que se realiza la lectura), otra corresponde a la letra y otra a la etiqueta de esta letra. Para realizar las pruebas se programa en Arduino IDE, una lectura cada segundo de la seña de la letra a realizar. Esta lectura se repite 5 veces, para después pasar a la siguiente lectura de la seña dando un espacio de tiempo de 3 segundos y permita el cambio. Son en total 62 señas, correspondientes a las 27 letras del alfabeto, los 12 meses del año, los 7 días de la semana, los números del 1 al 10 y las 6 palabras básicas. Por lo que, el tiempo de una prueba es de $(5s + 3s)(62) = 496s$, lo que corresponde a 8.26 minutos. En el *Apéndice 3* se muestra el código correspondiente a la programación en Netbeans IDE para la exportación de los datos a Excel.

Para ejecutar la librería de Keras se utiliza la plataforma de código abierto de TensorFlow, en donde esta librería ya esta configurada. Para codificar en la plataforma

TensorFlow se utiliza Colaboratory (también llamado Colab), que es un entorno gratuito de desarrollo de software de Google, que permite a los usuarios la programación en Python sin tener que instalar y configurar TensorFlow, pues ya está preinstalado. La versión de TensorFlow tiene que ser la misma que se utiliza en Arduino IDE, que es la versión 2.2.0.

Los datos para el entrenamiento de la RNA son almacenados en el archivo .csv; para leer los datos del archivo se requiere la librería 'pandas'. Las pruebas están divididas en los valores de los 18 datos y el número de etiqueta.

Para la fila correspondiente a los resultados de la etiqueta se propone implementar la estrategia "One Hot Encoding", la cual crea una columna para cada prueba realizada y para cada resultado de éstas, agrega un 1 a la columna a la que corresponde dicho resultado y las demás agrega un 0. La estrategia "One Hot Encoding" se implementa de la siguiente manera: la fila correspondiente a los valores de la etiqueta varían entre 0 y 62, por lo que se crea un vector de ceros con un tamaño de 62 y dependiendo de el número de etiqueta se le asigna un 1 a la posición del vector igual al de la etiqueta.

Para el entrenamiento y evaluación de la RNA se propone trabajar con un 20 % de los datos aleatoriamente. Para la definición del modelo se implementa un modelo secuencial (Sequential), que es el modelo de Keras que permite agregar capas de manera secuencial. Para las entradas de la primer capa se ocupan 18 datos (por los datos de los sensores), los cuales son definidos por **input_dim**. Se definen tres capas en la red, para evaluar el desempeño de ésta.

La clase Dense, define el número de neuronas en el primer argumento, el método de inicialización en el segundo y la función de activación en el tercero. Para la primera y segunda capa se define una función de activación **relu** y para la tercer capa una función **sigmoid**. El número de neuronas definido es de 16. El diagrama del modelo de la RNA es el mostrado en la *Imagen 2.23*.

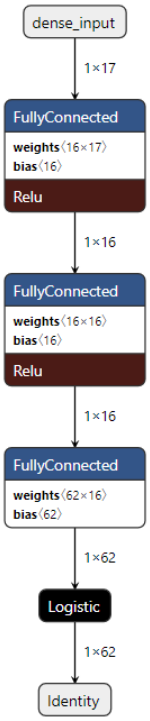


Imagen 2.23: Diagrama del modelo de la RNA.

En redes neuronales, una época es el número de iteración realizada para ajustar los pesos. Se asigna un número de 1500 épocas. Para la validación del entrenamiento se realiza una simulación de la RNA con 100 épocas, lo que genera un resultado de 100 % de exactitud. La pérdida en el entrenamiento se muestra en la *Imagen 2.24*.

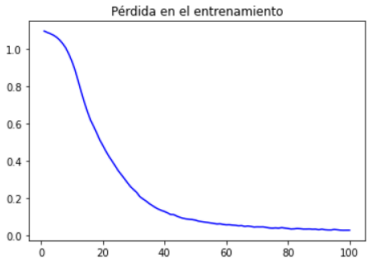


Imagen 2.24: Gráfica de pérdida en el entrenamiento.

Los datos de la RNA ya entrenada en TensorFlow se convierten a datos para TensorFlow Lite, el cual es una herramienta de TensorFlow para microcontroladores

y otros dispositivos. Esta herramienta permite ejecutar los modelos de aprendizaje profundo en un espacio de memoria en kilobytes y requiere una plataforma de 32 bits, la cual cumple la tarjeta ESP32 con programación en Arduino IDE.

Para trabajar con TensorFlow Lite en Arduino IDE, el modelo de la red ya entrenada se codifica a valores hexadecimales y se guardan en un archivo .cc para guardarlo en el archivo de Arduino. El código en Colab de todo el modelo de la RNA hasta la codificación a hexadecimal se muestra en el *Apéndice 4*.

Para la programación en Arduino, se necesita la librería de TensorFlow Lite que Arduino ya tiene preinstalada y editar la programación en el folder: arduino_output_handler.cpp, ya que, inicialmente estaba programada para la tarjeta de Arduino Nano.

Para realizar el proceso de traducción de TensorFlow Lite se requiere reservar memoria para pasar el modelo al microcontrolador, ya que, se cuenta con una memoria pequeña. En un paso posterior se asignan variables pero no se le asignan valores, ya que este, es el siguiente paso, asignar el valor de una entrada, después de este paso, se invoca al modelo para que finalmente se obtenga el valor de salida. En la *Imagen 2.25* se muestra este proceso.

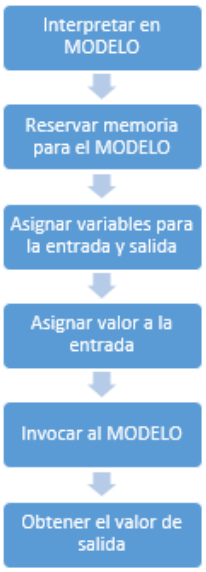


Imagen 2.25: Diagrama "Proceso de traducción en TensorFlow Lite" [49].

Las especificaciones recomendadas por TensorFlow Lite para el microcontrolador son:

- Arquitectura de 32-bit
- Mínimo 50 Mhz
- Mínimo 100 kB RAM

Estas especificaciones se cumplen con la tarjeta ESP32.

2.2.4. Módulo de comunicación con el usuario

Para la comunicación con el usuario se utiliza una interfaz gráfica donde se muestra la traducción; el diagrama de alto nivel de la ventana principal se muestra en la *Imagen 2.26*. Dentro de la ventana principal se encuentran botones para iniciar, pausar y borrar el contenido. La traducción se muestra en un recuadro grande y a lado se muestra la imagen de la seña correspondiente a dicha traducción. Para una visualización de los datos que van siendo adquiridos de los sensores se utiliza un botón que al seleccionarlo se dirige a otra ventana donde muestra estos datos. También se utiliza un botón para dirigir a la ventana de validación de la letra a identificar.

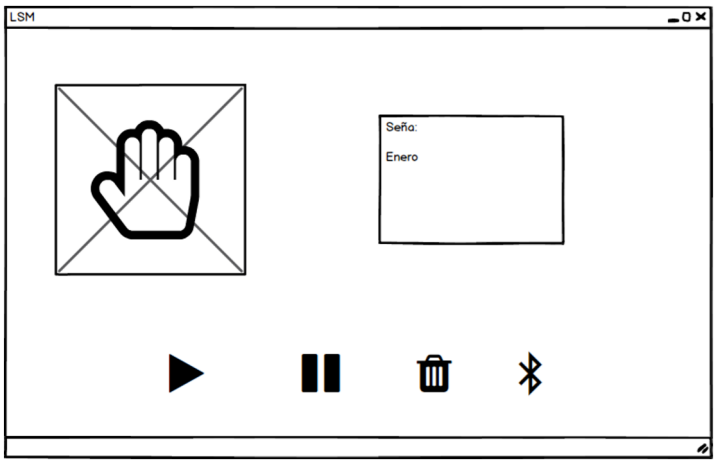


Imagen 2.26: Ventana principal.

La programación de la interfaz se realiza en Netbeans IDE. Para la lectura de los datos se utiliza nuevamente la librería PanamaHitek_Arduino (utilizada para

la exportación de Arduino a Excel). En este caso se utiliza solamente las clases de Arduino y MultiMessage, donde se crea un objeto para cada una. Para la lectura de los datos se utiliza un *EventListener* del puerto serie. La visualización de la ventana principal se muestra en *Imagen 2.27*.

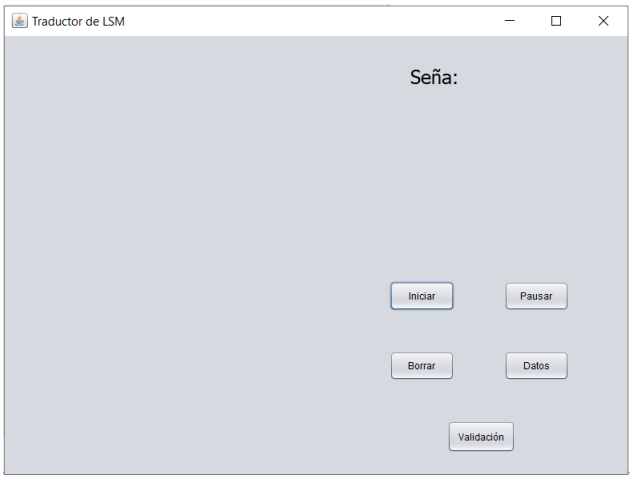


Imagen 2.27: Visualización de ventana principal.

En la *Imagen 2.28* se muestra el diagrama de alto nivel correspondiente a la ventana donde se visualizan que datos están entrando. En esta ventana se tiene un botón para eliminar el contenido.

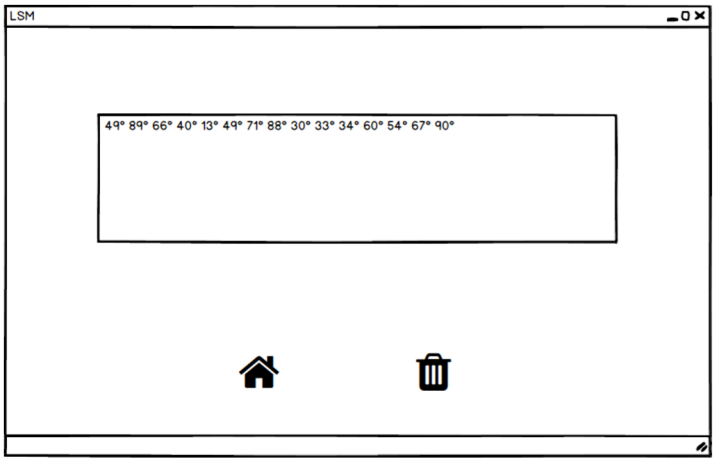


Imagen 2.28: Ventana de datos entrantes.

Los datos leídos se almacenan en un tabla, en la cual se leen los 18 datos de entrada y la traducción de la seña. La ventana principal se puede visualizar al mismo tiempo, así como realizar alguna interacción (por ejemplo, pausar). Si se desea visualizar únicamente la ventana principal, se cierra la ventana de lectura. En la *Imagen 2.29* se muestra la visualización de esta ventana.

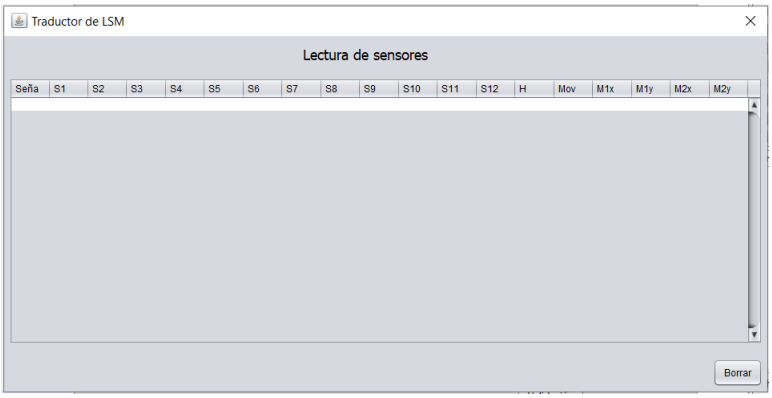


Imagen 2.29: Visualización de ventana de datos entrantes.

Por último se visualiza en la *Imagen 2.30* la ventana correspondiente a la validación de las señas realizadas. En ella se muestra un cuadro (*jTextField*) donde se escribe la seña que se quiere validar. Al realizarse la lectura correcta, aparece un mensaje con la indicación que la lectura ha sido correcta, de lo contrario, aparece una ventana emergente (*jOptionPnae*) que indica un error en la lectura. Esta última ventana puede aparecer si la seña que se realiza no coincide con la seña requerida, si no se realiza ninguna seña o el programa está en pausa.

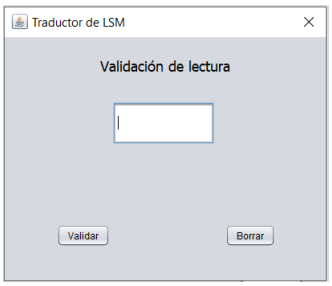


Imagen 2.30: Ventana de validación.

2.2.5. Módulo de suministro de energía

La batería 18650 de Li-ion tiene las características mencionadas en la *Tabla 2.31* para alimentar la tarjeta ESP32.

Especificaciones técnicas
Capacidad nominal de la batería: 3000 mAh
Composición: Li-ion
Peso: >60g
Voltaje nominal: 3.7V
Dimensiones: 69.2 mm x 18.3 mm
Temperatura de operación: 0 a 50°C

Tabla 2.31: Especificaciones técnicas de la batería 18650 [1].

La conexión con la ESP32 se realiza mediante una tarjeta de batería portable 18650 V3. En la *Imagen 2.31* se muestran las conexiones a esta tarjeta. El puerto USB es la salida de alimentación para la tarjeta ESP32 y la entrada Micro USB es la entrada para el cable de carga de la tarjeta misma. Adicionalmente cuenta con salidas de voltaje de 3V y 5V (3 salidas para cada voltaje).

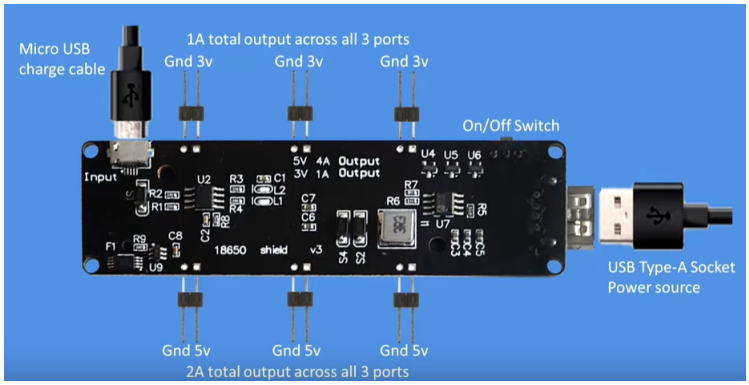


Imagen 2.31: Battery shield 18650 para ESP32 [1].

Los componentes en la tarjeta de la batería se muestran en la *Imagen 2.32* y en la *Tabla 2.32* se indica la descripción de estos componentes.

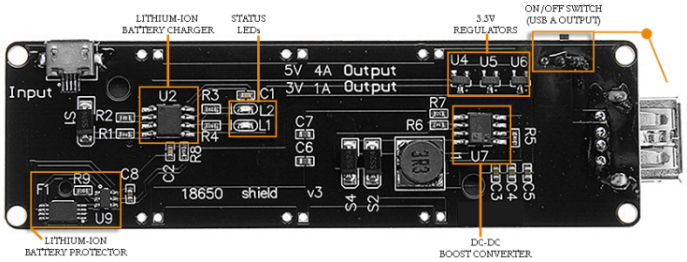


Imagen 2.32: Componentes en tarjeta de batería [11].

Componente	**Descripción**
U9 = DW01V	1S Li-Ion battery protection chip
F1 = 8205A	Dual N-Channel MOSFET
U2 = TC4056A	1S Li-Ion battery charger chip
U7 = FP6298	4.5A current mode dc-dc boost converter chip
U4, U5, U6 = 662K (XC6206xxxx)	Positive voltage regulator chip (3.3V)
L1 = Green LED	Battery CHGD indicator
L2 = Red LED	Battery CHRG indicator
S1, S2, S4 = Schottky Diodes	S1 = SS14 and S2, S4 = SS24

Tabla 2.32: Componentes en la tarjeta de batería [11].

En la *Imagen 2.33* e *Imagen 2.34* se muestra el diseño de la caja para la tarjeta de batería (*Apéndice 6*).

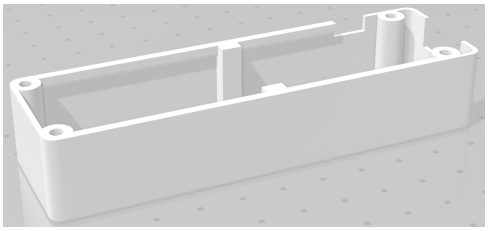


Imagen 2.33: Parte inferior de la caja para la tarjeta de batería [51].

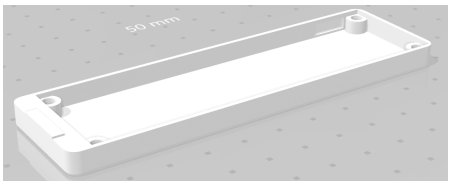


Imagen 2.34: Parte superior de la caja para la tarjeta de batería [51].

En la *Imagen 2.35* e *Imagen 2.36* se muestra el diseño de la caja para la tarjeta ESP32 (*Apéndice 6*).

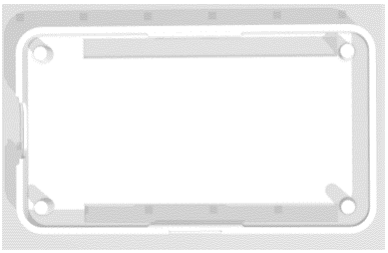


Imagen 2.35: Parte inferior de la caja para la tarjeta ESP32 [52].

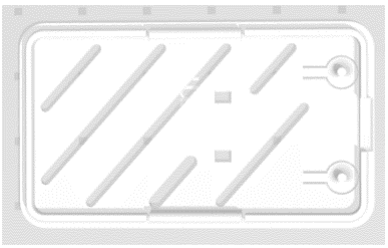


Imagen 2.36: Parte superior de la caja para la tarjeta ESP32 [52].

2.2.6. Integración sistema mecatrónico

La integración del sistema mecatrónico se muestra en la *Imagen 2.37.* donde se muestra la integración de cada uno de los módulos.

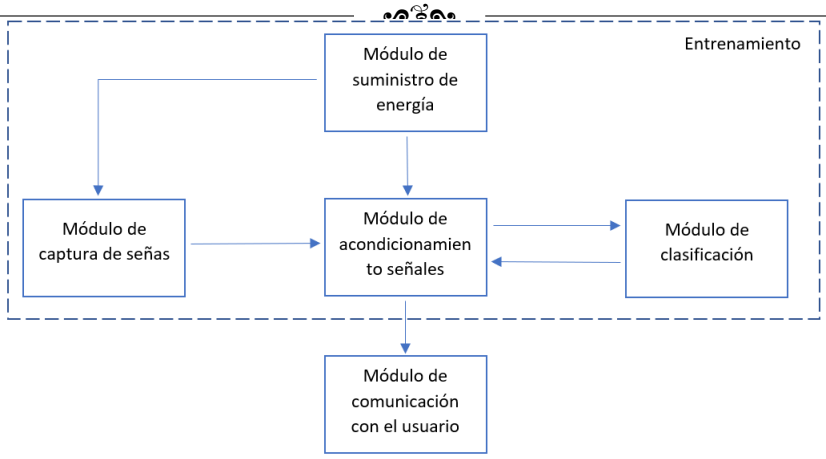


Imagen 2.37: Integración de los módulos en el sistema.

La fase de entrenamiento se inicia con el módulo de captura de señas, módulo donde se trabajan los componentes electro-mecánicos que ayudan a la medición de flexión y orientación del brazo. Para una correcta lectura, se trabaja en conjunto con el módulo de acondicionamiento de señales, que es el módulo que permite procesar estos datos; en este módulo se trabaja con los protocolos de comunicación correspondientes para cada componente. El microcontrolador es el medio para realizar todo el proceso. El siguiente módulo es el que hace la clasificación de los datos resultantes del proceso en el microcontrolador. En este módulo se trabaja con la arquitectura del microcontrolador, la arquitectura de software y el aprendizaje automático. El módulo de suministro de energía es el que alimenta a los primeros dos módulos.

Una vez terminada la fase de entrenamiento, se integran los módulos de clasificación y de acondicionamiento de señales, donde el microcontrolador trabaja directamente con el modelo de la RNA. Esta parte del proceso se realiza de forma iterativa, integrando inicialmente el módulo de captura de señas, el módulo de acondicionamiento de señales y el módulo de clasificación se integran simultáneamente y se finaliza integrando el modulo de comunicación con el usuario, que es el que tiene control de todos los módulos anteriores para su visualización. El módulo de suministro de energía sigue siendo el que alimenta a los primeros dos módulos. Sin alguno de los cinco módulos el sistema no cumple con su función.

Para la validación de la integración de todos los módulos se utilizó la validación del entrenamiento simulada para las letras A, B y C únicamente, la cual tuvo una exactitud del 100 % con 100 épocas; se muestra la pérdida de entrenamiento en la *Imagen 2.24*. Una vez teniendo el modelo, se generó en un archivo .cc, el cual contiene el modelo en codificación hexadecimal para que Arduino pueda trabajar con éste. En Arduino, se siguieron los pasos del proceso de la *Imagen 2.25*. Para la visualización de los resultados interpretados por Arduino, se diseñó y programó una interfaz, donde se visualizaran las imágenes de la seña que se estuviera realizando junto con la letra que el modelo estuviera clasificando (*Imagen 2.38*, *Imagen 2.39* e *Imagen 2.40*). Los botones de iniciar, pausar, borrar y datos fueron únicamente puestos para el diseño de la interfaz final.

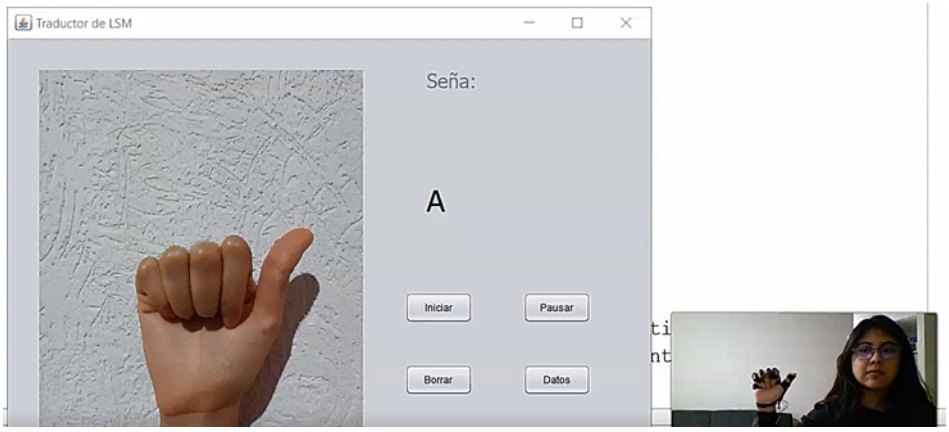


Imagen 2.38: Letra A mostrada en interfaz.

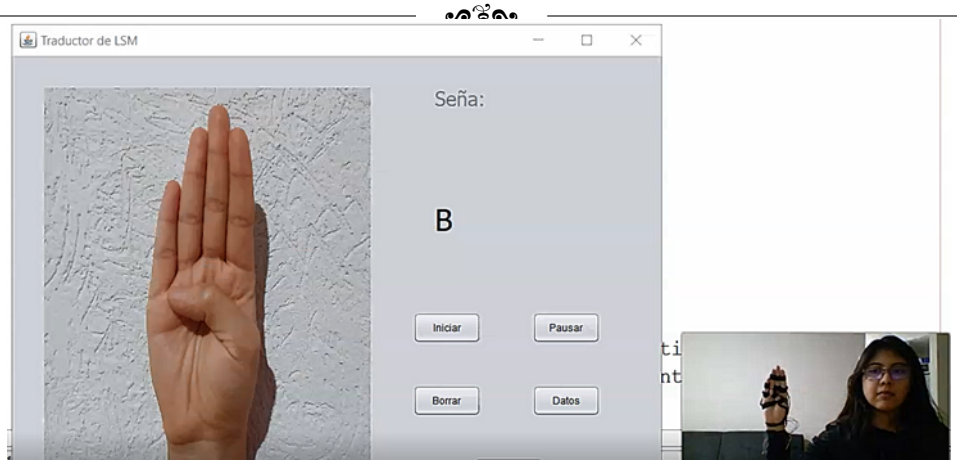


Imagen 2.39: Letra B mostrada en interfaz.

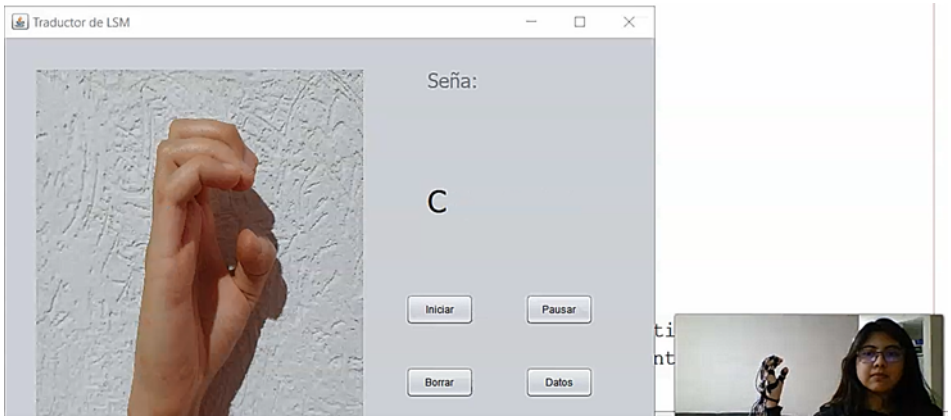


Imagen 2.40: Letra C mostrada en interfaz.

CAPÍTULO 3

Implementación del sistema

3.1. Módulo de captura de señas

En la *Imagen 3.1* se muestra la estructura de Sintra PVC previamente moldeada con pistola de calor sobre un molde de mano de yeso, con los 10 sensores flex acomodados de acuerdo a las medidas de la *Imagen 2.7*, ajustados con las medidas dadas para las tiras de velcro. Sobre la estructura también se muestran los soportes para los sensores flex.

Imagen 3.1: Estructura de la mano con los 10 sensores colocados.

Para validar que los sensores flex se encuentran colocados correctamente como se indica en las medidas de la *Imagen 2.7*, se realiza una prueba de medición.

Prueba de medición de sensores flex.

Descripción: Validación de las medidas correspondientes a la *Imagen 2.7*.
La prueba consiste en medir con una regla, la posición de los sensores flex respecto a su colocación sobre el velcro, estas medidas deben coincidir con una tolerancia de ± 10 % de las medidas correspondientes por cada sensor. Las medidas se colocan en la *Tabla 3.1* para su comparación con las medidas esperadas.
Resultado: Todas las medidas entran en el rango de tolerancia.

	Valor medido	**Valor esperado**
S1	1.7 cm	1.69 cm
S2	1.1 cm	1.12 cm
S3	0.5 cm	0.54 cm
S4	0.7 cm	0.69 cm
S5	0.15 cm	0.14 cm
S6	0.45 cm	0.47 cm
S7	0.35 cm	0.39 cm
S8	0.7 cm	0.74 cm
S9	0.85 cm	0.94 cm
S10	1.1 cm	1.17 cm

Tabla 3.1: Prueba de medición de sensores flex.

El hilo conductivo está cosido en los velcros de la parte de superior de los dedos índice y medio para la verificación del contacto entre estos mismos. El sensor MPU6050 correspondiente al del dorso de la mano, se coloca sobre los soportes de los sensores flex. Adicionalmente se colocan tiras de velcro para el ajuste de la parte del dispositivo en la mano y muñeca.

Sobre el brazo se coloca una cinta elástica ajustable con velcro, donde el sensor MPU6050 correspondiente está cosido, así como la tarjeta ESP32 acomodada para

una correcta lectura de los 15 sensores. Para verificar de que se estén tomando correctamente las medidas de todos los sensores, se realiza una prueba cuando la mano realiza algún movimiento.

Prueba de medición de los 15 sensores.

Descripción: Validación de las medidas de los sensores dentro de su rango.

La prueba consiste en realizar algún movimiento con el dispositivo puesto. La programación de lectura de los 15 sensores se realiza en Arduino, donde se muestra los valores de resistencia de los sensores flex, los tres ejes de aceleración y los tres del giroscopio con su correcta calibración de los MPU6050 y el valor analógico del hilo conductivo. Los valores de la resistencia de los sensores flex deben variar en un rango de 25kΩ hasta 70kΩ, que es cuando se encuentra el sensor totalmente doblado. Para la lectura del MPU6050, la unidad de medida de la aceleración es de m/s^2, la cual varía en valores positivos entres 0 y 1 para un movimiento de la mano. Y la medida del giroscopio varía en función de que tan rápido se realiza un giro en la mano, varía en valores tanto negativos como positivos, generalmente en un rango de -30 y 50. El hilo conductivo genera un 0 al no tener contacto entre dedos y un 1 al tener contacto.

Resultado: Los resultados de cuatro mediciones de esta prueba, se muestran a continuación, donde las resistencias R1 a R12 corresponden a los sensores S1 a S12 respectivamente. Para los resultados de los MPU6050, los primeros dos dígitos son del número de sensor (M1 o M2), el tercer dígito indica A y G, mediciones del acelerómetro y giroscopio respectivamente y el último dígito indica el eje (x,y,z). Los valores de cada sensor que se muestran en las cuatro pruebas entran en el rango de mediciones esperado.

```
R1: 28393       R2: 27297       R3: 28093       R4: 34660       R5: 26313       R6: 28522
R7: 26993       R8: 29462       R9: 28942       R10: 28427      R11: 26767      R12: 29035
M1Ax : 0.91     M1Ay : 0.01     M1Az : 0.30
M1Gx : 11.85    M1Gy : 29.52    M1Gz : 10.65
M2Ax : -0.24    M2Ay : 0.93     M2Ay : 0.12
M2Gx: -14.83    M2Gy : 6.26     M2Gz : -2.45
Hilo : 1
R1: 28522       R2: 27432       R3: 28359       R4: 34625       R5: 26286       R6: 28495
R7: 26407       R8: 28557       R9: 28536       R10: 27769      R11: 26784      R12: 28829
M1Ax : 0.87     M1Ay : 0.02     M1Az : 0.45
M1Gx : 1.04     M1Gy : 7.81     M1Gz : 0.43
M2Ax : -0.32    M2Ay : 0.87     M2Ay : 0.26
M2Gx: -9.29     M2Gy : 0.96     M2Gz : -5.66
Hilo : 1
```

```
R1: 27512       R2: 26993       R3: 25861       R4: 33852       R5: 24485       R6: 27593
R7: 32648       R8: 25939       R9: 25887       R10: 26412      R11: 26563      R12: 29357
M1Ax : 0.94     M1Ay : 0.09     M1Az : 0.33
M1Gx : -0.68    M1Gy : 1.84     M1Gz : 1.73
M2Ax : -0.30    M2Ay : 0.93     M2Ay : 0.13
M2Gx: -3.88     M2Gy : 2.96     M2Gz : 1.78
Hilo : 0
R1: 27693       R2: 27017       R3: 26029       R4: 34071       R5: 24614       R6: 27731
R7: 30175       R8: 26579       R9: 26173       R10: 26658      R11: 26568      R12: 29372
M1Ax : 0.94     M1Ay : 0.08     M1Az : 0.32
M1Gx : 1.16     M1Gy : -3.06    M1Gz : -1.19
M2Ax : -0.29    M2Ay : 0.92     M2Ay : 0.13
M2Gx: -1.18     M2Gy : 0.02     M2Gz : 3.41
Hilo : 0
```

3.2. Módulo de acondicionamiento de señales

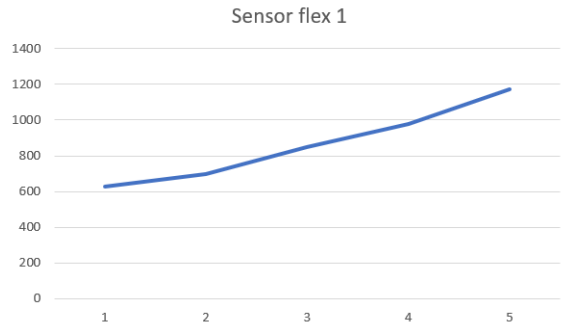
La implementación del módulo de acondicionamiento de señales consiste en la lectura de los sensores y su transmisión. Se realiza una prueba correspondiente a los sensores flex, donde se obtienen las gráficas de comportamiento.

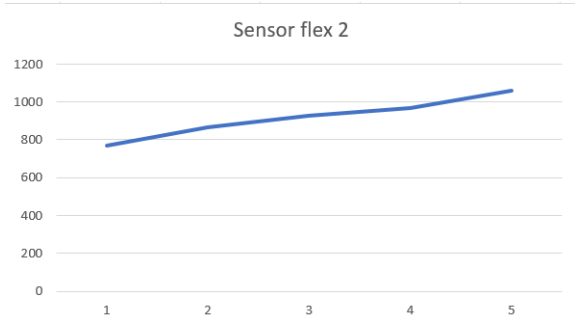
Prueba de gráficas de comportamiento de sensores flex.

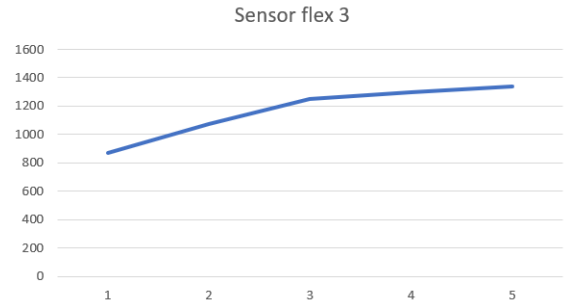
Descripción: Verificación del comportamiento de sensores flex de la mano.
La prueba consiste en realizar 5 flexiones para cada dedo, donde se registren las lecturas de los sensores flex. Las 5 flexiones varían desde el dedo completamente estirado hasta el dedo completamente flexionado. Se registra el valor máximo y mínimo para cada sensor, dependiendo de la flexión de cada dedo con el dispositivo puesto.
Resultado: En la *Tabla 3.2* se registran las 5 lecturas, así como sus valores mínimos y máximos para cada sensor. Las gráficas de comportamiento para cada sensor se muestran a continuación.

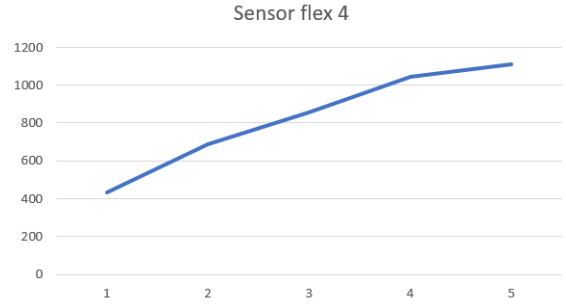
S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**
625	767	872	432	446	765	318	705	539	455
700	865	1075	687	576	923	453	746	759	784
847	926	1250	854	807	1133	625	863	896	979
977	970	1298	1043	1045	1163	779	1069	1220	1107
1175	1062	1338	1109	1124	1184	992	1209	1424	1221

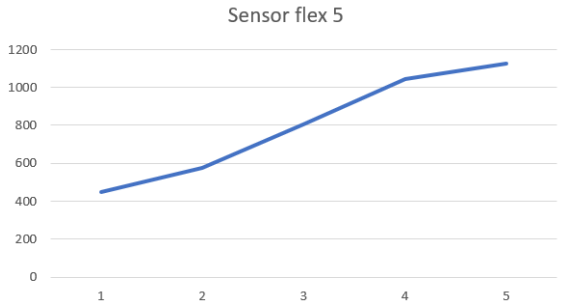
Tabla 3.2: Lectura de 5 flexiones en sensores flex.

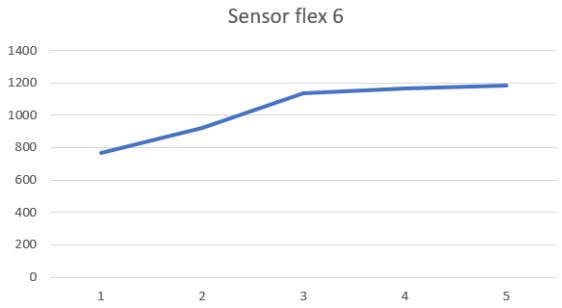

Sensor flex 1
1400
1200
1000
800
600
400
200
0
1
2
3
4
5

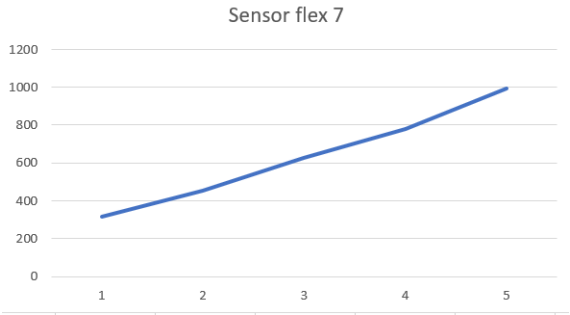

Sensor flex 2
1200
1000
800
600
400
200
0
1
2
3
4
5

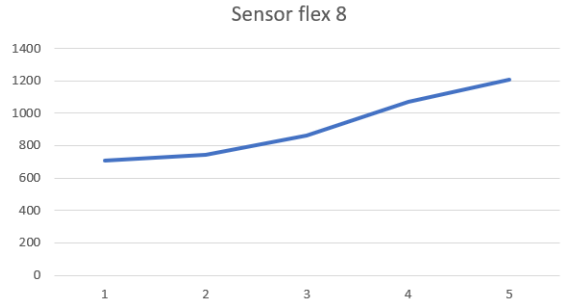

Sensor flex 3
1600
1400
1200
1000
800
600
400
200
0
1
2
3
4
5

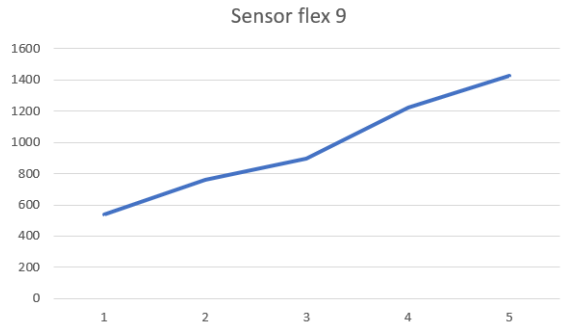

Sensor flex 4
1200
1000
800
600
400
200
0
1
2
3
4
5

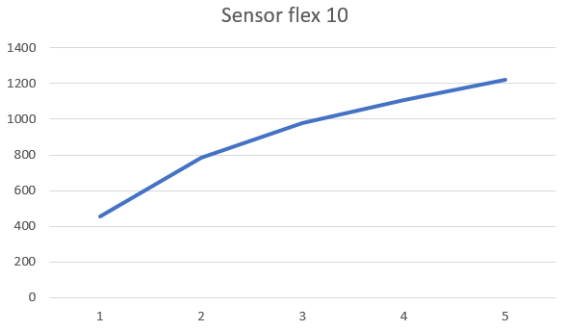

Sensor flex 5
1200
1000
800
600
400
200
0
1
2
3
4
5


Sensor flex 6
1400
1200
1000
800
600
400
200
0
1
2
3
4
5


Sensor flex 7
1200
1000
800
600
400
200
0
1
2
3
4
5


Sensor flex 8
1400
1200
1000
800
600
400
200
0
1
2
3
4
5





Para la verificación de los ángulos de rotación, se realiza una prueba con 4 configuraciones distintas de la mano.

Prueba de ángulos de rotación en los MPU6050.

Descripción: Verificación de los ángulos x y y de rotación medidos en el MPU6050. La prueba consiste en realizar 4 configuraciones de la mano correspondientes a la *Imagen 1.1*, donde se verifica que los ángulos cambien dependiendo del eje en el que se muevan.

Resultado: En la *Tabla 3.3* se registran los valores de los ángulos por número de configuración.

Configuración	7	5	8	6
M1X	84.39	79.115	45.5625	-0.38
M1Y	4.768	9.615	29.385	57.73
M2X	0.098	13.53	52.315	88.46
M2Y	-79.218	-75.915	-51.735	3.84

Tabla 3.3: Ángulos de rotación de configuraciones de la mano.

3.3. Módulo de clasificación

La implementación del módulo de clasificación se divide en tres secciones: la exportación de las lecturas a un archivo .csv, la generación del modelo de la RNA y la codificación del modelo en Arduino IDE. Las prueba realizada en este módulo corresponden a una clasificación únicamente de A,B y C para los 10 ángulos de los sensores de la mano.

Para la generación del archivo .csv, se implementa el mismo código del *Apéndice 3*, donde se genera la lectura de las 3 señas cada segundo, con un intervalo de 3 segundos cada 5 lecturas de la misma seña. Se obtiene un tamaño de pruebas de 587 lecturas.

Para la generación del modelo de la RNA, se programa en Colab, el mismo código utilizado en el *Apéndice 4*, el cual genera un modelo que se codifica en Arduino IDE.

Prueba del modelo de la RNA en Arduino IDE.

Descripción: Verificación de la codificación del modelo de la RNA con TensorFlow Lite en Arduino IDE.

La prueba consiste en la codificación del modelo de la RNA utilizando TensorFlow Lite, se tienen 10 pruebas de datos a evaluar en Arduino IDE, así como su etiqueta. El resultado de la RNA se muestra en un vector con tres datos, estos mismos corresponden a la codificación "One Hot Encoding", por lo que para su verificación se identifica el valor mayor en el vector y se compara con la etiqueta real. Si la posición del valor más grande coincide con el de la etiqueta real, se concluye que la traducción es correcta.

Resultado: En la *Imagen 3.2* se muestran los resultados de los vectores de salida de la RNA, así como su comparación con su etiqueta real. Se concluye que todas las clasificaciones son correctas.

```
Número de entradas: 10
Tipo: float
Número de salidas: 3
Tipo: float
|           RNA                      |           Valor real        |
|   0.00000     0.00020     0.00020  |    0.00    0.00    1.00     |
|   0.00000     0.00017     0.00003  |    0.00    1.00    0.00     |
|   0.00000     0.00005     0.00320  |    0.00    0.00    1.00     |
|   0.00000     0.00023     0.00000  |    0.00    1.00    0.00     |
|   0.88702     0.00029     0.00000  |    1.00    0.00    0.00     |
|   0.95970     0.00014     0.00000  |    1.00    0.00    0.00     |
|   0.00000     0.00027     0.00001  |    0.00    1.00    0.00     |
|   0.32643     0.00017     0.00000  |    1.00    0.00    0.00     |
|   0.00000     0.00000     0.94700  |    0.00    0.00    1.00     |
|   0.54462     0.00083     0.00000  |    1.00    0.00    0.00     |
```

Imagen 3.2: Resultados de prueba del modelo de la RNA.

Los valores correspondientes a la letra A: 1 0 0, letra B: 0 1 0, letra C: 0 0 1.

3.4. Módulo de comunicación con el usuario

Para la verificación del módulo de comunicación con el usuario se genera un código en Arduino IDE donde se visualiza una letra o palabra y 18 datos cada segundo.

Prueba de interfaz.

Descripción: Verificación de visualización de datos del puerto serie de Arduino a interfaz.

La prueba consiste en la visualización de las tres ventanas de la interfaz. En la ventana principal la verificación de la visualización de los datos al iniciar el programa, la función de borrar los datos al presionar el botón correspondiente, así como la verificación de los botones correspondientes para abrir las otras dos ventanas. En la ventana de lectura de datos, la verificación del botón para eliminar los datos y en la ventana de validación dos ejemplos de una validación correcta y otra incorrecta.

Los datos entrantes recorren las 62 señas en orden (alfabeto, meses, días, números y palabras) y 18 datos fijos codificados al azar.

Resultados:

Visualización de la interfaz principal al presionar el botón de iniciar:

Imagen 3.3: Interfaz principal.

Visualización de la interfaz principal al presionar el botón de borrar:

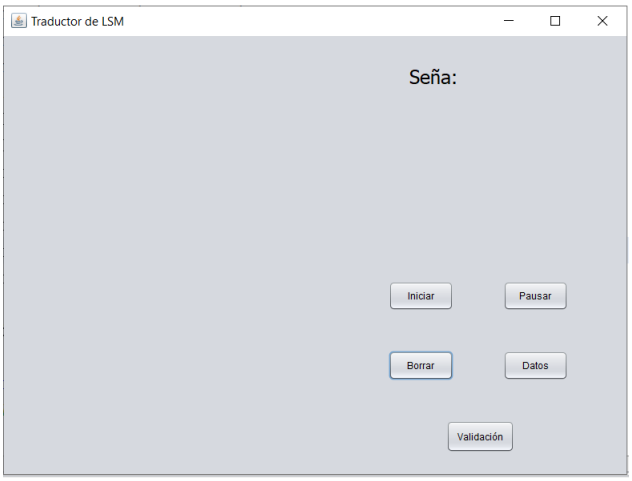


Imagen 3.4: Interfaz principal sin contenido.

Visualización de la ventana de lectura de datos (presionando el botón en la ventana principal):

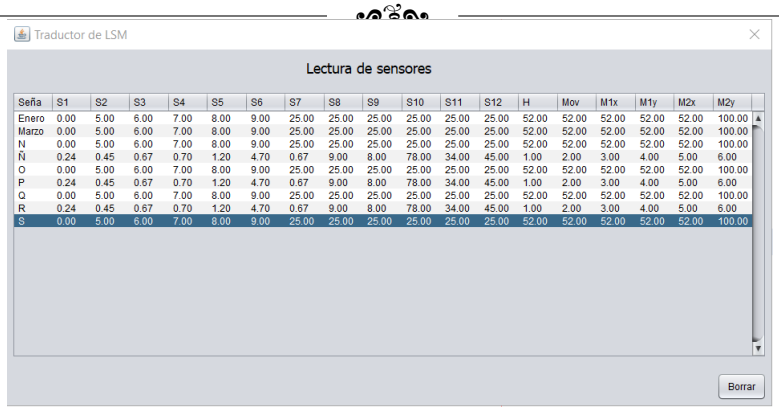


Imagen 3.5: Interfaz de lectura de datos.

Visualización de la ventana de lectura de datos al presionar el botón de borrar:

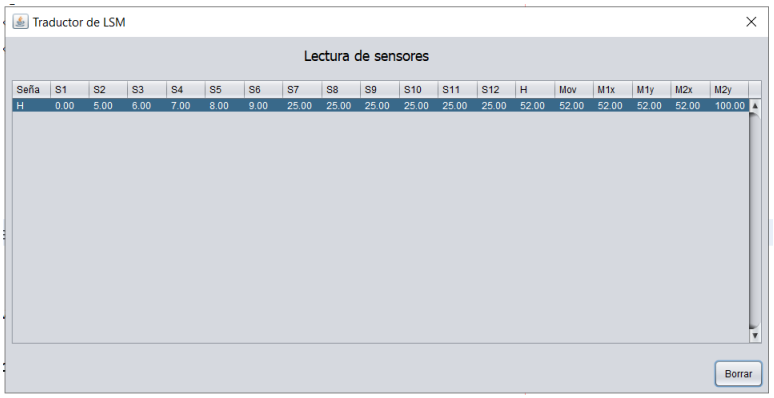


Imagen 3.6: Interfaz de lectura de datos sin contenido.

Visualización de la ventana de validación (presionando el botón en la ventana principal):

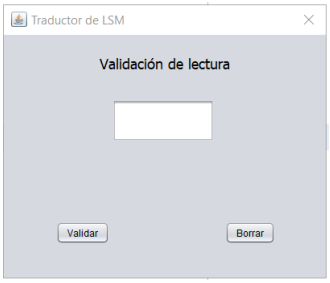


Imagen 3.7: Interfaz de validación.

Visualización de la ventana de validación al presionar el botón de validar y que la lectura es correcta:

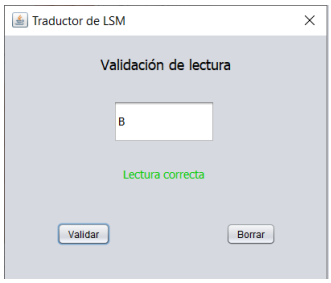


Imagen 3.8: Interfaz de validación con lectura correcta.

Visualización de la ventana de validación al presionar el botón de validar y que la lectura es incorrecta:

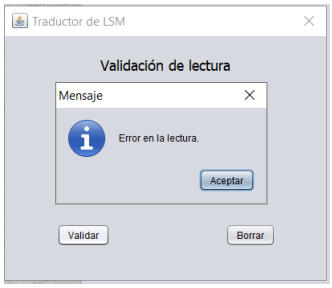


Imagen 3.9: Interfaz de validación con lectura incorrecta.

3.5. Módulo de suministro de energía

Para la verificación del módulo de suministro de energía se mide el voltaje de la tarjeta de batería con la batería 18650 cuando la batería se encuentra cargada (cuando el led indicador es verde).

Prueba de funcionamiento de tarjeta de batería 18650 V3.

Descripción: Verificación del voltaje de salida de la tarjeta de batería en la entrada de la tarjeta ESP32.

La prueba consiste en conectar la tarjeta de batería con la tarjeta ESP32. Se hace la

medición del voltaje con un multímetro desde la tarjeta ESP32, este voltaje debe ser un valor aproximado a 5V.

Resultados: En la *Imagen 3.10* se muestra la foto de la medida del multímetro al medirlo desde la tarjeta ESP32, la medida es de 4.67V, por lo que muestra que el voltaje es correcto

Imagen 3.10: Voltaje de alimentación de la tarjeta ESP32.

La implementación de la tarjeta de batería en su caja se muestra en la *Imagen 3.11* y en la *Imagen 3.12* la implementación junto con la batería.

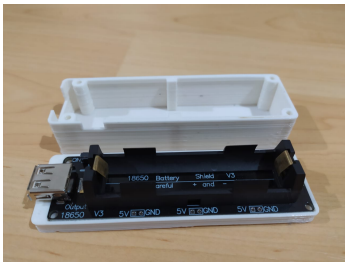


Imagen 3.11: Tarjeta de batería 18650 V3 con caja.

Imagen 3.12: Tarjeta de batería 18650 con caja y batería.

3.6. Integración sistema mecatrónico

Para la integración del sistema se realizan una serie de pruebas para cada seña, comprobando para cada una, la verificación de su traducción y resultados en el sistema. Las pruebas se realizan por cada integración de los módulos en el sistema.
Para la integración del módulo de captura de señas y el módulo de acondicionamiento de señales se realiza la conexión mostrada en el diagrama de la *Imagen 2.18*, en la cual se relaciona cada sensor a una entrada de la tarjeta ESP32 para una correcta lectura. La conexión se realiza sobre una placa fenólica perforada con pistas de 7.5 cm x 4.5 cm, la caja de la tarjeta ESP32 se coloca antes de aplicar la soldadura a cada componente.
Una vez realizada la conexión se verifica con el mismo código que utiliza la RNA para su clasificación, la lectura para las pruebas de la RNA. El código para la recolección de datos varía en la impresión de los datos, ya que aplica para cada iteración en el puerto serie el siguiente pseudocódigo:

```
Para i=0 hasta 61
      contador = 0
      ESPERAR 2s
      HACER
            Lectura de sensores
            IMPRIMIR CON SALTO DE LÍNEA (Letra)
            Para i=0 hasta 17
                  IMPRIMIR CON SALTO DE LÍNEA (Lectura de sensores)
            IMPRIMIR CON SALTO DE LÍNEA (Etiqueta)
            ESPERAR 1s
            cont++
      MIENTRAS (contador<5)
```

Donde aplica 5 lecturas de una misma seña, recorriendo las 62 señas a traducir. Los datos que se imprimen cada segundo son un total de 20 datos. El primero corresponde a la letra a traducir, seguido de las 18 lecturas de los sensores con 10 decimales para una mayor precisión y por último la etiqueta correspondiente a la letra. Estas etiquetas son mostradas en la *Tabla 3.4* asociadas a las señas correspondientes. Cuando se termina con las 5 lecturas de una misma seña se da un tiempo de 2 segundos, se reinicia el contador y se vuelve a realizar el mismo proceso. El código en Arduino

IDE se muestra en el *Apéndice 7*, correspondiente a “RECOLECCION_DATOS".

Señas	**Etiquetas**
Alfabeto	0 - 26
Meses del año	27 - 38
Días de la semana	39 - 45
Números	46 - 55
Palabras	56 - 61

Tabla 3.4: Señas - Etiquetas.

Para la transferencia de los datos al archivo .csv, se ocupa el mismo código ya utilizado para las pruebas en los módulos anteriores, ubicado en el *Apéndice 3*, correspondiente a "package examples.excel". Al generar las pruebas, se generan errores en algunas lecturas al realizar erróneamente las señas correspondientes. Estos errores se deben a equivocaciones del usuario, ya sea por el tiempo o por una mala configuración. Para identificar estos errores, se hace una comparación en el archivo de Excel, antes de su conversión a .csv, donde se eliminan los valores que no corresponden al rango promedio en el que se mantiene una misma seña.

Prueba de lectura de sensores para cada seña.

Descripción: Verificación del rango de los ángulos de rotación, mostrados en la *Tabla 2.1* de requerimientos y verificación de la lectura de los sensores flex por seña.
Procedimiento: La prueba consiste en realizar la lectura de sensores, así como su transferencia al archivo Excel para obtener aproximadamente 15 lecturas por cada seña (esto quiere decir que se realice 3 veces la iteración del código de “RECOLECCION_DATOS"). Se registra el tiempo de muestreo total de las pruebas. En una tabla se registra el número de pruebas totales realizadas, el número de pruebas totales obtenidas después de la evaluación de errores, el número de pruebas realizadas por letra y el número de pruebas realizadas por letra después de la evaluación de errores, obteniendo así el porcentaje de pruebas eficientes. En otras tablas se registran los valores mínimos y máximos de la lectura de cada seña, donde se puede realizar

fácilmente la comparación con los valores mínimos y máximos esperados.

Resultado: El tiempo de muestreo fue de 00:26:05. En la *Tabla 3.5* se muestran los números de pruebas realizadas totales y por seña, así como el porcentaje de eficiencia. En el *Apéndice 8* se muestran los resultados de los valores mínimos y máximos de cada lectura por seña, comparando y evaluando que se encuentran dentro del rango esperado.

Seña	**Total**	**Resultante**	%	**Seña**	**Total**	**Resultante**	%
A	15	12	80	Mayo	15	15	100
B	15	13	86.7	Junio	15	15	100
C	15	15	100	Julio	15	12	80
D	15	9	60	Agosto	15	15	100
E	15	14	93.3	Septiembre	15	13	86.7
F	20	13	65	Octubre	15	10	66.7
G	20	13	65	Noviembre	15	12	80
H	20	15	75	Diciembre	15	13	86.7
I	20	13	65	Lunes	15	14	93.3
J	20	12	60	Martes	15	15	100
K	20	10	50	Miércoles	15	13	86.7
L	20	9	45	Jueves	15	14	93.3
M	20	15	75	Viernes	15	15	100
N	15	15	100	Sábado	15	12	80
Ñ	15	15	100	Domingo	15	15	100
O	15	13	86.7	Uno	15	13	86.7
P	15	9	60	Dos	15	13	86.7
Q	15	10	66.7	Tres	15	14	93.3
R	15	12	80	Cuatro	15	15	100
S	15	13	86.7	Cinco	15	15	100

(pasa a la página siguiente)

Seña	**Total**	**Resultante**	%	**Seña**	**Total**	**Resultante**	%
T	15	15	100	Seis	15	13	86.7
U	15	15	100	Siete	15	11	73.3
V	15	11	73.3	Ocho	15	15	100
W	15	13	86.7	Nueve	15	15	100
X	15	12	80	Diez	15	13	86.7
Y	15	14	93.3	Hola	15	9	60
Z	15	13	86.7	Bien	15	12	80
Enero	15	8	53.3	Mal	15	13	86.7
Febrero	15	9	60	Perdón	15	14	93.3
Marzo	15	11	73.3	No	15	12	80
Abril	15	11	73.3	Si	15	15	100
				Totales	970	797	82

Tabla 3.5: Número de pruebas por seña.

El módulo de suministro de energía se integra al sistema realizando la conexión mediante un cable USB a micro USB para la alimentación de la tarjeta ESP32, misma que se utiliza para alimentar a los sensores. La tarjeta de la batería junto con la batería se colocan dentro de la caja, posteriormente se colocan dentro del soporte del brazo, ajustando a la medida del brazo con el velcro. Se realiza una prueba de verificación del voltaje de salida de la tarjeta ESP32 al estar alimentada por la tarjeta de batería con la batería y al estar alimentando a todos los sensores utilizados.

Prueba de medición de voltaje de salida de la tarjeta ESP32.

Descripción: Verificación del voltaje de salida de la tarjeta ESP32.

Procedimiento: La prueba consiste en realizar la medición del voltaje con un multímetro en los pines de voltaje de salida y tierra de la tarjeta ESP32. Este voltaje corresponde a un valor aproximado a 3.3V, por lo que el valor medido debe corres-

ponder a esta medida.

Resultado: En la *Imagen 3.13* se muestra la foto de la medida del multímetro, la medida es de 3.278V, por lo que se muestra que es correcto.

Imagen 3.13: Voltaje de salida de la tarjeta ESP32.

La integración del módulo de clasificación en el sistema se realiza con los datos correspondientes a la prueba de lectura de sensores para cada seña, en la cual se registra la lectura de los 18 datos de sensores para cada seña (relacionada con su etiqueta). La programación del código para la generación del modelo de la RNA se realiza en Colab, donde se implementa el código utilizado en la validación del módulo de clasificación, código correspondiente al *Apéndice 4*.

Prueba de RNA.

Descripción: Verificación del funcionamiento de la RNA en el sistema.

Procedimiento: La prueba consiste en implementar el archivo .csv generado de la prueba de lectura de sensores para cada seña, en el código para la generación del modelo de la RNA en Colab. Se muestra como resultado, una gráfica de pérdida en el entrenamiento, así como su porcentaje de exactitud. Posteriormente se evalúa un resultado aleatorio de la clasificación en TensorFlow Lite desde Colab y se genera el código en hexadecimal para la codificación en Arduino IDE, registrando el tamaño en bytes de este archivo.

Resultado: Porcentaje de exactitud de 94.38 %. En la *Imagen 3.14* se muestra la pérdida en el entrenamiento con 1500 épocas.

Imagen 3.14: Gráfica de pérdida en el entrenamiento del sistema.

Para la evaluación del modelo en TensorFlow Lite, se muestra en la *Imagen 3.15* el diagrama del proceso para un resultado, anexado igualmente en el código del *Apéndice 4*. Se escoge la muestra 40 de la evaluación. Los resultados corresponden a un vector de 62 valores, donde se visualiza que la posición del vector mayor corresponda a la posición de la codificación "One Hot Encoding" de la etiqueta.

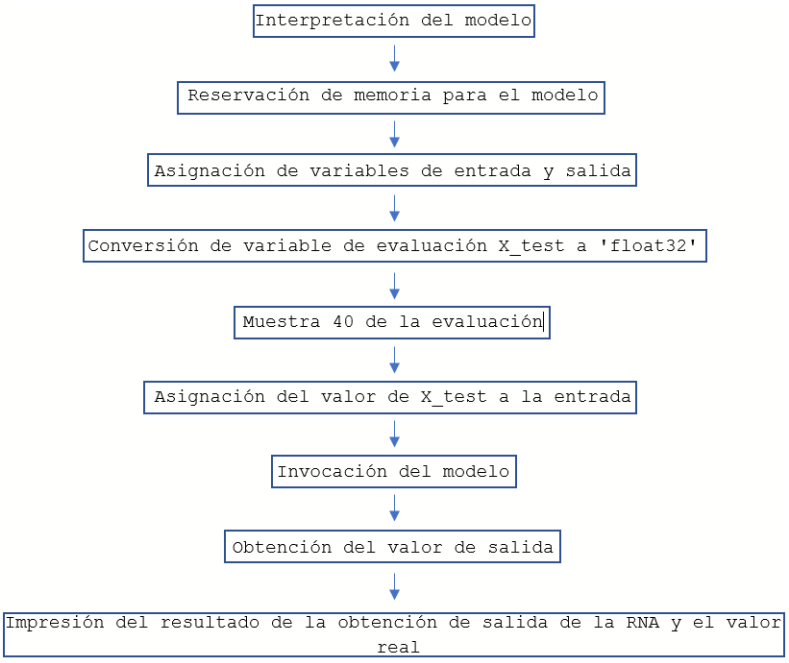


Imagen 3.15: Proceso de un resultado en TensorFlow Lite.

```
[[0.0000000e+00 0.0000000e+00 0.0000000e+00 3.2954341e-33 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 4.5471198e-34
  6.4652881e-30 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 1.2330725e-34 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 1.9091080e-32 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 1.5794640e-21 8.2958745e-20 0.0000000e+00
  0.0000000e+00 0.0000000e+00 3.6131222e-31 1.0107180e-25 5.6211943e-29
  8.7732679e-26 4.6810753e-26 0.0000000e+00 2.0840927e-14 2.6349343e-35
  0.0000000e+00 0.0000000e+00 0.0000000e+00 1.1518004e-24 6.8614064e-35
  3.2860623e-22 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 4.1140567e-23
  7.7906858e-38 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 2.3546089e-34]]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

La posición del vector mayor del resultado es 39, lo mismo que en la etiqueta, por lo que se verifica el funcionamiento del modelo de la RNA en TensorFlow Lite. La codificación en hexadecimal para Arduino IDE se muestra en el *Apéndice 5* con un tamaño de 7848 bytes.

Terminada la fase de entrenamiento del sistema, se integra el módulo de comunicación con el usuario para la visualización del resultado de todo el sistema en conjunto. El modelo de la RNA se descarga del folder de Colab, donde se genera el archivo. El código en Arduino IDE utilizado es el mismo que se usó para la recolección de datos, pero sin la modificación de la impresión de datos, el código se muestra en el *Apéndice 7*, correspondiente a " EVALUACION_DATOS".

La visualización se realiza al conectar el puerto serie de Arduino IDE con la interfaz en Java. La programación utilizada en la interfaz se muestra en el *Apéndice 3*, correspondiente a "package com.mycompany.interfazfinal".

Prueba de traducción de cada seña.

Descripción: Verificar el funcionamiento del sistema para la traducción de cada seña.

Procedimiento: La prueba consiste en la implementación de todos los módulos en conjunto, donde se registra en una tabla la traducción de las 62 señas, repitiendo 10 veces cada seña. Se verifica el porcentaje de eficiencia por cada seña.

Resultado: En la *Tabla 3.6* se muestran los resultados de la prueba por número de etiqueta.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	%
0	0	0	0	0	0	0	0	0	0	0	100
1	1	1	1	1	1	1	1	1	1	1	100
2	2	2	2	2	2	2	2	2	2	2	100
3	3	3	3	3	3	3	3	3	3	3	100
4	4	4	4	4	4	4	4	4	4	4	100
5	5	5	5	5	5	5	5	5	5	5	100
6	6	6	6	6	6	6	6	6	6	6	100
7	7	7	7	7	7	7	7	7	7	7	100
8	8	8	8	8	8	8	8	8	8	8	100
9	9	9	9	9	9	9	9	9	9	9	100
10	10	10	10	10	10	10	10	10	10	10	100
11	11	11	11	11	11	11	11	11	11	11	100
12	12	7	12	12	12	12	12	12	12	47	80
13	13	13	13	13	13	13	7	12	13	13	80
14	14	40	40	14	14	40	40	14	14	14	100
15	15	15	15	15	15	15	15	15	15	15	100
16	16	16	16	16	16	3	16	3	3	3	60
17	17	17	17	17	17	17	17	17	17	17	100
18	18	18	18	18	18	18	18	18	18	18	100
19	19	19	19	19	19	19	19	19	19	19	100
20	20	20	20	20	20	20	20	20	20	20	100
21	21	21	21	21	21	21	21	21	21	21	100
22	22	22	22	22	22	22	22	22	22	22	100
23	23	22	23	43	22	22	23	23	23	23	60
24	24	24	24	24	24	24	24	24	24	24	100
25	25	25	25	25	25	25	25	25	25	25	100
26	26	26	26	26	26	26	26	26	26	26	100

(pasa a la página siguiente)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	%
27	27	27	27	27	27	27	27	4	4	27	80
28	28	28	28	28	33	28	28	28	28	28	90
29	29	29	29	29	29	29	29	29	29	29	100
30	30	30	30	30	30	30	30	30	30	30	100
31	31	31	31	31	31	31	31	31	31	31	100
32	32	32	32	32	32	32	32	9	32	25	80
33	33	33	33	33	33	33	33	33	33	33	100
34	34	0	30	34	34	34	11	44	34	34	60
35	35	35	44	35	44	35	35	44	44	35	60
36	36	36	36	36	36	36	36	36	36	36	100
37	37	37	37	37	37	37	37	37	37	37	100
38	38	38	38	38	38	38	38	38	38	38	100
39	39	39	39	39	39	39	39	39	39	39	100
40	40	40	40	40	40	40	40	40	40	40	100
41	41	41	41	41	41	41	41	41	41	41	100
42	42	42	42	42	42	42	42	42	42	42	100
43	43	43	43	43	43	43	43	43	43	43	100
44	44	44	44	44	44	44	44	44	44	44	100
45	45	45	39	45	39	38	45	45	45	45	70
46	52	46	46	46	46	46	46	46	46	46	90
47	47	47	47	47	47	43	53	53	47	47	70
48	48	48	48	48	48	48	48	48	48	48	100
49	1	49	49	49	49	49	59	49	49	49	80
50	50	50	50	50	50	50	50	50	50	50	100
51	51	51	51	51	51	51	51	51	51	51	100
52	52	52	52	52	52	52	52	52	52	52	100
53	53	53	53	53	53	53	53	53	53	53	100

(pasa a la página siguiente)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	%
54	54	54	59	54	54	54	59	54	54	54	80
55	55	55	55	55	55	55	55	55	55	55	100
56	56	56	56	56	40	56	40	40	40	56	60
57	57	57	57	57	57	57	57	57	57	57	100
58	58	58	58	58	58	58	58	58	58	58	100
59	59	59	59	59	59	59	59	59	59	59	100
60	60	60	60	60	60	60	60	60	13	60	90
61	61	61	61	61	61	61	61	61	61	61	100
										% total	93.38709677

Tabla 3.6: Resultados de 10 pruebas de la traducción por seña.

Prueba de funcionamiento general del sistema.

Descripción: Verificar el porcentaje final del funcionamiento del sistema para la traducción de cada seña.

Procedimiento: Aplicar el mismo proceso de la prueba "Prueba de traducción de cada seña", sin la visualización en la interfaz, únicamente con el código correspondiente al *Apéndice 7* a EVALUACION_DATOS. Se utiliza el mismo algoritmo con la impresión del porcentaje de cada seña en el puerto serie. Se realizaron 100 pruebas por cada seña, por lo que se registran únicamente los porcentajes finales. **Resultado:** En la *Tabla 3.7* se muestran los resultados por porcentaje de precisión de la traducción de cada seña en el dispositivo.

Seña	**Porcentaje**	**Seña**	**Porcentaje**
A	94	Mayo	99
B	100	Junio	82
C	100	Julio	96
D	100	Agosto	54

(pasa a la página siguiente)

Seña	Porcentaje	Seña	Porcentaje
E	100	Septiembre	58
F	100	Octubre	100
G	99	Noviembre	98
H	100	Diciembre	94
I	100	Lunes	100
J	93	Martes	100
K	95	Miércoles	99
L	100	Jueves	96
M	89	Viernes	100
N	66	Sábado	95
Ñ	91	Domingo	71
O	95	Uno	83
P	84	Dos	64
Q	84	Tres	96
R	93	Cuatro	84
S	96	Cinco	100
T	98	Seis	100
U	99	Siete	100
V	92	Ocho	100
W	68	Nueve	79
X	93	Diez	100
Y	97	Hola	43
Z	95	Bien	97
Enero	82	Mal	94
Febrero	83	Perdón	98
Marzo	97	No	71
Abril	98	Si	87

(pasa a la página siguiente)

Seña	Porcentaje	Seña	Porcentaje
		Total	90.62903226

Tabla 3.7: Resultados de porcentajes de precisión de cada seña.

CAPÍTULO 4

Análisis de resultados

4.1. Análisis de ingeniería

La RNA tuvo mejores resultados al aumentar el número de pruebas por cada seña, dando como resultado final del dispositivo, 100 pruebas por cada seña. Las pruebas se realizaron con un algoritmo que permite capturar 5 veces los valores de los 18 datos (sensores) por cada una de las 62 señas. Se realizó un proceso de pruebas con 6 iteraciones, se retiró el guante y se volvió a repetir el mismo proceso 4 veces, el último con un número de iteraciones de 2.

Las pruebas se realizaron únicamente con un usuario, por lo que, el dispositivo se podría complementa, realizando pruebas con diferentes usuarios con una medida XS, capturando así, diferentes datos de cada seña con distintos usuarios. Los resultados tenderían a ser más exactos cuando el dispositivo se porte por un usuario nuevo, ya que, cada usuario puede realizar una misma seña con una variación pequeña en los ángulos, lo que ayuda al entrenamiento de la RNA a identificar la seña cuando la porte un usuario que no haya participado en el las pruebas de entrenamiento.

Las pruebas tuvieron dos tipos de errores: los causados por el usuario y los esperado del sistema. Para el entrenamiento de la RNA, los errores causados por el usuario fueron eliminados y los errores del sistema persistieron en el entrenamiento, ya que, son errores que se deben considerar.

El porcentaje de eficiencia se realizó con el entrenamiento de la RNA de 970 pruebas registradas en la *Tabla 3.5*, inicialmente, realizando la seña 10 veces, donde el usuario identificaba que la seña que estuviera realizando coincidiera con la validación de la ventana en la interfaz correspondiente. Se anotaron los resultados de cada prueba en la *Tabla 3.6*. Posteriormente se generó un algoritmo que permitiera identificar en Arduino IDE la coincidencia de la seña realizada con la seña a evaluar 100 veces, esta evaluación fue generada para el resultado final de la eficiencia del dispositivo y en la *Tabla 3.7* se registraron estos porcentajes. Para esta tabla se puede considerar que 53 de las 62 señas fueron traducidas correctamente (señas que tienen un porcentaje mayor a 80 %). Las señas de Agosto y Septiembre son señas en las que el ángulo del dedo pulgar debía ser lo que las diferenciará. Al contar con dos sensores en el dedo pulgar, los sensores no llegan a su flexión total, por lo que, una solución sería implementar un solo sensor a ese dedo. La seña correspondiente a "hola", al ser la que tiene el porcentaje menor, tuvo gran similitud con la seña Martes. Al analizar la similitud, se concluyeron dos soluciones: la primer solución es, agregar al código el eje en el que existe movimiento y no sólo comparar si existe algún cambio de posición de ejes en cualquiera de los ejes; la segunda solución es, en el sensor flex del codo, comparar la resistencia en un segundo y detectar si existe un cambio de flexión.

La RNA debe calibrarse cada determinado tiempo, ya que, el desgaste de los sensores flex provoca que éstos reduzcan su resistencia con el paso del tiempo. El tiempo dependerá del uso constante o no del dispositivo. El número de ciclos de flexión es mayor a 1 millón para llegar al desgaste total de los sensores flex.

Al implementar el dispositivo, se enfrentaron las siguientes problemáticas:

- Se deshabilitaron los pines analógicos de la tarjeta ESP32 al hacer la conexión con Bluetooth, por lo cual, se buscó un módulo lo suficientemente pequeño para adoptarlo al dispositivo y que cumpliera con la función de generar en una única salida que considerará las 8 salidas de los sensores, también se implementó el módulo Cd4051.

- Inicialmente se propuso la implementación de 3 MPU6050, pero al ser imple-

mentados en el dispositivo, uno en el dorso de la mano, otro en la muñeca y otro en el antebrazo, se observó que la lectura del módulo ubicado en la muñeca era innecesaria al ya contar con los otros dos.

- La tarjeta ESP32 cuenta con 2 protocolos de comunicación I^2C, para realizar la comunicación se requería configurar otro pin en la tarjeta, lo que llevaba a más cables y líneas de código, así que, finalmente se optó por quitarlo.

- Se tuvo que cambiar de IDE para el desarrollo de la tarjeta, ya que se trabajó con TensorFlow Lite, una versión ligera especial para microcontroladores, ideal para el proyecto. Inicialmente, se planeaba usar Zerynth Studio, pero esta IDE no se encuentra dentro de las IDE con las que trabaja TensorFlow Lite. La página oficial de TensorFlow Lite trabaja con Arduino IDE, ESP IDF y Mbed. Por lo que se optó trabajar con Arduino IDE.

- Se modificó el diseño del dispositivo para mejorar su ergonomía en la parte del soporte de los sensores flex en los dedos. Inicialmente se contemplaba colocar Sintra pvc para cada dedo, pero al colocarlos y tratar de realizar los movimientos por un tiempo, la presión en cada uno de estos soportes en los dedos llegaba a ser muy molesta. Se optó por cambiar los soportes únicamente con tiras de velcro, ya que es un material cómodo para el ajuste.

El dispositivo cumplió con sus principales criterios considerados en el diseño conceptual, siendo uno de estos que el dispositivo debía ser ligero y permitir un libre movimiento para realizar las señas. El dispositivo pesó 360 g, lo que representa un peso menor al límite propuesto de 450 g, siendo éste un peso soportable para portar durante 4 horas distribuido en el miembro superior derecho. La parte con más peso del dispositivo se localizó en el brazo, diseñado para que en esta parte fuera colocada la pila. En la parte de la mano y antebrazo se tiene menos peso y al estar realizando las señas no se presentó ningún inconveniente.

Para una expansión de señas en el dispositivo, se necesitaría un análisis para verificar que los sensores son suficientes en la traducción. Empleando algunas configuraciones del hilo conductivo para la detección de contactos entre algunos dedos o

con la mano izquierda, con lo cual se implementaría un guante en la mano izquierda para la lectura de más datos para el caso de señas bimanuales.

4.2. Análisis de costos

En la *Tabla 4.1* se reflejan los costos de todos los componentes del dispositivo en MXN, clasificado por módulos, siendo los módulos: 1 Captura de señas, 2 Acondicionamiento de señales y 3 Suministro de energía.

Módulo	**Componentes**	**Cantidad**	**Unidad**	**Precio**	**Envío**	**Total**
1	Sensor flex	12	pza.	320	0	3840
	MPU6050	2	pza.	29	0	58
	Hilo conductivo	1	rollo	99	0	99
	Sintra PVC espumado	1	lámina	360	0	360
	Impresión 3D soportes	4	pza.	15	0	60
	Tiras de velcro	2	rollo	54.34	0	108.68
	Cinta elástica	1	pza.	149	90	239
	Hilo para coser	1	rollo	10	0	10
2	Placa Cd4051	1	pza.	71.5	90	161.5
	Tarjeta ESP32	1	pza.	169	72	241
	Placa fenólica perforada	1	pza.	19	0	19
	Resistencias 15K	20	pza.	1	0	20
	Impresión 3D caja ESP32	1	pza.	56	90	146
3	Impresión 3D caja batería	1	pza.	213	90	303
	Cable USB micro USB	1	pza.	60	60	120
	Tarjeta 18650 V3	1	pza.	179.99	60	239.99
	Batería 18650	1	pza.	150	0	150
					342	6175.17

Tabla 4.1: Costos de componentes del dispositivo por módulo.

Los costos del material para la implementación del dispositivo en MXN fueron identificados en la *Tabla 4.2*, clasificados por módulos, siendo los módulos: 1 Captura de señas y 2 Acondicionamiento de señales. El módulo de captura de señas tiene principalmente material para el molde de yeso de la mano y el módulo de acondicionamiento de señales material para soldar/desoldar componentes.

Módulo	**Material**	**Precio**
1	Pistola de calor	357.18
	Kola Loka	22
	Alginato	120
	Yeso	20
	Vaselina	30
2	Pistola de calor	47
	Pasta para soldar	19
	Cinta de aislar	19
	Desoldador	89
	Malla para desoldar	39
	Cautin	339.99
		1102.17

Tabla 4.2: Material para implementación del dispositivo por módulo.

Considerando los costos variables como son los honorarios, servicio de luz, servicio de internet y gastos de papelería, en la *Tabla 4.3* se muestra el costo total del desarrollo del dispositivo.

Descripción	**Precio**
Costo total de componentes	6175.17
Costo total del material para implementación	1102.17
Costos variables	5000.00

(pasa a la página siguiente)

Descripción	Precio
	12277.34

Tabla 4.3: Costo total del dispositivo.

4.3. Análisis de valor

El dispositivo puede ser usado para la validación de cualquiera de las 62 señas. Usado para la enseñanza de palabras básicas, donde la persona que porte el dispositivo pueda visualizar que la seña que realiza es correcta, así como los movimientos, ángulos de flexión y ángulos de rotación. La característica que tiene este dispositivo es la identificación de señas similares por su configuración en la mano pero con distintos movimientos, un ejemplo es la letra L, donde la configuración en la mano es la misma que para la palabra Lunes pero el movimiento realizado para la segunda seña es lo que hace que el dispositivo identifique movimiento en alguno de los 3 ejes y se puede diferenciar de la primer seña. Una continuación del proyecto puede ser la implementación de pruebas con distintas personas, así como poder agregar más señas con la implementación de otro guante en la mano izquierda. Igualmente se puede combinar con un sistema de procesamiento de imágenes adicional para un mejor resultado de eficiencia de la clasificación, ya que la LSM es una lengua que no solo involucra el movimiento de las manos, la expresión corporal y los gestos son parte igualmente importante de esta lengua.

Conclusiones

Al realizarse el desarrollo del dispositivo en medio de la pandemia se presentaron problemáticas que afectaron en los tiempos de análisis, desarrollo y pruebas, dado que, fue necesario sustituir algunos componentes por falla o averío. Debido a la alta demanda de los servicios de paquetería los componentes tardaron 2 - 3 veces más del tiempo esperado. Algunos otros componentes fueron comprados desde China, sin embargo debido a que fue uno de los países más afectados y que pararon sus actividades casi totalmente, los envíos fueron cancelados. En ese caso, se adquirieron los componentes con vendedores nacionales, pero al triple del precio inicial adicional a los costos de envío.

Se tenía previsto realizar las conexiones a la tarjeta ESP32 mediante una placa impresa en PCB, misma que estaba prevista realizarse con las herramientas ubicadas en los laboratorios de UPIITA; al cotizar con proveedores locales, el precio era desmesurado, por lo que se optó por una placa fenólica perforada, esta quedó más grande a las medidas de la cinta elástica.

Cuando no se contaba con algún componente y era indispensable para continuar con la implementación, se adaptó el dispositivo temporalmente para poder seguir con las pruebas.

Para una correcta visualización de la traducción hay que esperar alrededor de 2 segundos. Algunas señas como Domingo, Junio, s y m necesitan esperar hasta 3 segundos para obtener una lectura correcta. Las letras con menor porcentaje de eficiencia son la P, (que se confunde con la letra D), la letra W, hola, Agosto y Septiembre. El tiempo mínimo para realizar la seña es de 1 segundo, por lo que puede ser que los movimientos de una persona sorda sean demasiado rápidos, el dispositivo no identificaría todas las señas.

Cabe mencionar que el resultado de la traducción del dispositivo es 3 % menos de lo esperado del modelo de la RNA, según la Prueba de funcionamiento general. Como se indica en la *Tabla 3.5*, puede deberse al error del usuario que realiza la seña, ya sea por falta de experiencia en la LSM ó la manera en que realicé la seña.

Inicialmente se identificó una lista de señas básicas a nivel nacional. Se analizaron e implementaron los componentes para el dispositivo. Se programó una RNA para la identificación de las señas seleccionadas. Se integro la interfaz gráfica, donde se puede observar el resultado de la seña identificada.

Aún con las problemáticas presentadas se logró totalmente diseñar y construir un dispositivo para la extremidad superior derecha que obtiene mediante sensores, los datos de posicionamiento para su identificación mediante redes neuronales y transcripción mostrada en una interfaz gráfica de las señas manuales que caracterizan al alfabeto dactilológico, además de la lista de palabras básicas de la LSM.

Referencias

[1] Arduino project power source, using a single 18650 cell. https://www.youtube.com/watch?v=gG82$_fG4VM8$.

[2] Ciclo de vida en "v". http://ingsoftware.weebly.com/ciclo-de-vida-en-v.html.

[3] Diseño conceptual - INGENIERIA DEL DISEÑO. https://sites.google.com/site/ingenieriadeldiseno/diseno-conceptual.

[4] Especificación del diseño de producto (PDS).

[5] Guía para determinar tallas de guantes. https://rwoperu.com/assets/guia-de-tallas-y-cuidados-en-espanol.pdf?fbclid=IwAR2vstSM$_9$8$djXOTCUV7RXlyzdCv0fAp6TuLX7brFlD6t8KyHB-MkgrQN1Y$.

[6] Imagine Cup 2012 - Finalist Presentations: Team quadSquad, Ukraine.

[7] Introducción a las redes neuronales aplicadas. http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf.

[8] Keras | TensorFlow core. https://www.tensorflow.org/guide/keras?hl=es.

[9] Mgsystem sensor flex 2.2 guante mano robot resistivo. https://articulo.mercadolibre.com.ec/MEC-422160749-mgsystem-sensor-flex-22-guante-mano-robot-resistivo-$_JM$.

[10] Módulo ESP32 ESP-WROOM-32. https://naylampmechatronics.com/espressif-esp/382-modulo-esp32-esp-wroom-32.html.

[11] Portable power- 18650 battery shield for raspberry pi & arduino. https://www.electroschematics.com/battery-shield/.

[12] Primeros pasos con TensorFlow: Kit de herramientas. https://developers.google.com/machine-learning/crash-course/first-steps-with-tensorflow/toolkit?hl=es-419.

[13] Tallas 101: Guía sobre tallas de guantes y mangas. https://www.superiorglove.com/es/work-gloves-101/sizing?fbclid=IwAR3ICYXIOhtm69PlfH1qIonQl19IGK8vJhy1L7WX2Y6AioXcKyx9XQAvirE.

[14] Tutorial 8. lectura del sensor MPU6050 (acelerómetro y giroscópio) en microcontrolador de la serie STM32f4. https://www.intesc.mx/tutorial-8-lectura-del-sensor-mpu6050-acelerometro-y-giroscopio-en-microcontrolador-de-la-serie-stm32f4/.

[15] UNIDAD 4 REDES NEURONALES - INTELIGENCIA ARTIFICIAL. https://sites.google.com/site/mayinteligenciartificial/unidad-4-redes-neuronales.

[16] Wired golf glove translates sign language. https://www.youtube.com/watch?v=oQObpghc9yMfeature=youtu.

[17] Estudiante de Electromecánica desarrolla guante inteligente que traduce la lengua de señas. https://ingenieria.uaq.mx/estudiante-de-electromecanica-desarrolla-guante-inteligente-que-traduce-la-lengua-de-senas/, 2015.

[18] La discapacidad en México, datos 2014. 2016. Perfil sociodemográfico de la población 362.4021, INEGI, Aguascalientes, México, 2016.

[19] Talking Hands | Signs can raise their voice! http://www.limix.it/talking-hands-en/, 2018.

[20] *Audismo, por una cultura de inclusión* (Cenlex, Zacatenco, GAM, CDMX, Mar. 2019).

[21] Abdessamad, T. Capitulo 4 el método AHP. http://bibing.us.es/proyectos/abreproy/70496/fichero/Capitulo+4+El+m%C3%A9todo+AHP.pdf.

[22] Ashlock, D., and Warren, A. Guía de acondicionamiento de señales para ingenieros.

[23] Avetis Kalustyan. EnableTalk. https://www.youtube.com/watch?v=HCAwPBbDkhkfeature=youtu.be, May 2012.

[24] Aznar Bellver, J. Proceso analítico jerárquico. AHP (analytic hierarchy process) | | UPV. https://www.youtube.com/watch?v=gaML3XIHiGc.

[25] Balet, O. Figure 42 : Le dextrous hand master d'exos inc. https://www.researchgate.net/figure/Le-Dextrous-Hand-Master-dExos-Inc-extrait-de-Burdea93$_{f}ig25_{2}$88977556.

[26] CDMX, I. *Diccionario de Lengua de Señas Mexicana de la Ciudad de México*. No. 1 in DLSM.COMISA. CDMX, Sept. 2017, p. 505.

[27] Daniela Vega. Tock Tock: El guante traductor de señas. https://www.unotv.com/noticias/portal/investigaciones-especiales/detalle/tock-tock-guante-traductor-senas-979154/, July 2016.

[28] Dariel Torres Araoz, Ana Cecilia García de León Jiménez, Eduardo Humberto González Moreno, Karla Verónica Hernández Torres, and Andrea Carmona Aguirre. TockTcok. http://emprendeweb.mx/index.php/tocktock, 2016.

[29] Dehoff, B., Rhodes, R. E., and Levack, D. La estructura de descomposición funcional (FBS) y su relación con el costo del ciclo de vida. https://www.researchgate.net/publication/268324898$_{T}he_{F}unctional_{B}reakdown_{S}tructure_{F}BS_{a}nd_{I}ts_{R}elationship_{t}o_{L}ife_{C}ycle_{C}ostprev = search$.

[30] Eduardo Buendía. Sordos en México: sin educación ni trabajo. https://www.eluniversal.com.mx/articulo/periodismo-de-datos/2017/04/2/sordos-en-mexico-sin-educacion-ni-trabajo, Apr. 2017.

[31] Fuente Externa. Ingeniero mexicano crea guante que traduce lenguaje de señas. https://www.megapractical.com/noticias/ingeniero-mexicano-crea-guante-que-traduce-lenguaje-de-se %C3 %B1as, May 2016.

[32] García, A. PanamaHitek_arduino. https://github.com/PanamaHitek/PanamaHitek_Arduino. original-date: 2016-11-08T15:35:14Z.

[33] Goli Mohammadi. Talking Hands Translates Sign Language to Voice. https://makezine.com/2018/06/18/talking-hands/, June 2018.

[34] Grifantini, K. Guantes de datos de código abierto. https://www.technologyreview.es/s/560/guantes-de-datos-de-codigo-abierto.

[35] Isermann, R. MECHATRONIC SYSTEMS - INNOVATE PRODUCTS WITH EMBEDDED CONTROL-. https://www.scribd.com/document/367308869/MECHATRONIC-SYSTEMS-by-Isermann-pdf.

[36] Jiménez Molina, E., and Contreras Garrido, N. Ingeniería de software - modelo en v. https://prezi.com/ryyutemqk5go/ingenieria-de-software-modelo-en-v/5.

[37] Jose L. Hernandez-Rebollar. METHOD AND APPARATUS FOR TRANSLATINGHAND GESTURES. https://patentimages.storage.googleapis.com/8b/9b/13/2d2307bf51b376/US7565295.pdf, July 2009.

[38] José Manuel Infante. Flex Sensor. https://rambal.com/presion-peso-nivel-flex/250-sensor-flex.html.

[39] José Ángel Avelar Barragán. *Prototipo traductor de lenguaje de señas mexicano mediante el uso de redes neuronale.* PhD thesis, UPIITA, IPN, CDMX, Dec. 2018.

[40] Kolban, N. *Kolban´s Book on ESP32*.

[41] LiMiX Srl. Introducing Talking Hands SUB ITA. https://www.youtube.com/watch?v=2MUGsCDbjTc, Jan. 2018.

[42] Machado, A. Luva de dados (dataglove). http://aplinformaticas.blogspot.com/2010/09/luva-de-dados-dateglove.html.

[43] Meza, G. G. Tu primer red neuronal usando keras. https://medium.com/@gogasca/$tu-primer-red-neuronal-usando-keras-72d36130ee6c$.

[44] Montserrat Sánchez Maldonado. Así es la vida de personas sordas en México. https://www.animalpolitico.com/2016/09/lenguaje-personas-sordas-mexico/, Sept. 2016.

[45] Padilla Montiel, O. Manipulador teleoperado inalámbricamente. http://catarina.udlap.mx/u$_{dl_a}$/$tales/documentos/lmt/padilla_{mo}/capitulo3.pdf?fbclid=IwAR3Rt5keGowDzalyy4u5bcA2XIEcPvrd5IK25w7c-NSaKS2nIyKg51y3CCk$.

[46] Patterson, R. R. SIGN LANGUAGE TRANSLATOR. https://patentimages.storage.googleapis.com/b1/0c/ae/373520c5ca1dc1/US20020152077A1.pdf, Oct. 2002.

[47] Ruvalcaba, D. G., Ruvalcaba, M., Orozco, J. P., López, R., and Cañedo, C. E. Prototipo de Guantes Traductores de la Lengua de Señas Mexicana para Personas con Discapacidad Auditiva y del Habla. *MEMORIAS DEL XLI CONGRESO NACIONAL DE INGENIERÍA BIOMÉDICA/ 5*, 1 (Oct. 2018), 350–353.

[48] Samuel Castro Solis, and Samuel Santiago Sanjuan. *PROTOTIPO DE UN SISTEMA ELECTRÓNICO INTÉRPRETE DE LA LENGUA DE SEÑAS MEXICANA*. PhD thesis, UPIITA, IPN, CDMX, 2018.

[49] Sandro, A. 2-TUTORIAL TENSORFLOW LITE ARDUINO DUE: testing the model, python. https://www.youtube.com/watch?v=aYBUpt2w5Dgt=498s.

[50] Saul Gausin. Uso de Flex Sensor con Arduino. http://computointegrado.blogspot.com/2012/04/uso-de-flex-sensor-con-arduino.html, Apr. 2012.

[51] Thingiverse.com. ESP32 DevKit case. https://www.thingiverse.com/thing:4125952.

[52] Thingiverse.com. WeMos 18650 battery shield box w/modified lid. https://www.thingiverse.com/thing:2953929.

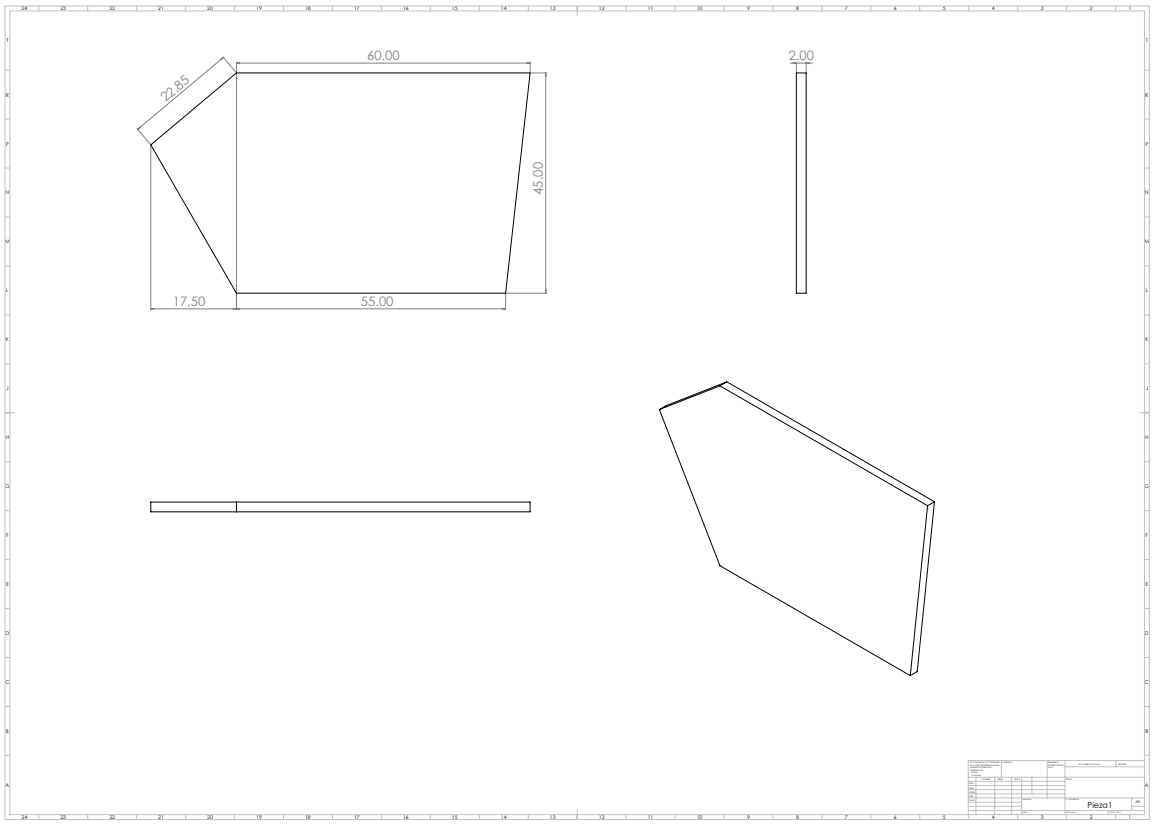
[53] Tutoriales. Tutorial MPU6050, Acelerómetro y Giroscopio. https://naylampmechatronics.com/blog/45$_{T}utorial$ – $MPU6050$ – $Aceler\,\%C3\,\%B3metro - y - Giroscopio.html$, 2016.

[54] UnoTV. Tock Tock: El guante traductor de señas. https://www.youtube.com/watch?v=1UhgcccdAm8, July 2016.

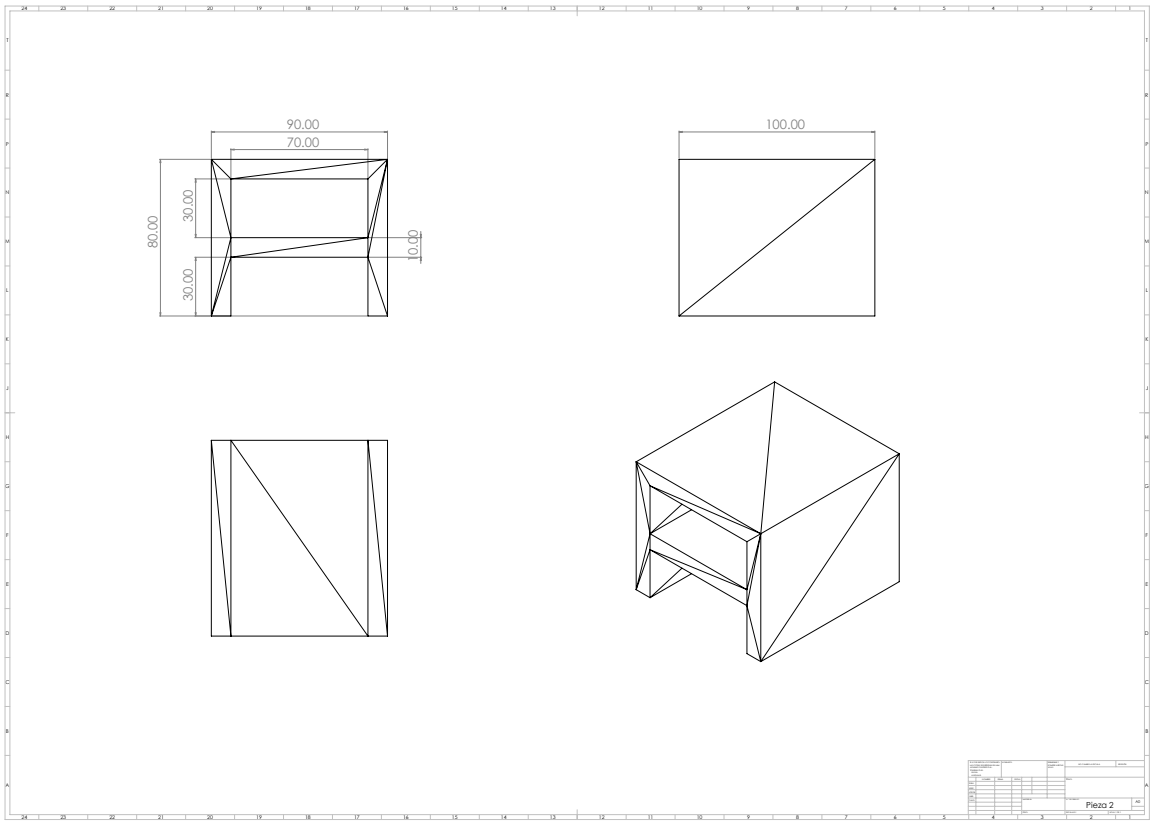
[55] Álvarez, J. G. M., Gómez, M. , Ferrara, J. G. T., Olvera, F. J. P., León, M. R. T., Vieyra, G. Q., Torres, F. J. A., and Díaz, C. A. CREA EGRESADO DE LA UPIIZ GUANTE QUE TRADUCE EL LENGUAJE EN SEÑAS. https://www.repositoriodigital.ipn.mx/bitstream/123456789/23795/1/G-sem1239.pdf, May 2016.

Apéndices

Apéndice 1. Estructura de Sintra PVC espumado.


60.00
22.85
45.00
17.50
55.00
2.00
Pieza1

Apéndice 2. Soporte para los sensores flex.


90.00
70.00
80.00
30.00
10.00
30.00
100.00
Pieza 2

Apéndice 3. Códigos en Netbeans IDE.

```
package examples.excel;

import com.panamahitek.ArduinoException;
import com.panamahitek.PanamaHitek_Arduino;
import com.panamahitek.PanamaHitek_DataBuffer;
import com.panamahitek.PanamaHitek_MultiMessage;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import jssc.SerialPortEvent;
import jssc.SerialPortEventListener;
import jssc.SerialPortException;

public class ExcelExport extends javax.swing.JFrame {

  //Objeto para la gestion de la conexion con Arduino
  PanamaHitek_Arduino ino = new PanamaHitek_Arduino();
  //Objeto para la gestion de multiples mensajes recibidos desde Arduino
  PanamaHitek_MultiMessage multi = new PanamaHitek_MultiMessage(19, ino);
  //Objeto para la gestion y almacenamiento de datos recibidos
  PanamaHitek_DataBuffer buffer = new PanamaHitek_DataBuffer();

  public ExcelExport() {
    initComponents();

    /*Se añaden en el objeto buffer las columnas necesarias, definiendo el
    número de columna, nombre y tipo de dato que se almacena */
    buffer.addTimeColumn(0, "Tiempo"); //Indica la hora de lectura
    buffer.addColumn(1, "Letra", String.class);
    buffer.addColumn(2, "S1", Double.class);
```

```
buffer.addColumn(3, "S2", Double.class);
buffer.addColumn(4, "S3", Double.class);
buffer.addColumn(5, "S4", Double.class);
buffer.addColumn(6, "S5", Double.class);
buffer.addColumn(7, "S6", Double.class);
buffer.addColumn(8, "S7", Double.class);
buffer.addColumn(9, "S8", Double.class);
buffer.addColumn(10, "S9", Double.class);
buffer.addColumn(11, "S10", Double.class);
buffer.addColumn(12, "S11", Double.class);
buffer.addColumn(13, "H", Double.class);
buffer.addColumn(14, "Mov", Double.class);
buffer.addColumn(15, "M1x", Double.class);
buffer.addColumn(16, "M1y", Double.class);
buffer.addColumn(17, "M2x", Double.class);
buffer.addColumn(18, "M2y", Double.class);
buffer.addColumn(19, "Etiqueta", Double.class);
//Se inserta el objeto buffer en un jPanel
buffer.insertToPanel(jPanel1);

/**
 * Con este objeto se gestiona la recepcion de datos. El evento
 * serialEvent se "disparara" cada vez que se reciba un dato desde
 * Arduino enviado a traves del puerto serie
 */
SerialPortEventListener listener = new SerialPortEventListener() {
   @Override
   public void serialEvent(SerialPortEvent serialPortEvent) {
      try {
         if (multi.dataReceptionCompleted()) {
```

```
            buffer.addValue(1, multi.getMessage(0));
            buffer.addValue(2, Double.parseDouble(multi.getMessage(1)));
            buffer.addValue(3, Double.parseDouble(multi.getMessage(2)));
            buffer.addValue(4, Double.parseDouble(multi.getMessage(3)));
            buffer.addValue(5, Double.parseDouble(multi.getMessage(4)));
            buffer.addValue(6, Double.parseDouble(multi.getMessage(5)));
            buffer.addValue(7, Double.parseDouble(multi.getMessage(6)));
            buffer.addValue(8, Double.parseDouble(multi.getMessage(7)));
            buffer.addValue(9, Double.parseDouble(multi.getMessage(8)));
            buffer.addValue(10, Double.parseDouble(multi.getMessage(9)));
            buffer.addValue(11, Double.parseDouble(multi.getMessage(10)));
            buffer.addValue(12, Double.parseDouble(multi.getMessage(11)));
            buffer.addValue(13, Double.parseDouble(multi.getMessage(12)));
            buffer.addValue(14, Double.parseDouble(multi.getMessage(13)));
            buffer.addValue(15, Double.parseDouble(multi.getMessage(14)));
            buffer.addValue(16, Double.parseDouble(multi.getMessage(15)));
            buffer.addValue(17, Double.parseDouble(multi.getMessage(16)));
            buffer.addValue(18, Double.parseDouble(multi.getMessage(17)));
            buffer.addValue(19, Double.parseDouble(multi.getMessage(18)));
            //Se salta un "renglon" en el buffer de datos
            buffer.printRow();
            //Para que se obtengan datos distintos en cada iteración
            multi.flushBuffer();
        }
    } catch (ArduinoException ex) {
        Logger.getLogger(ExcelExport.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SerialPortException ex) {
        Logger.getLogger(ExcelExport.class.getName()).log(Level.SEVERE, null, ex);
    } catch (Exception ex) {
        Logger.getLogger(ExcelExport.class.getName()).log(Level.SEVERE, null, ex);
    }
```

```
        }
    };

    try {
        //Se inicia la conexion con el puerto COM3 a 115200 baudios
        ino.arduinoRX("COM3", 115200, listener);
    } catch (ArduinoException | SerialPortException ex) {
        Logger.getLogger(ExcelExport.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    jButton1 = new javax.swing.JButton();

    jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 1532, Short.MAX_VALUE)
```

```
);
jPanel1Layout.setVerticalGroup(
   jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGap(0, 547, Short.MAX_VALUE)
);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jButton1.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
jButton1.setText("Exportar a Excel");
jButton1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton1ActionPerformed(evt);
   }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(layout.createSequentialGroup()
      .addGap(133, 133, 133)
      .addComponent(jButton1)
      .addContainerGap(1306, Short.MAX_VALUE))
);
layout.setVerticalGroup(
   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(layout.createSequentialGroup()
      .addContainerGap(572, Short.MAX_VALUE)
      .addComponent(jButton1)
      .addGap(4, 4, 4))
```

```
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed
        /**
         * Este boton permite finalizar la conexion con Arduino y exportar los
         * datos a Excel
         */
        try {
            //Se exportan los datos a Excel
            buffer.exportExcelFile();
            //Se finaliza la conexion con Arduino
            ino.killArduinoConnection();
            //Se muestra un mensaje
            JOptionPane.showMessageDialog(this, "Conexión con Arduino Finalizada");
            //Se cierra el programa
            System.exit(0);
        } catch (ArduinoException | IOException ex) {
            Logger.getLogger(ExcelExport.class.getName()).log(Level.SEVERE, null, ex);
        }
    }//GEN-LAST:event_jButton1ActionPerformed
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {

        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
```

```
            if ("Metal".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(ExcelExport.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
    } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(ExcelExport.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
    } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(ExcelExport.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(ExcelExport.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ExcelExport().setVisible(true);
        }
    });
}

// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton jButton1;
```

```
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables
}
```

```
package com.mycompany.interfazfinal;

import com.panamahitek.ArduinoException;
import com.panamahitek.PanamaHitek_Arduino;
import com.panamahitek.PanamaHitek_MultiMessage;
import java.awt.Image;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import jssc.SerialPortEvent;
import jssc.SerialPortEventListener;
import jssc.SerialPortException;

public class Principal extends javax.swing.JFrame {

  public static String[] getDatos() {
    return Datos;
  }

  public static void setDatos(String[] Datos) {
    Principal.Datos = Datos;
  }

  private static String[] Datos = new String[19];
  public static boolean dato = true;
  PanamaHitek_Arduino ino = new PanamaHitek_Arduino();
  PanamaHitek_MultiMessage multi = new PanamaHitek_MultiMessage(19, ino);
  SerialPortEventListener listener = new SerialPortEventListener() {
    @Override
    public void serialEvent(SerialPortEvent spe) {

      try {
```

```
        if (multi.dataReceptionCompleted()) {
            if (jLabel2.isEnabled() == true) {
                dato = false;
                do {
                    jLabel1.setText(jLabel1.getText());
                    if (jLabel3.isEnabled() == true) {
                        jLabel1.setText(null);
                        jLabel4.setIcon(null);
                        jLabel3.setEnabled(false);
                    }
                } while (dato == false);
            }
            if (jLabel3.isEnabled() == true) {
                dato = false;
                do {
                    jLabel1.setText(null);
                    jLabel4.setIcon(null);
                } while (dato == false);
            }
            jLabel1.setText(multi.getMessage(0));
            for (int i = 0; i < 19; i++) {
                Datos[i] = multi.getMessage(i);
            }

            switch (jLabel1.getText()) {
                case "A":
                    ImageIcon imagen = new ImageIcon("src/Imagenes/A.jpg");
                    Icon icon = new ImageIcon(imagen.getImage().getScaledInstance(jLabel4.getWidth(), jLabel4.getHeight(),
Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon);
                    break;
                case "B":
                    ImageIcon imagen2 = new ImageIcon("src/Imagenes/B.jpg");
```

```
                    Icon icon2 = new ImageIcon(imagen2.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon2);

                    break;

                case "C":

                    ImageIcon imagen3 = new ImageIcon("src/Imagenes/C.jpg");

                    Icon icon3 = new ImageIcon(imagen3.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon3);

                    break;

                case "D":

                    ImageIcon imagen4 = new ImageIcon("src/Imagenes/D.jpg");

                    Icon icon4 = new ImageIcon(imagen4.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon4);

                    break;

                case "E":

                    ImageIcon imagen5 = new ImageIcon("src/Imagenes/E.jpg");

                    Icon icon5 = new ImageIcon(imagen5.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon5);

                    break;

                case "F":

                    ImageIcon imagen6 = new ImageIcon("src/Imagenes/F.jpg");

                    Icon icon6 = new ImageIcon(imagen6.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon6);

                    break;

                case "G":

                    ImageIcon imagen7 = new ImageIcon("src/Imagenes/G.jpg");

                    Icon icon7 = new ImageIcon(imagen7.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon7);

                    break;

                case "H":
```

```
                ImageIcon imagen8 = new ImageIcon("src/Imagenes/H.jpg");

                Icon icon8 = new ImageIcon(imagen8.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon8);

                break;

              case "I":

                ImageIcon imagen9 = new ImageIcon("src/Imagenes/I.jpg");

                Icon icon9 = new ImageIcon(imagen9.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon9);

                break;

              case "J":

                ImageIcon imagen10 = new ImageIcon("src/Imagenes/J.gif");

                Icon icon10 = new ImageIcon(imagen10.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon10);

                break;

              case "K":

                ImageIcon imagen11 = new ImageIcon("src/Imagenes/K.gif");

                Icon icon11 = new ImageIcon(imagen11.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon11);

                break;

              case "L":

                ImageIcon imagen12 = new ImageIcon("src/Imagenes/L.jpg");

                Icon icon12 = new ImageIcon(imagen12.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon12);

                break;

              case "M":

                ImageIcon imagen13 = new ImageIcon("src/Imagenes/M.jpg");

                Icon icon13 = new ImageIcon(imagen13.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon13);

                break;
```

```
                case "N":
                    ImageIcon imagen14 = new ImageIcon("src/Imagenes/N.jpg");
                    Icon icon14 = new ImageIcon(imagen14.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon14);
                    break;
                case "O":
                    ImageIcon imagen16 = new ImageIcon("src/Imagenes/O.jpg");
                    Icon icon16 = new ImageIcon(imagen16.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon16);
                    break;
                case "P":
                    ImageIcon imagen17 = new ImageIcon("src/Imagenes/P.jpg");
                    Icon icon17 = new ImageIcon(imagen17.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon17);
                    break;
                case "Q":
                    ImageIcon imagen18 = new ImageIcon("src/Imagenes/Q.gif");
                    Icon icon18 = new ImageIcon(imagen18.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon18);
                    break;
                case "R":
                    ImageIcon imagen19 = new ImageIcon("src/Imagenes/R.jpg");
                    Icon icon19 = new ImageIcon(imagen19.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon19);
                    break;
                case "S":
                    ImageIcon imagen20 = new ImageIcon("src/Imagenes/S.jpg");
                    Icon icon20 = new ImageIcon(imagen20.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon20);
```

```
                    break;

                case "T":

                    ImageIcon imagen21 = new ImageIcon("src/Imagenes/T.jpg");

                    Icon icon21 = new ImageIcon(imagen21.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon21);

                    break;

                case "U":

                    ImageIcon imagen22 = new ImageIcon("src/Imagenes/U.jpg");

                    Icon icon22 = new ImageIcon(imagen22.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon22);

                    break;

                case "V":

                    ImageIcon imagen23 = new ImageIcon("src/Imagenes/V.jpg");

                    Icon icon23 = new ImageIcon(imagen23.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon23);

                    break;

                case "W":

                    ImageIcon imagen24 = new ImageIcon("src/Imagenes/W.jpg");

                    Icon icon24 = new ImageIcon(imagen24.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon24);

                    break;

                case "X":

                    ImageIcon imagen25 = new ImageIcon("src/Imagenes/X.gif");

                    Icon icon25 = new ImageIcon(imagen25.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                    jLabel4.setIcon(icon25);

                    break;

                case "Y":

                    ImageIcon imagen26 = new ImageIcon("src/Imagenes/Y.jpg");

                    Icon icon26 = new ImageIcon(imagen26.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
```

```
                jLabel4.setIcon(icon26);

                break;

            case "Z":

                ImageIcon imagen27 = new ImageIcon("src/Imagenes/Z.gif");

                Icon icon27 = new ImageIcon(imagen27.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon27);

                break;

            case "Enero":

                ImageIcon imagen28 = new ImageIcon("src/Imagenes/Enero.gif");

                Icon icon28 = new ImageIcon(imagen28.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon28);

                break;

            case "╣Ä":

                jLabel1.setText("Ñ");

                ImageIcon imagen15 = new ImageIcon("src/Imagenes/NN.gif");

                Icon icon15 = new ImageIcon(imagen15.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon15);

                break;

            case "Febrero":

                ImageIcon imagen29 = new ImageIcon("src/Imagenes/Febrero.gif");

                Icon icon29 = new ImageIcon(imagen29.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon29);

                break;

            case "Marzo":

                ImageIcon imagen30 = new ImageIcon("src/Imagenes/Marzo.gif");

                Icon icon30 = new ImageIcon(imagen30.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon30);

                break;

            case "Abril":
```

```
                ImageIcon imagen31 = new ImageIcon("src/Imagenes/Abril.gif");

                Icon icon31 = new ImageIcon(imagen31.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon31);

                break;

              case "Mayo":

                ImageIcon imagen32 = new ImageIcon("src/Imagenes/Mayo.gif");

                Icon icon32 = new ImageIcon(imagen32.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon32);

                break;

              case "Junio":

                ImageIcon imagen33 = new ImageIcon("src/Imagenes/Junio.gif");

                Icon icon33 = new ImageIcon(imagen33.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon33);

                break;

              case "Julio":

                ImageIcon imagen34 = new ImageIcon("src/Imagenes/Julio.gif");

                Icon icon34 = new ImageIcon(imagen34.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon34);

                break;

              case "Agosto":

                ImageIcon imagen35 = new ImageIcon("src/Imagenes/Agosto.gif");

                Icon icon35 = new ImageIcon(imagen35.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon35);

                break;

              case "Septiembre":

                ImageIcon imagen36 = new ImageIcon("src/Imagenes/Septiembre.gif");

                Icon icon36 = new ImageIcon(imagen36.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon36);

                break;
```

```
                case "Octubre":
                    ImageIcon imagen37 = new ImageIcon("src/Imagenes/Octubre.gif");
                    Icon icon37 = new ImageIcon(imagen37.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon37);
                    break;
                case "Noviembre":
                    ImageIcon imagen38 = new ImageIcon("src/Imagenes/Noviembre.gif");
                    Icon icon38 = new ImageIcon(imagen38.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon38);
                    break;
                case "Diciembre":
                    ImageIcon imagen39 = new ImageIcon("src/Imagenes/Diciembre.gif");
                    Icon icon39 = new ImageIcon(imagen39.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon39);
                    break;
                case "Lunes":
                    ImageIcon imagen40 = new ImageIcon("src/Imagenes/Lunes.gif");
                    Icon icon40 = new ImageIcon(imagen40.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon40);
                    break;
                case "Martes":
                    ImageIcon imagen41 = new ImageIcon("src/Imagenes/Martes.gif");
                    Icon icon41 = new ImageIcon(imagen41.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon41);
                    break;
                case "Miercoles":
                    jLabel1.setText("Miércoles");
                    ImageIcon imagen42 = new ImageIcon("src/Imagenes/Miercoles.gif");
                    Icon icon42 = new ImageIcon(imagen42.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
```

```
                jLabel4.setIcon(icon42);

                break;

            case "Jueves":

                ImageIcon imagen43 = new ImageIcon("src/Imagenes/Jueves.gif");

                Icon icon43 = new ImageIcon(imagen43.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon43);

                break;

            case "Viernes":

                ImageIcon imagen44 = new ImageIcon("src/Imagenes/Viernes.gif");

                Icon icon44 = new ImageIcon(imagen44.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon44);

                break;

            case "Sabado":

                jLabel1.setText("Sábado");

                ImageIcon imagen45 = new ImageIcon("src/Imagenes/Sabado.gif");

                Icon icon45 = new ImageIcon(imagen45.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon45);

                break;

            case "Domingo":

                ImageIcon imagen46 = new ImageIcon("src/Imagenes/Domingo.gif");

                Icon icon46 = new ImageIcon(imagen46.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon46);

                break;

            case "Uno":

                ImageIcon imagen47 = new ImageIcon("src/Imagenes/Uno.jpg");

                Icon icon47 = new ImageIcon(imagen47.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon47);

                break;

            case "Dos":

                ImageIcon imagen48 = new ImageIcon("src/Imagenes/Dos.jpg");
```

```
                ImageIcon icon48 = new ImageIcon(imagen48.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon48);

                break;

            case "Tres":

                ImageIcon imagen49 = new ImageIcon("src/Imagenes/Tres.jpg");

                Icon icon49 = new ImageIcon(imagen49.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon49);

                break;

            case "Cuatro":

                ImageIcon imagen50 = new ImageIcon("src/Imagenes/Cuatro.jpg");

                Icon icon50 = new ImageIcon(imagen50.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon50);

                break;

            case "Cinco":

                ImageIcon imagen51 = new ImageIcon("src/Imagenes/Cinco.jpg");

                Icon icon51 = new ImageIcon(imagen51.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon51);

                break;

            case "Seis":

                ImageIcon imagen52 = new ImageIcon("src/Imagenes/Seis.jpg");

                Icon icon52 = new ImageIcon(imagen52.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon52);

                break;

            case "Siete":

                ImageIcon imagen53 = new ImageIcon("src/Imagenes/Siete.jpg");

                Icon icon53 = new ImageIcon(imagen53.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon53);

                break;

            case "Ocho":
```

```
                ImageIcon imagen54 = new ImageIcon("src/Imagenes/Ocho.jpg");

                Icon icono54 = new ImageIcon(imagen54.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icono54);

                break;

              case "Nueve":

                ImageIcon imagen55 = new ImageIcon("src/Imagenes/Nueve.gif");

                Icon icon55 = new ImageIcon(imagen55.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon55);

                break;

              case "Diez":

                ImageIcon imagen56 = new ImageIcon("src/Imagenes/Diez.gif");

                Icon icon56 = new ImageIcon(imagen56.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon56);

                break;

              case "Hola":

                ImageIcon imagen57 = new ImageIcon("src/Imagenes/Hola.gif");

                Icon icon57 = new ImageIcon(imagen57.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon57);

                break;

              case "Bien":

                ImageIcon imagen58 = new ImageIcon("src/Imagenes/Bien.gif");

                Icon icon58 = new ImageIcon(imagen58.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon58);

                break;

              case "Mal":

                ImageIcon imagen59 = new ImageIcon("src/Imagenes/Mal.gif");

                Icon icon59 = new ImageIcon(imagen59.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));

                jLabel4.setIcon(icon59);

                break;
```

```
                case "Perdon":
                    jLabel1.setText("Perdón");
                    ImageIcon imagen60 = new ImageIcon("src/Imagenes/Perdon.jpg");
                    Icon icon60 = new ImageIcon(imagen60.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon60);
                    break;
                case "Si":
                    ImageIcon imagen61 = new ImageIcon("src/Imagenes/Si.gif");
                    Icon icon61 = new ImageIcon(imagen61.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon61);
                    break;
                case "No":
                    ImageIcon imagen62 = new ImageIcon("src/Imagenes/No.gif");
                    Icon icon62 = new ImageIcon(imagen62.getImage().getScaledInstance(jLabel4.getWidth(),
jLabel4.getHeight(), Image.SCALE_DEFAULT));
                    jLabel4.setIcon(icon62);
                    break;
                default:
                    jLabel4.setIcon(null);
                    break;
            }

            multi.flushBuffer();
          }
        } catch (ArduinoException ex) {
          Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SerialPortException ex) {
          Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
        }
      }
    };
```

```
public Principal() {
    initComponents();
    this.setLocationRelativeTo(null);
    this.setTitle("Traductor de LSM");
    jLabel5.setEnabled(false);
}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jDialog1 = new javax.swing.JDialog();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    jButton4 = new javax.swing.JButton();
    jLabel3 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jButton5 = new javax.swing.JButton();

    javax.swing.GroupLayout jDialog1Layout = new javax.swing.GroupLayout(jDialog1.getContentPane());
    jDialog1.getContentPane().setLayout(jDialog1Layout);
    jDialog1Layout.setHorizontalGroup(
        jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 400, Short.MAX_VALUE)
    );
    jDialog1Layout.setVerticalGroup(
        jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 300, Short.MAX_VALUE)
    );
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 36)); // NOI18N

jLabel2.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N
jLabel2.setText("Seña:");

jButton1.setText("Iniciar");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Pausar");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton3.setText("Borrar");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jButton4.setText("Datos");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
```

```
        }
    });

    jButton5.setText("Validación");
    jButton5.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton5ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(38, 38, 38)
            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 391,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                    .addGap(51, 51, 51)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE, 81, Short.MAX_VALUE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE, 81, Short.MAX_VALUE)
                        .addComponent(jButton4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGap(73, 73, 73))
                .addGroup(layout.createSequentialGroup()
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                .addGroup(layout.createSequentialGroup()
                    .addGap(77, 77, 77)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 219,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 235,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jLabel3))))
                .addGroup(layout.createSequentialGroup()
                    .addGap(123, 123, 123)
                    .addComponent(jButton5)))
                .addContainerGap(30, Short.MAX_VALUE))))
        .addGroup(layout.createSequentialGroup()
            .addGap(64, 64, 64)
            .addComponent(jLabel5)
            .addGap(0, 0, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addGroup(layout.createSequentialGroup()
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 88,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()
                            .addGap(44, 44, 44)
                            .addComponent(jLabel3)))
                    .addGap(64, 64, 64)
```

```
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(73, 73, 73)

                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

                  .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE, 37, Short.MAX_VALUE)

                  .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

                .addGap(50, 50, 50)

                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

                  .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE, 37, Short.MAX_VALUE)

                  .addComponent(jButton4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE))

              .addGroup(layout.createSequentialGroup()

                .addGap(38, 38, 38)

                .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 482,
javax.swing.GroupLayout.PREFERRED_SIZE)))

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addComponent(jLabel5)

            .addContainerGap(21, Short.MAX_VALUE))

      );


      pack();

    }// </editor-fold>


    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

      dato = true;

      jLabel2.setEnabled(false);

      jLabel3.setEnabled(false);


      try {

        ino.arduinoRX("COM3", 115200, listener);
```

```
    } catch (ArduinoException ex) {
        Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SerialPortException ex) {
        Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    jLabel3.setEnabled(true);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    jLabel2.setEnabled(true);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    (new Lectura(this, false)).setVisible(true);
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    (new Validacion(this, false)).setVisible(true);
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
```

```
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Principal.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Principal.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Principal.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Principal.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            new Principal().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
```

```
    private javax.swing.JButton jButton5;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    // End of variables declaration
}
package com.mycompany.interfazfinal;

import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JScrollPane;
import javax.swing.Timer;
import javax.swing.table.DefaultTableModel;

public class Lectura extends javax.swing.JDialog {
    public static DefaultTableModel modelo;
    String label = "";
    String Tabla_valor = "";
    int fila = 0;
    String[] array = new String[19];
    Timer timer = new Timer(50, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            fila = modelo.getRowCount();
            Tabla_valor = (String) modelo.getValueAt(fila - 1, 0);
            if (Tabla_valor.equals(label)) {
                array = Principal.getDatos();
                if (array[0].equals("╠Д")) {
```

```
            array[0] = "Ñ";
          }
          label = array[0];
        } else {

          array = Principal.getDatos();
          if (array[0].equals("╠Д")) {
            array[0] = "Ñ";
          }
          label = array[0];
          modelo.addRow(array);
        }
        jTable1.setRowSelectionInterval(fila - 1, fila - 1);
        int y = fila * jTable1.getRowHeight();
        int w = jTable1.getWidth();
        int h = jTable1.getHeight();
        jScrollPane1.getViewport().scrollRectToVisible(new Rectangle(0, y, w, h));
      }
    });

    public Lectura(java.awt.Frame parent, boolean modal) {
      super(parent, modal);
      initComponents();
      this.setLocationRelativeTo(null);
      this.setTitle("Traductor de LSM");

      jScrollPane1.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
      jScrollPane1.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

      timer.start();

      modelo = new DefaultTableModel();
```

```
    modelo.addColumn("Seña");
    modelo.addColumn("S1");
    modelo.addColumn("S2");
    modelo.addColumn("S3");
    modelo.addColumn("S4");
    modelo.addColumn("S5");
    modelo.addColumn("S6");
    modelo.addColumn("S7");
    modelo.addColumn("S8");
    modelo.addColumn("S9");
    modelo.addColumn("S10");
    modelo.addColumn("S11");
    modelo.addColumn("S12");
    modelo.addColumn("H");
    modelo.addColumn("Mov");
    modelo.addColumn("M1x");
    modelo.addColumn("M1y");
    modelo.addColumn("M2x");
    modelo.addColumn("M2y");
    this.jTable1.setModel(modelo);
    modelo.addRow(Principal.getDatos());
    label = (String) modelo.getValueAt(0, 0);
}
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jScrollPane1 = new javax.swing.JScrollPane();
    jTable1 = new javax.swing.JTable();
    jButton1 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
```

```
jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {

    }
));
jScrollPane1.setViewportView(jTable1);

jButton1.setText("Borrar");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
jLabel1.setText("Lectura de sensores");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 941, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton1)))
```

```
            .addContainerGap())
        .addGroup(layout.createSequentialGroup()
            .addGap(373, 373, 373)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 226,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 332, Short.MAX_VALUE)
            .addGap(18, 18, 18)
            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap())
    );

    pack();
  }// </editor-fold>

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    modelo.setNumRows(0);
    modelo.addRow(Principal.getDatos());
    label = (String) modelo.getValueAt(0, 0);
  }
  /**
   * @param args the command line arguments
   */
  public static void main(String args[]) {
    /* Set the Nimbus look and feel */
```

```
//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
/* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
 * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
 */
try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(Lectura.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(Lectura.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(Lectura.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(Lectura.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the dialog */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        Lectura dialog = new Lectura(new javax.swing.JFrame(), true);
        dialog.addWindowListener(new java.awt.event.WindowAdapter() {
            @Override
            public void windowClosing(java.awt.event.WindowEvent e) {
```

```
                System.exit(0);
            }
        });
        dialog.setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
// End of variables declaration
}

package com.mycompany.interfazfinal;

import javax.swing.JOptionPane;

public class Validacion extends javax.swing.JDialog {

String[] lectura = new String[19];
String validar = "";

public Validacion(java.awt.Frame parent, boolean modal) {
    super(parent, modal);
    initComponents();
    this.setLocationRelativeTo(null);
    this.setTitle("Traductor de LSM");
    jLabel1.setVisible(false);
}
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jTextField1 = new javax.swing.JTextField();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

    jTextField1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

    jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    jLabel1.setForeground(new java.awt.Color(0, 204, 0));
    jLabel1.setText("Lectura correcta");

    jLabel2.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
    jLabel2.setText("Validación de lectura");

    jButton1.setText("Borrar");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });
```

```
        jButton2.setText("Validar");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(65, 65, 65)
                .addComponent(jButton2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addGap(63, 63, 63))
            .addGroup(layout.createSequentialGroup()
                .addGap(119, 119, 119)
                .addComponent(jLabel2)
                .addGap(23, 119, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 126,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 114,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(139, 139, 139))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

```
            .addGap(25, 25, 25)
            .addComponent(jLabel2)
            .addGap(34, 34, 34)
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 52,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 40, Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(jButton2)
              .addComponent(jButton1))
            .addGap(42, 42, 42))
      );

      pack();
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      lectura = Principal.getDatos();
      validar = jTextField1.getText();
      if(lectura[0].equals("╣Ñ")){
        lectura[0]="Ñ";
      }
      if(validar.equals(lectura[0])){
        jLabel1.setVisible(true);
      }else
        JOptionPane.showMessageDialog(null, "Error en la lectura. ");
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
jTextField1.setText(null);
    }
```

```
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Validacion.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Validacion.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Validacion.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Validacion.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the dialog */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            Validacion dialog = new Validacion(new javax.swing.JFrame(), true);
            dialog.addWindowListener(new java.awt.event.WindowAdapter() {
```

```
                @Override
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            dialog.setVisible(true);
        }
    });
  }

  // Variables declaration - do not modify
  private javax.swing.JButton jButton1;
  private javax.swing.JButton jButton2;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel2;
  private javax.swing.JTextField jTextField1;
  // End of variables declaration
}
```

Apéndice 4. Código en Colab de la RNA.

```
%tensorflow_version 2.x


import tensorflow as tf
from tensorflow.keras import layers
from keras.utils import to_categorical
import pandas as pd
import numpy as np
import io
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split


!python --version
print ('Tensorflow '+ tf.__version__)
print ('Keras '+ tf.keras.__version__)


from google.colab import files
uploaded = files.upload()


datos = pd.read_csv(io.BytesIO(uploaded['LA_FINAL_COMPARACION(PRUEBA10).csv']))


sensor= datos.values[:,:17]
resultado= datos.values[:,17]
resultado__OHE = to_categorical(resultado)
print ('Número de pruebas: ', len(resultado))


X_train, X_test, y_train, y_test = train_test_split(sensor, resultado__OHE,
                                                    test_size=0.2,
                                                    random_state=21)


model= tf.keras.Sequential()
model.add(layers.Dense(16, activation='relu', input_dim=17))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(62, activation='sigmoid'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=1500, batch_size=5)


loss = history.history['loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'b')
plt.title('Pérdida en el entrenamiento')


res = model.evaluate(X_test, y_test)
```

```
print("\n%s: %.2f%%" % (model.metrics_names[1], res[1]*100))


converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
tflite_model = converter.convert()
open("modelo.tflite", "wb").write(tflite_model)


ejercicio= tf.lite.Interpreter('modelo.tflite')
ejercicio.allocate_tensors()
input_details = ejercicio.get_input_details()
output_details = ejercicio.get_output_details()
X_test = X_test.astype('float32')
test = 40
input_data = X_test[test].reshape((1,17))
ejercicio.set_tensor(input_details[0]['index'],input_data )
ejercicio.invoke()
output_data = ejercicio.get_tensor(output_details[0]['index'])
print(output_data)
print(y_test[test])


def hex_to_c_array(hex_data, var_name):

  c_str = ''

  c_str += '#ifndef ' + var_name.upper() + '_H\n'
  c_str += '#define ' + var_name.upper() + '_H\n\n'

  c_str += '\nunsigned int ' + var_name + '_len = ' + str(len(hex_data)) + ';\n'

  c_str += 'unsigned char ' + var_name + '[] = {'
  hex_array = []
  for i, val in enumerate(hex_data) :
    hex_str = format(val, '#04x')
    if (i + 1) < len(hex_data):
      hex_str += ','
    if (i + 1) % 12 == 0:
      hex_str += '\n '
    hex_array.append(hex_str)
  c_str += '\n ' + format(' '.join(hex_array)) + '\n};\n\n'
  c_str += '#endif //' + var_name.upper() + '_H'

  return c_str


c_model_name='modelo_colab'
with open(c_model_name + '.h', 'w') as file:
  file.write(hex_to_c_array(tflite_model, c_model_name))
```

Apéndice 5. Modelo de la RNA.

```
#ifndef MODELO_COLAB_H

#define MODELO_COLAB_H

unsigned int modelo_colab_len = 7848;

unsigned char modelo_colab[] = {

0x08, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0xf6, 0xe5, 0xff, 0xff, 0x03, 0x00, 0x00, 0x00, 0x68, 0x1e, 0x00, 0x00, 0xf4, 0x19,
0x00, 0x00, 0xdc, 0x19, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0xcc, 0x19, 0x00, 0x00, 0xc4, 0x19, 0x00, 0x00,
0x74, 0x19, 0x00, 0x00, 0x1c, 0x19, 0x00, 0x00, 0x14, 0x18, 0x00, 0x00, 0xc4, 0x13, 0x00, 0x00, 0xb4, 0x0f, 0x00, 0x00, 0x24, 0x00,
0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0xe2, 0xff, 0xff,
0x04, 0xe2, 0xff, 0xff, 0x08, 0xe2, 0xff, 0xff, 0x0c, 0xe2, 0xff, 0xff, 0xca, 0xe6, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x80, 0x0f, 0x00,
0x00, 0x45, 0x66, 0x92, 0xbe, 0x30, 0xd3, 0x17, 0xbe, 0x52, 0x48, 0x57, 0xbf, 0x90, 0x01, 0xaf, 0xbf, 0x06, 0x0c, 0xac, 0xbe, 0x00,
0x1f, 0xb9, 0xc0, 0x3b, 0x78, 0xa4, 0xbe, 0x28, 0xb3, 0xf0, 0xbf, 0xda, 0x66, 0x0a, 0x3e, 0x3c, 0x6e, 0xd3, 0xbf, 0x53, 0x74, 0x94,
0x3e, 0xfa, 0xba, 0x46, 0x3f, 0x98, 0x94, 0x53, 0x3f, 0x18, 0xa8, 0xb6, 0xbf, 0xe7, 0x40, 0xe6, 0xc0, 0x22, 0xd0, 0x03, 0xc0, 0x79,
0x5a, 0xbe, 0x3e, 0xa0, 0x15, 0x5d, 0x3f, 0x42, 0xeb, 0x6d, 0xc0, 0xc7, 0xaa, 0x8f, 0xbf, 0xd9, 0x25, 0x8a, 0x3e, 0xd0, 0x8d, 0xf8,
0xbe, 0x19, 0x4e, 0x38, 0xbe, 0x07, 0xf1, 0x10, 0xc0, 0x61, 0x77, 0x49, 0x3f, 0x4d, 0x84, 0x77, 0x3f, 0xa9, 0x50, 0xe3, 0x3d, 0x0c,
0x16, 0x97, 0xc0, 0x4e, 0xaf, 0xa1, 0xc0, 0x31, 0x6f, 0x58, 0xbf, 0xbf, 0xd6, 0xb6, 0xbd, 0x41, 0x5e, 0x6d, 0x3f, 0x9b, 0x0d, 0x0f,
0x3f, 0xc3, 0xd4, 0xd7, 0xbe, 0x97, 0x60, 0x21, 0xc1, 0xe3, 0x16, 0x00, 0xc0, 0x38, 0x8a, 0x2f, 0xbe, 0xd1, 0x2b, 0x32, 0xbf, 0xda,
0x45, 0x7d, 0xbf, 0x6f, 0xf4, 0x53, 0xc0, 0xe0, 0x1c, 0xdf, 0xbe, 0x37, 0x01, 0x61, 0x3f, 0x2b, 0xc6, 0xa3, 0x3e, 0xa8, 0xda, 0xf2,
0xbf, 0x96, 0x49, 0x88, 0xbf, 0x8e, 0xf4, 0xf2, 0x3e, 0x7f, 0x82, 0xa7, 0xbd, 0x1a, 0xae, 0x6a, 0x3f, 0xf1, 0x13, 0x2d, 0xbf, 0xaf,
0xf4, 0xe2, 0x3c, 0xc3, 0x69, 0x78, 0xbf, 0x61, 0xb6, 0x4a, 0x3e, 0x08, 0xf2, 0x1b, 0x3f, 0xea, 0xb0, 0x4e, 0xc0, 0x98, 0xb3, 0xac,
0xbe, 0x73, 0x67, 0xe4, 0xbe, 0xae, 0x2e, 0x86, 0x3e, 0x92, 0x9c, 0x47, 0xbf, 0x4d, 0x78, 0x75, 0xbf, 0x23, 0x5b, 0x6f, 0x3b, 0x9b,
0x7e, 0x1b, 0xbf, 0xe7, 0xda, 0xb8, 0xbf, 0xc8, 0xd7, 0xd5, 0xbd, 0xcc, 0x60, 0x2e, 0xc0, 0xcc, 0x03, 0x1a, 0xc0, 0xed, 0x31, 0xec,
0x3e, 0x44, 0x58, 0x3f, 0xc0, 0x67, 0x02, 0xb7, 0x3f, 0x0e, 0x30, 0x98, 0x3f, 0x29, 0x8f, 0xcc, 0xc0, 0xb0, 0x50, 0x87, 0xbf, 0x8a,
0x8a, 0x1e, 0xbf, 0x07, 0xf9, 0x79, 0x3f, 0x5e, 0x42, 0xf6, 0xbe, 0x7e, 0x29, 0x11, 0x3e, 0x63, 0x13, 0x47, 0x3f, 0x68, 0xdc, 0xc5,
0xbf, 0xc6, 0x1f, 0x91, 0xc0, 0x49, 0x1c, 0x20, 0xbd, 0xbd, 0x53, 0x5e, 0xc0, 0xa9, 0x4d, 0x3c, 0xc0, 0x91, 0x50, 0x9a, 0x3e, 0x7f,
0x31, 0x89, 0xc0, 0x5f, 0x5d, 0x50, 0x3f, 0x41, 0xe6, 0x7a, 0x3f, 0x7b, 0xe3, 0xb5, 0xbf, 0x6d, 0x60, 0x24, 0xc0, 0x3a, 0x16, 0x4c,
0xbf, 0xf7, 0x95, 0x3a, 0x3f, 0x96, 0xcc, 0x8d, 0x3f, 0xc6, 0xcf, 0xbc, 0xbf, 0xb3, 0xf5, 0xa9, 0xbf, 0x2f, 0xd0, 0xce, 0xc0, 0xd7,
0x95, 0xf4, 0xbf, 0x25, 0xba, 0x99, 0x3f, 0xcb, 0x2f, 0x78, 0xbf, 0xdf, 0x1c, 0x6d, 0xbe, 0x48, 0x6b, 0x0f, 0x3f, 0x6f, 0x9d, 0xd4,
0xbf, 0xcb, 0x11, 0x01, 0xc0, 0x49, 0x24, 0x79, 0x3f, 0x1b, 0xbc, 0x0e, 0xc1, 0x39, 0x85, 0x9c, 0xbe, 0x67, 0x17, 0xc9, 0xbe, 0x78,
0x61, 0x3c, 0x3f, 0x4d, 0x38, 0x0b, 0xc1, 0x26, 0x76, 0x22, 0xc1, 0x55, 0xea, 0x11, 0xbd, 0xdd, 0x84, 0x56, 0xbe, 0x50, 0x28, 0xfd,
0xc0, 0x32, 0x90, 0x24, 0x40, 0x9f, 0x31, 0x8e, 0xbf, 0x34, 0x0d, 0x53, 0x3f, 0xac, 0x8b, 0x8b, 0x3f, 0xd0, 0x9f, 0xf0, 0xbf, 0x7d,
0x19, 0xc2, 0xc0, 0x87, 0xa0, 0x47, 0xc0, 0x6d, 0x3d, 0x96, 0xbf, 0xfb, 0x72, 0xc2, 0xbe, 0xe5, 0xe7, 0x96, 0xbe, 0x1d, 0x00, 0xf0,
0xbe, 0xe8, 0x15, 0xf8, 0x3e, 0x62, 0xaf, 0x5b, 0xc0, 0x6c, 0x38, 0xd2, 0xbf, 0x4a, 0x7f, 0xda, 0x3e, 0x74, 0xec, 0x9b, 0xbf, 0x69,
0xc6, 0x40, 0xbf, 0x4d, 0x89, 0x78, 0x3f, 0xe0, 0xdb, 0xdb, 0xbf, 0xf5, 0x1c, 0xae, 0xbf, 0x96, 0x00, 0xd5, 0xc0, 0x6b, 0x88, 0x49,
0xbf, 0xdb, 0x8a, 0xeb, 0x3e, 0x1d, 0xaf, 0x54, 0x3e, 0x51, 0xbd, 0x48, 0x3f, 0xcf, 0xd1, 0xeb, 0xc0, 0x8f, 0x20, 0x17, 0x3f, 0x6d,
0xd1, 0x68, 0x3e, 0xda, 0x46, 0x9d, 0x3e, 0x69, 0x42, 0xd2, 0xbe, 0x69, 0x17, 0xde, 0xbf, 0xf6, 0xd0, 0xaf, 0xc0, 0x2b, 0xa4, 0xb0,
0xbe, 0x3e, 0x20, 0x0f, 0x3f, 0x79, 0x74, 0x12, 0xc0, 0x74, 0xd7, 0x96, 0x3f, 0xa3, 0x0d, 0x09, 0x3f, 0xc6, 0xcf, 0x99, 0xbe, 0x49,
0x1a, 0xe0, 0xbe, 0x3d, 0xd1, 0xba, 0x3e, 0x68, 0xa0, 0xc5, 0x3e, 0x3e, 0xbe, 0xdc, 0xbe, 0xde, 0xb2, 0x64, 0x3e, 0xac, 0xf0, 0x58,
0xbd, 0x0d, 0x9e, 0xdf, 0xbe, 0x3b, 0x47, 0x9e, 0xbf, 0x18, 0xe5, 0x15, 0xc0, 0x25, 0xa8, 0x8b, 0xc0, 0xa8, 0x27, 0x0a, 0x3f, 0xdd,
0x5e, 0x57, 0xbe, 0x8e, 0x31, 0x96, 0xbe, 0x6b, 0xe0, 0x63, 0xbe, 0x61, 0xa3, 0x40, 0x3f, 0xce, 0xee, 0xee, 0x3d, 0xfb, 0x1e, 0xad,
0xbf, 0x3d, 0xb9, 0x12, 0xbe, 0xc9, 0x94, 0x94, 0xbe, 0xf3, 0xb3, 0x2b, 0x3f, 0x87, 0x1c, 0xcc, 0xc0, 0xc1, 0xf1, 0x3f, 0xc0, 0x9f,
0xb2, 0xbc, 0xbf, 0xd9, 0xa0, 0x8e, 0xbe, 0x0d, 0x22, 0x33, 0xbe, 0xe3, 0x6b, 0x69, 0xbe, 0x93, 0xf8, 0x3b, 0xbe, 0x8b, 0xb8, 0x0f,
0xc0, 0x57, 0x92, 0x74, 0xbf, 0x73, 0xc0, 0x82, 0xbe, 0xdd, 0x60, 0x2a, 0xbf, 0x36, 0xbc, 0x98, 0x3e, 0xb2, 0x69, 0x8a, 0x3e, 0x3e,
0xa7, 0xb2, 0xc0, 0xe9, 0x0f, 0x4f, 0xbf, 0xcd, 0xa8, 0x25, 0xbf, 0xb1, 0xba, 0x5b, 0x3e, 0xae, 0xd9, 0x09, 0xc0, 0x73, 0x5f, 0xa9,
0x3e, 0xb1, 0x16, 0x43, 0x3f, 0xce, 0x7e, 0xf8, 0x3d, 0x1e, 0x19, 0x89, 0xbf, 0x29, 0x76, 0xe7, 0xbf, 0x91, 0xf1, 0xd5, 0xc0, 0xfe,
0x6c, 0x59, 0x3e, 0xc2, 0x1b, 0x96, 0x3d, 0x4a, 0xda, 0x5d, 0xbf, 0xf2, 0xf3, 0x41, 0xc0, 0xfa, 0xde, 0xa7, 0xbf, 0x46, 0xcb, 0x27,
0xbf, 0xdb, 0xcd, 0x0c, 0xbf, 0xdb, 0x0d, 0xe2, 0x3d, 0xa6, 0x9b, 0x97, 0xbf, 0xad, 0xb7, 0x9c, 0xbe, 0x4b, 0x93, 0x00, 0xbf, 0xc7,
0x81, 0xbd, 0xbf, 0xcd, 0xad, 0x18, 0x3d, 0x7d, 0xb3, 0xb1, 0x3e, 0x0c, 0x03, 0x4d, 0xbf, 0xe7, 0xbb, 0x5b, 0x3e, 0x3a, 0x5a, 0x39,
0x3f, 0x44, 0x4e, 0x4e, 0xbd, 0xb5, 0x1c, 0x5c, 0xbf, 0xda, 0xbc, 0x15, 0xc1, 0xfe, 0xb3, 0xbb, 0xbf, 0x41, 0xc7, 0xb0, 0xbf, 0xc9,
0x9c, 0x37, 0xbf, 0x30, 0x62, 0x86, 0xbf, 0x82, 0xf7, 0x23, 0xbf, 0x97, 0x9e, 0x3f, 0xbf, 0xa6, 0xfa, 0xa1, 0x3d, 0x2f, 0x62, 0x99,
0xbe, 0xcf, 0xf2, 0xfe, 0x3e, 0xfc, 0xbf, 0xbc, 0x3e, 0x78, 0xeb, 0xc2, 0xc0, 0xd7, 0xd3, 0x9e, 0xbd, 0xe2, 0x8f, 0x0c, 0x3f, 0x0e,
0x69, 0xaf, 0xbf, 0x32, 0xdb, 0x61, 0xbd, 0x3b, 0x8a, 0x8a, 0xc0, 0xfd, 0x9b, 0xe9, 0xc0, 0xe0, 0x33, 0x77, 0x3e, 0x78, 0xdb, 0x1e,
0x3f, 0xfb, 0x45, 0x7f, 0xbf, 0xe8, 0x90, 0x9f, 0x3e, 0xe4, 0x13, 0x9f, 0xc0, 0x62, 0x42, 0x53, 0xbf, 0xd4, 0xf8, 0x50, 0xbf, 0xb9,
0xbf, 0x94, 0x3d, 0x09, 0xc1, 0x7b, 0xc0, 0x7b, 0xf3, 0x60, 0xc0, 0x53, 0xa7, 0x00, 0x3f, 0x47, 0x00, 0x39, 0xbf, 0xf1, 0x10, 0x76,
0xbf, 0xf6, 0xb2, 0x33, 0xc0, 0x8c, 0xe8, 0x34, 0x3f, 0x57, 0x63, 0x83, 0x3d, 0xa7, 0x65, 0xab, 0xbf, 0x84, 0x55, 0xfd, 0xbf, 0x6f,
0x4a, 0x0f, 0xc0, 0xec, 0xd3, 0xff, 0xbe, 0x42, 0x08, 0x5a, 0x3e, 0x8c, 0x9e, 0x0f, 0x3f, 0xa5, 0xac, 0xce, 0xbe, 0xd1, 0xb9, 0xa3,
0xbf, 0xd9, 0x21, 0x69, 0x3f, 0xcc, 0x6b, 0x48, 0xbe, 0x13, 0x17, 0x44, 0xbf, 0x3c, 0xe2, 0x08, 0xbd, 0x94, 0xb1, 0x8a, 0x3d, 0x8e,
0xce, 0x9a, 0xbe, 0x95, 0xf5, 0xf3, 0x3d, 0x2a, 0xa3, 0x3b, 0xbf, 0x30, 0x59, 0xad, 0xbf, 0x72, 0x8c, 0x46, 0xc0, 0xa3, 0x07, 0xc2,
0xbd, 0x76, 0x5b, 0x43, 0xc0, 0x4e, 0x5e, 0x8c, 0xbf, 0x78, 0x0b, 0x1b, 0xbf, 0x28, 0x06, 0xd4, 0xbe, 0x8f, 0xd4, 0x2f, 0xbe, 0x31,
0x45, 0x89, 0x3f, 0xe8, 0xc7, 0x4d, 0xbf, 0xbd, 0x89, 0xc9, 0xbf, 0x67, 0x3c, 0x01, 0x3d, 0x8e, 0xe4, 0x55, 0xbc, 0x84, 0xb5, 0x6e,
0xbe, 0xf9, 0xa9, 0x62, 0x3d, 0x8a, 0x59, 0x8f, 0xbe, 0x56, 0xe7, 0x87, 0xbf, 0xf6, 0x31, 0x6a, 0x3e, 0x93, 0x2a, 0x0d, 0x3f, 0xcc,
0x6f, 0x81, 0xbc, 0x91, 0x41, 0x1d, 0xc1, 0x4f, 0x7b, 0x9d, 0xc0, 0x77, 0x9a, 0x97, 0xbe, 0x0e, 0x3c, 0x9e, 0xbe, 0xf4, 0x3c, 0x53,
0xbf, 0xd7, 0x06, 0xc3, 0x3f, 0x23, 0x11, 0x6a, 0xc0, 0x93, 0xf7, 0x51, 0xbf, 0x86, 0xb1, 0xc1, 0xbe, 0x4e, 0x78, 0x96, 0xbf, 0x12,
0xb4, 0xa6, 0xbd, 0x13, 0x88, 0xa0, 0x3c, 0xbc, 0xfc, 0x23, 0xbf, 0x9b, 0x54, 0x26, 0xc1, 0xd5, 0x05, 0xa8, 0xbf, 0x73, 0x5b, 0x01,
```

0xc0, 0x58, 0xa9, 0x94, 0xbe, 0x4c, 0xc2, 0x94, 0x3f, 0x97, 0x5d, 0x58, 0x3d, 0xcd, 0xda, 0xa4, 0xbf, 0x16, 0x1e, 0xb0, 0x3f, 0x0b,
0xbd, 0x59, 0xbf, 0x71, 0xf3, 0x0f, 0xbf, 0xdf, 0x37, 0x37, 0xbf, 0x92, 0x96, 0x99, 0xbf, 0xdc, 0x86, 0x9b, 0xbf, 0xff, 0x66, 0x90,
0xbe, 0x95, 0xd5, 0x17, 0xbe, 0xeb, 0xae, 0x25, 0xc0, 0x53, 0xbe, 0x0c, 0xc0, 0x2e, 0xf7, 0x21, 0xc0, 0x69, 0x72, 0xf0, 0xbe, 0x2c,
0x19, 0x0d, 0xbf, 0xf6, 0xe3, 0x7f, 0x3f, 0x1f, 0x72, 0x3d, 0x3e, 0x69, 0x19, 0xea, 0xbd, 0x35, 0xe9, 0xb3, 0x3f, 0xd8, 0x9d, 0x24,
0xc0, 0x43, 0x76, 0x96, 0xbf, 0x8e, 0x50, 0x8d, 0xbf, 0x7f, 0xee, 0x2b, 0x3f, 0x6f, 0x16, 0xfb, 0xbf, 0xfc, 0xea, 0xeb, 0x3e, 0x7c,
0xb3, 0x48, 0x3f, 0x4b, 0x56, 0xc7, 0xc0, 0xda, 0xe0, 0x46, 0xc0, 0x79, 0xa1, 0x05, 0xbe, 0x05, 0xdf, 0xec, 0xbe, 0x8b, 0x02, 0x42,
0x3e, 0x8a, 0x5a, 0x27, 0xbf, 0xe8, 0x29, 0x15, 0xbd, 0x0d, 0xc1, 0x99, 0xbf, 0xd7, 0x6d, 0xb9, 0x3e, 0x0d, 0x5a, 0x06, 0x3f, 0x58,
0xcf, 0xca, 0xc0, 0xe2, 0x75, 0x96, 0x3f, 0x25, 0xb4, 0x19, 0xbf, 0x72, 0xed, 0xf1, 0xbf, 0xc1, 0x48, 0x7b, 0xc0, 0x73, 0xff, 0xc5,
0xbf, 0xde, 0xc9, 0xe6, 0x3e, 0x66, 0xbf, 0x03, 0xc0, 0xc9, 0x58, 0x1b, 0xc0, 0x46, 0x60, 0xd2, 0xbf, 0xf2, 0x59, 0x80, 0x3f, 0xdf,
0x89, 0x91, 0x3f, 0xb9, 0xd1, 0x66, 0xc0, 0x05, 0x3b, 0x4d, 0xbf, 0x13, 0x0f, 0x0a, 0x40, 0x5f, 0x39, 0xa9, 0xbf, 0x00, 0x8c, 0x8f,
0x3f, 0x95, 0x51, 0x21, 0xbe, 0x20, 0xc4, 0xe1, 0xbe, 0xea, 0x7f, 0x1d, 0xbf, 0x2d, 0x10, 0xdd, 0xbc, 0xc0, 0x2e, 0xe3, 0xbe, 0x6b,
0x3d, 0x0c, 0xbe, 0xdc, 0x75, 0x44, 0xc0, 0x1a, 0x05, 0x85, 0xbe, 0xfa, 0xc5, 0xb6, 0xbf, 0x26, 0x07, 0x0a, 0x3c, 0xd4, 0xd5, 0xf9,
0xbe, 0x95, 0xf6, 0x71, 0xbf, 0x0e, 0x01, 0x61, 0xbf, 0x4d, 0x7a, 0xe7, 0x3e, 0xb7, 0xce, 0x97, 0xbe, 0x0c, 0x9d, 0x7c, 0xbe, 0x48,
0x2a, 0xb5, 0xbe, 0xcb, 0x4a, 0xac, 0x3e, 0xd0, 0x6c, 0x08, 0xbf, 0x9b, 0x04, 0x20, 0x3f, 0x5f, 0x50, 0x27, 0xbf, 0x0a, 0x44, 0xe9,
0xbe, 0xd3, 0x98, 0xaf, 0xc0, 0x86, 0xae, 0x43, 0x3f, 0x6d, 0x45, 0x9b, 0xc0, 0x38, 0xda, 0x12, 0xbe, 0x5a, 0x56, 0x1f, 0xc0, 0x45,
0x1f, 0xb7, 0xbf, 0xd7, 0x63, 0xf5, 0xbf, 0xbe, 0x17, 0x92, 0x3f, 0xdf, 0x2c, 0x24, 0x3f, 0x65, 0x98, 0x77, 0xbf, 0xa8, 0x85, 0xb0,
0x3d, 0x0d, 0x21, 0x43, 0x3e, 0xd4, 0x6d, 0x86, 0x3b, 0x7f, 0x48, 0x3b, 0xbe, 0x5c, 0x06, 0x7e, 0xbf, 0xbc, 0x4a, 0x62, 0xc0, 0xf9,
0x6f, 0x65, 0x3e, 0xfb, 0x5b, 0x9d, 0x3e, 0xe9, 0xa9, 0xc4, 0xbe, 0x9f, 0xf3, 0x9b, 0xc0, 0xf4, 0xec, 0x13, 0xc0, 0x03, 0x99, 0xa7,
0x3e, 0x58, 0x44, 0x5f, 0x3e, 0xd5, 0xd9, 0x85, 0xc0, 0xfd, 0x18, 0x8c, 0xc0, 0xe2, 0x3b, 0xea, 0xc0, 0xe5, 0xc6, 0x8b, 0x3e, 0x70,
0xbd, 0x2a, 0x3f, 0xbb, 0xd9, 0x81, 0xc0, 0x04, 0x8f, 0x60, 0xc0, 0xad, 0xdd, 0xda, 0xbe, 0x2d, 0xa9, 0x74, 0xc0, 0x9d, 0x25, 0x0a,
0x3e, 0x5d, 0xc6, 0xa4, 0xbf, 0x2e, 0xfa, 0x2e, 0x3f, 0x6d, 0xf2, 0x43, 0xbf, 0xd4, 0xd6, 0x3d, 0xc0, 0x55, 0xfd, 0x2c, 0xbf, 0x04,
0xde, 0xda, 0xbe, 0xfa, 0x93, 0x3b, 0xc0, 0xc5, 0x41, 0xfd, 0xbe, 0x71, 0xc9, 0x07, 0xbc, 0xfd, 0x61, 0x0f, 0x3f, 0x25, 0x5d, 0x27,
0x3f, 0xd8, 0xf9, 0x54, 0xbe, 0x2c, 0x41, 0x7c, 0xbe, 0x3e, 0x22, 0xc1, 0xbf, 0x9c, 0x33, 0xf2, 0xbd, 0x03, 0xb2, 0xcd, 0xbe, 0xd0,
0x07, 0xb1, 0x3e, 0xa6, 0xa8, 0xe3, 0xbf, 0xf0, 0x7c, 0x1a, 0xc1, 0x91, 0x3e, 0xce, 0xc0, 0x37, 0x85, 0x5c, 0xbf, 0xc1, 0xe8, 0x59,
0xbf, 0x27, 0x3e, 0xb0, 0xbf, 0xf3, 0xaa, 0x04, 0x40, 0xed, 0xe4, 0xa6, 0x3f, 0x09, 0xd6, 0x7e, 0xc0, 0xab, 0x49, 0x9c, 0xbf, 0x37,
0x3a, 0xb2, 0xbe, 0xf4, 0x68, 0xeb, 0x3e, 0xee, 0xf2, 0x3f, 0x3f, 0xcf, 0x35, 0xf9, 0xbf, 0x76, 0xc9, 0x65, 0xc0, 0xf7, 0x97, 0x77,
0xbc, 0x80, 0x90, 0x34, 0xbe, 0xa8, 0x41, 0x8a, 0xbf, 0x35, 0xa7, 0x95, 0x3e, 0xe1, 0x7c, 0xfa, 0x3e, 0x82, 0x99, 0xd4, 0xbe, 0x58,
0x55, 0xdb, 0xbe, 0x37, 0x1c, 0x2b, 0xbe, 0x0b, 0x94, 0x34, 0xc1, 0x9d, 0xbd, 0xc3, 0xbf, 0xcf, 0x6c, 0x90, 0xbf, 0xfa, 0xc3, 0x8f,
0xbe, 0x16, 0xcb, 0xd3, 0x3e, 0xca, 0x91, 0x71, 0x3e, 0x77, 0x4b, 0xb2, 0xbe, 0x70, 0xe6, 0x6a, 0xc0, 0x89, 0x9b, 0x2d, 0xbe, 0xa6,
0xd7, 0xce, 0xbf, 0xd0, 0x1e, 0xf4, 0x3e, 0x4a, 0x6b, 0x19, 0x3d, 0x2a, 0xd4, 0x58, 0xbf, 0x66, 0xb3, 0x09, 0xbf, 0x58, 0xe4, 0x0f,
0x3f, 0xcd, 0x34, 0x77, 0x3e, 0x70, 0x1b, 0x3c, 0xc0, 0xff, 0x25, 0x72, 0xbe, 0xdd, 0x7a, 0x1e, 0xbf, 0x13, 0xd9, 0x20, 0x3f, 0x94,
0x42, 0x15, 0xbc, 0xa6, 0x51, 0x65, 0xc0, 0x5b, 0x34, 0x5a, 0x3e, 0xc2, 0x73, 0x7b, 0x3d, 0xee, 0x3e, 0x93, 0x3e, 0x92, 0x77, 0x2b,
0xc0, 0xdc, 0x37, 0x7f, 0xbf, 0x2e, 0xfb, 0xba, 0xbf, 0x97, 0x01, 0x00, 0xc0, 0x8c, 0x24, 0xb7, 0x3d, 0xd6, 0xd8, 0x23, 0xbe, 0xa9,
0xf2, 0xa4, 0x3e, 0x4f, 0x8a, 0x29, 0xbd, 0xa8, 0x13, 0x10, 0xc0, 0xb1, 0x32, 0x1c, 0xc1, 0x9b, 0x84, 0x55, 0x3e, 0x1b, 0x42, 0xc3,
0xbe, 0xbb, 0xae, 0x77, 0x3e, 0xf7, 0x31, 0xa6, 0xbe, 0xc6, 0x19, 0x86, 0x3e, 0x9c, 0x97, 0x04, 0xc0, 0x73, 0xac, 0x44, 0x3f, 0x73,
0x87, 0x0b, 0xc0, 0x4d, 0xa8, 0x18, 0x3f, 0x1e, 0xf6, 0x14, 0x3f, 0xf6, 0x39, 0x99, 0xbe, 0xe9, 0x37, 0x0d, 0xbf, 0xde, 0x16, 0xbe,
0xc0, 0x9f, 0x2e, 0xd3, 0xbf, 0x25, 0xbb, 0x0e, 0xbf, 0xf7, 0xff, 0x05, 0xbf, 0x19, 0x6d, 0x59, 0x3e, 0x5f, 0x5e, 0x36, 0x3f, 0x76,
0x97, 0x02, 0xbf, 0x5c, 0x4d, 0x6f, 0x3e, 0xe8, 0xc6, 0xef, 0xbf, 0x2c, 0xef, 0x8b, 0xbd, 0xde, 0xcf, 0xc4, 0xc0, 0xc8, 0x3c, 0x87,
0x3e, 0x50, 0x4f, 0x1a, 0x3f, 0x28, 0xe3, 0x43, 0xc0, 0xcd, 0xde, 0x20, 0xc0, 0xcb, 0x56, 0xd7, 0x3e, 0x86, 0xe2, 0xf7, 0x3c, 0x51,
0x48, 0xf4, 0xbe, 0xd8, 0x94, 0x9a, 0xbf, 0x94, 0xa9, 0xc8, 0xbf, 0xd6, 0xc2, 0x67, 0xbf, 0x13, 0xb4, 0x11, 0xbe, 0xcd, 0xb7, 0x0e,
0x3f, 0x30, 0x85, 0xa4, 0xbf, 0xcc, 0xf5, 0x08, 0xbf, 0x56, 0xb9, 0xba, 0xbf, 0xaa, 0x99, 0x21, 0xbd, 0xaf, 0x81, 0x11, 0x3e, 0x87,
0x73, 0x50, 0x3f, 0x3f, 0x9b, 0x55, 0xbe, 0x7c, 0x7b, 0x94, 0xc0, 0x1a, 0xa3, 0x91, 0x3f, 0x56, 0xc9, 0x60, 0xbe, 0xcc, 0x2c, 0x0d,
0xbf, 0x87, 0x8d, 0x58, 0xc0, 0xfc, 0x31, 0x66, 0x3f, 0x28, 0x17, 0x14, 0xbf, 0xb9, 0x74, 0xec, 0x3e, 0x28, 0xf5, 0x85, 0x3f, 0xda,
0x2d, 0xc4, 0xbf, 0xa6, 0x83, 0xda, 0xbe, 0xeb, 0xb4, 0x88, 0xbe, 0x47, 0x6b, 0xa3, 0x3f, 0x33, 0x2b, 0x27, 0xbf, 0x50, 0x3c, 0xaf,
0xbf, 0x99, 0x98, 0x0f, 0x3f, 0x68, 0x0f, 0x80, 0xc0, 0xc5, 0x47, 0x3b, 0xc0, 0x84, 0x6e, 0x52, 0x3f, 0x8e, 0xc9, 0x7e, 0xc0, 0xe0,
0xe0, 0xe1, 0xbf, 0x45, 0xcf, 0x45, 0xbf, 0x09, 0xa5, 0x14, 0x3f, 0xcd, 0xd8, 0x89, 0x3d, 0x9b, 0xff, 0x84, 0xbf, 0x5c, 0x9d, 0x39,
0xbf, 0xd3, 0x57, 0x59, 0xbf, 0x38, 0x63, 0x0d, 0xbf, 0x13, 0x58, 0xf9, 0xbf, 0xd3, 0x95, 0x05, 0xc0, 0x98, 0x92, 0xcf, 0x3e, 0x5d,
0xff, 0x4e, 0x3e, 0xe3, 0x94, 0xb5, 0x3e, 0x9a, 0x52, 0x0f, 0x3f, 0xa1, 0xa0, 0x10, 0xc0, 0x42, 0x13, 0x33, 0xc0, 0x65, 0x41, 0x9a,
0xbf, 0xe9, 0x9e, 0x0d, 0xc0, 0xe7, 0xab, 0x32, 0x3e, 0x8d, 0x61, 0x25, 0x3e, 0x05, 0x02, 0xb6, 0xbe, 0x07, 0xd4, 0x17, 0xbf, 0x96,
0x31, 0xa6, 0xbf, 0xec, 0x06, 0x50, 0xbf, 0x78, 0xdf, 0xc4, 0xbf, 0x01, 0x9b, 0xb3, 0xbe, 0x3a, 0x2b, 0x17, 0x3f, 0x1e, 0x37, 0x73,
0xbf, 0x39, 0x0a, 0xc6, 0xbe, 0x54, 0x7e, 0x88, 0x3f, 0x7a, 0x88, 0x61, 0xbf, 0x5a, 0xa4, 0x8e, 0xbf, 0x50, 0x1a, 0xaf, 0xbf, 0x50,
0x3a, 0xbd, 0xbf, 0xc1, 0x48, 0xc1, 0xbe, 0x4b, 0xc1, 0x95, 0x3d, 0x0c, 0x46, 0x80, 0xbe, 0x3d, 0xd9, 0x9b, 0xbe, 0x7c, 0x04, 0xa0,
0xbf, 0xdd, 0x86, 0xa2, 0xbf, 0xcb, 0x5c, 0x36, 0xbf, 0xcf, 0xf1, 0x8b, 0x3e, 0x9e, 0xa9, 0x65, 0x3f, 0x09, 0x39, 0x41, 0xbf, 0xb9,
0xd9, 0xef, 0xbf, 0x55, 0x03, 0x84, 0x3f, 0x42, 0x02, 0x2b, 0xbe, 0xde, 0xb2, 0xf2, 0xbe, 0x0e, 0x58, 0x8c, 0xbe, 0x84, 0x74, 0x86,
0xbe, 0x6b, 0x3f, 0x36, 0xbe, 0x91, 0x4b, 0xbd, 0xbe, 0xdc, 0xa3, 0x96, 0xbf, 0x15, 0x29, 0x8b, 0x3e, 0x5d, 0xea, 0x1c, 0xc0, 0x55,
0xf3, 0xb3, 0xbf, 0x13, 0x26, 0xca, 0xc0, 0x6b, 0xce, 0x8b, 0x3e, 0xfe, 0xf2, 0x90, 0x3f, 0x1a, 0xed, 0xc8, 0xbf, 0x28, 0x9a, 0x75,
0xbf, 0x08, 0xb6, 0xb8, 0x3f, 0x0b, 0xc6, 0x99, 0xbf, 0x70, 0x21, 0x3b, 0x3e, 0xcf, 0x14, 0xa6, 0xbf, 0x07, 0x2b, 0x87, 0xbe, 0xc4,
0xa7, 0xae, 0x3d, 0x73, 0x17, 0x50, 0x3e, 0x7d, 0x1f, 0x1d, 0x3e, 0x60, 0x84, 0x4c, 0xbe, 0x6f, 0x36, 0x4a, 0x3e, 0xce, 0x48, 0x7e,
0x3e, 0x00, 0x76, 0x94, 0xbe, 0x48, 0x3e, 0x4c, 0xbf, 0x98, 0x77, 0xc2, 0xbf, 0x37, 0x95, 0xf7, 0xbe, 0xc1, 0x18, 0x93, 0xbf, 0xbb,
0xa6, 0x73, 0xbf, 0x55, 0x64, 0x35, 0x3f, 0xf5, 0x2e, 0x78, 0xc0, 0xa9, 0x46, 0x14, 0xbf, 0x78, 0xd8, 0xe1, 0xbe, 0x67, 0xed, 0xd4,
0x3d, 0x2a, 0x65, 0x8d, 0x3d, 0xda, 0x4b, 0x9a, 0xbe, 0x9d, 0x77, 0x4e, 0xbe, 0xec, 0x87, 0x87, 0xbd, 0xf1, 0x8b, 0x3f, 0xbd, 0x82,
0x2d, 0xf2, 0xbe, 0x2b, 0x29, 0xf9, 0xc0, 0xfa, 0x9a, 0x8a, 0x3e, 0x05, 0x8d, 0x14, 0x3d, 0xb1, 0x22, 0x86, 0xbe, 0x54, 0xe7, 0x29,
0xbe, 0x9c, 0xcb, 0xc9, 0xbf, 0x79, 0xa7, 0x40, 0xc0, 0x4e, 0xe2, 0x71, 0xbb, 0xbc, 0xa4, 0x9d, 0xbf, 0x6d, 0x26, 0x62, 0xbe, 0xa9,
0xe7, 0x59, 0xc0, 0x57, 0x7b, 0x38, 0xc0, 0xcc, 0x92, 0xb4, 0x3e, 0x31, 0x14, 0x33, 0x3e, 0xe5, 0x32, 0x94, 0xbf, 0xd2, 0xb5, 0x1b,
0x3f, 0x55, 0xa2, 0x00, 0x3f, 0xe6, 0x5e, 0x43, 0x3e, 0xf9, 0x76, 0xab, 0xbf, 0x90, 0x96, 0x07, 0xbf, 0x06, 0x7a, 0x1f, 0xbd, 0xa1,
0x8e, 0x87, 0xbf, 0xb9, 0x61, 0x50, 0x3f, 0x1f, 0x65, 0x81, 0xbb, 0x2a, 0x19, 0x28, 0xbf, 0xf2, 0xe5, 0x75, 0x3e, 0x8b, 0x33, 0xe6,

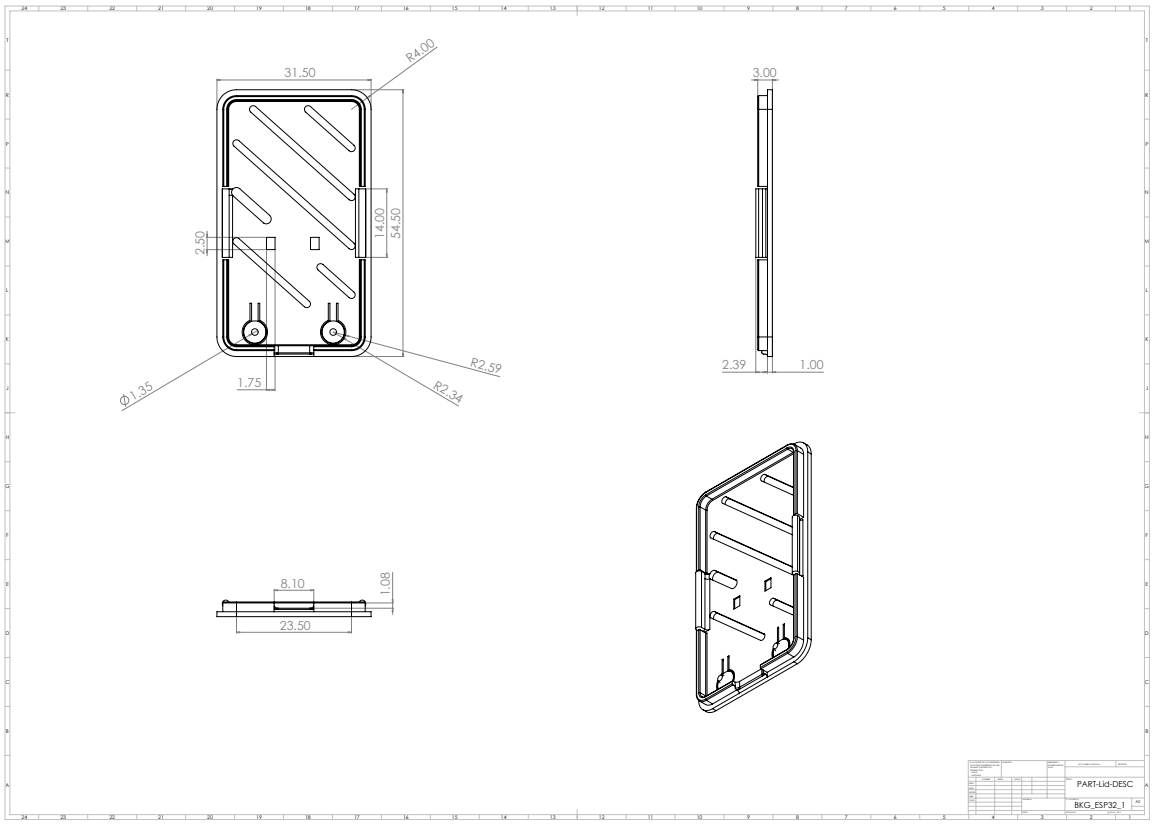
0xbe, 0x27, 0x0d, 0xea, 0xbf, 0xb5, 0x49, 0x0e, 0x3e, 0xd9, 0x7c, 0x63, 0x3e, 0x06, 0x6d, 0x80, 0xbe, 0xfd, 0x4d, 0xa3, 0xbf, 0xfc,
0x9c, 0x57, 0xbd, 0x6a, 0xbe, 0x86, 0xbe, 0x06, 0xb0, 0x08, 0xc1, 0x2b, 0xcc, 0xfa, 0xbe, 0x51, 0xf3, 0xc7, 0x3d, 0x3c, 0xb7, 0xcd,
0x3d, 0xec, 0x83, 0xbc, 0x3e, 0x99, 0x22, 0x29, 0xbf, 0x00, 0xc8, 0x57, 0xc0, 0x5b, 0x6c, 0x86, 0xbf, 0x0a, 0x2a, 0xa5, 0xbf, 0x15,
0x53, 0xdb, 0xbd, 0x95, 0x49, 0x08, 0xbf, 0xf8, 0x73, 0xfb, 0x3e, 0xaa, 0xcf, 0x44, 0xc0, 0x8e, 0x1f, 0x31, 0x3f, 0x2f, 0xfd, 0x62,
0xbe, 0x98, 0xa7, 0xf2, 0xbd, 0x56, 0x3d, 0xa8, 0xbe, 0xbd, 0x1a, 0x39, 0xbf, 0x2b, 0x32, 0x76, 0x3f, 0x95, 0xe1, 0x10, 0xbf, 0x84,
0x4f, 0x67, 0x3e, 0xa3, 0xfc, 0x98, 0xbe, 0x3b, 0x3d, 0x03, 0xbf, 0x4d, 0xb1, 0x8b, 0x3e, 0x34, 0x6f, 0x6b, 0x3d, 0x1e, 0xc1, 0x8e,
0xbf, 0xe5, 0xb9, 0xfd, 0x3e, 0x1a, 0xb4, 0x97, 0xbc, 0x7d, 0x3d, 0x2d, 0xbe, 0xca, 0x60, 0x33, 0xc0, 0x3a, 0x7c, 0x34, 0xbf, 0x23,
0xa9, 0xb4, 0xbf, 0x45, 0x9e, 0xd7, 0xbe, 0x70, 0x40, 0xa7, 0xbf, 0x8e, 0x95, 0xb6, 0xbe, 0x60, 0x9e, 0x20, 0x3e, 0xd0, 0x13, 0xfb,
0x3e, 0x41, 0x34, 0x5a, 0xbf, 0x9e, 0x73, 0x39, 0xc0, 0x09, 0x6f, 0x04, 0xbf, 0x73, 0x41, 0xad, 0xbe, 0x0e, 0x89, 0xd7, 0x3d, 0x94,
0x2c, 0x84, 0xbf, 0xb7, 0xe6, 0x14, 0x3e, 0x65, 0x43, 0xb1, 0xbf, 0xb2, 0xe4, 0x9f, 0x3f, 0x9a, 0xd8, 0x8e, 0xbf, 0x93, 0xff, 0xc4,
0x3e, 0xa6, 0xd4, 0x10, 0x3e, 0x08, 0x3a, 0xf4, 0x3d, 0x33, 0x72, 0x68, 0xbf, 0x9a, 0x9c, 0x1a, 0xc0, 0xee, 0x1a, 0x97, 0xbf, 0x1e,
0x40, 0x49, 0xbf, 0x0d, 0x54, 0x5a, 0xbf, 0x03, 0x91, 0x1a, 0x3e, 0x98, 0x2c, 0x45, 0xbd, 0x10, 0xb0, 0x80, 0xbe, 0x35, 0x9e, 0xc3,
0x3e, 0x39, 0xf7, 0x93, 0x3d, 0x52, 0x9e, 0x9b, 0xbe, 0xe7, 0xb9, 0xb7, 0xbd, 0x99, 0xa5, 0x3f, 0xc0, 0x94, 0x40, 0x8c, 0xbf, 0x45,
0x08, 0xd8, 0xbe, 0x60, 0x23, 0x39, 0xbf, 0x3e, 0x3c, 0x03, 0xbf, 0x1f, 0x9e, 0x65, 0x3e, 0x60, 0x1f, 0x27, 0xbf, 0x64, 0x64, 0x85,
0xbd, 0x88, 0x03, 0x97, 0x3e, 0xc5, 0x0e, 0x3a, 0xbf, 0xfa, 0xe4, 0xcf, 0x3e, 0xea, 0x12, 0xa2, 0xbd, 0xd3, 0x3c, 0xd2, 0xc0, 0x83,
0xca, 0x52, 0xbf, 0xdc, 0x6c, 0x47, 0x3e, 0xf7, 0x80, 0x19, 0xbf, 0x52, 0xbc, 0xae, 0xbf, 0x18, 0x30, 0xb2, 0xc0, 0x7d, 0xe9, 0x41,
0xbd, 0xf6, 0xe8, 0x78, 0xbe, 0x1c, 0x22, 0x27, 0xbf, 0xc7, 0x45, 0x10, 0x3f, 0x79, 0x8b, 0x0f, 0xc0, 0x3b, 0xcd, 0x84, 0x3f, 0x77,
0xe5, 0x2e, 0x3e, 0x26, 0xa0, 0x87, 0xbf, 0x4d, 0x8c, 0xb1, 0xbf, 0x31, 0x12, 0x00, 0xc0, 0x87, 0x85, 0xb7, 0x3c, 0x49, 0xae, 0x96,
0xbf, 0xfd, 0xde, 0xac, 0x3e, 0x89, 0xa3, 0x38, 0xc0, 0x9c, 0xa9, 0x44, 0xbe, 0xe8, 0xb3, 0x9a, 0xc0, 0xf3, 0xe7, 0xbd, 0xbf, 0xe1,
0x8b, 0xa9, 0xbc, 0x8e, 0x1e, 0x96, 0x3f, 0x9d, 0x6d, 0x22, 0xbf, 0x18, 0xd2, 0x31, 0x3f, 0xcc, 0xfe, 0x74, 0x3f, 0xff, 0x8e, 0x59,
0x3f, 0x57, 0xaf, 0x46, 0xbf, 0x4f, 0x55, 0x49, 0xbe, 0x75, 0xd7, 0x04, 0xc0, 0x8d, 0xb7, 0x27, 0xbf, 0x3d, 0x60, 0x00, 0xc0, 0xd4,
0x7d, 0x52, 0x3f, 0xa8, 0xe7, 0x8c, 0xc0, 0xd5, 0x82, 0x6a, 0xbe, 0xce, 0x58, 0xe4, 0xc0, 0x02, 0x18, 0x2a, 0xc0, 0x83, 0x19, 0x89,
0xbe, 0x0a, 0x80, 0x89, 0x3e, 0x52, 0xa4, 0x3e, 0x3f, 0xb8, 0xff, 0xc9, 0x3d, 0xf4, 0x66, 0x6b, 0xbd, 0x76, 0x3d, 0xd4, 0x3d, 0xcf,
0xab, 0xf1, 0xbf, 0x0e, 0xfa, 0xb2, 0xbe, 0x6c, 0x57, 0xdd, 0xbe, 0xbe, 0x3d, 0x6c, 0xbe, 0xc9, 0x1e, 0xa2, 0xbf, 0xc6, 0xc7, 0x8f,
0x3d, 0x1c, 0x87, 0x79, 0xbf, 0x92, 0xac, 0x1a, 0x3f, 0xa0, 0x63, 0xf4, 0xc0, 0x21, 0x24, 0x26, 0xc0, 0xb0, 0x0f, 0x4f, 0xc0, 0x0c,
0xa8, 0xde, 0x3e, 0xc0, 0xf1, 0x08, 0x3f, 0x37, 0x8d, 0x9b, 0x3e, 0x5b, 0x4f, 0xb5, 0xbd, 0x1b, 0xd8, 0x9c, 0x3d, 0xc6, 0xe7, 0xcf,
0xbf, 0xaa, 0x5b, 0xc7, 0x3d, 0x2d, 0x34, 0x89, 0x3e, 0xee, 0x46, 0xb6, 0xbf, 0xd1, 0x76, 0xc4, 0xbe, 0x7a, 0xe8, 0xf1, 0xbe, 0xec,
0xe9, 0x9e, 0xbd, 0xee, 0x1c, 0x69, 0xbe, 0xc2, 0xe7, 0xe5, 0xc0, 0x12, 0xa5, 0xde, 0xbe, 0x7f, 0xc5, 0x01, 0xbf, 0x6a, 0x89, 0x0e,
0xc0, 0x0b, 0x2a, 0xe0, 0xbd, 0x5c, 0x7f, 0xea, 0xbe, 0x90, 0xcd, 0x14, 0x3e, 0xb6, 0xac, 0xba, 0x3e, 0x73, 0xe5, 0xfd, 0xbf, 0x64,
0xda, 0x8e, 0xc0, 0x53, 0x24, 0x85, 0xbf, 0x9d, 0xfe, 0xb6, 0xc0, 0x87, 0x5b, 0xee, 0x3e, 0x55, 0x55, 0x9f, 0xbf, 0x17, 0x7a, 0x8f,
0x3e, 0x9f, 0xac, 0x38, 0xc0, 0xef, 0x4c, 0x94, 0xbf, 0x53, 0x13, 0xad, 0x3d, 0x5e, 0xfb, 0xfb, 0x3e, 0xd2, 0x4c, 0x0d, 0xc0, 0x30,
0x28, 0x92, 0xc0, 0x02, 0x95, 0x37, 0x3e, 0x38, 0x8e, 0xc9, 0xbe, 0x0a, 0xd6, 0x5e, 0x3e, 0x1e, 0xcf, 0x96, 0xbf, 0x37, 0x58, 0xae,
0x3d, 0x34, 0xa6, 0x43, 0x3f, 0x34, 0xd5, 0x79, 0xc1, 0xe3, 0x69, 0x30, 0xbf, 0x04, 0x37, 0x00, 0xbf, 0x4b, 0xf0, 0x18, 0x3f, 0xf1,
0xa9, 0xa3, 0xbf, 0xcb, 0x2b, 0x2e, 0xbf, 0xbc, 0x42, 0xec, 0x3e, 0x70, 0x2c, 0xcb, 0x3c, 0xff, 0xb2, 0x80, 0xc0, 0x34, 0x45, 0x3f,
0xbf, 0xa8, 0x6f, 0x1c, 0xc0, 0x74, 0xf0, 0x9f, 0x3e, 0x8d, 0x72, 0x8e, 0x3e, 0x46, 0xf7, 0xb5, 0xbf, 0x42, 0x22, 0x03, 0xbf, 0x69,
0x87, 0x0d, 0xbf, 0xac, 0x6e, 0xea, 0xbf, 0x30, 0x59, 0x9c, 0xbf, 0x4e, 0x65, 0x98, 0xbe, 0x23, 0xdb, 0xd8, 0xbe, 0xfc, 0x25, 0x34,
0xbf, 0x27, 0xc8, 0x10, 0xc0, 0x94, 0xf4, 0x34, 0xbe, 0x57, 0x72, 0xaf, 0xbe, 0x5c, 0x63, 0x34, 0xbf, 0x17, 0x64, 0x07, 0xbf, 0xc9,
0x2d, 0x11, 0xbf, 0x14, 0x05, 0x05, 0xbf, 0x91, 0xc3, 0x9e, 0xbf, 0xa3, 0xea, 0x4b, 0xbc, 0xe2, 0xee, 0xa5, 0xbe, 0xf0, 0x70, 0xe3,
0xbe, 0x0e, 0x64, 0x06, 0x3b, 0xd0, 0xcd, 0x2f, 0x3f, 0x5d, 0x99, 0x0c, 0x3c, 0xde, 0x7e, 0xdc, 0xbe, 0x66, 0x87, 0x14, 0xbe, 0xcc,
0x8e, 0x5c, 0xbf, 0x62, 0xfe, 0x5c, 0xbf, 0xf5, 0x23, 0x13, 0xc0, 0xbf, 0xd6, 0x49, 0xbd, 0xab, 0x55, 0xfc, 0x3c, 0xcf, 0x28, 0xf3,
0x3d, 0x3a, 0x4c, 0xa1, 0xbf, 0x92, 0x11, 0xa2, 0xbe, 0xac, 0xd8, 0x33, 0x3f, 0x23, 0x28, 0x4f, 0x3f, 0x24, 0x35, 0x62, 0xbf, 0xde,
0xa8, 0x21, 0x3e, 0xf5, 0x3a, 0x23, 0x3f, 0xe8, 0x60, 0x5f, 0x3f, 0x20, 0xed, 0x05, 0x3c, 0xa8, 0x02, 0xac, 0xbf, 0x9b, 0x3b, 0xcc,
0xc0, 0x0c, 0xdc, 0x4f, 0xbf, 0x0f, 0x15, 0x19, 0xc0, 0xac, 0x53, 0x5e, 0xc0, 0x2f, 0xf1, 0x78, 0x3f, 0x11, 0xbb, 0xa1, 0xbe, 0x90,
0xc2, 0xc6, 0x3d, 0x98, 0xc3, 0x12, 0xc0, 0x51, 0xf7, 0x3f, 0x3f, 0x8e, 0x38, 0x8a, 0xc0, 0x93, 0x22, 0xf4, 0xc0, 0x51, 0x4d, 0xa0,
0x3e, 0xab, 0x27, 0x24, 0xbe, 0xc5, 0x17, 0xc8, 0xbe, 0x51, 0x7b, 0x4f, 0xbf, 0x2e, 0xae, 0xd7, 0xbf, 0x85, 0xbd, 0xb8, 0x3d, 0xe2,
0x90, 0x0a, 0xbf, 0x8b, 0x5e, 0xd5, 0xbd, 0x33, 0xf3, 0x8d, 0x3e, 0x68, 0x69, 0x1c, 0xc0, 0xb0, 0x71, 0x9f, 0xbc, 0x1b, 0xc4, 0xa4,
0x3d, 0xfc, 0xf6, 0x52, 0x40, 0x28, 0x6c, 0x13, 0xbe, 0x45, 0x32, 0xa8, 0x3e, 0x6a, 0x6a, 0x5d, 0x3d, 0xaa, 0xdd, 0x6d, 0xbe, 0x45,
0x8e, 0x82, 0xbf, 0xf3, 0xbf, 0x0d, 0xbe, 0xbf, 0xc9, 0x0b, 0xbf, 0x8f, 0x4e, 0x69, 0xbd, 0xb3, 0x20, 0x49, 0xbe, 0xae, 0x76, 0x03,
0xc1, 0xc5, 0x2e, 0x51, 0xc1, 0xbc, 0x77, 0x42, 0x3e, 0x77, 0x39, 0xae, 0x3e, 0xa3, 0x7b, 0xce, 0xbe, 0x8e, 0x7b, 0xa7, 0xbf, 0x7d,
0xdf, 0x69, 0xc0, 0xf6, 0xac, 0x51, 0xbe, 0xc1, 0x1b, 0x1b, 0xbf, 0x77, 0x12, 0x5a, 0xbe, 0xe3, 0x6a, 0xc8, 0x3e, 0xd9, 0xd1, 0x61,
0x3f, 0x5b, 0x00, 0xa0, 0xbf, 0x0e, 0x6b, 0x46, 0x3f, 0x5a, 0x17, 0x06, 0x3e, 0x6b, 0x02, 0xa9, 0xbe, 0x01, 0xa3, 0xf2, 0xc0, 0x04,
0xcc, 0xe5, 0x3f, 0x6a, 0x8b, 0x30, 0xbf, 0xa5, 0x75, 0x37, 0x3e, 0xbe, 0xa3, 0x9e, 0x3e, 0x31, 0xe7, 0x6a, 0xbf, 0xb9, 0x4e, 0xac,
0xbe, 0x64, 0x4f, 0x28, 0xc1, 0xa4, 0x3d, 0xb4, 0x3e, 0x53, 0x2d, 0xf5, 0x3e, 0xdc, 0x90, 0x54, 0xbf, 0xb5, 0x47, 0x2d, 0xbf, 0x86,
0xbf, 0xbe, 0xbe, 0xe7, 0xcf, 0xa7, 0x3d, 0xc7, 0x04, 0xcc, 0x3e, 0xa8, 0x8f, 0xb6, 0xbe, 0x55, 0x58, 0x2f, 0xbf, 0x40, 0x60, 0x5f,
0xc0, 0xe2, 0x7e, 0xdc, 0xbf, 0xed, 0xc7, 0x08, 0x3f, 0x08, 0x16, 0x26, 0xbe, 0xfc, 0xe0, 0x5a, 0x3f, 0xac, 0x72, 0x81, 0xc0, 0x51,
0x95, 0x9a, 0x3e, 0x15, 0x7e, 0xab, 0xbf, 0xe3, 0x42, 0x0a, 0xc0, 0x20, 0xe8, 0x1a, 0xbf, 0x4c, 0x91, 0x07, 0xbf, 0xf1, 0x5a, 0x05,
0x3d, 0xe4, 0x31, 0xf0, 0xbf, 0xca, 0x2d, 0x01, 0xc0, 0x66, 0xab, 0xb4, 0xbd, 0x78, 0x3f, 0xa6, 0xbe, 0x5e, 0xb1, 0x0b, 0x3f, 0x13,
0xf7, 0x3b, 0x3f, 0x4c, 0x2e, 0x82, 0xc0, 0x5f, 0x24, 0x59, 0xbe, 0x62, 0x08, 0x0d, 0xbe, 0xdb, 0x28, 0x27, 0xc1, 0x41, 0xc8, 0x85,
0xbc, 0x30, 0xe3, 0x71, 0xbf, 0x5a, 0x65, 0xd4, 0xbe, 0xc2, 0x70, 0x8a, 0xbe, 0x5c, 0x04, 0x1f, 0x3f, 0xe7, 0xb5, 0xda, 0x3d, 0x75,
0x98, 0x89, 0x3e, 0x21, 0xee, 0x48, 0xbf, 0x06, 0xf3, 0x43, 0xbf, 0xf7, 0xb7, 0x36, 0xbe, 0xa6, 0x61, 0x03, 0x3d, 0xb2, 0x62, 0x5e,
0xc0, 0x2d, 0x94, 0x40, 0xbe, 0x12, 0x30, 0xf5, 0x3d, 0x56, 0xf6, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0xb0,
0xaf, 0x46, 0xbf, 0x86, 0xe4, 0x2a, 0x3e, 0x12, 0xc4, 0x3d, 0x3e, 0x17, 0xe9, 0x92, 0x3e, 0x22, 0xd7, 0x07, 0x40, 0x70, 0xef, 0xb0,
0xbe, 0xfa, 0x44, 0x3c, 0xc0, 0xf0, 0x89, 0x0d, 0xbf, 0xb1, 0x8a, 0xbd, 0x3f, 0x0e, 0x6f, 0x89, 0xbe, 0x9f, 0x4a, 0x06, 0xbf, 0x89,
0xdb, 0x41, 0x3d, 0xd2, 0x31, 0xa4, 0x3f, 0x35, 0xfa, 0xbf, 0x3e, 0x1d, 0xb8, 0xac, 0x3e, 0x5a, 0x86, 0xdc, 0x3f, 0x5e, 0x02, 0xf2,
0xbf, 0xf0, 0x69, 0x5f, 0x3d, 0x03, 0xe9, 0xa3, 0x3f, 0x10, 0xdf, 0xc4, 0xbe, 0x21, 0x92, 0x0c, 0x40, 0x2e, 0x63, 0xb2, 0xbf, 0x9e,
0x6e, 0xd4, 0xbf, 0x4f, 0x5d, 0x93, 0x3e, 0x01, 0x1f, 0x82, 0x3f, 0x1a, 0x49, 0x23, 0x3f, 0x85, 0xf2, 0xc7, 0x3e, 0x36, 0x7d, 0xa9,

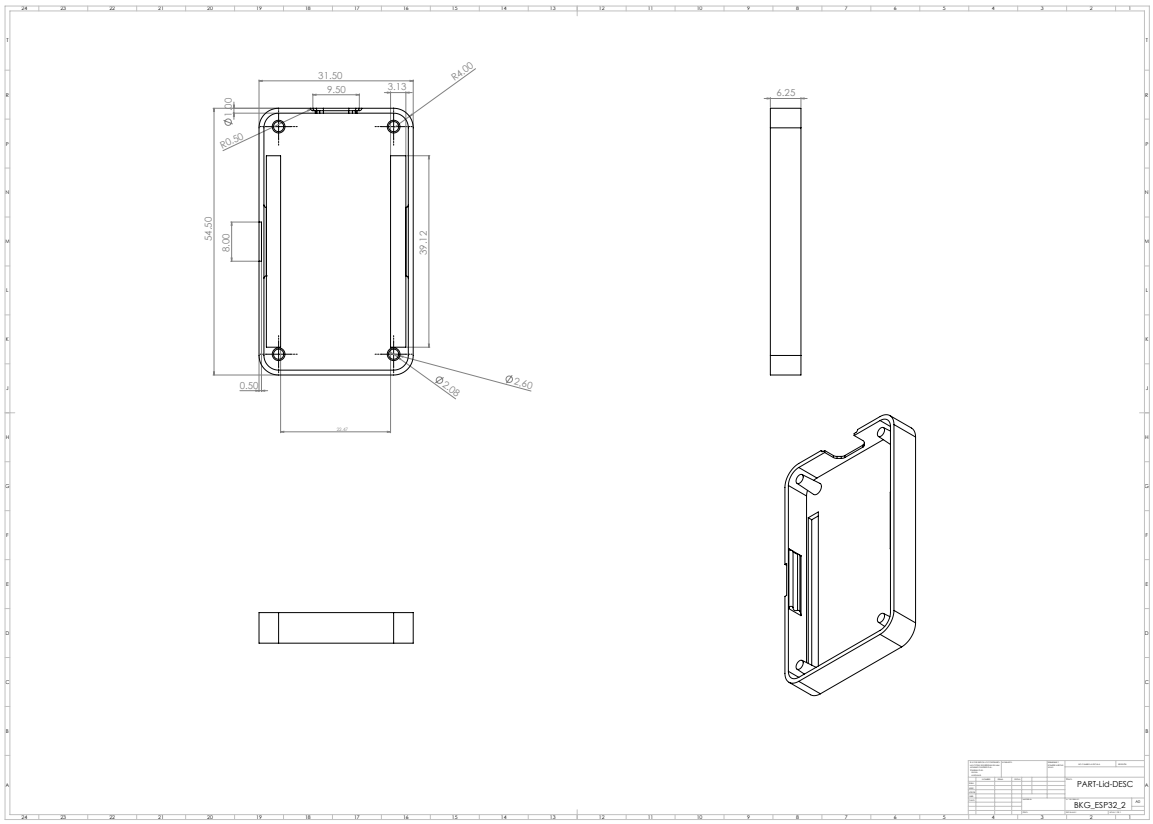
0x3e, 0x3c, 0x6b, 0x8f, 0x3e, 0x0a, 0x85, 0x7b, 0x3e, 0x51, 0xf8, 0xb2, 0xbe, 0x40, 0x57, 0x5e, 0x3e, 0x76, 0xbe, 0x04, 0x40, 0x46,
0x27, 0x53, 0xbe, 0x27, 0x20, 0x42, 0xc0, 0x6e, 0x28, 0xb5, 0xbe, 0xd6, 0x43, 0x8b, 0xbd, 0x47, 0xfd, 0x2b, 0x3f, 0xe8, 0x0a, 0x13,
0xbf, 0x15, 0xf0, 0x2f, 0x40, 0xc4, 0x77, 0x08, 0x3d, 0x2f, 0x43, 0x8c, 0xbd, 0x85, 0x15, 0x43, 0x3e, 0x04, 0x4c, 0xcc, 0x3e, 0xa4,
0x17, 0xd7, 0x3f, 0xfb, 0x70, 0x62, 0xbe, 0xfe, 0x93, 0x83, 0xbe, 0x13, 0x1a, 0xb5, 0x3f, 0x6d, 0x71, 0xba, 0xbe, 0x77, 0x7d, 0xda,
0x3e, 0x89, 0x2f, 0xc2, 0x3e, 0x2a, 0xc5, 0x7d, 0xbe, 0xc5, 0x8d, 0x83, 0xbe, 0x31, 0x4d, 0x70, 0xbe, 0xc2, 0x75, 0x00, 0x40, 0xb2,
0xa6, 0xa7, 0x3f, 0xe2, 0x6c, 0xc2, 0xbf, 0xaf, 0xab, 0xad, 0x3f, 0x91, 0xaa, 0xaa, 0xbf, 0xaf, 0x80, 0xb8, 0x3e, 0xae, 0xd0, 0xf5,
0x3f, 0xd0, 0xac, 0x41, 0x3d, 0x96, 0x0a, 0x74, 0xbe, 0xed, 0x7d, 0xd4, 0x3c, 0xac, 0xce, 0x41, 0xbe, 0x51, 0x64, 0xc1, 0x3e, 0x36,
0xb2, 0x72, 0x40, 0x98, 0xdb, 0x8c, 0xbe, 0x96, 0xb6, 0xb3, 0x3e, 0x7b, 0xea, 0xab, 0x3e, 0x7e, 0x1c, 0x62, 0x40, 0x98, 0xd6, 0x41,
0xbe, 0x04, 0xa6, 0xf9, 0x3e, 0x0c, 0x43, 0xd8, 0x3f, 0xc2, 0xf0, 0xaa, 0x3f, 0x09, 0x2b, 0xc0, 0xbd, 0x5a, 0x97, 0xe8, 0xbd, 0xa9,
0xdb, 0x9e, 0x3e, 0x7b, 0xa3, 0xc0, 0x3e, 0x6e, 0xa3, 0x94, 0xbf, 0xd7, 0x1f, 0x12, 0x40, 0x89, 0xa0, 0xd6, 0x3e, 0x70, 0xd4, 0x4c,
0xbf, 0xc9, 0xb7, 0xc4, 0x3e, 0x37, 0xad, 0x5b, 0x3f, 0x6c, 0xbf, 0x21, 0x40, 0x84, 0xbf, 0x8d, 0xbf, 0xe1, 0x4f, 0x7e, 0xbf, 0xf2,
0xb8, 0xb9, 0xbf, 0xcc, 0x24, 0x19, 0x3f, 0x5f, 0x57, 0xb7, 0xbf, 0xa4, 0xf6, 0xc4, 0x3e, 0x82, 0xc4, 0xca, 0x3e, 0xbc, 0x47, 0x83,
0xbd, 0x70, 0xa3, 0x86, 0x3d, 0x27, 0x8f, 0x85, 0x3f, 0xf4, 0x0b, 0x9d, 0x3f, 0xfe, 0x7b, 0x5e, 0x3e, 0x92, 0xe3, 0xab, 0xbe, 0xea,
0x4a, 0xd0, 0xbe, 0xcc, 0x5e, 0xe8, 0x3f, 0x79, 0x79, 0x5b, 0x3f, 0x95, 0x5b, 0x20, 0xbf, 0xa7, 0xbf, 0x2f, 0x3f, 0x24, 0x82, 0x24,
0x3f, 0x1f, 0xf9, 0x2b, 0xbf, 0xed, 0x26, 0x4c, 0x40, 0x96, 0x91, 0x93, 0xbe, 0x33, 0xb5, 0xe0, 0xbf, 0xa2, 0x47, 0xad, 0xbe, 0x2c,
0xc2, 0xa8, 0x3d, 0x98, 0x94, 0xa5, 0xbe, 0x44, 0xea, 0x01, 0x3e, 0x10, 0xa4, 0x92, 0x3c, 0xed, 0x3d, 0x9e, 0xbf, 0x4c, 0x34, 0x96,
0x3d, 0x2d, 0x80, 0xf5, 0x3f, 0x02, 0x5d, 0x5c, 0xbf, 0x5e, 0x3d, 0x8b, 0xbf, 0x1d, 0x7f, 0xe8, 0x3f, 0x06, 0x20, 0x93, 0xbf, 0xc2,
0xec, 0xa9, 0x3e, 0x63, 0x60, 0xa9, 0xbf, 0xeb, 0x6b, 0xca, 0xbe, 0x01, 0x02, 0x40, 0x40, 0xc8, 0xd6, 0xd9, 0xbd, 0xca, 0x0d, 0x5c,
0xbe, 0x9d, 0x8b, 0xa5, 0x3e, 0x0b, 0x3f, 0x0e, 0x3f, 0x65, 0x79, 0x87, 0xbe, 0xde, 0xa4, 0x0f, 0x40, 0x00, 0x84, 0x75, 0xba, 0x40,
0x4c, 0x89, 0x3f, 0xca, 0xb1, 0x69, 0x3f, 0x43, 0x08, 0x6d, 0x3f, 0x48, 0x46, 0x3f, 0xbf, 0x32, 0xba, 0x65, 0x3f, 0xa7, 0xfe, 0xf8,
0x3d, 0x70, 0x5f, 0x76, 0x40, 0x5f, 0x9f, 0x0a, 0xbd, 0x25, 0xc1, 0x03, 0xc0, 0x5a, 0x33, 0x1e, 0x3e, 0x71, 0xac, 0xaf, 0x3e, 0x85,
0xed, 0x7b, 0xbf, 0x67, 0xcc, 0xa2, 0x3e, 0x54, 0x12, 0xaf, 0xbe, 0x28, 0xa6, 0x54, 0x40, 0xc7, 0xec, 0x8c, 0xbe, 0x56, 0xae, 0x93,
0x3f, 0xad, 0xd6, 0x01, 0x40, 0x79, 0x3d, 0x9f, 0x3f, 0x83, 0xf3, 0xf2, 0xbf, 0xd3, 0xe2, 0xba, 0xbf, 0xe6, 0xb9, 0xde, 0x3e, 0xe6,
0xc3, 0x5e, 0xc0, 0x84, 0x2a, 0xdd, 0xbd, 0x28, 0x1f, 0x2c, 0x3f, 0xd7, 0x14, 0x3d, 0xbe, 0xf7, 0xde, 0xbe, 0xbe, 0x30, 0xe3, 0xf1,
0x3e, 0x0f, 0x8e, 0x10, 0xbc, 0x73, 0xce, 0xc1, 0x3e, 0x55, 0x36, 0xfd, 0x3f, 0xa0, 0xa3, 0xcc, 0xbc, 0x3c, 0x8e, 0x1c, 0xc0, 0x42,
0x93, 0x43, 0x3f, 0x80, 0x55, 0xf7, 0x3f, 0x81, 0xd9, 0x74, 0xbe, 0xb6, 0x2c, 0x06, 0x40, 0x77, 0xd1, 0xa9, 0x3d, 0x61, 0xf1, 0x6f,
0xba, 0xcd, 0xad, 0xda, 0xbd, 0xf0, 0xb2, 0x16, 0xbe, 0x5d, 0xf2, 0xc0, 0x3e, 0x80, 0x87, 0xd5, 0x3b, 0x13, 0x2c, 0xce, 0x3f, 0x9e,
0x71, 0x37, 0xbf, 0x97, 0x03, 0xa2, 0x3e, 0x5e, 0x3d, 0x47, 0x3f, 0xb3, 0xa0, 0x9b, 0xbe, 0xe8, 0x37, 0x5e, 0xbf, 0x18, 0x06, 0x4e,
0xbf, 0x99, 0x20, 0x49, 0x3f, 0x99, 0x9b, 0x53, 0x40, 0x7e, 0x4f, 0x74, 0x40, 0x6f, 0x78, 0xe2, 0x3f, 0x4f, 0x47, 0x90, 0x40, 0x6a,
0xe2, 0x3f, 0x3e, 0x47, 0x0a, 0x7e, 0xbf, 0x71, 0x16, 0xcb, 0xbe, 0xc3, 0x4e, 0xc9, 0xbe, 0xeb, 0x8c, 0xd8, 0x3e, 0x42, 0x3b, 0x4e,
0xbf, 0xd7, 0xf7, 0xa5, 0x3e, 0xc4, 0x86, 0x48, 0xbf, 0xf3, 0xdd, 0xd5, 0xbe, 0x3f, 0x13, 0x04, 0x3f, 0x3e, 0x3d, 0x38, 0xbf, 0x7c,
0x11, 0xb8, 0xbf, 0x27, 0xd0, 0x97, 0x3f, 0x68, 0x2e, 0x6f, 0x40, 0xa0, 0x66, 0x91, 0xbf, 0x06, 0xf3, 0x16, 0x40, 0x5c, 0x72, 0x58,
0xbe, 0x1d, 0x93, 0x56, 0xbe, 0xb3, 0xcc, 0x65, 0xbe, 0x2d, 0x7d, 0xb1, 0x3e, 0x56, 0xca, 0xec, 0x3f, 0xd8, 0xb6, 0xbd, 0x3d, 0x0b,
0xd7, 0xba, 0xbe, 0x63, 0xbe, 0x02, 0x3f, 0x94, 0x13, 0xc7, 0xbe, 0xd6, 0x9e, 0xae, 0xbf, 0x67, 0x49, 0xb1, 0xbe, 0x4d, 0x6d, 0x17,
0x3f, 0xd8, 0x1d, 0x98, 0x3e, 0xe9, 0xb1, 0xae, 0x3e, 0xcb, 0xf7, 0x2b, 0x3f, 0xa2, 0x4f, 0x4b, 0xc0, 0x90, 0xeb, 0x5a, 0x3e, 0x40,
0xd8, 0xb4, 0x3f, 0xf1, 0xda, 0xc7, 0x3e, 0x94, 0x62, 0xcf, 0xbe, 0x05, 0x57, 0xfc, 0x3f, 0x32, 0x38, 0x01, 0x3e, 0xaf, 0xb6, 0xc9,
0x3e, 0xca, 0xb2, 0x16, 0x3f, 0x34, 0xdc, 0xc0, 0x3d, 0xd3, 0x5f, 0x0c, 0x40, 0xf4, 0x59, 0xbc, 0x3f, 0xe7, 0x70, 0xc9, 0x3f, 0xf4,
0x4c, 0xeb, 0x3d, 0x0f, 0xf6, 0x1f, 0xc0, 0x42, 0x0d, 0x36, 0x3f, 0x7c, 0x59, 0x03, 0xc0, 0xc8, 0xc3, 0xe1, 0x3e, 0x47, 0x20, 0xaf,
0x3f, 0x3d, 0xd1, 0xb5, 0xbe, 0x80, 0xdf, 0xb2, 0xbe, 0x19, 0x68, 0x1e, 0xbf, 0x28, 0x2b, 0x89, 0x3f, 0xb9, 0x40, 0xa6, 0x3e, 0xc4,
0xe4, 0xae, 0x3f, 0x5e, 0x6c, 0x46, 0xbe, 0x0f, 0x68, 0x0b, 0x40, 0x6d, 0x27, 0x15, 0x40, 0xb5, 0x4b, 0x13, 0xc0, 0x14, 0x3c, 0x1d,
0xc0, 0x04, 0x23, 0x91, 0xbd, 0x04, 0x7d, 0xbe, 0xbe, 0xe9, 0x48, 0xbe, 0xbf, 0xbc, 0x3c, 0xb0, 0x3d, 0x48, 0xe1, 0x88, 0xbf, 0xdc,
0xf5, 0xa0, 0xbd, 0x3c, 0x29, 0xda, 0xbe, 0x70, 0x1d, 0xb6, 0x3f, 0x62, 0xfa, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x40, 0x04, 0x00,
0x00, 0x0f, 0x64, 0x83, 0xbe, 0x4e, 0xd9, 0x0a, 0xbf, 0xb6, 0x8f, 0xd2, 0x3f, 0x33, 0xff, 0xbc, 0x3e, 0xb2, 0x21, 0xaa, 0x3f, 0xcf,
0x63, 0x26, 0xbf, 0x79, 0x06, 0xc9, 0x3f, 0x01, 0x6f, 0x95, 0xbe, 0x75, 0x8d, 0x4d, 0x3f, 0x47, 0x64, 0x28, 0x3f, 0xa0, 0xb5, 0xe4,
0x3e, 0xf0, 0x9e, 0xb2, 0xbe, 0x3e, 0x45, 0xe5, 0x3f, 0x97, 0x0f, 0x25, 0x3e, 0x89, 0x53, 0xc0, 0x3d, 0xd5, 0x37, 0x40, 0xc0, 0x06,
0x4f, 0xb9, 0x3e, 0x8b, 0xee, 0xcd, 0xbe, 0x67, 0xcd, 0x3a, 0xbe, 0xda, 0xa0, 0x93, 0xbe, 0x14, 0xd0, 0x83, 0xbe, 0xe0, 0x15, 0xa2,
0x3d, 0x1b, 0x1b, 0xd6, 0xbe, 0xf8, 0x58, 0xd9, 0xbd, 0x04, 0xc7, 0x96, 0xbe, 0xe0, 0xaa, 0x55, 0x3d, 0x48, 0x91, 0x7d, 0x3e, 0x8c,
0x80, 0xd0, 0xbe, 0x90, 0xba, 0xa7, 0x3d, 0xd0, 0xcb, 0x41, 0xbd, 0x7c, 0xcb, 0x1c, 0x3e, 0x88, 0x56, 0xa2, 0x3d, 0x94, 0xec, 0x74,
0x3e, 0xca, 0x82, 0x81, 0xbe, 0xee, 0xef, 0x4a, 0x40, 0x10, 0xba, 0xcf, 0xbd, 0x81, 0xec, 0xb5, 0x3d, 0x56, 0x06, 0x34, 0x3f, 0x6f,
0x7c, 0xb8, 0xbf, 0x30, 0x01, 0x83, 0x3f, 0xa1, 0xa0, 0xf7, 0x3f, 0x14, 0x0f, 0x9a, 0x40, 0x7d, 0x80, 0xc3, 0xbf, 0xc7, 0x6e, 0x03,
0x3f, 0x72, 0x26, 0xa2, 0xbf, 0xb8, 0x44, 0x18, 0xbd, 0x3e, 0x7a, 0x9d, 0xbf, 0x00, 0x2f, 0x97, 0x3e, 0xd2, 0xd8, 0xca, 0x3f, 0x58,
0x0a, 0x83, 0x3f, 0x60, 0xbe, 0x15, 0xc0, 0x40, 0xb9, 0xec, 0xbc, 0x40, 0xd3, 0x15, 0xbc, 0x18, 0xfe, 0xaa, 0x3e, 0xc6, 0x09, 0xd8,
0xbe, 0x4a, 0x80, 0x71, 0xbe, 0x80, 0xc2, 0x9b, 0xbd, 0x10, 0x0d, 0x9d, 0x3d, 0xd0, 0x6e, 0x03, 0x3d, 0xf0, 0x7b, 0x8f, 0xbe, 0xc0,
0x8a, 0x09, 0x3d, 0x62, 0xe1, 0xc4, 0xbe, 0x66, 0xba, 0xa5, 0xbe, 0x60, 0x32, 0xaf, 0xbc, 0x24, 0xb5, 0x0e, 0xbe, 0x18, 0x63, 0xa8,
0x3d, 0xa0, 0x8b, 0x11, 0x3e, 0x6b, 0x56, 0xc9, 0xbe, 0x0c, 0x16, 0x8c, 0xbf, 0xaa, 0xc2, 0x26, 0xbf, 0xd4, 0xfa, 0xb3, 0x3f, 0x3e,
0x93, 0x28, 0x3f, 0xd6, 0xec, 0x91, 0x40, 0x6f, 0x51, 0x0a, 0x3f, 0x8e, 0xa2, 0x16, 0x40, 0x24, 0xa3, 0x1b, 0x40, 0x91, 0xc8, 0x04,
0x40, 0x3e, 0xb3, 0xd6, 0x3d, 0xc6, 0x39, 0x8a, 0xbe, 0x7b, 0xb4, 0x82, 0x3e, 0xf4, 0xc1, 0xb7, 0xbf, 0x60, 0xc5, 0xaa, 0xbf, 0x00,
0x2d, 0x81, 0x3f, 0xc8, 0x26, 0x0b, 0x40, 0x06, 0x55, 0xb7, 0x40, 0x8b, 0x8b, 0x7d, 0x3f, 0x71, 0x93, 0x82, 0xbf, 0x07, 0x9d, 0x0f,
0x40, 0xbb, 0x32, 0xae, 0x3f, 0xc9, 0xe8, 0xea, 0x3f, 0x7a, 0xe9, 0x05, 0x3f, 0xc8, 0xef, 0x06, 0x40, 0x9f, 0x57, 0x1e, 0x40, 0x51,
0xa7, 0xc1, 0x3f, 0x65, 0xde, 0xd4, 0x3f, 0x5a, 0x82, 0x84, 0xbe, 0x43, 0xeb, 0x2b, 0xbf, 0x47, 0xfa, 0x6c, 0x3f, 0xea, 0xe5, 0x79,
0x3e, 0x4c, 0xfd, 0xcf, 0xbe, 0x30, 0xff, 0x20, 0xc0, 0x4b, 0x14, 0x62, 0x3e, 0xdd, 0x0f, 0xea, 0x3f, 0x0a, 0x5f, 0x61, 0x3f, 0x8c,
0x8b, 0xf7, 0xbf, 0xf1, 0xaa, 0x19, 0x3f, 0x86, 0xea, 0x17, 0x3f, 0x33, 0x19, 0x58, 0x3f, 0x0a, 0x06, 0xb9, 0x3f, 0x87, 0x20, 0x2b,
0x40, 0x31, 0x83, 0xc8, 0xbf, 0xc9, 0x2e, 0xd3, 0xbe, 0x10, 0xd6, 0xcf, 0x3e, 0xd2, 0xe0, 0x7f, 0x3e, 0x72, 0xa4, 0x52, 0x3f, 0x34,
0xfc, 0xc0, 0x3f, 0x46, 0x84, 0xff, 0xbe, 0x65, 0xcd, 0x4e, 0xbf, 0xf1, 0xce, 0x7b, 0xc0, 0xb1, 0xec, 0xde, 0xbe, 0xb0, 0x14, 0x4e,
0x40, 0xe5, 0x00, 0x33, 0xc0, 0x62, 0xc4, 0x5f, 0xbf, 0x3a, 0x7f, 0x2b, 0xbe, 0x70, 0x7c, 0x6a, 0xbf, 0x45, 0xc3, 0x2f, 0xbf, 0x51,
0xc4, 0x42, 0xc0, 0x6a, 0x43, 0x0b, 0xbe, 0x7e, 0x50, 0x0f, 0xc0, 0xc0, 0xb6, 0x07, 0x40, 0x7b, 0xef, 0x59, 0x3f, 0x72, 0xd1, 0xb3,

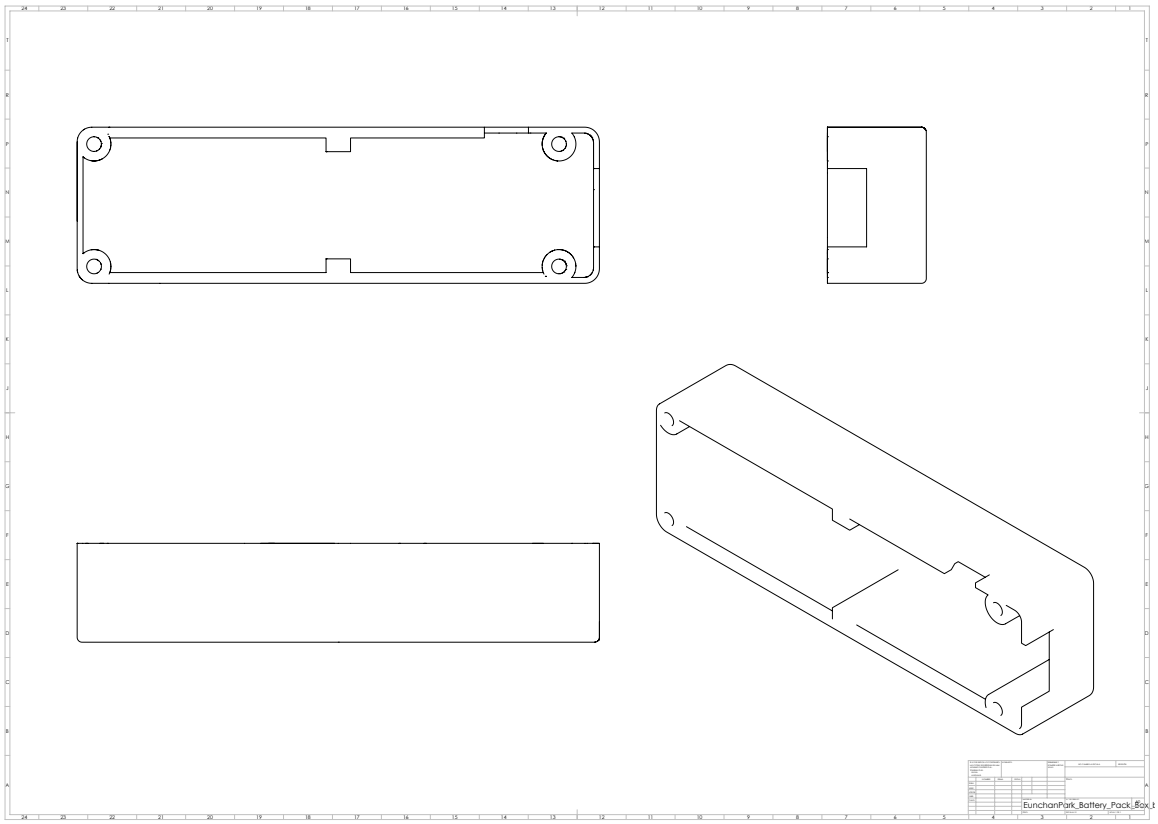
0x3e, 0xca, 0x85, 0x4c, 0x3f, 0x43, 0xa0, 0xba, 0xbe, 0xeb, 0x56, 0x11, 0x3f, 0x13, 0xa3, 0x43, 0x3f, 0xc1, 0x53, 0xe4, 0x3f, 0xe0,
0x62, 0x2d, 0x40, 0x38, 0x6a, 0x8c, 0xbf, 0xca, 0xa7, 0x30, 0xbe, 0xb2, 0xa0, 0xaf, 0xc0, 0x23, 0x88, 0xb5, 0x3e, 0x18, 0x42, 0xd0,
0xbf, 0x6f, 0x9b, 0x65, 0xbf, 0x4b, 0x25, 0x72, 0xbe, 0x3d, 0x5b, 0xfc, 0x3f, 0x94, 0x74, 0x08, 0x40, 0x02, 0x3b, 0xc6, 0xbd, 0x85,
0x4d, 0x5c, 0xbf, 0xda, 0x21, 0x57, 0x3e, 0x71, 0x36, 0x5d, 0x3f, 0x82, 0xe9, 0x58, 0x3f, 0x2f, 0xa8, 0xac, 0x3f, 0xa7, 0x3e, 0xda,
0x3f, 0x23, 0x42, 0xfe, 0x3e, 0x14, 0xe0, 0x7b, 0xbe, 0x24, 0x7b, 0x22, 0x3e, 0x7e, 0xaf, 0x73, 0x3f, 0x1d, 0x53, 0xaa, 0xbe, 0xf0,
0xfd, 0xd5, 0x3f, 0xf3, 0xc0, 0x92, 0x3f, 0xfe, 0x38, 0x15, 0xbf, 0x1e, 0xc7, 0xa7, 0xbf, 0x3b, 0x7f, 0x3e, 0x3e, 0xae, 0x93, 0x05,
0x40, 0x01, 0x02, 0xf1, 0xbe, 0xc8, 0x34, 0x8b, 0x3f, 0x6a, 0x7e, 0xae, 0x3e, 0x0d, 0x06, 0x54, 0x3f, 0xc6, 0x9c, 0x58, 0xc0, 0xd4,
0x02, 0xb7, 0x3f, 0xe5, 0x89, 0x79, 0x40, 0xa1, 0x12, 0x62, 0xc0, 0xe4, 0x25, 0xcb, 0xbe, 0xa4, 0x22, 0x3d, 0xc0, 0xbd, 0xd0, 0xc2,
0xbf, 0x65, 0xae, 0x81, 0x3f, 0xbe, 0x32, 0x6b, 0xbe, 0xa2, 0x0b, 0x89, 0xbf, 0x5a, 0xe5, 0x45, 0x3f, 0x08, 0x37, 0x39, 0x40, 0x18,
0x44, 0xf3, 0x3e, 0xda, 0x8c, 0x7d, 0x3e, 0x7b, 0x41, 0x91, 0xbf, 0xe7, 0xe3, 0x26, 0x3e, 0x36, 0xf3, 0xde, 0xbd, 0x95, 0xe2, 0x22,
0x40, 0x53, 0x56, 0xbe, 0xbe, 0xb6, 0x15, 0xf9, 0xbc, 0x34, 0x97, 0x17, 0x3e, 0x58, 0xfa, 0xa2, 0x3e, 0x33, 0xc3, 0xd4, 0xbd, 0x67,
0x8b, 0xb3, 0xbc, 0x8a, 0x26, 0x83, 0x3d, 0xba, 0x66, 0x0c, 0xbe, 0x76, 0xc2, 0xd7, 0xbe, 0xbf, 0xc6, 0xb3, 0x3e, 0xf7, 0x03, 0xe1,
0xbe, 0xcb, 0xce, 0xb7, 0x3e, 0xec, 0x69, 0xe7, 0x3d, 0xaa, 0x91, 0xa7, 0xbe, 0x74, 0x5c, 0xa4, 0xbe, 0x08, 0x90, 0x4b, 0xbe, 0xe0,
0xbe, 0x21, 0x3e, 0x66, 0x96, 0x14, 0xc0, 0x6e, 0xc5, 0xac, 0xbf, 0x02, 0x5b, 0xa5, 0x3f, 0x26, 0x0a, 0x24, 0x3f, 0xf0, 0xd7, 0x6b,
0x40, 0x16, 0x83, 0x41, 0x3f, 0xf6, 0xc4, 0xc2, 0xbf, 0xeb, 0x20, 0x05, 0xc0, 0xe5, 0x26, 0x5e, 0x3f, 0xe3, 0x40, 0xe9, 0xbf, 0xfc,
0xd3, 0x0d, 0xbf, 0xb7, 0x84, 0x9b, 0x3f, 0xe9, 0xa0, 0x58, 0x3f, 0xba, 0xef, 0x80, 0x3f, 0x71, 0x08, 0xb7, 0x3d, 0xec, 0x61, 0xfc,
0x3f, 0x35, 0x42, 0x05, 0xc0, 0x58, 0x13, 0x71, 0xbe, 0xea, 0xa0, 0x2f, 0xbe, 0xd0, 0x7e, 0xf4, 0x3d, 0xa8, 0xeb, 0xa1, 0x3d, 0x68,
0xf6, 0x0c, 0x3e, 0x68, 0x59, 0x29, 0x3e, 0x10, 0x9c, 0x88, 0x3e, 0xc0, 0x4f, 0x1a, 0x3d, 0x2c, 0xd2, 0xa1, 0xbd, 0x76, 0x53, 0xb7,
0x3e, 0x78, 0x43, 0x82, 0x3d, 0x00, 0xa8, 0x7c, 0x3e, 0x1a, 0x4d, 0xb1, 0xbe, 0x6e, 0x52, 0xb3, 0xbe, 0xbe, 0xef, 0x89, 0xbe, 0x0a,
0xa7, 0x85, 0xbe, 0x40, 0x76, 0xfe, 0x3b, 0x5b, 0x1b, 0xc1, 0xbe, 0xe6, 0x4e, 0x90, 0xbe, 0xd0, 0x27, 0x05, 0x3e, 0x90, 0x53, 0xc3,
0x3d, 0x40, 0xa9, 0x21, 0x3d, 0xd2, 0xfa, 0xa1, 0xbe, 0x25, 0x7c, 0xa8, 0xbe, 0xe8, 0x27, 0xab, 0xbe, 0x2c, 0x9f, 0xc4, 0x3e, 0x13,
0xd0, 0x44, 0xbe, 0xf0, 0xcf, 0xc5, 0x3e, 0xcb, 0x89, 0x60, 0xbe, 0x91, 0x01, 0x49, 0xbe, 0x40, 0x66, 0xf7, 0xbb, 0x28, 0x9b, 0x4c,
0xbd, 0x4c, 0xb7, 0xad, 0xbe, 0x68, 0x15, 0xc8, 0x3d, 0x67, 0x5c, 0x06, 0x3f, 0xfa, 0xc9, 0x83, 0xbe, 0xb4, 0xd9, 0x80, 0x40, 0x81,
0x6c, 0xac, 0x3f, 0x2d, 0xb8, 0x15, 0xc0, 0x13, 0x45, 0x13, 0x40, 0xe9, 0x32, 0x82, 0xc0, 0xef, 0x8c, 0x81, 0xc0, 0xdc, 0xd6, 0x2c,
0x40, 0x6f, 0x2e, 0x35, 0x3f, 0xac, 0x7c, 0x7e, 0xbd, 0xf9, 0x5d, 0x53, 0xbf, 0x0c, 0xfc, 0x74, 0x3e, 0x75, 0x2e, 0x8d, 0x3f, 0xb0,
0xa4, 0xbf, 0x3f, 0x55, 0xb1, 0xf0, 0x3e, 0x06, 0x32, 0xaa, 0x3e, 0xae, 0xfe, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00,
0x00, 0x1a, 0x54, 0x28, 0xbd, 0x0d, 0x3e, 0xb3, 0xbe, 0xb8, 0x9c, 0x14, 0xbe, 0x7c, 0x44, 0xa0, 0xbe, 0xf1, 0x3d, 0x28, 0xbf, 0xe7,
0x6e, 0xc5, 0xbe, 0xdb, 0xaa, 0x72, 0xbf, 0x56, 0xd4, 0x69, 0xbf, 0xc1, 0xe7, 0x5d, 0x3e, 0xf2, 0x99, 0x2d, 0xbd, 0x29, 0x1b, 0x35,
0x3e, 0xb9, 0xda, 0x88, 0xbe, 0x25, 0xe6, 0x60, 0xbd, 0x65, 0xc9, 0x9b, 0xbe, 0x20, 0x9e, 0xd1, 0x3e, 0xdf, 0x42, 0x97, 0xbf, 0xb2,
0x79, 0x3e, 0x3e, 0x22, 0x84, 0x7b, 0x3d, 0xd1, 0x66, 0x5f, 0xbe, 0x4b, 0x18, 0x3e, 0x3f, 0x85, 0x02, 0x13, 0x3f, 0x93, 0x11, 0xfa,
0xbe, 0xf3, 0x17, 0x12, 0xbf, 0xbb, 0x40, 0x37, 0xbb, 0x5a, 0x37, 0x23, 0x3e, 0xeb, 0xc4, 0x10, 0xbf, 0x3a, 0x69, 0x01, 0xbf, 0x66,
0x03, 0xa2, 0xbe, 0x0d, 0x44, 0xc9, 0xbe, 0xf5, 0x15, 0x39, 0x3e, 0x59, 0x92, 0x23, 0x3f, 0x4f, 0xfb, 0x13, 0xbd, 0x4b, 0x76, 0xa9,
0xbe, 0xa2, 0xa7, 0xd2, 0xbe, 0x78, 0xff, 0x39, 0xbe, 0x9a, 0x9c, 0x09, 0x3f, 0x86, 0x84, 0x24, 0xbf, 0x91, 0x57, 0x35, 0x3c, 0x15,
0x38, 0xc9, 0xbe, 0x42, 0xee, 0x8d, 0xbe, 0x87, 0xd8, 0x55, 0xbf, 0x52, 0xb2, 0xa0, 0xbe, 0xad, 0xda, 0x63, 0xbf, 0x23, 0xe4, 0x17,
0xbf, 0x63, 0x75, 0xf9, 0xbd, 0xe1, 0xab, 0x41, 0xbe, 0xff, 0xde, 0x44, 0x3e, 0x7f, 0xa8, 0x97, 0xbe, 0xdc, 0x0a, 0xaa, 0xbe, 0x25,
0x30, 0xd1, 0xbe, 0x55, 0xfb, 0x3e, 0xbf, 0x50, 0xc0, 0x70, 0x3e, 0x3b, 0x1c, 0xb1, 0xbd, 0x9f, 0x3f, 0x77, 0xbf, 0xbc, 0x49, 0x39,
0x3f, 0xe8, 0xac, 0x6a, 0xbf, 0x30, 0xec, 0xeb, 0x3d, 0xb2, 0x24, 0xcf, 0xbe, 0x41, 0xf5, 0xd1, 0xbe, 0x9d, 0xf0, 0x12, 0xbf, 0x06,
0x2c, 0x7a, 0xbe, 0x46, 0x18, 0xca, 0x3d, 0xb2, 0xff, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0xf7, 0x86, 0xaf,
0x3e, 0x48, 0xcb, 0xa7, 0x3d, 0xdd, 0x80, 0xa2, 0x3f, 0x61, 0x1c, 0xb6, 0xbc, 0x98, 0x38, 0xcc, 0x3d, 0xe0, 0x24, 0x1c, 0x3f, 0xde,
0xcf, 0x1a, 0x3e, 0xc5, 0xf0, 0x46, 0x3f, 0x6f, 0x56, 0x13, 0xbf, 0x32, 0x00, 0xa8, 0xbc, 0x84, 0xff, 0x52, 0x3e, 0xa7, 0xae, 0x76,
0x3f, 0xe1, 0x10, 0xef, 0x3e, 0xb8, 0x83, 0xed, 0x3e, 0xc3, 0xe1, 0xad, 0xbe, 0x29, 0x1d, 0xef, 0x3c, 0x00, 0x00, 0x06, 0x00, 0x08,
0x00, 0x04, 0x00, 0x06, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x35, 0xa9, 0xec, 0x3e, 0x00, 0x00, 0x00,
0x00, 0x71, 0xad, 0x0e, 0xbf, 0x00, 0x00, 0x00, 0x00, 0xe2, 0xc0, 0x4a, 0xbe, 0xd4, 0x3d, 0xf6, 0xbc, 0x8f, 0x46, 0x20, 0x3d, 0xe6,
0xff, 0x88, 0x3f, 0xf3, 0xea, 0x20, 0x3f, 0x01, 0xae, 0xb8, 0xbc, 0x49, 0x3b, 0x07, 0x3f, 0xcc, 0x1d, 0x23, 0xbd, 0xdc, 0xa2, 0x6b,
0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x54, 0x2a, 0x44, 0x3f, 0x98, 0xfb, 0xff, 0xff, 0x9c, 0xfb, 0xff, 0xff, 0x0f, 0x00,
0x00, 0x00, 0x4d, 0x4c, 0x49, 0x52, 0x20, 0x43, 0x6f, 0x6e, 0x76, 0x65, 0x72, 0x74, 0x65, 0x64, 0x2e, 0x00, 0x01, 0x00, 0x00, 0x00,
0x14, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x18, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00, 0x0e, 0x00,
0x00, 0x00, 0x30, 0x01, 0x00, 0x00, 0x24, 0x01, 0x00, 0x00, 0x18, 0x01, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00,
0x00, 0x04, 0x00, 0x00, 0x00, 0x6d, 0x61, 0x69, 0x6e, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x78,
0x00, 0x00, 0x00, 0x34, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x10, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c,
0x00, 0x0a, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x0a, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0xce, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x08, 0x18, 0x00,
0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x44, 0xfc, 0xff, 0xff, 0x01, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00,
0x03, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x14,
0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x07, 0x00, 0x10, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x1c, 0x00, 0x00,
0x00, 0x10, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xba, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x01, 0x01, 0x00, 0x00, 0x00, 0x08,
0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00,
0x0e, 0x00, 0x16, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x07, 0x00, 0x10, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08,
0x24, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00, 0x08, 0x00, 0x07, 0x00, 0x06, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0b, 0x00, 0x00, 0x00, 0xe8, 0x02, 0x00, 0x00, 0x88, 0x02, 0x00, 0x00, 0x34, 0x02, 0x00, 0x00, 0xe0,
0x01, 0x00, 0x00, 0x9c, 0x01, 0x00, 0x00, 0x54, 0x01, 0x00, 0x00, 0x0c, 0x01, 0x00, 0x00, 0xc8, 0x00, 0x00, 0x00, 0x84, 0x00, 0x00,
0x00, 0x3c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x52, 0xfd, 0xff, 0xff, 0x24, 0x00, 0x00, 0x00, 0x0b, 0x00, 0x00, 0x00, 0x0c,
0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x44, 0xfd, 0xff, 0xff, 0x08, 0x00, 0x00, 0x00, 0x49, 0x64, 0x65, 0x6e, 0x74, 0x69, 0x74,
0x79, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x86, 0xfd, 0xff, 0xff, 0x34,

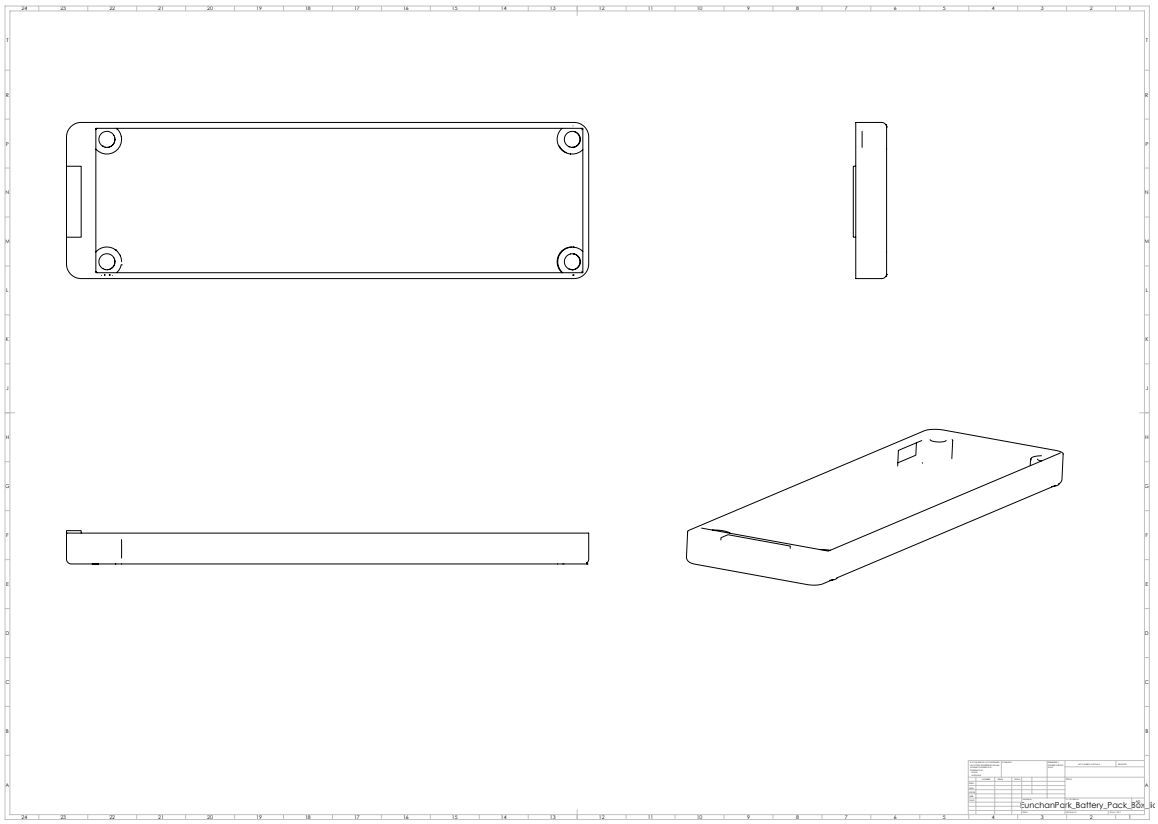
```
0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x78, 0xfd, 0xff, 0xff, 0x1a, 0x00, 0x00,
0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x32, 0x2f, 0x42, 0x69,
0x61, 0x73, 0x41, 0x64, 0x64, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0xca, 0xfd, 0xff,
0xff, 0x30, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xbc, 0xfd, 0xff, 0xff, 0x17,
0x00, 0x00, 0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x31, 0x2f,
0x52, 0x65, 0x6c, 0x75, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0a, 0xfe, 0xff, 0xff, 0x30,
0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xfc, 0xfd, 0xff, 0xff, 0x15, 0x00, 0x00,
0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x2f, 0x52, 0x65, 0x6c, 0x75,
0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x4a, 0xfe, 0xff, 0xff, 0x34, 0x00, 0x00,
0x00, 0x07, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x3c, 0xfe, 0xff, 0xff, 0x19, 0x00, 0x00, 0x00, 0x73,
0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x32, 0x2f, 0x4d, 0x61, 0x74, 0x4d,
0x75, 0x6c, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x8e, 0xfe, 0xff, 0xff, 0x34,
0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x80, 0xfe, 0xff, 0xff, 0x19, 0x00, 0x00,
0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x31, 0x2f, 0x4d, 0x61,
0x74, 0x4d, 0x75, 0x6c, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0xd2, 0xfe, 0xff,
0xff, 0x30, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xc4, 0xfe, 0xff, 0xff, 0x17,
0x00, 0x00, 0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x2f, 0x4d, 0x61,
0x74, 0x4d, 0x75, 0x6c, 0x00, 0x02, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x11, 0x00, 0x00, 0x00, 0x12, 0xff, 0xff, 0xff, 0x44,
0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x04, 0xff, 0xff, 0xff, 0x29, 0x00, 0x00,
0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x32, 0x2f, 0x42, 0x69,
0x61, 0x73, 0x41, 0x64, 0x64, 0x2f, 0x52, 0x65, 0x61, 0x64, 0x56, 0x61, 0x72, 0x69, 0x61, 0x62, 0x6c, 0x65, 0x4f, 0x70, 0x00, 0x00,
0x00, 0x01, 0x00, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x62, 0xff, 0xff, 0xff, 0x44, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x0c,
0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x54, 0xff, 0xff, 0xff, 0x29, 0x00, 0x00, 0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74,
0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x31, 0x2f, 0x42, 0x69, 0x61, 0x73, 0x41, 0x64, 0x64, 0x2f, 0x52, 0x65,
0x61, 0x64, 0x56, 0x61, 0x72, 0x69, 0x61, 0x62, 0x6c, 0x65, 0x4f, 0x70, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00,
0x00, 0xb2, 0xff, 0xff, 0xff, 0x40, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xa4,
0xff, 0xff, 0xff, 0x27, 0x00, 0x00, 0x00, 0x73, 0x65, 0x71, 0x75, 0x65, 0x6e, 0x74, 0x69, 0x61, 0x6c, 0x2f, 0x64, 0x65, 0x6e, 0x73,
0x65, 0x2f, 0x42, 0x69, 0x61, 0x73, 0x41, 0x64, 0x64, 0x2f, 0x52, 0x65, 0x61, 0x64, 0x56, 0x61, 0x72, 0x69, 0x61, 0x62, 0x6c, 0x65,
0x4f, 0x70, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x14, 0x00, 0x04, 0x00, 0x00, 0x00, 0x08,
0x00, 0x0c, 0x00, 0x10, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x08,
0x00, 0x00, 0x00, 0x04, 0x00, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00, 0x0b, 0x00, 0x00, 0x00, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x69,
0x6e, 0x70, 0x75, 0x74, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x11, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
0x20, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00, 0x06, 0x00, 0x05, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x0e,
0x0a, 0x00, 0x0c, 0x00, 0x07, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x09, 0x03, 0x00, 0x00, 0x00

};


#endif //MODELO_COLAB_H
```

Apéndice 6. Cajas.


31.50
R4.00
14.00
54.50
2.50
Ø1.35
1.75
R2.59
R2.34
3.00
2.39
1.00
8.10
1.08
23.50
PART-Lid-DESC
BKG_ESP32_1


31.50
9.50
3.13
R4.00
Ø1.00
R0.50
54.50
8.00
39.12
0.50
Ø2.08
Ø2.60
6.25
PART-Lid-DESC
BKG_ESP32_2


EunchanPark_Battery_Pack_Box_t



Apéndice 7. Códigos en Arduino IDE.

```
//RECOLECCION_DATOS

#include <Wire.h>
#include  "BluetoothSerial.h"
#include <TensorFlowLite.h>
#include "tensorflow/lite/experimental/micro/kernels/all_ops_resolver.h"
#include "tensorflow/lite/experimental/micro/micro_error_reporter.h"
#include "tensorflow/lite/experimental/micro/micro_interpreter.h"
#include "tensorflow/lite/experimental/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"
#include "tensorflow/lite/experimental/micro/kernels/micro_ops.h"
#include "modelo_colab.h"
#define Sensor1 39
#define Sensor2 35
#define Sensores 33
#define Sensor11 34
#define Sensor12 32
#define Hilo 36
#define Selector0 25
#define Selector1 26
#define Selector2 27
#define MPU 0x68
#define MPU2 0x69
#define FS_A 16384.0
#define FS_G 131.0
#define R_G 57.295779

namespace {
const tflite::Model* model = nullptr;
tflite::ErrorReporter* error_reporter = nullptr;
tflite::MicroInterpreter* interpreter = nullptr;
TfLiteTensor* input = nullptr;
TfLiteTensor* output = nullptr;
constexpr int kTensorArenaSize = 7 * 1024;
```

```
uint8_t tensor_arena[kTensorArenaSize];

}

BluetoothSerial SerialBT;

char *vector[] = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "Ñ", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y",
"Z", "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre",  "Octubre", "Noviembre", "Diciembre",
"Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo", "Uno", "Dos", "Tres", "Cuatro", "Cinco", "Seis", "Siete",
"Ocho", "Nueve", "Diez", "Hola", "Bien", "Mal", "Perdon", "No", "Si"

        };

const int selectPins[3] = {Selector0, Selector1, Selector2};

float mpu [4];

float mpu2 [4];

float resta [4];

float entradas[18];

float salidas[62];

String letra;

float numero;

float in [13];

float Acc[2];

float Gy[2];

float Acc2[2];

float Gy2[2];

int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;

int16_t AcX2, AcY2, AcZ2, GyX2, GyY2, GyZ2;

int cont;

void Multiplexor(byte pin) {

 for (int i = 0; i < 3; i++)

 {

  if (pin & (1 << i))

   digitalWrite(selectPins[i], HIGH);

  else

   digitalWrite(selectPins[i], LOW);

 }

}


void setup() {

 Serial.begin(115200);
```

```
SerialBT.begin("ESP32");
static tflite::MicroErrorReporter micro_error_reporter;
error_reporter = &micro_error_reporter;
model = tflite::GetModel(modelo_colab);
if (model->version() != TFLITE_SCHEMA_VERSION) {
 error_reporter->Report("La versión de TF Lite del modelo y arduino es distinta");
}
static tflite::MicroMutableOpResolver micro_mutable_op_resolver;
micro_mutable_op_resolver.AddBuiltin(
 tflite::BuiltinOperator_FULLY_CONNECTED,
 tflite::ops::micro::Register_FULLY_CONNECTED(),
 1, 3);
static tflite::MicroInterpreter static_interpreter( model,
   micro_mutable_op_resolver,
   tensor_arena,
   kTensorArenaSize,
   error_reporter);
interpreter = &static_interpreter;
TfLiteStatus allocate_status = interpreter-> AllocateTensors();
if (allocate_status != kTfLiteOk) {
 error_reporter->Report("AllocateTensors() failed");
}
input = interpreter->input(0);
output = interpreter->output(0);
pinMode(Sensores, INPUT);
pinMode(Sensor1, INPUT);
pinMode(Sensor2, INPUT);
pinMode(Sensor11, INPUT);
pinMode(Sensor12, INPUT);
pinMode(Hilo, INPUT);
for (int i = 0; i < 3; i++)
{
 pinMode(selectPins[i], OUTPUT); // Como pines de salida
 digitalWrite(selectPins[i], HIGH); // Como nivel lógico "H" - "alto"
}
```

```
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Wire.begin();
  Wire.beginTransmission(MPU2);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
}
void loop() {
  for (int i = 0; i < 62; i++) {
    cont = 0;
    delay(2000);
    do {
      Wire.beginTransmission(MPU);
      Wire.write(0x3B);
      Wire.endTransmission(false);
      Wire.requestFrom(MPU, 6, true);
      AcX = Wire.read() << 8 | Wire.read();
      AcY = Wire.read() << 8 | Wire.read();
      AcZ = Wire.read() << 8 | Wire.read();
      Acc[1] = atan(-1 * (AcX / FS_A) / sqrt(pow((AcY / FS_A), 2) + pow((AcZ / FS_A), 2))) * R_G;
      Acc[0] = atan((AcY / FS_A) / sqrt(pow((AcX / FS_A), 2) + pow((AcZ / FS_A), 2))) * R_G;
      Wire.beginTransmission(MPU);
      Wire.write(0x43);
      Wire.endTransmission(false);
      Wire.requestFrom(MPU, 4, true);
      GyX = Wire.read() << 8 | Wire.read();
      GyY = Wire.read() << 8 | Wire.read();

      Gy[0] = GyX / FS_G;
      Gy[1] = GyY / FS_G;
      mpu[0] = 0.98 * (mpu[0] + Gy[0] * 0.010) + 0.02 * Acc[0];
```

```
mpu[1] = 0.98 * (mpu[1] + Gy[1] * 0.010) + 0.02 * Acc[1];
delay(10);
Wire.beginTransmission(MPU2);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU2, 6, true);
AcX2 = Wire.read() << 8 | Wire.read();
AcY2 = Wire.read() << 8 | Wire.read();
AcZ2 = Wire.read() << 8 | Wire.read();
Acc2[1] = atan(-1 * (AcX2 / FS_A) / sqrt(pow((AcY2 / FS_A), 2) + pow((AcZ2 / FS_A), 2))) * R_G;
Acc2[0] = atan((AcY2 / FS_A) / sqrt(pow((AcX2 / FS_A), 2) + pow((AcZ2 / FS_A), 2))) * R_G;
Wire.beginTransmission(MPU2);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU2, 4, true);
GyX2 = Wire.read() << 8 | Wire.read();
GyY2 = Wire.read() << 8 | Wire.read();
Gy2[0] = GyX2 / FS_G;
Gy2[1] = GyY2 / FS_G;
mpu[2] = 0.98 * (mpu[2] + Gy2[0] * 0.3) + 0.02 * Acc2[0];
mpu[3] = 0.98 * (mpu[3] + Gy2[1] * 0.3) + 0.02 * Acc2[1];
delay(300);
Wire.beginTransmission(MPU);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true);
AcX = Wire.read() << 8 | Wire.read();
AcY = Wire.read() << 8 | Wire.read();
AcZ = Wire.read() << 8 | Wire.read();
Acc[1] = atan(-1 * (AcX / FS_A) / sqrt(pow((AcY / FS_A), 2) + pow((AcZ / FS_A), 2))) * R_G;
Acc[0] = atan((AcY / FS_A) / sqrt(pow((AcX / FS_A), 2) + pow((AcZ / FS_A), 2))) * R_G;

Wire.beginTransmission(MPU);
Wire.write(0x43);
Wire.endTransmission(false);
```

```
Wire.requestFrom(MPU, 4, true);
GyX = Wire.read() << 8 | Wire.read();
GyY = Wire.read() << 8 | Wire.read();
Gy[0] = GyX / FS_G;
Gy[1] = GyY / FS_G;
mpu2[0] = 0.98 * (mpu2[0] + Gy[0] * 0.010) + 0.02 * Acc[0];
mpu2[1] = 0.98 * (mpu2[1] + Gy[1] * 0.010) + 0.02 * Acc[1];
delay(10);
Wire.beginTransmission(MPU2);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU2, 6, true);
AcX2 = Wire.read() << 8 | Wire.read();
AcY2 = Wire.read() << 8 | Wire.read();
AcZ2 = Wire.read() << 8 | Wire.read();
Acc2[1] = atan(-1 * (AcX2 / FS_A) / sqrt(pow((AcY2 / FS_A), 2) + pow((AcZ2 / FS_A), 2))) * R_G;
Acc2[0] = atan((AcY2 / FS_A) / sqrt(pow((AcX2 / FS_A), 2) + pow((AcZ2 / FS_A), 2))) * R_G;
Wire.beginTransmission(MPU2);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU2, 4, true);
GyX2 = Wire.read() << 8 | Wire.read();
GyY2 = Wire.read() << 8 | Wire.read();
Gy2[0] = GyX2 / FS_G;
Gy2[1] = GyY2 / FS_G;
mpu2[2] = 0.98 * (mpu2[2] + Gy2[0] * 0.01) + 0.02 * Acc2[0];
mpu2[3] = 0.98 * (mpu2[3] + Gy2[1] * 0.01) + 0.02 * Acc2[1];
delay(10);
resta[0] = abs ( mpu[0] - mpu2[0]);
resta[1] = abs ( mpu[1] - mpu2[1]);
resta[2] = abs ( mpu[2] - mpu2[2]);
resta[3] = abs ( mpu[3] - mpu2[3]);
entradas[12] = 0;
if (resta[0] > 15) {
  entradas[12] = 1;
```

```
}
else if (resta[1] > 15) {
 entradas[12] = 1;
}
else if (resta[2] > 15) {
 entradas[12] = 1;
}
else if (resta[3] > 15) {
 entradas[12] = 1;
}
in[0] = analogRead(Sensor1);
entradas[0] = in[0] / 4095;
in[1] = analogRead(Sensor2);
entradas[1] = in[1] / 4095;
in[10] = analogRead(Sensor11);
entradas[10] = in[10] / 4095;
in[11] = analogRead(Sensor12);
entradas[11] = in [11] / 4095;
in[12] = analogRead(Hilo);
for (byte pin = 2; pin <= 9; pin++)
{
 Multiplexor(pin);
 in [pin] = analogRead(Sensores);
 entradas[pin] = in[pin] / 4095;
}
int x = 0;
do {
 Wire.beginTransmission(MPU);
 Wire.write(0x3B);
 Wire.endTransmission(false);
 Wire.requestFrom(MPU, 6, true);
 AcX = Wire.read() << 8 | Wire.read();
 AcY = Wire.read() << 8 | Wire.read();
 AcZ = Wire.read() << 8 | Wire.read();
 Acc[1] = atan(-1 * (AcX / FS_A) / sqrt(pow((AcY / FS_A), 2) + pow((AcZ / FS_A), 2))) * R_G;
```

```
Acc[0] = atan((AcY / FS_A) / sqrt(pow((AcX / FS_A), 2) + pow((AcZ / FS_A), 2))) * R_G;
Wire.beginTransmission(MPU);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU, 4, true);
GyX = Wire.read() << 8 | Wire.read();
GyY = Wire.read() << 8 | Wire.read();
Gy[0] = GyX / FS_G;
Gy[1] = GyY / FS_G;
entradas[13] = 0.98 * (entradas[13] + Gy[0] * 0.010) + 0.02 * Acc[0];
entradas[13] = (entradas[13] + 100) / 200;
entradas[14] = 0.98 * (entradas[14] + Gy[1] * 0.010) + 0.02 * Acc[1];
entradas[14] = (entradas[14] + 100) / 200;
delay(10);
Wire.beginTransmission(MPU2);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU2, 6, true);
AcX2 = Wire.read() << 8 | Wire.read();
AcY2 = Wire.read() << 8 | Wire.read();
AcZ2 = Wire.read() << 8 | Wire.read();
Acc2[1] = atan(-1 * (AcX2 / FS_A) / sqrt(pow((AcY2 / FS_A), 2) + pow((AcZ2 / FS_A), 2))) * R_G;
Acc2[0] = atan((AcY2 / FS_A) / sqrt(pow((AcX2 / FS_A), 2) + pow((AcZ2 / FS_A), 2))) * R_G;
Wire.beginTransmission(MPU2);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU2, 4, true);
GyX2 = Wire.read() << 8 | Wire.read();
GyY2 = Wire.read() << 8 | Wire.read();
Gy2[0] = GyX2 / FS_G;
Gy2[1] = GyY2 / FS_G;

entradas[15] = 0.98 * (entradas[15] + Gy2[0] * 0.001) + 0.02 * Acc2[0];
entradas[15] = (entradas[15] + 100) / 200;
entradas[16] = 0.98 * (entradas[16] + Gy2[1] * 0.001) + 0.02 * Acc2[1];
```

```
      entradas[16] = (entradas[16] + 100) / 200;
      x++;
      delay (1);
     }
     while (x != 150);
     for (int u = 0; u < 18; u++) {
      input->data.f[u] = entradas[u];
     }
     TfLiteStatus invoke_status = interpreter->Invoke();
     if (invoke_status != kTfLiteOk) {
      error_reporter->Report("Invoke failed\n");
     }
     for (int u = 0; u < 18; u++) {
      salidas [i] = output->data.f[u];
     }
     numero = salidas[0];
     letra = vector [0];
     for (int i = 0; i <= 61; i++) {
      if (salidas[i] >= numero) {
        numero = salidas[i];
        letra = vector[i];
      }
     }
     Serial.println(vector[i]);
     for (int i = 0; i < 18; i++) {
      Serial.println(entradas[i], 10);
     }
     Serial.println(i);
     delay(300);
     cont++;
    } while (cont < 5);
   }
}
```

Apéndice 8. Valores mínimos y máximos por seña.

APÉNDICE 8. VALORES MÍNIMOS Y MÁXIMOS POR SEÑA.

Valores máximos de cada sensor por seña														
Etiqueta	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**M1X**	**M1Y**	**M2X**	**M2Y**
0	1146	1021	519	903	457	720	657	555	688	801	81	12	-3	-76
1	926	736	1255	1133	1265	1106	1460	1331	1494	1011	83	9	-1	-75
2	983	852	669	1074	607	978	1153	1115	911	958	84	12	7	-69
3	1069	929	676	993	507	915	814	684	1469	1087	81	16	13	-66
4	1046	930	641	1161	538	1046	939	1034	733	1047	84	11	-3	-76
5	1093	1010	1206	1136	1263	1158	1467	1395	805	779	84	10	-5	-75
6	1185	1098	575	931	491	779	769	519	1499	999	10	37	102	7
7	1165	1072	1238	1121	422	848	800	531	1491	1051	11	37	104	4
8	1029	891	639	849	637	734	1484	1343	869	816	74	14	-8	-62
9	1058	925	671	932	592	740	1479	1326	912	834	55	85	97	-28
10	1115	965	1108	848	571	805	716	466	1494	1034	70	59	43	-41
11	1143	1031	559	932	489	742	728	493	1499	1008	76	17	-4	-78
12	879	750	745	956	636	944	656	615	894	806	68	24	-2	-55
13	906	781	790	926	396	816	677	499	901	834	65	26	-1	-53
14	948	816	895	964	427	819	717	535	1040	835	18	81	27	8
15	1088	1072	829	1005	663	912	1104	896	930	851	83	18	12	-77
16	1099	970	1035	858	590	721	763	493	1497	1008	78	17	12	-71
17	1083	1015	592	944	500	734	736	528	1011	881	36	96	122	35
18	942	803	1245	1122	413	881	816	588	1498	1079	84	8	-4	-84
19	1006	933	526	971	593	795	755	570	774	887	83	6	-7	-78
20	950	816	573	970	527	874	749	642	912	791	83	7	-8	-76
21	913	778	1243	1136	465	895	749	649	1503	1099	85	9	-3	-75
22	1086	910	1251	1173	611	1038	901	790	1434	1110	83	7	-1	-75
23	911	790	1213	1168	1278	1102	659	638	1476	1115	84	9	-1	-75
24	1090	1030	533	915	448	743	723	533	942	918	39	49	112	64
25	1155	1047	618	847	560	705	1482	1319	882	867	21	32	106	-10
26	1061	900	672	943	576	848	816	580	1502	1015	17	67	31	11
27	1030	913	608	1155	469	1088	876	895	693	1045	83	33	35	-76
28	1113	1057	1215	1135	1245	1168	1472	1341	1409	1030	84	34	39	-72
29	1010	938	1190	926	1169	983	814	724	1460	829	65	32	25	-12
30	1136	1014	501	935	402	729	693	548	656	830	66	17	-10	-53
31	943	820	1229	1141	1275	1091	747	925	1495	995	79	56	66	-69
32	1059	966	670	896	574	690	1495	1358	1010	825	81	67	59	-70
33	1142	1049	592	911	545	729	1477	1328	1499	1043	70	36	61	-59
34	1153	1065	528	925	441	779	690	544	1249	787	77	63	50	-63
35	1070	847	553	934	528	835	738	606	847	861	80	49	37	-64
36	1074	1008	927	990	701	1008	1058	819	995	821	81	47	46	-69
37	1014	880	1232	1137	466	907	791	607	1486	1053	78	55	66	-66
38	1119	1039	738	967	687	927	1094	752	1462	1047	79	58	64	-58
39	1137	1019	592	926	499	810	716	551	1488	1059	55	51	19	-27
40	1136	1067	1245	1122	432	874	773	629	1495	1072	69	52	26	-23
41	892	780	1247	1118	1302	1070	778	824	1524	1073	70	40	33	-27
42	1047	976	663	858	571	761	1479	1348	981	822	63	47	15	-24
43	971	890	1246	1136	449	917	813	604	1506	1055	62	51	28	-26
44	1055	1011	511	870	490	798	703	746	724	819	66	50	29	-19
45	1088	983	739	982	582	891	774	771	1510	1060	61	53	25	-25

46	1065	944	641	971	555	928	751	628	1478	1043	47	24	64	-47
47	978	807	1245	1133	461	942	738	651	1428	1073	48	21	60	-51
48	955	811	1243	1185	1264	1131	747	734	1424	1046	48	22	60	-51
49	916	816	1291	1178	1155	1127	1459	1303	1424	1041	47	24	59	-50
50	1148	1045	1219	1154	1129	1151	1487	1260	1409	1040	48	22	60	-52
51	989	890	527	922	405	784	695	597	1223	876	9	42	103	5
52	1152	1021	555	969	457	869	753	675	1449	1085	50	17	75	-52
53	1161	1029	1232	1174	480	942	794	678	1419	1062	52	16	68	-55
54	1038	896	1168	1123	1174	1133	1415	1205	1442	976	74	30	2	-62
55	1146	1040	1163	1158	1120	1150	1402	1189	1357	1054	34	88	37	22
56	1161	1087	1211	1123	397	858	805	535	1488	1007	43	55	20	-17
57	983	783	1257	1104	1248	1083	1464	1309	1488	898	92	34	82	-68
58	979	797	1191	1127	1216	1035	1469	1332	1488	956	74	65	18	-5
59	1150	1069	731	995	739	666	1485	1360	1381	807	80	16	7	-74
60	1150	1086	1217	1050	443	922	767	497	1488	974	69	24	-5	-57
61	1142	1072	989	955	638	817	1474	1314	1193	816	69	20	-12	-37

APÉNDICE 8. VALORES MÍNIMOS Y MÁXIMOS POR SEÑA.

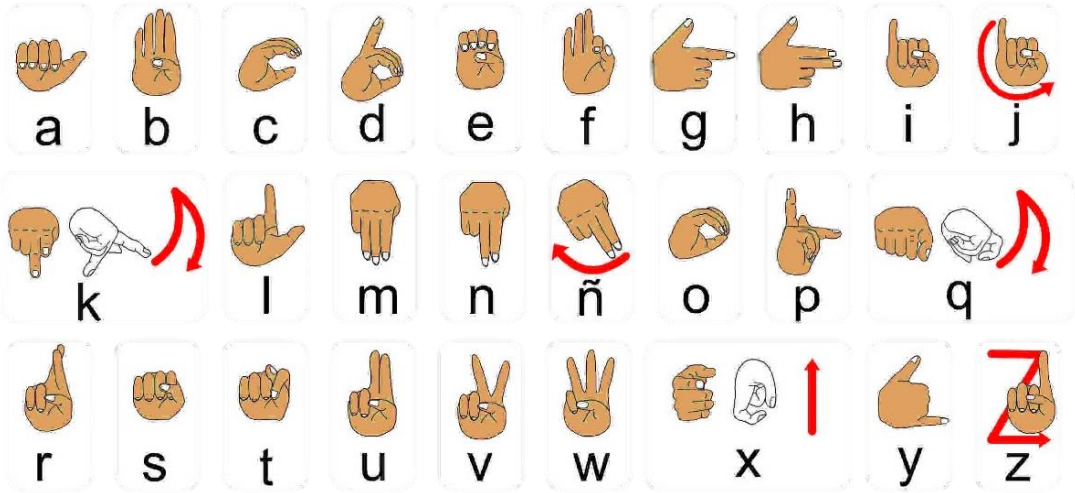
Valores mínimos de cada sensor por seña														
Etiqueta	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**M1X**	**M1Y**	**M2X**	**M2Y**
0	1104	983	438	880	384	650	528	423	626	783	77	2	-12	-82
1	862	688	1168	1107	1174	1010	1451	1246	1485	934	81	6	-5	-79
2	945	810	621	1043	490	930	979	1099	838	893	81	10	3	-77
3	1023	838	634	901	495	820	765	533	1401	1032	78	12	9	-71
4	1011	907	623	1138	470	993	877	991	661	1033	82	8	-6	-80
5	1051	915	1181	1111	1199	1115	1454	1326	725	725	76	4	-10	-77
6	1142	1011	463	887	375	752	640	448	1483	959	4	29	92	0
7	1149	1023	1232	1104	368	784	720	497	1488	1010	6	27	92	-1
8	951	833	597	811	567	670	1471	1307	779	756	65	10	-14	-73
9	986	801	599	858	465	688	1471	1264	816	785	22	-17	-52	-55
10	1063	917	1057	790	490	695	558	382	1445	893	33	9	-10	-75
11	1136	976	455	912	438	714	615	471	1472	899	73	12	-11	-82
12	716	698	656	847	507	817	583	496	672	699	58	20	-7	-60
13	827	696	671	843	355	765	603	451	816	771	59	21	-4	-55
14	883	752	741	858	398	773	639	485	916	763	-5	23	-32	-7
15	1042	1001	769	954	563	894	949	807	851	803	77	11	4	-82
16	1058	944	848	845	567	674	688	467	1489	983	77	16	9	-74
17	1006	923	511	917	437	705	641	464	886	781	-16	18	-27	-45
18	879	753	1157	1072	387	827	721	551	1488	1034	82	5	-6	-87
19	958	853	459	881	481	730	656	545	675	845	81	3	-9	-85
20	813	707	500	921	453	794	688	528	835	766	80	5	-10	-83
21	871	740	1171	1120	385	859	694	578	1481	1003	81	5	-6	-84
22	1001	822	1168	1146	492	966	791	639	1418	1087	82	7	-2	-79
23	863	742	1193	1147	1201	1007	576	591	1424	957	81	7	-3	-80
24	1036	901	448	848	337	683	653	464	816	815	-12	20	72	-29
25	1137	1023	573	813	511	621	1470	1232	656	775	17	26	94	-18
26	1007	835	640	922	526	763	683	506	1491	960	-5	44	-12	-9
27	998	867	573	1131	427	939	763	758	610	979	76	-4	-15	-81
28	989	825	627	971	451	902	739	656	701	683	74	-16	-32	-82
29	991	841	1164	849	1013	894	706	591	1398	787	27	21	-8	-40
30	1130	976	413	843	320	672	589	428	558	767	46	-29	-74	-103
31	867	737	1200	1104	1223	962	624	743	1477	944	68	-15	-18	-82
32	990	883	496	851	417	624	1459	1307	739	741	71	-28	-34	-86
33	1111	946	559	879	474	685	1465	1251	1469	1002	58	-23	-19	-104
34	1096	962	464	862	373	738	618	454	631	406	66	-28	-39	-86
35	902	742	522	896	411	784	630	503	741	767	67	-7	-28	-83
36	1001	960	829	944	596	912	941	752	886	790	70	-15	-34	-81
37	906	757	1200	1099	375	856	642	545	1482	1008	64	-18	-24	-83
38	1088	1009	639	923	570	869	912	640	1446	1031	63	-41	-38	-75
39	1109	976	496	901	400	766	612	496	1457	960	17	36	-14	-65
40	1123	990	1225	1092	400	847	675	581	1473	1039	28	29	-17	-65
41	848	752	1232	1101	1270	996	656	768	1474	1017	31	15	-23	-64
42	995	823	541	750	410	699	1458	1313	826	777	28	25	-13	-65
43	915	813	1227	1066	387	803	662	501	1498	1029	27	30	-10	-62
44	951	887	442	772	423	686	620	601	624	767	28	27	-9	-78
45	946	849	624	951	501	841	697	611	1421	950	24	36	-5	-56

46	1031	855	576	918	449	865	610	540	1399	995	44	20	57	-54
47	955	782	1229	1121	395	901	674	569	1409	1021	47	19	56	-57
48	927	767	1217	1137	1209	1089	675	656	1411	1011	46	19	56	-57
49	885	734	1199	1141	1135	1103	1435	1245	1409	1011	46	19	56	-58
50	1134	1001	1152	1146	1086	1110	1429	1214	1389	993	46	15	58	-62
51	927	849	439	835	347	707	600	540	576	467	4	36	94	1
52	1130	965	526	922	405	799	699	588	1393	994	45	10	59	-64
53	1133	959	1199	1125	357	917	769	639	1392	1015	47	10	56	-64
54	893	741	674	801	592	747	1007	626	794	656	62	-16	-41	-72
55	1121	1001	1118	1136	1038	1092	1377	1105	1319	1001	-19	52	-3	-24
56	1125	1071	1171	998	320	807	641	478	1472	959	18	4	-35	-28
57	928	713	1177	1059	1156	995	1453	1232	1470	870	41	-5	5	-121
58	879	731	1177	1089	1171	880	1456	1297	1470	891	20	8	-15	-82
59	1139	1011	667	884	599	624	1478	1323	967	528	79	14	4	-81
60	1069	877	737	754	357	761	671	416	883	694	62	-3	-27	-69
61	982	787	445	819	387	672	634	542	789	703	42	6	-36	-59

Anexos

Anexo 1. Señas en la LSM.

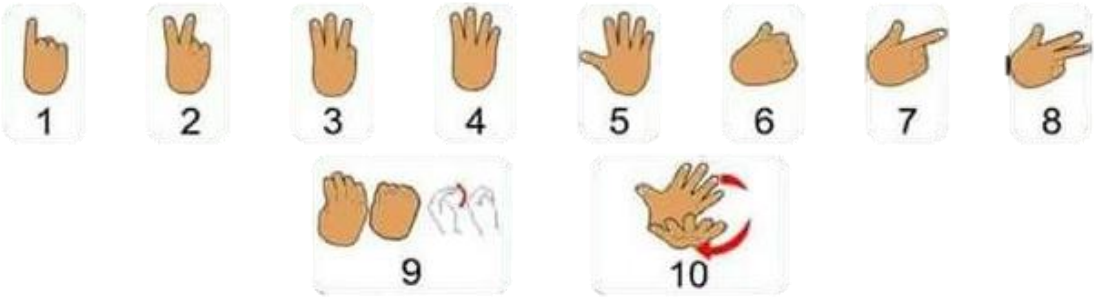
Abecedario



Días de la semana

Meses del año

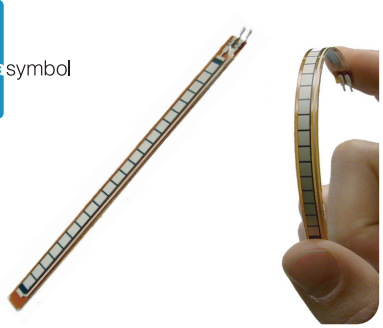
Números 1-10



Palabras

Anexo 2. Hojas de datos.

spectrasymbol



FLEX SENSOR FS

Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
  - Robotics
  - Gaming (Virtual Motion)
  - Medical Devices
  - Computer Peripherals
  - Musical Instruments
  - Physical Therapy
- Simple Construction
- Low Profile

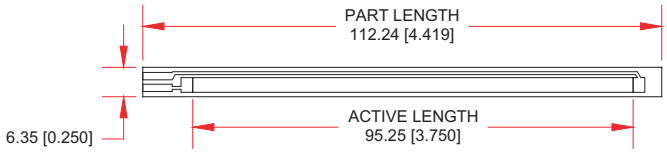
Mechanical Specifications

-Life Cycle: >1 million
-Height: ≤0.43mm (0.017")
-Temperature Range: -35°C to +80°C

Electrical Specifications

-Flat Resistance: 10K Ohms
-Resistance Tolerance: ±30%
-Bend Resistance Range: 60K to 110K Ohms
-Power Rating : 0.50 Watts continuous. 1 Watt Peak

Dimensional Diagram - Stock Flex Sensor



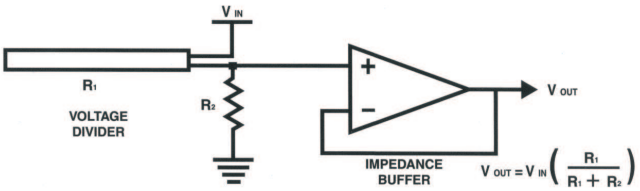
How to Order - Stock Flex Sensor

FS	L	0095	103	ST
Series	Model	Active Length	Resistance	Connectors
FS = Flex Sensor	L = Linear	0095 = 95.25mm	103 = 10 KOhms	ST = Solder Tab

How It Works

Schematics

BASIC FLEX SENSOR CIRCUIT:

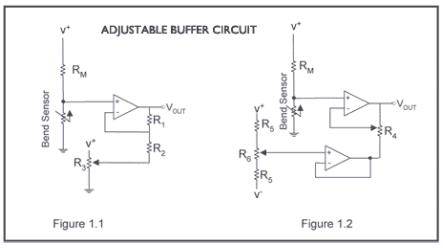


Following are notes from the ITP Flex Sensor Workshop

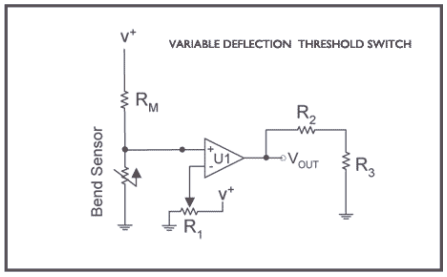
"The impedance buffer in the [Basic Flex Sensor Circuit] (above) is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces errer due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324."

"You can also test your flex sensor using the simplest circut, and skip the op amp."

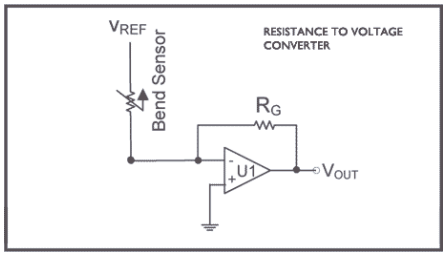
"Adjustable Buffer - a potentiometer can be added to the circuit to adjust the sensitivity range."



"Variable Deflection Threshold Switch - an op amp is used and outputs either high or low depending on the voltage of the inverting input. In this way you can use the flex sensor as a switch without going through a microcontroller."



"Resistance to Voltage Converter - use the sensor as the input of a resistance to voltage converter using a dual sided supply op-amp. A negative reference voltage will give a positive output. Should be used in situations when you want output at a low degree of bending."



ESP32 Series

Datasheet

Including:

ESP32-D0WD-V3

ESP32-D0WDQ6-V3

ESP32-D0WD

ESP32-D0WDQ6

ESP32-D2WD

ESP32-S0WD

ESP32-U4WDH

Version 3.4

Espressif Systems

www.espressif.com

Table 2: Description of ESP32 Power-up and Reset Timing Parameters

Parameters	Description	Min.	Unit
t_0	Time between the 3.3 V rails being brought up and CHIP_PU being activated	50	μs
t_1	Duration of CHIP_PU signal level < V_{IL_nRST} (refer to its value in Table 13 DC Characteristics) to reset the chip	50	μs

- In scenarios where ESP32 is powered on and off repeatedly by switching the power rails, while there is a large capacitor on the VDD33 rail and CHIP_PU and VDD33 are connected, simply switching off the CHIP_PU power rail and immediately switching it back on may cause an incomplete power discharge cycle and failure to reset the chip adequately.
  An additional discharge circuit may be required to accelerate the discharge of the large capacitor on rail VDD33, which will ensure proper power-on-reset when the ESP32 is powered up again. Please find the discharge circuit in Figure **ESP32-WROOM-32 Peripheral Schematics**, in *ESP32-WROOM-32 Datasheet*.
- When a battery is used as the power supply for the ESP32 series of chips and modules, a supply voltage supervisor is recommended, so that a boot failure due to low voltage is avoided. Users are recommended to pull CHIP_PU low if the power supply for ESP32 is below 2.3 V. For the reset circuit, please refer to Figure **ESP32-WROOM-32 Peripheral Schematics**, in *ESP32-WROOM-32 Datasheet*.

Notes on power supply:

- The operating voltage of ESP32 ranges from 2.3 V to 3.6 V. When using a single-power supply, the recommended voltage of the power supply is 3.3 V, and its recommended output current is 500 mA or more.
- When VDD_SDIO 1.8 V is used as the power supply for external flash/PSRAM, a 2-kohm grounding resistor should be added to VDD_SDIO. For the circuit design, please refer to Figure **ESP32-WROVER Schematics**, in *ESP32-WROVER Datasheet*.
- When the three digital power supplies are used to drive peripherals, e.g., 3.3 V flash, they should comply with the peripherals' specifications.

2.4 Strapping Pins

There are five strapping pins:

- MTDI
- GPIO0
- GPIO2
- MTDO
- GPIO5

Software can read the values of these five bits from register "GPIO_STRAPPING".

During the chip's system reset release (power-on-reset, RTC watchdog reset and brownout reset), the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device's boot mode, the operating voltage of VDD_SDIO and other initial system settings.

3. Functional Description

This chapter describes the functions integrated in ESP32.

3.1 CPU and Memory

3.1.1 CPU

ESP32 contains one or two low-power Xtensa® 32-bit LX6 microprocessor(s) with the following features:

- 7-stage pipeline to support the clock frequency of up to 240 MHz (160 MHz for ESP32-S0WD, ESP32-D2WD, and ESP32-U4WDH)
- 16/24-bit Instruction Set provides high code-density
- Support for Floating Point Unit
- Support for DSP instructions, such as a 32-bit multiplier, a 32-bit divider, and a 40-bit MAC
- Support for 32 interrupt vectors from about 70 interrupt sources

The single-/dual-CPU interfaces include:

- Xtensa RAM/ROM Interface for instructions and data
- Xtensa Local Memory Interface for fast peripheral register access
- External and internal interrupt sources
- JTAG for debugging

3.1.2 Internal Memory

ESP32's internal memory includes:

- 448 KB of ROM for booting and core functions
- 520 KB of on-chip SRAM for data and instructions
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.
- Embedded flash

Note:
Products in the ESP32 series differ from each other, in terms of their support for embedded flash and the size of it. For details, please refer to Section 7 *Part Number and Ordering Information*.

3.1.3 External Flash and SRAM

ESP32 supports multiple external QSPI flash and SRAM chips. More details can be found in Chapter SPI in the *ESP32 Technical Reference Manual*. ESP32 also supports hardware encryption/decryption based on AES to protect developers' programs and data in flash.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- Up to 16 MB of external flash can be mapped into CPU instruction memory space and read-only memory space simultaneously.
  - When external flash is mapped into CPU instruction memory space, up to 11 MB + 248 KB can be mapped at a time. Note that if more than 3 MB + 248 KB are mapped, cache performance will be reduced due to speculative reads by the CPU.
  - When external flash is mapped into read-only data memory space, up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads are supported.
- External SRAM can be mapped into CPU data memory space. SRAM up to 8 MB is supported and up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads and writes are supported.

Note:
After ESP32 is initialized, firmware can customize the mapping of external SRAM or flash into the CPU address space.

3.1.4 Memory Map

The structure of address mapping is shown in Figure 7. The memory and peripheral mapping of ESP32 is shown in Table 5.

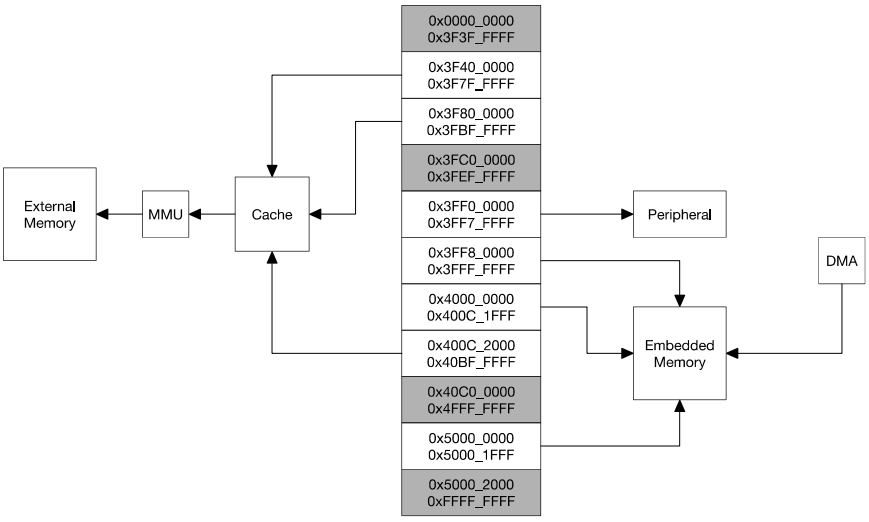


Figure 7: Address Mapping Structure

Table 5: Memory and Peripheral Mapping

Category	Target	Start Address	End Address	Size
Embedded Memory	Internal ROM 0	0x4000_0000	0x4005_FFFF	384 KB
	Internal ROM 1	0x3FF9_0000	0x3FF9_FFFF	64 KB
	Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
	Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
		0x400A_0000	0x400B_FFFF	
	Internal SRAM 2	0x3FFA_E000	0x3FFD_FFFF	200 KB
	RTC FAST Memory	0x3FF8_0000	0x3FF8_1FFF	8 KB
		0x400C_0000	0x400C_1FFF	
	RTC SLOW Memory	0x5000_0000	0x5000_1FFF	8 KB
External Memory	External Flash	0x3F40_0000	0x3F7F_FFFF	4 MB
		0x400C_2000	0x40BF_FFFF	11 MB+248 KB
	External RAM	0x3F80_0000	0x3FBF_FFFF	4 MB
Peripheral	DPort Register	0x3FF0_0000	0x3FF0_0FFF	4 KB
	AES Accelerator	0x3FF0_1000	0x3FF0_1FFF	4 KB
	RSA Accelerator	0x3FF0_2000	0x3FF0_2FFF	4 KB
	SHA Accelerator	0x3FF0_3000	0x3FF0_3FFF	4 KB
	Secure Boot	0x3FF0_4000	0x3FF0_4FFF	4 KB
	Cache MMU Table	0x3FF1_0000	0x3FF1_3FFF	16 KB
	PID Controller	0x3FF1_F000	0x3FF1_FFFF	4 KB
	UART0	0x3FF4_0000	0x3FF4_0FFF	4 KB
	SPI1	0x3FF4_2000	0x3FF4_2FFF	4 KB
	SPI0	0x3FF4_3000	0x3FF4_3FFF	4 KB
	GPIO	0x3FF4_4000	0x3FF4_4FFF	4 KB
	RTC	0x3FF4_8000	0x3FF4_8FFF	4 KB
	IO MUX	0x3FF4_9000	0x3FF4_9FFF	4 KB
	SDIO Slave	0x3FF4_B000	0x3FF4_BFFF	4 KB
	UDMA1	0x3FF4_C000	0x3FF4_CFFF	4 KB
	I2S0	0x3FF4_F000	0x3FF4_FFFF	4 KB
	UART1	0x3FF5_0000	0x3FF5_0FFF	4 KB
	I2C0	0x3FF5_3000	0x3FF5_3FFF	4 KB
	UDMA0	0x3FF5_4000	0x3FF5_4FFF	4 KB
	SDIO Slave	0x3FF5_5000	0x3FF5_5FFF	4 KB
	RMT	0x3FF5_6000	0x3FF5_6FFF	4 KB
	PCNT	0x3FF5_7000	0x3FF5_7FFF	4 KB
	SDIO Slave	0x3FF5_8000	0x3FF5_8FFF	4 KB
	LED PWM	0x3FF5_9000	0x3FF5_9FFF	4 KB
	Efuse Controller	0x3FF5_A000	0x3FF5_AFFF	4 KB
	Flash Encryption	0x3FF5_B000	0x3FF5_BFFF	4 KB
	PWM0	0x3FF5_E000	0x3FF5_EFFF	4 KB
	TIMG0	0x3FF5_F000	0x3FF5_FFFF	4 KB
	TIMG1	0x3FF6_0000	0x3FF6_0FFF	4 KB

  - Enhanced power control
  - Ping
- Bluetooth Low Energy
  - Advertising
  - Scanning
  - Simultaneous advertising and scanning
  - Multiple connections
  - Asynchronous data reception and transmission
  - Adaptive Frequency Hopping and Channel assessment
  - Connection parameter update
  - Data Length Extension
  - Link Layer Encryption
  - LE Ping

3.7 RTC and Low-Power Management

With the use of advanced power-management technologies, ESP32 can switch between different power modes.

- Power modes
  - **Active mode**: The chip radio is powered on. The chip can receive, transmit, or listen.
  - **Modem-sleep mode**: The CPU is operational and the clock is configurable. The Wi-Fi/Bluetooth baseband and radio are disabled.
  - **Light-sleep mode**: The CPU is paused. The RTC memory and RTC peripherals, as well as the ULP co-processor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
  - **Deep-sleep mode**: Only the RTC memory and RTC peripherals are powered on. Wi-Fi and Bluetooth connection data are stored in the RTC memory. The ULP co-processor is functional.
  - **Hibernation mode**: The internal 8-MHz oscillator and ULP co-processor are disabled. The RTC recovery memory is powered down. Only one RTC timer on the slow clock and certain RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

Table 6: Power Consumption by Power Modes

Power mode	Description			Power consumption
Active (RF working)	Wi-Fi Tx packet			Please refer to Table 15 for details.
	Wi-Fi/BT Tx packet			
	Wi-Fi/BT Rx and listening			
Modem-sleep	The CPU is powered on.	240 MHz *	Dual-core chip(s)	30 mA ~ 68 mA
			Single-core chip(s)	N/A
		160 MHz *	Dual-core chip(s)	27 mA ~ 44 mA
			Single-core chip(s)	27 mA ~ 34 mA
		Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA

Power mode	Description			Power consumption
			Single-core chip(s)	20 mA ~ 25 mA
Light-sleep	-			0.8 mA
Deep-sleep	The ULP co-processor is powered on.			150 μA
	ULP sensor-monitored pattern			100 μA @1% duty
	RTC timer + RTC memory			10 μA
Hibernation	RTC timer only			5 μA
Power off	CHIP_PU is set to low level, the chip is powered off.			1 μA

Note:

- * Among the ESP32 series of SoCs, ESP32-D0WD-V3, ESP32-D0WDQ6-V3, ESP32-D0WD, and ESP32-D0WDQ6 have a maximum CPU frequency of 240 MHz, ESP32-D2WD, ESP32-S0WD, and ESP32-U4WDH have a maximum CPU frequency of 160 MHz.
- When Wi-Fi is enabled, the chip switches between Active and Modem-sleep modes. Therefore, power consumption changes accordingly.
- In Modem-sleep mode, the CPU frequency changes automatically. The frequency depends on the CPU load and the peripherals used.
- During Deep-sleep, when the ULP co-processor is powered on, peripherals such as GPIO and I^2C are able to operate.
- When the system works in the ULP sensor-monitored pattern, the ULP co-processor works with the ULP sensor periodically and the ADC works with a duty cycle of 1%, so the power consumption is 100 μA.

4. Peripherals and Sensors

4.1 Descriptions of Peripherals and Sensors

4.1.1 General Purpose Input / Output Interface (GPIO)

ESP32 has 34 GPIO pins which can be assigned various functions by programming the appropriate registers. There are several kinds of GPIOs: digital-only, analog-enabled, capacitive-touch-enabled, etc. Analog-enabled GPIOs and Capacitive-touch-enabled GPIOs can be configured as digital GPIOs.

Most of the digital GPIOs can be configured as internal pull-up or pull-down, or set to high impedance. When configured as an input, the input value can be read through the register. The input can also be set to edge-trigger or level-trigger to generate CPU interrupts. Most of the digital IO pins are bi-directional, non-inverting and tristate, including input and output buffers with tristate control. These pins can be multiplexed with other functions, such as the SDIO, UART, SPI, etc. (More details can be found in the Appendix, Table IO_MUX.) For low-power operations, the GPIOs can be set to hold their states.

4.1.2 Analog-to-Digital Converter (ADC)

ESP32 integrates 12-bit SAR ADCs and supports measurements on 18 channels (analog-enabled pins). The ULP-coprocessor in ESP32 is also designed to measure voltage, while operating in the sleep mode, which enables low-power consumption. The CPU can be woken up by a threshold setting and/or via other triggers.

With appropriate settings, the ADCs can be configured to measure voltage on 18 pins maximum.

Table 7 describes the ADC characteristics.

Table 7: ADC Characteristics

Parameter	Description	Min	Max	Unit
DNL (Differential nonlinearity)	RTC controller; ADC connected to an external 100 nF capacitor; DC signal input; ambient temperature at 25 °C; Wi-Fi&BT off	–7	7	LSB
INL (Integral nonlinearity)		–12	12	LSB
Sampling rate	RTC controller	-	200	ksps
	DIG controller	-	2	Msps

Notes:

- When atten=3 and the measurement result is above 3000 (voltage at approx. 2450 mV), the ADC accuracy will be worse than described in the table above.
- To get better DNL results, users can take multiple sampling tests with a filter, or calculate the average value.
- The input voltage range of GPIO pins within VDD3P3_RTC domain should strictly follow the DC characteristics provided in Table 13. Otherwise, measurement errors may be introduced, and chip performance may be affected.

By default, there are ±6% differences in measured results between chips. ESP-IDF provides couple of calibration methods for ADC1. Results after calibration using eFuse Vref value are shown in Table 8. For higher accuracy, users may apply other calibration methods provided in ESP-IDF, or implement their own.

Reliability tests	Standards	Test conditions	Result
Moisture Sensitivity Level (MSL)	J-STD-020, MSL 3	30 °C, 60% RH, 192 hours, IR × 3 @260 °C	Pass

1. JEDEC document JEP157 states that 250 V CDM allows safe manufacturing with a standard ESD control process.
2. JEDEC document JEP155 states that 500 V HBM allows safe manufacturing with a standard ESD control process.

5.5 RF Power-Consumption Specifications

The power consumption measurements are taken with a 3.3 V supply at 25 °C of ambient temperature at the RF port. All transmitters' measurements are based on a 50% duty cycle.

Table 15: RF Power-Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

5.6 Wi-Fi Radio

Table 16: Wi-Fi Radio Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Operating frequency range ^{note1}	-	2412	-	2484	MHz
Output impedance ^{note2}	-	-	*note 2*	-	Ω
TX power ^{note3}	11n, MCS7	12	13	14	dBm
	11b mode	18.5	19.5	20.5	dBm
Sensitivity	11b, 1 Mbps	-	–98	-	dBm
	11b, 11 Mbps	-	–88	-	dBm
	11g, 6 Mbps	-	–93	-	dBm
	11g, 54 Mbps	-	–75	-	dBm
	11n, HT20, MCS0	-	–93	-	dBm
	11n, HT20, MCS7	-	–73	-	dBm
	11n, HT40, MCS0	-	–90	-	dBm
	11n, HT40, MCS7	-	–70	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	27	-	dB
	11g, 54 Mbps	-	13	-	dB
	11n, HT20, MCS0	-	27	-	dB
	11n, HT20, MCS7	-	12	-	dB

1. Device should operate in the frequency range allocated by regional regulatory authorities. Target operating frequency range is configurable by software.
2. The typical value of ESP32's Wi-Fi radio output impedance is different between chips in different QFN packages. For ESP32 chips with a QFN 6×6 package, the value is 30+j10 Ω. For ESP32 chips with a QFN 5×5 package, the value is 35+j10 Ω.
3. Target TX power is configurable based on device or certification requirements.

5.7 Bluetooth Radio

5.7.1 Receiver – Basic Data Rate

Table 17: Receiver Characteristics – Basic Data Rate

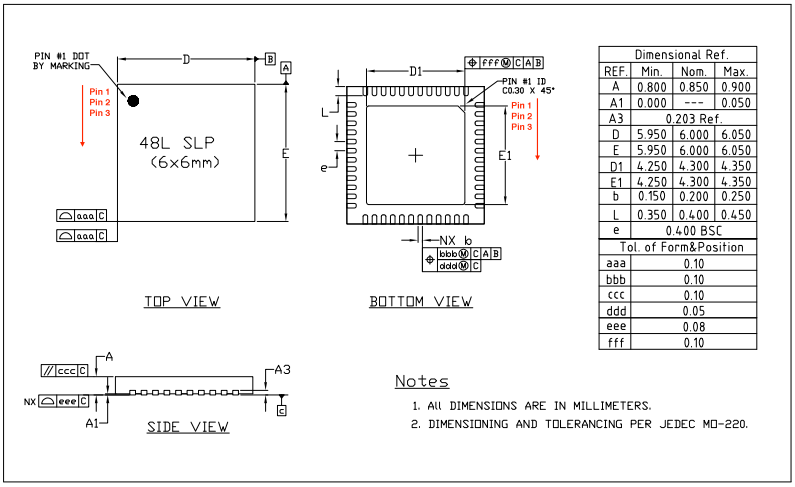
Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @0.1% BER	-	–90	–89	–88	dBm
Maximum received signal @0.1% BER	-	0	-	-	dBm
Co-channel C/I	-	-	+7	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	-	–6	dB
	F = F0 – 1 MHz	-	-	–6	dB
	F = F0 + 2 MHz	-	-	–25	dB
	F = F0 – 2 MHz	-	-	–33	dB
	F = F0 + 3 MHz	-	-	–25	dB
	F = F0 – 3 MHz	-	-	–45	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

5.7.2 Transmitter – Basic Data Rate

Table 18: Transmitter Characteristics – Basic Data Rate

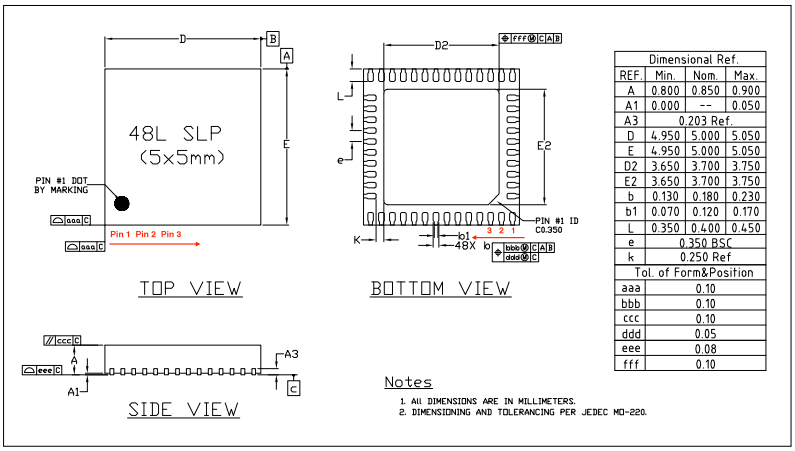
Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power (see note under Table 18)	-	-	0	-	dBm
Gain control step	-	-	3	-	dB
RF power control range	-	–12	-	+9	dBm
+20 dB bandwidth	-	-	0.9	-	MHz
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–47	-	dBm
	F = F0 ± 3 MHz	-	–55	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	155	kHz
$\Delta f2_{max}$	-	133.7	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	0.92	-	-
ICFT	-	-	–7	-	kHz

6. Package Information



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	0.000	---	0.050
A3		0.203 Ref.	
D	5.950	6.000	6.050
E	5.950	6.000	6.050
D1	4.250	4.300	4.350
E1	4.250	4.300	4.350
b	0.150	0.200	0.250
L	0.350	0.400	0.450
e		0.400 BSC	
Tol. of Form&Position			
aaa		0.10	
bbb		0.10	
ccc		0.10	
ddd		0.05	
eee		0.08	
fff		0.10	

Figure 8: QFN48 (6x6 mm) Package



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	0.000	--	0.050
A3		0.203 Ref.	
D	4.950	5.000	5.050
E	4.950	5.000	5.050
D2	3.650	3.700	3.750
E2	3.650	3.700	3.750
b	0.130	0.180	0.230
b1	0.070	0.120	0.170
L	0.350	0.400	0.450
e		0.350 BSC	
k		0.250 Ref	
Tol. of Form&Position			
aaa		0.10	
bbb		0.10	
ccc		0.10	
ddd		0.05	
eee		0.08	
fff		0.10	

Figure 9: QFN48 (5x5 mm) Package

Note:
The pins of the chip are numbered in an anti-clockwise direction from Pin 1 in the top view.

Signal No.	Input signals	Default value if unassigned	Same input signal from IO_MUX core	Output signals	Output enable of output signals
78	pcnt_ctrl_ch1_in6	0	no	ledc_hs_sig_out7	1'd1
79	pcnt_sig_ch0_in7	0	no	ledc_ls_sig_out0	1'd1
80	pcnt_sig_ch1_in7	0	no	ledc_ls_sig_out1	1'd1
81	pcnt_ctrl_ch0_in7	0	no	ledc_ls_sig_out2	1'd1
82	pcnt_ctrl_ch1_in7	0	no	ledc_ls_sig_out3	1'd1
83	rmt_sig_in0	0	no	ledc_ls_sig_out4	1'd1
84	rmt_sig_in1	0	no	ledc_ls_sig_out5	1'd1
85	rmt_sig_in2	0	no	ledc_ls_sig_out6	1'd1
86	rmt_sig_in3	0	no	ledc_ls_sig_out7	1'd1
87	rmt_sig_in4	0	no	rmt_sig_out0	1'd1
88	rmt_sig_in5	0	no	rmt_sig_out1	1'd1
89	rmt_sig_in6	0	no	rmt_sig_out2	1'd1
90	rmt_sig_in7	0	no	rmt_sig_out3	1'd1
91	-	-	-	rmt_sig_out4	1'd1
92	-	-	-	rmt_sig_out6	1'd1
94	-	-	-	rmt_sig_out7	1'd1
95	I2CEXT1_SCL_in	1	no	I2CEXT1_SCL_out	1'd1
96	I2CEXT1_SDA_in	1	no	I2CEXT1_SDA_out	1'd1
97	host_card_detect_n_1	0	no	host_ccmd_od_pullup_en_n	1'd1
98	host_card_detect_n_2	0	no	host_rst_n_1	1'd1
99	host_card_write_prt_1	0	no	host_rst_n_2	1'd1
100	host_card_write_prt_2	0	no	gpio_sd0_out	1'd1
101	host_card_int_n_1	0	no	gpio_sd1_out	1'd1
102	host_card_int_n_2	0	no	gpio_sd2_out	1'd1
103	pwm1_sync0_in	0	no	gpio_sd3_out	1'd1
104	pwm1_sync1_in	0	no	gpio_sd4_out	1'd1
105	pwm1_sync2_in	0	no	gpio_sd5_out	1'd1
106	pwm1_f0_in	0	no	gpio_sd6_out	1'd1
107	pwm1_f1_in	0	no	gpio_sd7_out	1'd1
108	pwm1_f2_in	0	no	pwm1_out0a	1'd1
109	pwm0_cap0_in	0	no	pwm1_out0b	1'd1
110	pwm0_cap1_in	0	no	pwm1_out1a	1'd1
111	pwm0_cap2_in	0	no	pwm1_out1b	1'd1
112	pwm1_cap0_in	0	no	pwm1_out2a	1'd1
113	pwm1_cap1_in	0	no	pwm1_out2b	1'd1
114	pwm1_cap2_in	0	no	pwm2_out1h	1'd1
115	pwm2_flta	1	no	pwm2_out1l	1'd1
116	pwm2_fltb	1	no	pwm2_out2h	1'd1
117	pwm2_cap1_in	0	no	pwm2_out2l	1'd1
118	pwm2_cap2_in	0	no	pwm2_out3h	1'd1

InvenSense Inc.
1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A.
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104
Website: www.invensense.com

Document Number: PS-MPU-6000A-00
Revision: 3.4
Release Date: 08/19/2013

MPU-6000 and MPU-6050
Product Specification
Revision 3.4

Primary Differences between MPU-6000 and MPU-6050

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I^2C, SPI	I^2C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/*g*	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		μg/ √ Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.4 Electrical Specifications, Continued

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
SERIAL INTERFACE						
SPI Operating Frequency, All Registers Read/Write	MPU-6000 only, Low Speed Characterization		100 ±10%		kHz	
	MPU-6000 only, High Speed Characterization		1 ±10%		MHz	
SPI Operating Frequency, Sensor and Interrupt Registers Read Only	MPU-6000 only		20 ±10%		MHz	
I^2C Operating Frequency	All registers, Fast-mode			400	kHz	
	All registers, Standard-mode			100	kHz	
I^2C ADDRESS	AD0 = 0		1101000			
	AD0 = 1		1101001			
DIGITAL INPUTS (SDI/SDA, AD0, SCLK/SCL, FSYNC, /CS, CLKIN)						
V_{IH}, High Level Input Voltage	MPU-6000	0.7*VDD			V	
	MPU-6050	0.7*VLOGIC			V	
V_{IL}, Low Level Input Voltage	MPU-6000			0.3*VDD	V	
	MPU-6050			0.3*VLOGIC	V	
C_I, Input Capacitance			< 5		pF	
DIGITAL OUTPUT (SDO, INT)						
V_{OH}, High Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000	0.9*VDD			V	
	R_{LOAD}=1MΩ; MPU-6050	0.9*VLOGIC			V	
V_{OL1}, LOW-Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000			0.1*VDD	V	
	R_{LOAD}=1MΩ; MPU-6050			0.1*VLOGIC	V	
$V_{OL.INT1}$, INT Low-Level Output Voltage	OPEN=1, 0.3mA sink Current			0.1	V	
Output Leakage Current	OPEN=1		100		nA	
t_{INT}, INT Pulse Width	LATCH_INT_EN=0		50		µs	

6.6 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

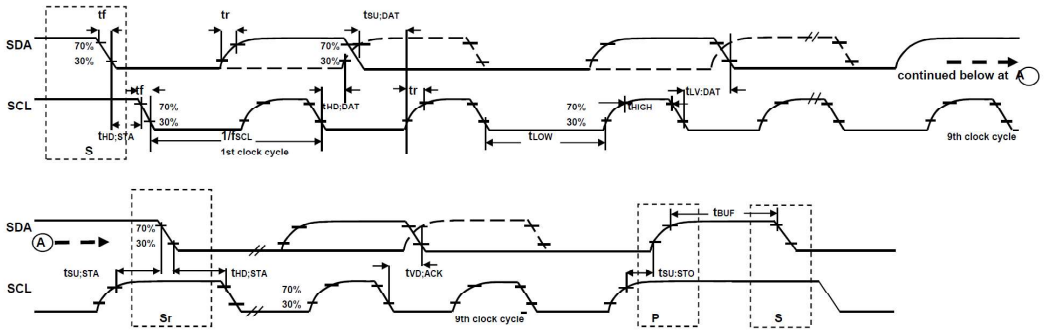
Parameters	Conditions	Min	Typical	Max	Units	Notes
INTERNAL CLOCK SOURCE	**CLK_SEL=0,1,2,3**					
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Clock Frequency Initial Tolerance	CLK_SEL=0, 25°C	-5		+5	%	
	CLK_SEL=1,2,3; 25°C	-1		+1	%	
Frequency Variation over Temperature	CLK_SEL=0		-15 to +10		%	
	CLK_SEL=1,2,3		±1		%	
PLL Settling Time	CLK_SEL=1,2,3		1	10	ms	
EXTERNAL 32.768kHz CLOCK	**CLK_SEL=4**					
External Clock Frequency			32.768		kHz	
External Clock Allowable Jitter	Cycle-to-cycle rms		1 to 2		µs	
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8.192		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1.024		kHz	
Accelerometer Sample Rate			1.024		kHz	
PLL Settling Time			1	10	ms	
EXTERNAL 19.2MHz CLOCK	**CLK_SEL=5**					
External Clock Frequency			19.2		MHz	
Gyroscope Sample Rate	Full programmable range	3.9		8000	Hz	
Gyroscope Sample Rate, Fast Mode	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow Mode	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
PLL Settling Time			1	10	ms	

6.7 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I²C Bus

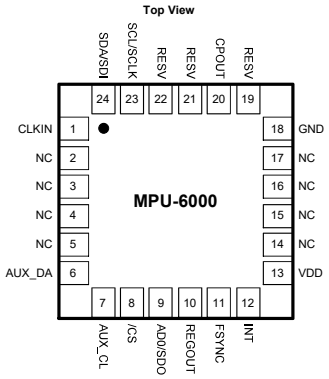


I²C Bus Timing Diagram

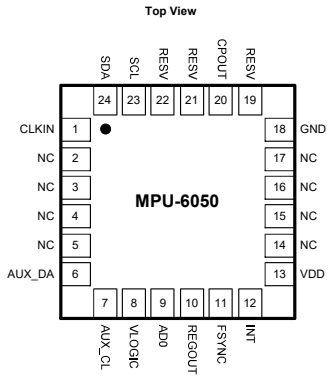
7 Applications Information

7.1 Pin Out and Signal Description

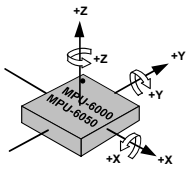
Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
1	Y	Y	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	Y	Y	AUX_DA	I^2C master serial data, for connecting to external sensors
7	Y	Y	AUX_CL	I^2C Master serial clock, for connecting to external sensors
8	Y		/CS	SPI chip select (0=SPI mode)
8		Y	VLOGIC	Digital I/O supply voltage
9	Y		AD0 / SDO	I^2C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I^2C Slave Address LSB (AD0)
10	Y	Y	REGOUT	Regulator filter capacitor connection
11	Y	Y	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	Y	Y	INT	Interrupt digital output (totem pole or open-drain)
13	Y	Y	VDD	Power supply voltage and Digital I/O supply voltage
18	Y	Y	GND	Power supply ground
19, 21	Y	Y	RESV	Reserved. Do not connect.
20	Y	Y	CPOUT	Charge pump capacitor connection
22	Y	Y	RESV	Reserved. Do not connect.
23	Y		SCL / SCLK	I^2C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I^2C serial clock (SCL)
24	Y		SDA / SDI	I^2C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I^2C serial data (SDA)
2, 3, 4, 5, 14, 15, 16, 17	Y	Y	NC	Not internally connected. May be used for PCB trace routing.



QFN Package
24-pin, 4mm x 4mm x 0.9mm

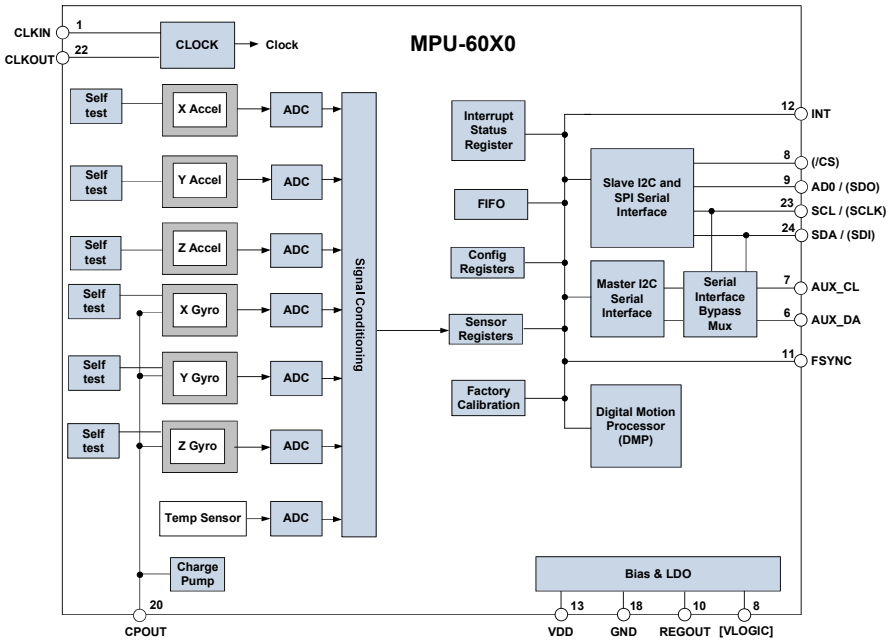


QFN Package
24-pin, 4mm x 4mm x 0.9mm



Orientation of Axes of Sensitivity and Polarity of Rotation

7.5 Block Diagram



Note: Pin names in round brackets ( ) apply only to MPU-6000
Pin names in square brackets [ ] apply only to MPU-6050

7.6 Overview

The MPU-60X0 is comprised of the following key blocks and functions:

- Three-axis MEMS rate gyroscope sensor with 16-bit ADCs and signal conditioning
- Three-axis MEMS accelerometer sensor with 16-bit ADCs and signal conditioning
- Digital Motion Processor (DMP) engine
- Primary I^2C and SPI (MPU-6000 only) serial communications interfaces
- Auxiliary I^2C serial interface for 3rd party magnetometer & other sensors
- Clocking
- Sensor Data Registers
- FIFO
- Interrupts
- Digital-Output Temperature Sensor
- Gyroscope & Accelerometer Self-test
- Bias and LDO
- Charge Pump

7.7 Three-Axis MEMS Gyroscope with 16-bit ADCs and Signal Conditioning

The MPU-60X0 consists of three independent vibratory MEMS rate gyroscopes, which detect rotation about the X-, Y-, and Z- Axes. When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis. The full-scale range of the gyro sensors may be digitally programmed to ±250, ±500, ±1000, or ±2000 degrees per second (dps). The ADC sample rate is programmable from 8,000 samples per second, down to 3.9 samples per second, and user-selectable low-pass filters enable a wide range of cut-off frequencies.

7.8 Three-Axis MEMS Accelerometer with 16-bit ADCs and Signal Conditioning

The MPU-60X0's 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The MPU-60X0's architecture reduces the accelerometers' susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0*g* on the X- and Y-axes and +1*g* on the Z-axis. The accelerometers' scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to ±2*g*, ±4*g*, ±8*g*, or ±16*g*.

7.9 Digital Motion Processor

The embedded Digital Motion Processor (DMP) is located within the MPU-60X0 and offloads computation of motion processing algorithms from the host processor. The DMP acquires data from accelerometers, gyroscopes, and additional 3rd party sensors such as magnetometers, and processes the data. The resulting data can be read from the DMP's registers, or can be buffered in a FIFO. The DMP has access to one of the MPU's external pins, which can be used for generating interrupts.

The purpose of the DMP is to offload both timing requirements and processing power from the host processor. Typically, motion processing algorithms should be run at a high rate, often around 200Hz, in order to provide accurate results with low latency. This is required even if the application updates at a much lower rate; for example, a low power user interface may update as slowly as 5Hz, but the motion processing should still run at 200Hz. The DMP can be used as a tool in order to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in the application.

7.10 Primary I^2C and SPI Serial Communications Interfaces

The MPU-60X0 communicates to a system processor using either a SPI (MPU-6000 only) or an I^2C serial interface. The MPU-60X0 always acts as a slave when communicating to the system processor. The LSB of the of the I^2C slave address is set by pin 9 (AD0).

The logic levels for communications between the MPU-60X0 and its master are as follows:

- MPU-6000: The logic level for communications with the master is set by the voltage on VDD
- MPU-6050: The logic level for communications with the master is set by the voltage on VLOGIC

For further information regarding the logic levels of the MPU-6050, please refer to Section 10.

9 Digital Interface

9.1 I^2C and SPI (MPU-6000 only) Serial Interfaces

The internal registers and memory of the MPU-6000/MPU-6050 can be accessed using either I^2C at 400 kHz or SPI at 1MHz (MPU-6000 only). SPI operates in four-wire mode.

Serial Interface

Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
8	Y		/CS	SPI chip select (0=SPI enable)
8		Y	VLOGIC	Digital I/O supply voltage. VLOGIC must be ≤ VDD at all times.
9	Y		AD0 / SDO	I^2C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I^2C Slave Address LSB
23	Y		SCL / SCLK	I^2C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I^2C serial clock
24	Y		SDA / SDI	I^2C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I^2C serial data

Note:

To prevent switching into I^2C mode when using SPI (MPU-6000), the I^2C interface should be disabled by setting the *I2C_IF_DIS* configuration bit. Setting this bit should be performed immediately after waiting for the time specified by the "Start-Up Time for Register Read/Write" in Section 6.3.

For further information regarding the *I2C_IF_DIS* bit, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document.

9.2 I^2C Interface

I^2C is a two-wire interface comprised of the signals serial data (SDA) and serial clock (SCL). In general, the lines are open-drain and bi-directional. In a generalized I^2C interface implementation, attached devices can be a master or a slave. The master device puts the slave address on the bus, and the slave device with the matching address acknowledges the master.

The MPU-60X0 always operates as a slave device when communicating to the system processor, which thus acts as the master. SDA and SCL lines typically need pull-up resistors to VDD. The maximum bus speed is 400 kHz.

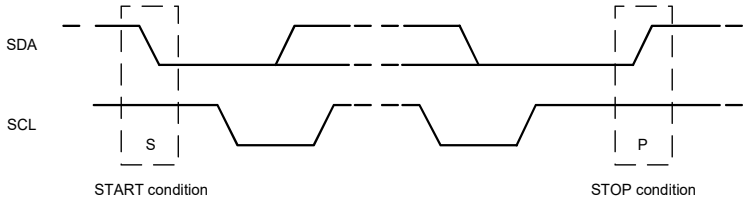
The slave address of the MPU-60X0 is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin AD0. This allows two MPU-60X0s to be connected to the same I^2C bus. When used in this configuration, the address of the one of the devices should be b1101000 (pin AD0 is logic low) and the address of the other should be b1101001 (pin AD0 is logic high).

9.3 I^2C Communications Protocol

START (S) and STOP (P) Conditions

Communication on the I^2C bus starts when the master puts the START condition (S) on the bus, which is defined as a HIGH-to-LOW transition of the SDA line while SCL line is HIGH (see figure below). The bus is considered to be busy until the master puts a STOP condition (P) on the bus, which is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH (see figure below).

Additionally, the bus remains busy if a repeated START (Sr) is generated instead of a STOP condition.

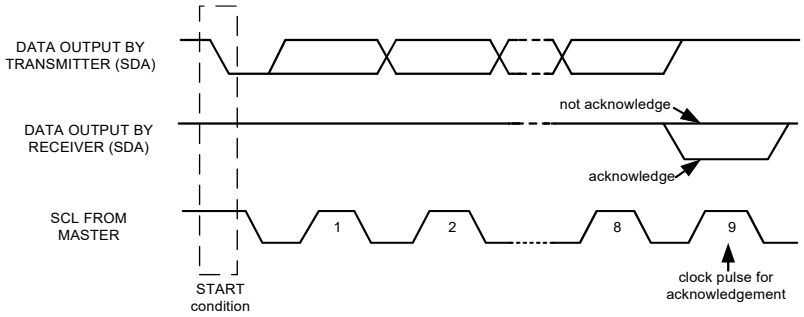


START and STOP Conditions

Data Format / Acknowledge

I^2C data bytes are defined to be 8-bits long. There is no restriction to the number of bytes transmitted per data transfer. Each byte transferred must be followed by an acknowledge (ACK) signal. The clock for the acknowledge signal is generated by the master, while the receiver generates the actual acknowledge signal by pulling down SDA and holding it low during the HIGH portion of the acknowledge clock pulse.

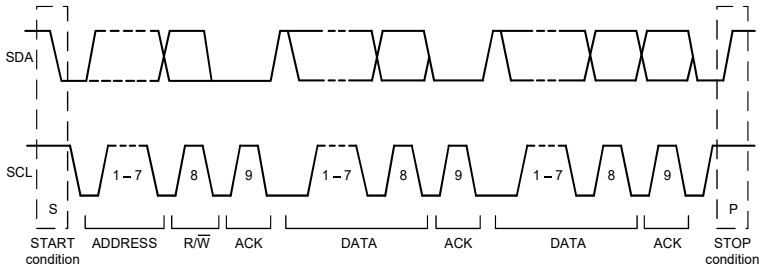
If a slave is busy and cannot transmit or receive another byte of data until some other task has been performed, it can hold SCL LOW, thus forcing the master into a wait state. Normal data transfer resumes when the slave is ready, and releases the clock line (refer to the following figure).



Acknowledge on the I^2C Bus

Communications

After beginning communications with the START condition (S), the master sends a 7-bit slave address followed by an 8th bit, the read/write bit. The read/write bit indicates whether the master is receiving data from or is writing to the slave device. Then, the master releases the SDA line and waits for the acknowledge signal (ACK) from the slave device. Each byte transferred must be followed by an acknowledge bit. To acknowledge, the slave device pulls the SDA line LOW and keeps it LOW for the high period of the SCL line. Data transmission is always terminated by the master with a STOP condition (P), thus freeing the communications line. However, the master can generate a repeated START condition (Sr), and address another slave without first generating a STOP condition (P). A LOW to HIGH transition on the SDA line while SCL is HIGH defines the stop condition. All SDA changes should take place when SCL is low, with the exception of start and stop conditions.



Complete I²C Data Transfer

To write the internal MPU-60X0 registers, the master transmits the start condition (S), followed by the I²C address and the write bit (0). At the 9th clock cycle (when the clock is high), the MPU-60X0 acknowledges the transfer. Then the master puts the register address (RA) on the bus. After the MPU-60X0 acknowledges the reception of the register address, the master puts the register data onto the bus. This is followed by the ACK signal, and data transfer may be concluded by the stop condition (P). To write multiple bytes after the last ACK signal, the master can continue outputting data rather than transmitting a stop signal. In this case, the MPU-60X0 automatically increments the register address and loads the data to the appropriate register. The following figures show single and two-byte write sequences.

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

To read the internal MPU-60X0 registers, the master sends a start condition, followed by the I^2C address and a write bit, and then the register address that is going to be read. Upon receiving the ACK signal from the MPU-60X0, the master transmits a start signal followed by the slave address and read bit. As a result, the MPU-60X0 sends an ACK signal and the data. The communication ends with a not acknowledge (NACK) signal and a stop bit from master. The NACK condition is defined such that the SDA line remains high at the 9^{th} clock cycle. The following figures show single and two-byte read sequences.

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

9.4 I^2C Terms

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I^2C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9^{th} clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9^{th} clock cycle
RA	MPU-60X0 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high