ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN GUANTE ELECTRÓNICO TRADUCTOR DE SEÑAS A TRAVÉS DE COMUNICACIÓN WIFI

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

Kenneth Christian Crespo Crespo
kenneth.crespo@epn.edu.ec

Edison David Males Maldonado
edison.males@epn.edu.ec

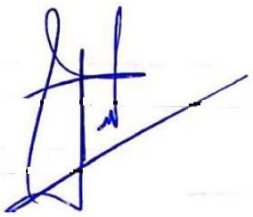
DIRECTOR: ING. ALAN DANIEL CUENCA SANCHEZ, MSC.
alan.cuenca@epn.edu.ec

CODIRECTOR: ING. MÓNICA DE LOURDES VINUEZA RHOR, MSC.
monica.vinueza@epn.edu.ec

Quito, Diciembre 2021

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por los Sres. Crespo Crespo Kenneth Christian y Males Maldonado Edison David como requerimiento parcial a la obtención del título de TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES, bajo nuestra supervisión:



Ing. Alan Cuenca MSc.

DIRECTOR DEL PROYECTO

Ing. Mónica Vinueza MSc.

CODIRECTORA DEL PROYECTO

DECLARACIÓN

Nosotros, Crespo Crespo Kenneth Christian con CI: 1724080740 y Males Maldonado Edison David con CI: declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que hemos consultado las referencias bibliográficas que se incluyen en este documento.

Sin prejuicio de los derechos reconocidos en el primer párrafo del artículo 144 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación – COESC-, somos titulares de la obra en mención y otorgamos una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional.

Entregamos toda la información técnica pertinente, en caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Kenneth Crespo

CI: 1724080740

Teléfono: 0978921083

Correo: kenneth.crespo@epn.edu.ec

Edison Males

CI: 1004845465

Teléfono: 0969308138

Correo: edison.males@epn.edu.ec

DEDICATORIA

Este proyecto es dedicado para todas las personas que carecen de sus facultades para hablar y escuchar, esperando que con este grano de arena se pueda contribuir en su desarrollo personal y profesional.

Kenneth Christian Crespo Crespo

AGRADECIMIENTO

Completamente agradecido con toda mi familia y amigos más cercanos por siempre ser un enorme apoyo para mí. Gracias a su sabiduría y a sus consejos pude culminar mi carrera de manera satisfactoria.

Expreso mi gratitud hacia mi compañero tesista Edison Males quien demostró dedicación y compromiso en la realización del proyecto.

Finalmente agradezco a nuestro director de tesis el Ingeniero Alan Cuenca por todo el apoyo ofrecido durante el desarrollo de nuestra tesis.

Kenneth Christian Crespo Crespo

DEDICATORIA

Este proyecto está dedicado a todas aquellas personas que día a día batallan en sus vidas diarias con esta discapacidad. Personas que lucha por un mejor trato.

Edison David Males Maldonado

AGRADECIMIENTO

Totales agradecimientos a toda mi familia, amigos y pareja, que, a pesar de las circunstancias o adversidades, me apoyaron por varios años y de varias maneras, hasta la culminación de mis estudios superiores.

Agradecimientos a mi compañero de tesis Kenneth Crespo, quien ha demostrado compromiso y constancia, no solo en la realización de este trabajo, si no también, en toda la carrera universitaria.

Para finalizar, mis agradecimientos con nuestro director de tesis el ingeniero Alan Cuenca, quien nos ha apoyado en gran manera tanto en el presente trabajo, como en nuestra formación profesional.

Edison David Males Maldonado

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo de titulación “Implementación de un guante electrónico traductor de señas a través de comunicación wifi” consiste en un dispositivo cuya función principal es traducir el lenguaje de señas a texto mediante los movimientos realizados con la mano derecha. La traducción es presentada en una plataforma *Web* a través de comunicación wifi. En base a los conocimientos adquiridos durante la carrera de Tecnología en Electrónica y Telecomunicaciones, y realizando investigación de ciertos temas relacionados al lenguaje de señas se ha cumplido con el desarrollo del sistema.

A continuación, se explican las cinco secciones del documento que detallan todos los procedimientos realizados durante el diseño e implementación del sistema traductor.

La primera sección contiene la introducción donde se describe el problema que se pretende resolver con la implementación del presente trabajo, además se puntualizan los objetivos planteados, así como los fundamentos necesarios para la realización del proyecto de titulación.

La segunda sección muestra la metodología utilizada para cumplir con los objetivos planteados y conseguir la construcción del guante electrónico.

En la tercera sección se detalla el procedimiento realizado para el desarrollo del presente trabajo de titulación, desde el diseño del dispositivo hasta las pruebas finales de funcionamiento.

En la cuarta sección se presentan las conclusiones y recomendaciones obtenidas durante el desarrollo del proyecto.

Finalmente, la quinta sección contiene toda la bibliografía utilizada a lo largo de todo el proceso de implementación del guante traductor.

PALABRAS CLAVE: Guante Electrónico, Wifi, Servidor *Web*, Sensores Flex, Lenguaje Dactilológico.

ABSTRACT

The present thesis "Implementation of an electronic glove translator of signs through wifi communication" consists of a device whose main function is to translate sign language into text through movements made with the right hand. The translation is presented on a Web platform through wifi communication. Based on the knowledge acquired during the career of Technology in Electronics and Telecommunications and conducting research on certain issues related to sign language, the development of the system has been completed.

The five sections of the document that detail all the procedures performed during the design and implementation of the translator system are explained below.

The first section contains the introduction where the problem that is intended to be solved with the implementation of this work is described, in addition the objectives set are specified, as well as the necessary foundations for the realization of the degree project.

The second section shows the methodology used to meet the objectives set and achieve the construction of the electronic glove.

The third section details the procedure carried out for the development of this degree work, from the design of the device to the final performance tests.

The fourth section presents the conclusions and recommendations obtained during the development of the project.

Finally, the fifth section contains all the bibliography used throughout the process of implementing the translator glove.

KEY WORDS: Electronic Glove, Wifi, Web Server, Flex Sensors, Fingerprint Language.

1 INTRODUCCIÓN

En nuestro entorno social, la gran mayoría de personas que tienen la capacidad de hablar no comprenden en absoluto el lenguaje de signos, por lo que para todos los individuos privados de escuchar y hablar les resulta complicado comunicarse con el resto de la sociedad.

Según datos del Consejo Nacional Para la Igualdad de Discapacidades (CONADIS), se estima que aproximadamente doscientos mil habitantes sufren de deficiencia auditiva y de lenguaje.

Para los individuos que padecen de esta discapacidad resulta difícil encontrar un trabajo digno, ya que cualquier actividad laboral requiere de una buena comunicación entre todas las áreas para cumplir con el objetivo de la empresa. Esto provoca que la gran mayoría sufra de escasos recursos, lo que limita mucho su desarrollo personal.

Por lo tanto, se plantea la construcción de un guante electrónico que posee cinco sensores flexibles que varían el valor de su resistencia al ser deformados. Cada uno se ubica en cada dedo de la mano para leer sus respectivas posiciones al momento de realizar alguna letra del lenguaje dactilológico.

Los datos tomados por los sensores pasan a ser procesados por el módulo ARDUINO UNO, donde se realiza la conversión del lenguaje de señas al de texto. La traducción se imprime en una pantalla LCD y, a su vez, es enviada al módulo wifi ESP8266 que es el encargado de subir la traducción al servidor *firebase* de *Google* en tiempo real. Cabe recalcar que cualquier persona con acceso a internet será capaz de conectarse a esta página *Web* culminando así el intercambio de información entre el usuario y el espectador.

De esta manera, se realizará la conversión directa del alfabeto de señas al alfabeto de letras del lenguaje castellano, dando así la oportunidad de comprender de mejor manera las ideas o sentimientos que las personas sordomudas están intentando transmitir.

En el Anexo 1 se encuentra el certificado de funcionamiento que establece la correcta implementación del guante electrónico traductor.

1.1 Objetivos

Objetivo general

Implementar un guante electrónico traductor de señas a través de comunicación wifi.

Objetivos específicos

Determinar los requerimientos necesarios para la construcción del guante electrónico.

Diseñar el prototipo para la traducción del lenguaje dactilológico.

Desarrollar el programa de medición y registro de información.

Establecer el enlace de red inalámbrica entre el módulo wifi y la plataforma *Web*.

Implementar el guante electrónico.

Realizar pruebas de funcionamiento.

Realizar un manual de usuario y mantenimiento.

1.2 Fundamentos

Lenguaje dactilológico

La dactilología es la representación manual de las letras del alfabeto. Gracias a ella es posible deletrear las palabras del lenguaje de señas. Es la forma más básica de esta lengua. Se ejecuta por la altura de la barbilla junto con expresiones faciales para un mejor entendimiento.

Para la comunicación mediante signos es frecuente utilizar la combinación de la primera o primeras letras de la palabra seguidas por un movimiento. En la Figura 1.1 se muestra un ejemplo en el cual se está realizando la letra “A” [1].

Figura 1.1 Ejemplo dactilología [1]

Sensores Flex

Son elementos piezorresistivos que dependen del grado en el que estén deformadas (dobladas). Por lo general una tira que mida 5 (cm) de largo variará entre 10 (kΩ) a 50 (kΩ) En la Figura 1.2 se observa la forma y el diseño del sensor [2].

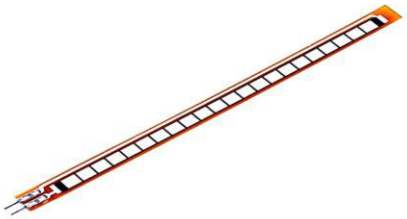


Figura 1.2 Sensor flex [2]

Este cambio de resistencia eléctrica se produce debido a las propiedades de los materiales de los sensores. Cuentan con elementos de carbono dentro de un sustrato delgado y flexible.

Al doblarse el sustrato del sensor provoca una salida de resistencia relacionada con el radio de curvatura. Mientras se concentre más carbono se obtendrá menos resistencia. [3].

Módulo wifi ESP8266

Dispositivo electrónico capaz de dar acceso a cualquier microcontrolador a su red wifi gracias a que tiene una pila integrada con protocolo TCP/IP. Este módulo viene preprogramado con comandos AT, los cuales se configuran en el propio software de Arduino por medio del puerto serial.

Este módulo posee una potente capacidad de almacenamiento y procesamiento, lo que permite conectar varios dispositivos como sensores, soporta aplicaciones VoIP, contiene auto calibrado RF, etc. En la Figura 1.3 se aprecia el módulo a utilizar [4].

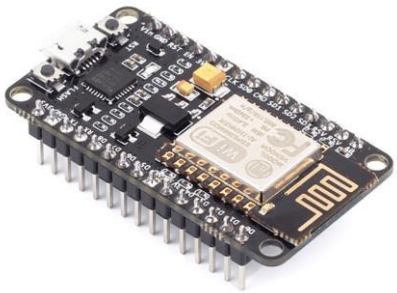


Figura 1.3 NodeMCU ESP8266 wifi [4]

Servidor *firebase*

Es una plataforma en la nube que tiene como función principal crear y desarrollar aplicaciones, tanto *Web* como móviles, procurando que el trabajo sea más rápido, pero sin perder la calidad requerida.

Existen 3 grupos donde se dividen las funcionalidades brindadas por *firebase* las cuales son: desarrollo, crecimiento y monetización [5].

Una de las herramientas a subrayar en este servidor *Web* son las bases de datos en tiempo real. En la Figura 1.4 se observa un ejemplo del entorno de *firebase*.

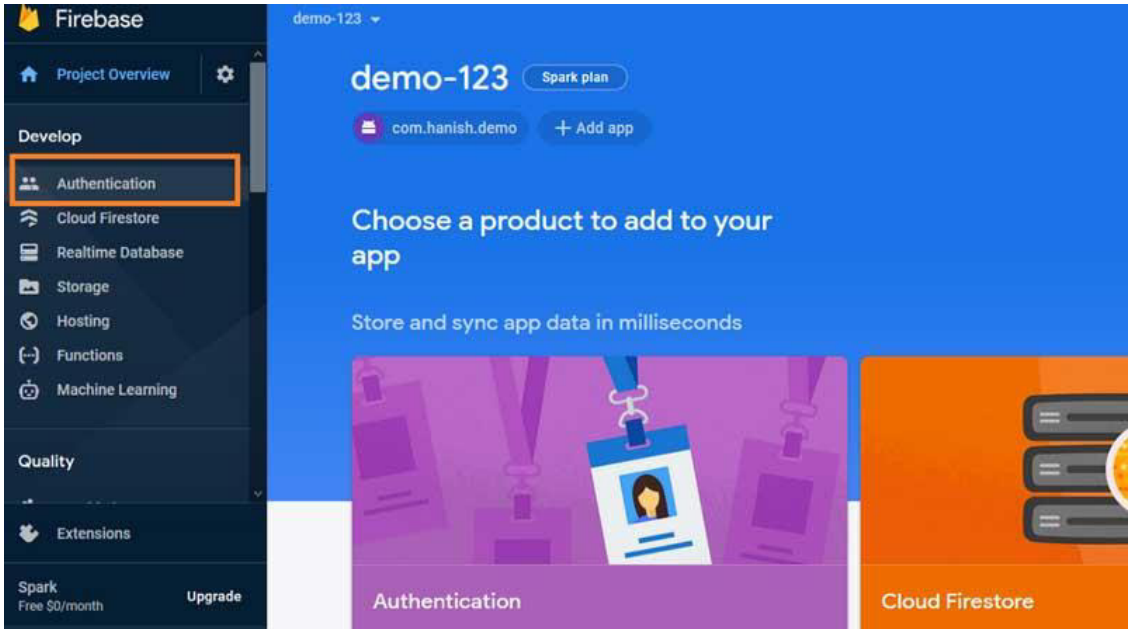


Figura 1.4 Entorno *firebase*

MIT *App Inventor*

MIT *App Inventor* es una plataforma creada por *Google Labs* utilizada en la creación de aplicaciones para software del sistema operativo Android. De una manera visual y partiendo de herramientas básicas, el usuario puede crear sus proyectos o aplicaciones simplemente enlazando una serie de bloques [6].

Una de las ventajas importantes de esta plataforma es que se pueden descargar los proyectos realizados de manera sencilla y gratuita desde la *Web*. A pesar de que esta plataforma tiene varias limitaciones debido a la simplicidad de su estructura, es una muy buena opción para proyectos iniciales en varios ámbitos, tanto en la educación como en la industria, pues permite cubrir gran parte de las necesidades en un dispositivo móvil [6]. En la Figura 1.5 se observan los diferentes pasos a seguir para crear una aplicación.

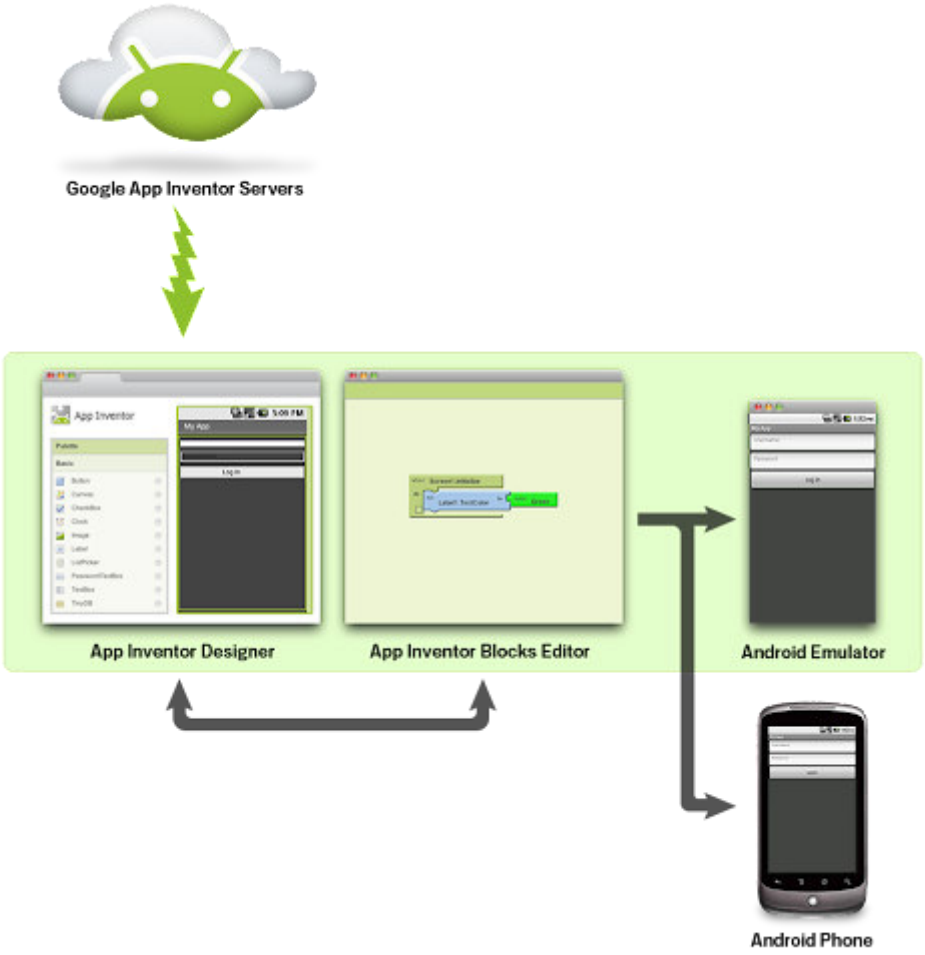


Figura 1.5 *App Inventor*

2 METODOLOGÍA

El presente proyecto consiste en la elaboración de un guante electrónico programado en la plataforma Arduino UNO, con una conexión inalámbrica entre un módulo wifi ESP8266 y el servidor *Web firebase* para la transmisión de la traducción del lenguaje de señas al alfabeto castellano.

2.1 Descripción de la metodología usada

Se realizó una investigación detallada de cada posición que la mano debe adoptar al momento de formar tanto, cada letra del alfabeto castellano, como los diez dígitos de los números naturales. Una vez establecido este método, se definieron los elementos electrónicos y las herramientas necesarias para desarrollar el proyecto. Además, se analizaron diferentes materiales para la elaboración del guante y su respectiva dimensión.

Se elaboró una simulación virtual del sistema electrónico del prototipo en un esquema que presenta la integración de cada elemento, los componentes utilizados, las conexiones inalámbricas y su interacción con la placa Arduino UNO. El circuito consiste en un sensor flex por cada dedo de la mano derecha, una shield wifi ESP8266, una pantalla LCD y una placa Arduino. Después de que los resultados de las pruebas en este software fueron satisfactorios, se procedió a diseñar el guante, analizando diferentes materiales para su elaboración, con el fin de que sea de fácil manipulación y cómodo para el usuario.

En base a programación en Arduino IDE, se desarrolló un código capaz de procesar las mediciones receptadas por los sensores flex al formar las posiciones de cada letra del abecedario, las cuales son almacenadas con los caracteres correspondientes. Esta es la manera en la que la traducción se ejecuta.

Se programó el módulo wifi ESP8266 que está conectado a la placa Arduino UNO, este se encarga de enviar los caracteres traducidos de manera inalámbrica al servidor *Web firebase*. Para ello se configuró la recepción y la visualización de la traducción directamente en esta plataforma con el fin de establecer la comunicación entre ambas partes.

Se procedió a realizar la placa electrónica diseñada previamente en el software Proteus 8.12. Luego, en el guante adquirido, se colocaron los sensores flex en cada dedo de la mano, los cuales se conectan directamente al Arduino. El resto de los componentes

como son: las placas, la pantalla LCD, la fuente de energía, los botones de control y el cableado, se colocaron en un compartimento ensamblado fuera del guante.

Se verificó el correcto ensamblaje y conexión del prototipo, con pruebas de funcionamiento con ayuda de una persona con pérdida auditiva que tenga conocimiento del lenguaje dactilológico, de esta manera se revisó si el prototipo cumple con su función de traductor de la manera más adecuada. Además, se revisó que los datos desplegados en el servidor y en la pantalla LCD correspondan a la traducción.

Finalmente, se realizó un video dedicado a personas con pérdida auditiva en el cual se muestra el uso del dispositivo, donde se especifican y se describen las partes y componentes principales del prototipo para su correcta manipulación.

.

3 RESULTADOS Y DISCUSIÓN

El funcionamiento del guante electrónico consiste en obtener la posición de cada letra del lenguaje dactilológico mediante sensores flex, los cuales están colocados en cada dedo de la mano derecha. La placa Arduino UNO es la encargada de procesar las mediciones tomadas por los sensores y traducir al lenguaje escrito.

La traducción es enviada al módulo wifi ESP8266, el cual está conectado directamente al servidor *firebase* en internet, en donde aparece en tiempo real el mensaje transmitido por el usuario. Cabe recalcar que cualquier persona que disponga de un dispositivo que tenga la capacidad de conectarse a internet, podrá ingresar a la página mencionada y observará el mensaje.

Adicionalmente, el mensaje aparecerá en una pantalla LCD, donde el usuario podrá verificar la letra que está realizando antes de imprimirla, esto con el fin de evitar que se envíen letras no deseadas. En esta pantalla también se imprimirá el mensaje, por lo que, en caso de no poseer conexión a internet, este medio también servirá como receptor del mensaje.

El mecanismo tiene tres botones de interacción, uno para imprimir la letra colocada, otro para colocar un espacio entre las palabras y el último para borrar todo lo que se colocó en la pantalla.

3.1 Requerimientos

Antes de elegir los componentes utilizados fue primordial realizar un estudio del lenguaje dactilológico para, posteriormente, analizar a detalle los elementos más adecuados para la construcción del proyecto.

Importancia del Lenguaje Dactilológico

Como el proyecto fue establecido para ser usado con una sola mano, utilizar lenguaje dactilológico cumple correctamente con este requerimiento, debido a que la dactilología fue creada para deletrear la lengua de signos precisamente con una mano. Se puede decir que es la forma más básica de comunicación para una persona privada de sus facultades del habla [7].

Cada letra del abecedario tiene su propia posición. En la Figura 3.1 se presenta un ejemplo de las tres primeras letras del abecedario con su respectiva posición.

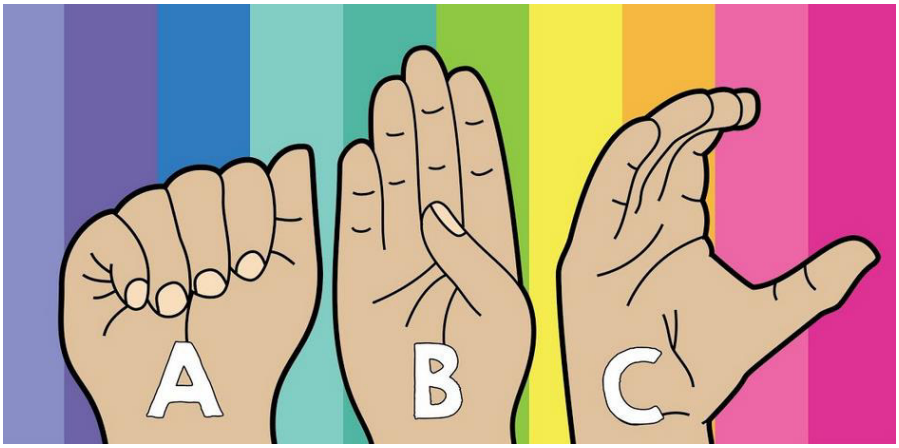


Figura 3.1 Ejemplos de lenguaje dactilológico [8]

Funciones principales del sensor flexible

Al ser un dispositivo maleable y que requiere tener una manipulación sencilla, es necesario contar con un diseño ergonómico, liviano y con características técnicas estables que impidan problemas de descalibración mientras el dispositivo esté en uso.

Según dichas consideraciones, la elección de utilizar un sensor flexible es precisa, debido a que sus características físicas y sistemáticas se adaptan totalmente a los requerimientos del proyecto.

Los parámetros más importantes para la elección del sensor flexible se indican a continuación.

- **Flexibilidad**, esta cualidad fue la más importante debido a que las posiciones que los dedos de la mano van a formar al momento de realizar una letra requieren que los sensores sean completamente flexibles. Esto hace que el guante sea completamente cómodo de usar [9].
- **Rango de medida**, estos sensores cambian su resistividad dependiendo del grado en que hayan sido doblados. Es importante recalcar que es un rango finito y solo puede ser flexionado hacia el costado contrario al que se encuentran las mallas de los sensores. Estos parámetros son completamente necesarios, porque en caso de que, al tener una posición completamente recta, los sensores podrían doblarse un poco hacia el lado contrario, pero al tener un límite en los 90°, no tomará valores más altos que los debidos. Así se evitan confusiones en la programación al detectar los dedos estirados o cerrados [9].

- **Estabilidad**, los valores arrojados por el sensor son muy estables. Existen variaciones mínimas al mantener una misma posición, estas disminuyen con un divisor de voltaje. Gracias a esto es posible generar una letra sin errores [9].
- **Alimentación**, como se ha escogido la plataforma Arduino debido a sus prestaciones, es de vital importancia una alimentación de 5 (V_{DC}), para evitar la utilización de fuentes externas [9].

En la Tabla 3.1 se muestran las principales características del sensor.

Tabla 3.1 Características eléctricas del sensor flexible [9]

Características del Sensor Flex	
Dimensiones	0.71 (cm) por 2.54 (cm)
Voltaje	5 (V_{DC}) – 12 (V_{DC})
Rango de resistencia	1.5 (kΩ) – 40 (kΩ)
Temperatura	-35 (ºC) – 80 (ºC)
Histéresis	7%
Tiempo de vida	Aproximadamente 1 millón de usos

Placa Arduino UNO

Teniendo en cuenta la facilidad de programación, la accesibilidad, el bajo coste de la placa y los requerimientos del proyecto; se optó por usar Arduino UNO [10]. Esta placa cuenta con características importantes como compatibilidad con otros módulos, almacenamiento suficiente para la extensión del código, la cantidad de pines y su tamaño. En la Figura 3.2 se aprecian las principales partes del módulo.

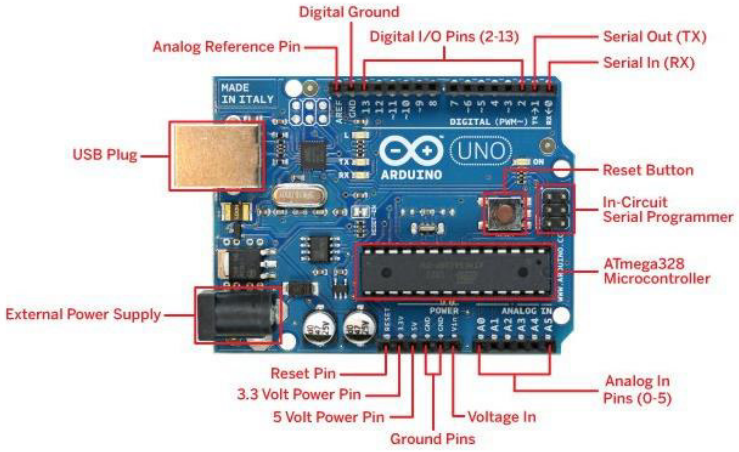


Figura 3.2 Arduino UNO [10]

Para observar de forma más detallada las características del Arduino UNO revisar el Anexo 2.

En la Tabla 3.2 se muestran las características generales con las que cuenta la placa.

Tabla 3.2 Características generales de Arduino Uno [11]

Características Arduino UNO	
Microcontrolador	ATMega328P
Voltaje de trabajo	5 (V_{DC})
Voltaje de entrada	7.5 (V_{DC}) a 12 (V_{DC})
Pines	14 digitales / 6 analógicos
Velocidad de reloj	16 (MHz)
Memoria Flash	32 (KB)
Memoria EEPROM	1 (KB)

Módulo wifi ESP8266

Para enviar los datos a través de internet se necesita de un dispositivo confiable capaz de recibir datos y transmitirlos a la red. Estas características las dispone el módulo ESP8266.

Entre las ventajas de usar este dispositivo se establece que es compatible con Arduino y está listo para ser usado en implementaciones de IoT (*Internet of Things*) [12]. Posee un regulador de voltaje integrado, un puerto USB de programación y puede ser programado mediante el IDE de Arduino. En la Figura 3.3 se observa la distribución de pines con sus respectivas funciones.

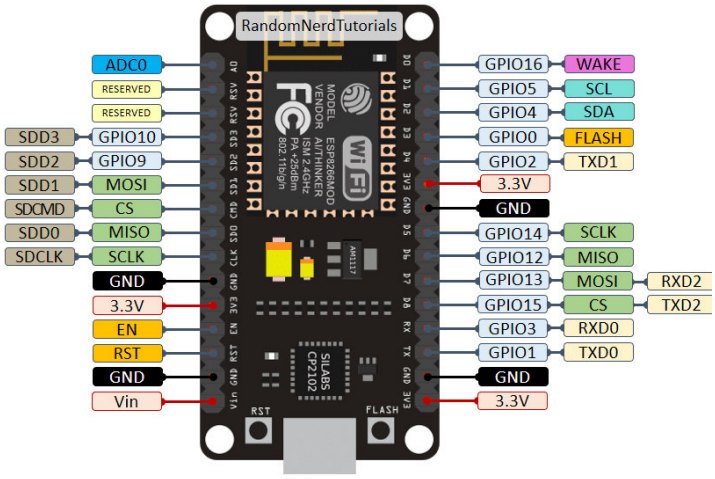


Figura 3.3 Distribución de pines ESP8266 [12]

En la Tabla 3.3 se muestran las principales características del módulo.

Tabla 3.3 Características NodeMCU ESP8266 [12]

Características ESP8266	
CPU	Tensilica Xtensa LX3 (32 bit)
Memoria Flash	4 (MB)
RAM	32 (kB)
Estándar	802.11 b/g/n
Protocolo	TCP/IP
Wi-Fi Direct	P2P soft-AP
Pines GPIO	17
Pin ADC	1
Voltaje de Entrada/Salida	3.3 (V_{DC})
Conversor USB Serial	CP2102
Dimensiones	49*26*12 (mm)
Pulsadores	Reset y Flash
Frecuencia de Reloj	80 (MHz)
Procesador	RISC

3.2 Diseño

Diseño del guante

Para dar inicio al proyecto fue vital comenzar por el diseño del prototipo del guante porque para registrar la posición de cada dedo en el módulo Arduino era necesario que los sensores flex tomen la forma precisa de cada letra.

Al inicio se estableció realizar un guante adaptable para cualquier tamaño de mano, pero después de verificar el funcionamiento de los sensores esto quedó descartado y se procedió a realizar un guante de una sola dimensión, ya que en 2 tipos de manos diferentes la disposición de los sensores no será la misma, dando como resultado cifras erróneas que no serán procesadas por la placa Arduino al no reconocer los valores guardados en la programación.

Se escogió un guante dieléctrico para evitar cortocircuitos que puedan dañar los elementos electrónicos del dispositivo. Para sostener los sensores se procedió a coser elásticos en cada dedo del guante, esto con el fin de que sea posible abrir o cerrar la

mano sin ningún tipo de inconvenientes. El guante utilizado para el diseño se observa en la Figura 3.4.

Figura 3.4 Diseño del Guante

Se debe tomar en cuenta que los sensores deben estar ubicados paralelamente con los dedos que les corresponden, por lo que, para evitar mediciones no deseadas se construyó el guante con características precisas para que permanezcan inmóviles en sus lugares, así no se presentarán problemas en los rangos de calibración de cada letra.

En cada dedo se cosieron telas elásticas para que los sensores no se muevan de su lugar, pero si se puedan estirar con los movimientos de la mano. Además, los sensores fueron fijados con silicona a la altura de los nudillos.

Diseño de la placa PCB

Se utilizó el programa Proteus 8.12 para realizar el diseño de la placa del circuito electrónico. Este programa sirve para la elaboración de circuitos y diseñar sus placas de una manera muy sencilla.

Se colocaron las respectivas resistencias, puentes, y los switches en la parte de la conexión del Arduino con el módulo wifi ESP8266. Se utilizaron resistencias de 47 (kΩ) debido a que el rango de trabajo de los sensores es más estable y se puede colocar de mejor manera las condiciones para la conversión de las letras [13].

En la Figura 3.5 se observan las conexiones correspondientes al circuito realizado.

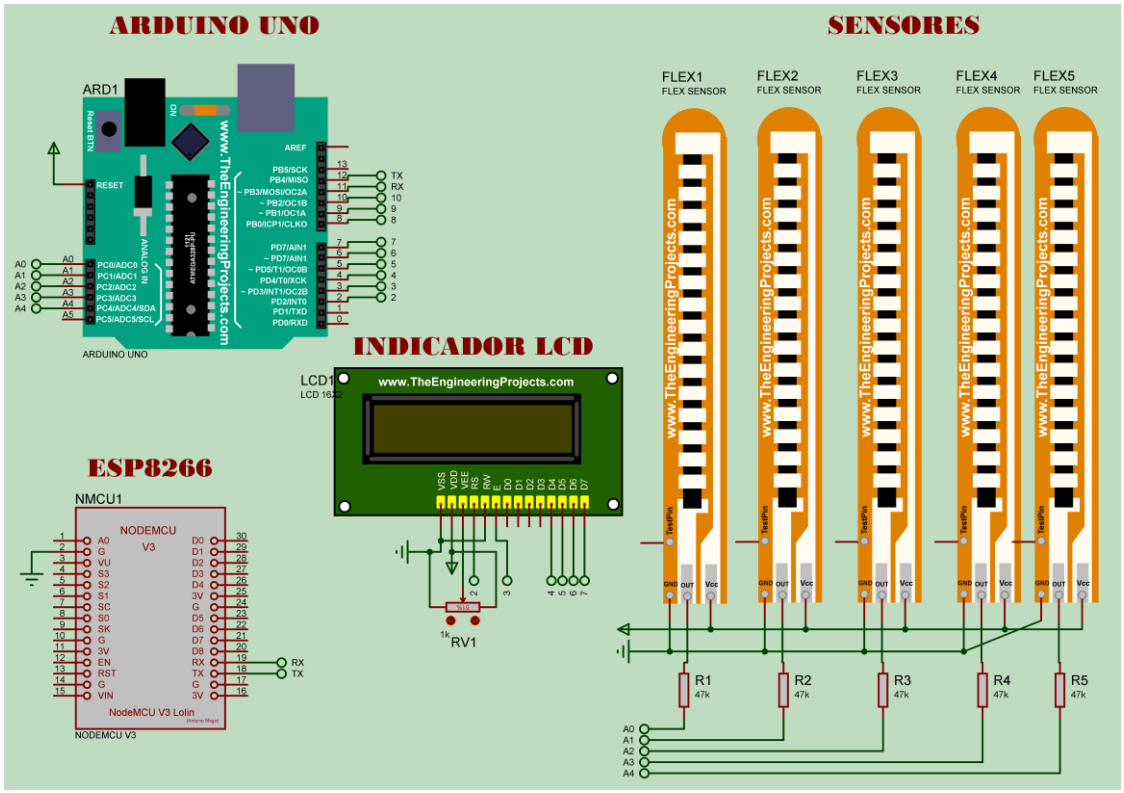


Figura 3.5 Circuito esquemático

Luego de finalizar las conexiones se debe exportar el circuito para que el programa Ares diseñe la placa PCB. Este resultado se observa en la Figura 3.6.

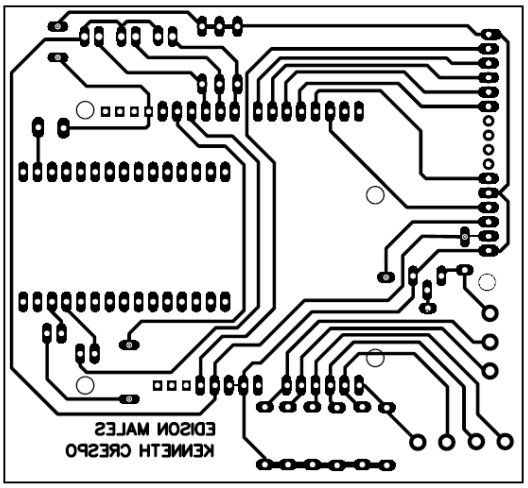


Figura 3.6 Diseño de pistas

Diseño del compartimento

Para mantener al circuito protegido se diseñó una caja de madera utilizando AutoCAD, acorde a las dimensiones de la placa y de los componentes utilizados.

En la Figura 3.7 se observa el diseño de la caja, con sus respectivas aberturas para interactuar con los botones, para las conexiones y para visualizar el mensaje en el LCD.

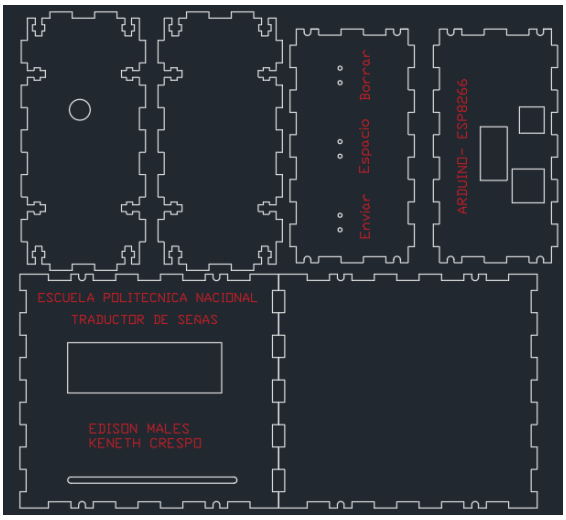


Figura 3.7 Diseño de la carcasa del circuito

3.3 Desarrollo del programa de medición y registro de información.

Programación en Arduino UNO

Una vez diseñado el prototipo se desarrolló el código de programación, cuya función principal es almacenar las mediciones generadas por los sensores al mantener las posiciones de las letras del lenguaje dactilológico. El proceso consistió en formar una letra con el guante puesto, medir las posiciones de los cinco dedos de la mano, registrar los valores generados y guardarlos en el programa. Este procedimiento fue repetido para cada letra y número del abecedario.

Para conectar los sensores flex al Arduino se necesita de una configuración sencilla, que consiste en conectar el VCC del sensor directamente a los 5 (V_{DC}) de la placa, y el GND conectado en serie a la entrada analógica junto con una resistencia, la cual va conectada de su otro extremo hacia tierra [13]. En la Figura 3.8 se puede apreciar la conexión del sensor.

Los valores para la resistencia de este circuito pueden variar entre 10 (kΩ) a 100 (kΩ), según el fabricante, es recomendable utilizar un valor intermedio, por lo que en este proyecto se utilizó el valor de 47 (kΩ).

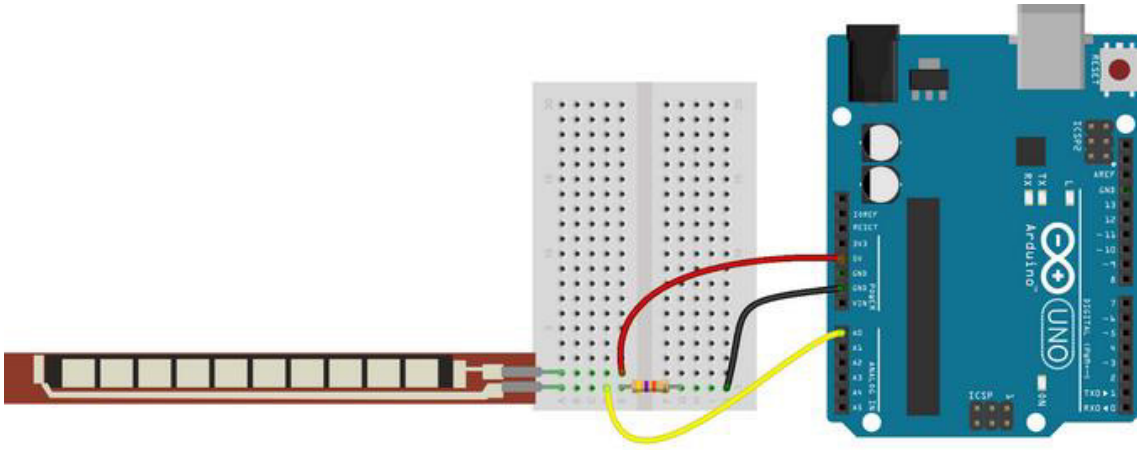


Figura 3.8 Conexión sensor flex [13]

Antes de empezar la programación fue indispensable convertir todos los valores generados por los sensores de unidades analógicas en unidades digitales (0 a 1023) con el conversor ADC del Arduino UNO, para luego registrar estos valores según la posición de cada dedo al realizar una letra.

En la Tabla 3.4 se muestran los valores en unidades digitales generadas por los sensores al momento de realizar la posición de cada letra.

Tabla 3.4 Registro de valores para cada símbolo.

Letra/Símbolo	Pulgar	Índice	Medio	Anular	Meñique
A	255	450	650	685	650
B	165	610	830	815	805
C	205	605	745	755	770
D	210	540	690	700	805
E	185	520	645	685	685
F	215	625	830	815	710
G	220	470	655	665	790
H	175	500	835	685	805
I	190	595	660	700	645
J	215	595	675	715	680
K	225	505	845	700	805
L	210	515	635	680	800
M	160	520	710	725	705
N	175	510	725	685	715
O	215	565	720	730	715

Letra/Símbolo	Pulgar	Índice	Medio	Anular	Meñique
P	230	455	740	700	800
Q	225	485	660	685	790
R	180	535	815	675	800
S	180	500	665	685	625
T	175	520	650	685	725
U	180	540	835	690	805
V	190	565	835	700	805
W	175	550	830	815	805
X	180	510	660	685	745
Y	200	595	670	700	670
Z	185	525	665	715	800
1	150	480	850	695	805
2	155	475	850	715	810
3	225	550	860	825	810
4	195	640	850	840	815
5	225	645	825	825	815
6	225	475	660	675	660
7	240	500	715	710	820
8	235	465	860	720	820
9	170	515	705	705	700

Cabe mencionar que se generó un rango alrededor del ± 10% de los valores registrados. Esto debido a que, al generar una posición, exactamente, más de una vez resulta poco probable de ejecutar la posición en el mismo rango, además no todas las manos son iguales, y entre usuarios diferentes, podría pasar que a algunos les funcione bien y a otros no se les genere ninguna letra.

También se debe tomar en cuenta que el número “0” se lo realiza con 2 manos. Al tener esta limitante en el proyecto se optó por realizarlo con la misma posición de la letra “O”.

A continuación, se detallan los pasos realizados en las líneas del código de programación:

- Se empezó incluyendo 2 librerías, la primera “*LiquidCrystal*”, que sirve iniciar el LCD; la segunda “*SoftwareSerial*” que convierte puertos normales a seriales. Seguido se escogen los pines donde van a ser colocadas las conexiones del

LCD y los pines que necesitan comunicación serial, tal cual se evidencia en la Figura 3.9.

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
SoftwareSerial mySerial(12, 11); //RX,TX
```

Figura 3.9 Librerías incluidas

- Después se colocan las variables de cada elemento a conectar. En la Figura 3.10 se observan los nombres para los pines 9, 10 y 11; en donde se conectan los pulsadores que generan las acciones de enviar la letra, borrarla y dar un espacio entre símbolos respectivamente. También se tienen variables adicionales necesarias para el programa.

```
short int pulsadorEnviar=8;
short int pulsadorBorrar=10;
short int pulsadorEspacio=9;
short int estado1,estado2,estado3;
short int contaPulsos1=0,contaPulsos2=0,contaPulsos3=0;
short int i=-1;
char letra = ' ';

const int FLEX_PIN0 = A0;
const int FLEX_PIN1 = A1;
const int FLEX_PIN2 = A2;
const int FLEX_PIN3 = A3;
const int FLEX_PIN4 = A4;
int P,I,M,A,Q;

int conta=0,retorno;
boolean LEDState = false;
long buttonTimer = 0;
long longPressTime = 250;
boolean buttonActive = false;
boolean longPressActive = false;
```

Figura 3.10 Variables del código de programación

- El siguiente paso fue inicializar los parámetros principales de la programación tales como el LCD, los pulsadores y las variables de los sensores flex. En la Figura 3.11 se observa este paso.

```
27 void setup() {
28   Serial.begin(9600);
29   lcd.begin(16, 2);
30   lcd.setCursor(2, 0); lcd.print("TRADUCTOR DE ");
31   lcd.setCursor(4, 1); lcd.print("SEÑAS");
32   delay(2000);
33   lcd.clear();
34   lcd.setCursor(0, 1); lcd.print("Letra: ");
35   pinMode(pulsadorEnviar, INPUT);
36   pinMode(pulsadorBorrar, INPUT);
37   pinMode(pulsadorEspacio, INPUT);
38   pinMode(FLEX_PIN0, INPUT);
39   pinMode(FLEX_PIN1, INPUT);
40   pinMode(FLEX_PIN2, INPUT);
41   pinMode(FLEX_PIN3, INPUT);
42   pinMode(FLEX_PIN4, INPUT);
43 }
```

Figura 3.11 Parámetros principales

- En la Figura 3.12 se observa la creación de nuevas variables para poder asignarlas a las entradas analógicas del Arduino que, en este caso, leen los valores generados por los sensores.

```
45 void loop(){
46     int P = analogRead(FLEX_PIN0);
47     int I = analogRead(FLEX_PIN1);
48     int M = analogRead(FLEX_PIN2);
49     int A = analogRead(FLEX_PIN3);
50     int Q = analogRead(FLEX_PIN4);
```

Figura 3.12 Lectura de entradas analógicas

- En la Figura 3.13 se observa la programación para registrar los valores medidos por los sensores. Se tiene como ejemplo la letra “U” que, para poder generarla como mensaje, es estrictamente necesario que cumpla con los cinco requerimientos colocados dentro el comando *if,* caso contrario no será posible obtenerla. El mismo proceso fue realizado para almacenar el resto de las letras del abecedario dentro del código. Tomar en cuenta en el ejemplo que en todos los parámetros se estableció un rango del ±10% para los valores generados.

```
123 if((P>180 && P<210) && (I>540 && I<570) && (M>835 && M<865) && (A>690 && A<720) && (Q>805 && Q<835))
124 {letra='U'; lcd.setCursor(6, 1); lcd.print(letra);}
```

Figura 3.13 Programación para registro de datos

- Cabe destacar que para registrar los datos fue necesario utilizar la Tabla 3.5, mencionada anteriormente, para colocar los valores para cada letra en el código.

Dando continuidad al ejemplo del paso anterior, en la Figura 3.14 se observa la posición pertinente a la letra “U” en lenguaje dactilológico.

Figura 3.14 Ejemplo letra “U”

- Una vez colocadas todas las letras, se procedió a desarrollar el código de programación para los botones del dispositivo, para lo cual se creó la siguiente función mostrada en la Figura 3.15. Se realizó un sistema de control anti-rebotes en los botones, para evitar posibles datos falsos en el envío de los datos al servidor *firebase*.

```
int presionarBoton(){
    if (digitalRead(pulsadorBorrar) == HIGH) {
        if (buttonActive == false) {
            buttonActive = true;
            buttonTimer = millis();}
    if ((millis() - buttonTimer > longPressTime) && (longPressActive == false)) {longPressActive = true;}
    }else{
    if (buttonActive == true) {
        if (longPressActive == true) {
            mySerial.print('l');
            lcd.clear();
            lcd.setCursor(0, 1); lcd.print("Letra: ");
            i=-1;
            longPressActive = false;
        }else{
             contaPulsos2++;
             lcd.setCursor(i, 0); lcd.print(" ");  i--; contaPulsos2=0;
        }
      buttonActive = false;
    }
    return conta;
  }
}
```

Figura 3.15 Función para botones del dispositivo

- Se inició con el botón para imprimir los símbolos. En esta parte del código el mensaje es enviado, tanto al LCD, como al servidor *Web*. Esto se puede evidenciar en la Figura 3.16.

```
//////////PULSADOR ENVIA LA LETRA A LA LCD/////////////////
if(digitalRead(pulsadorEnviar)==HIGH){estadol=1;}
if(estadol==HIGH && digitalRead(pulsadorEnviar)==LOW){contaPulsosl++; estadol=0;}

if(contaPulsosl==1){i++; lcd.setCursor(i, 0); lcd.print(letra); mySerial.print(letra); if(i>=15){i=0;} contaPulsosl=0;}
//if(contaPulsosl==1){mySerial.print('A'); contaPulsosl=0;}
```

Figura 3.16 Programación para el botón “Imprimir”

- El siguiente paso fue crear el botón para borrar el mensaje. Este botón permite eliminar absolutamente todo lo que esté impreso en la pantalla y no para hacerlo letra por letra, esto debido a que mientras el usuario está realizando la posición del símbolo que desea mostrar, este aparece en la segunda línea del LCD y solo si es la correcta, él elije si la imprime o no, por lo que no pueden generar errores. En la Figura 3.17 se observa esta línea de programación.

```
retorno=presionarBoton();

int presionarBoton(){       // empezamos una funcion int sin retorno
     if (digitalRead(pulsadorBorrar) == HIGH) { //condicion si se presiono el boton Borrar
         if (buttonActive == false) {           // colocamos la variable en falso
             buttonActive = true;               // colocamos la variable en verdadero
             buttonTimer = millis();}           // igualamos la funcion millis con la variable buttontimer
     if ((millis() - buttonTimer > longPressTime) && (longPressActive == false)) {longPressActive = true;}
     }else{
     if (buttonActive == true) {    // si la variable buttonActive esta en verdadero
         if (longPressActive == true) {   //y si la variable longPresActive esta en verdadero
             mySerial.print('#'); // imprimimos el simbolo (#) en el puerto serial
             lcd.clear();   // limpiamos el lcd
             lcd.setCursor(0, 1); lcd.print("Letra/Numero: ");    // imprimimos en el lcd
             i=-1;    // restamos 1 a la posicion del lcd
             longPressActive = false;     // volvemos a colocar la variable en falso
         }else{

              lcd.setCursor(i, 0); lcd.print(" ");  i--;     // imprimimos en espacio
         }
      buttonActive = false; // volvemos la variable a el estado falso
    }
    return conta;   //retornamos
  }
}
```

Figura 3.17 Programación para el botón “Borrar”

- Se debe recalcar que el código cuenta con líneas de programación para evitar el “efecto rebote” de los tres pulsadores, mediante el constante testeo de cada pulso.
- Finalmente, se programó el botón para dar espacio entre las letras. Esto se observa en la Figura 3.18.

```
if(digitalRead(pulsadorEspacio)==HIGH){estado3=1;}
if(estado3==HIGH && digitalRead(pulsadorEspacio)==LOW){contaPulsos3++; estado3=0;}

if(contaPulsos3==1){i++; lcd.setCursor(i, 0); lcd.print(" "); mySerial.print(' '); contaPulsos3=0;}
```

Figura 3.18 Programación para el botón “Espacio”

El código completo totalmente comentado se encuentra en el Anexo 4.

En la Figura 3.20 se aprecia el diagrama de flujo utilizado para el código desarrollado en Arduino.

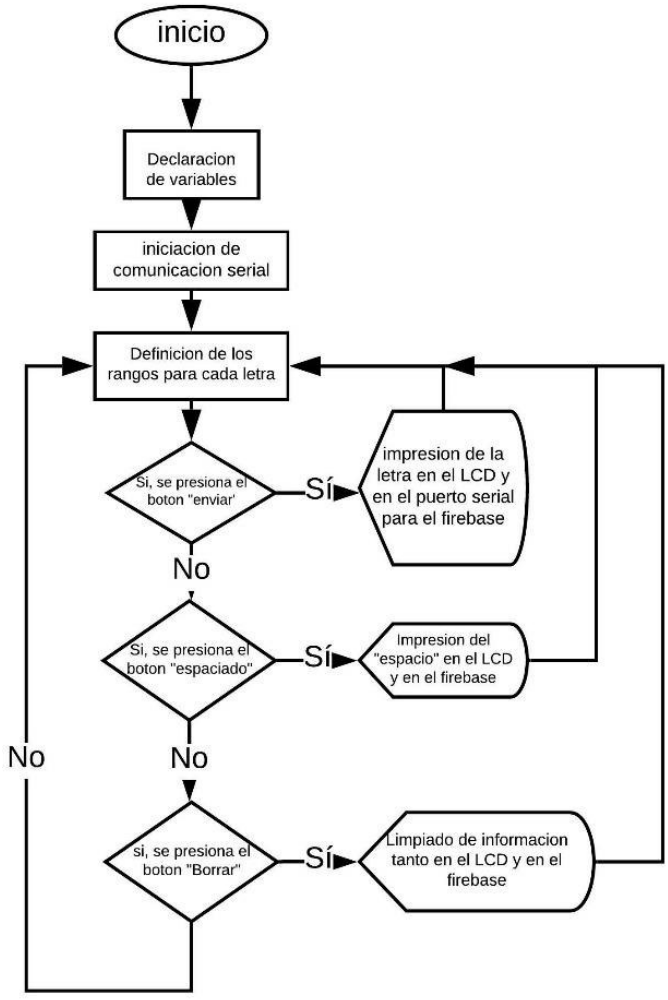


Figura 3.20 Diagrama de flujo del código de programación

3.4 Red inalámbrica

Programación NodeMCU

Con los datos medidos, listos para ser presentados el siguiente paso fue configurar el módulo ESP8266, el cual es el encargado de transmitir el mensaje a través de la red para que este pueda ser visualizado por cualquier persona con acceso a internet.

Una ventaja de haber utilizado este módulo es que su programación puede realizarse en el IDE Arduino.

A continuación, se describe la programación realizada:

- Se incluyeron las librerías del módulo ESP8266 y del servidor de *firebase* para poder utilizar estos elementos en la programación. Además, se definen las credenciales de la red wifi a la que se pretende conectar el módulo y de la página de *firebase* creada. Adicionalmente, se crearon dos variables para recibir las palabras en el servidor. En la Figura 3.19 se aprecia esta parte del código.

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

#define WIFI_SSID "NETLIFE-JAIRO MALES"  //Nombre dela RED WIFI
#define WIFI_PASSWORD "jairomales4383"   //Contraseña de la RED WIFI
#define FIREBASE_HOST "guante-traductor-default-rtdb.firebaseio.com"  //Database firebase
#define FIREBASE_AUTH "RFJIbzgqsqCl7QHARPaYENbaPvNc9YdaU4Nb6qrK"    //Contraseña de seguridad firebase

FirebaseData fbdo;

char recibir;
String palabra="";
```

Figura 3.19 Ingreso de credenciales

- Se inició la comunicación serial, que servirá para que se comuniquen la placa Arduino y el módulo ESP8266, que permitirá que se autentifiquen las credenciales de la red. Para que la conexión inalámbrica sea exitosa se utilizó la función "*whille*" para que mientras las credenciales colocadas en el programa sean las mismas que las de la red, la conexión sea exitosa. En la Figura 3.20 se observan las líneas de código correspondientes a lo descrito.

```
14 void setup(){
15      Serial.begin(9600);   delay(50);          //Inicializa puerto serial
16      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);     //Busca conectarse a la red WIFI
17      //Serial.print("Connecting to Wi-Fi");      //Impresion texto CONECTANDOSE
18      while (WiFi.status() != WL_CONNECTED){    //Conectando
19      //Serial.print(".");
20      delay(300);
21      }
22      //Serial.println();
23      //Serial.print("Connected with IP: ");
24      //Serial.println(WiFi.localIP());           //Impresion direccion ip de la red WIFI
25      //Serial.println();
```

Figura 3.20 Configuración del ESP8266

- En la Figura 3.21 se presenta el código principal, cuya función consiste en que, si el puerto serial está disponible, la variable que recibe el mensaje debe leerlo. Luego la función "*Firebase.setString*" se encarga de escribir el valor de la cadena en el nodo ubicado en la ruta de la página *Web*.

```
41 void loop() {
42      if(Serial.available()){
43       recibir=(char)Serial.read();
44       palabra+=recibir;
45       Firebase.setString(fbdo, "palabra", palabra);
46       //Serial.println(palabra);
47       if(recibir=='1'){palabra=""; Firebase.setString(fbdo, "palabra", palabra); }
48      }
49
50
51
52 }
```

Figura 3.21 Programa principal ESP8266

Comunicación con *firebase*

Para que el servidor de *firebase* y el módulo ESP6682 puedan establecer una comunicación fue necesario realizar las siguientes configuraciones:

- Se ingresó a la página principal de *firebase*, la cual se visualiza en la Figura 3.22. El primer paso es seleccionar el botón de comenzar. En caso de ingresar por primera vez, se cuenta con un tutorial de iniciación.

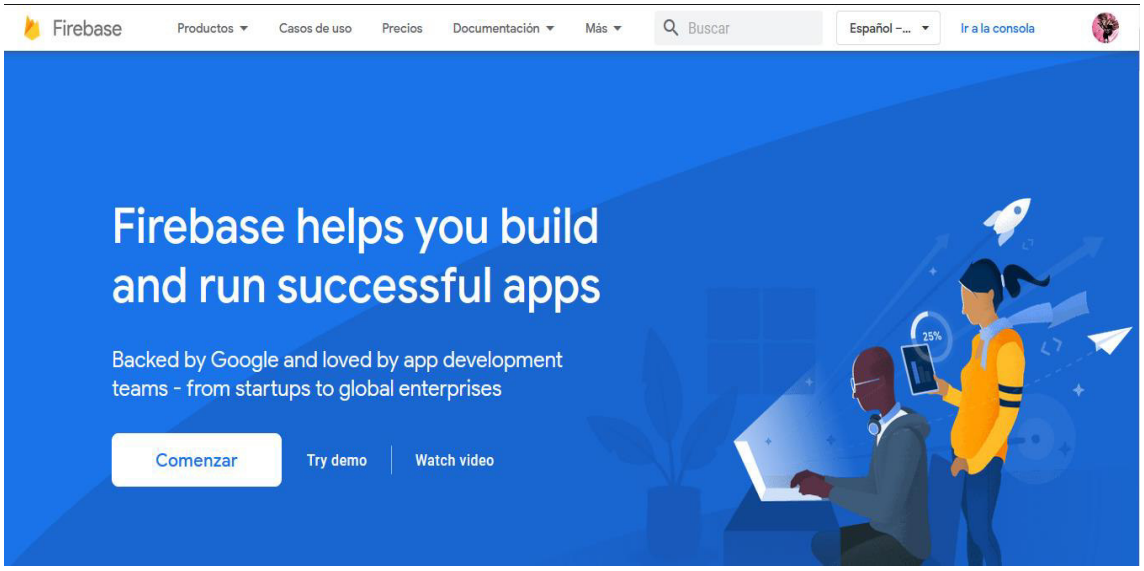


Figura 3.22 Página principal de *firebase*

- Se desplegará una pantalla con el mensaje “Crear un proyecto”, tal cual se aprecia en la Figura 3.23.

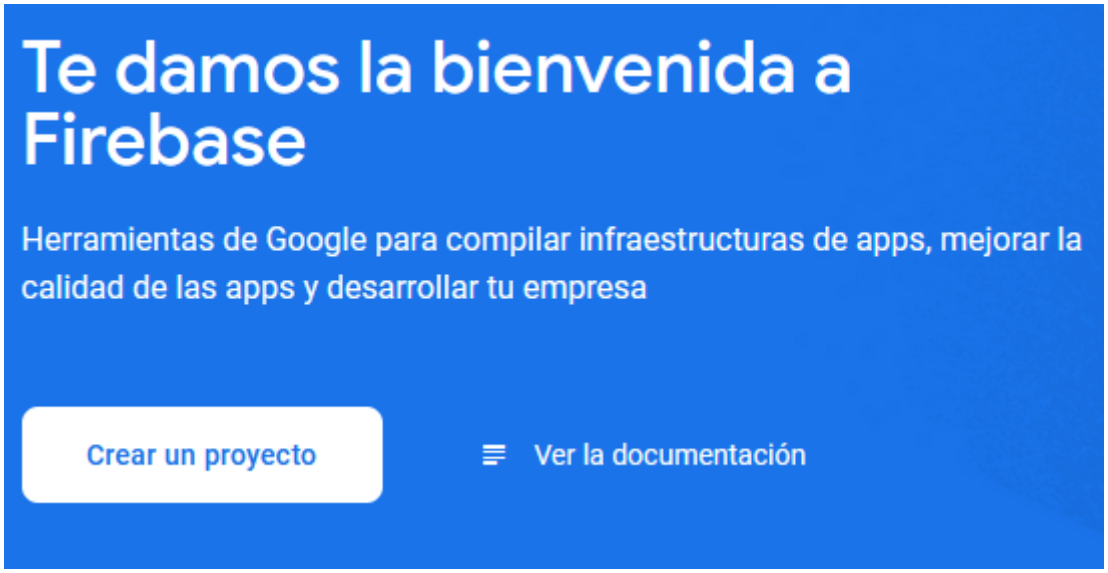


Figura 3.23 Crear proyecto en *firebase*

- El siguiente paso es ponerle nombre al proyecto y aceptar las condiciones, así como se observa en la Figura 3.24.

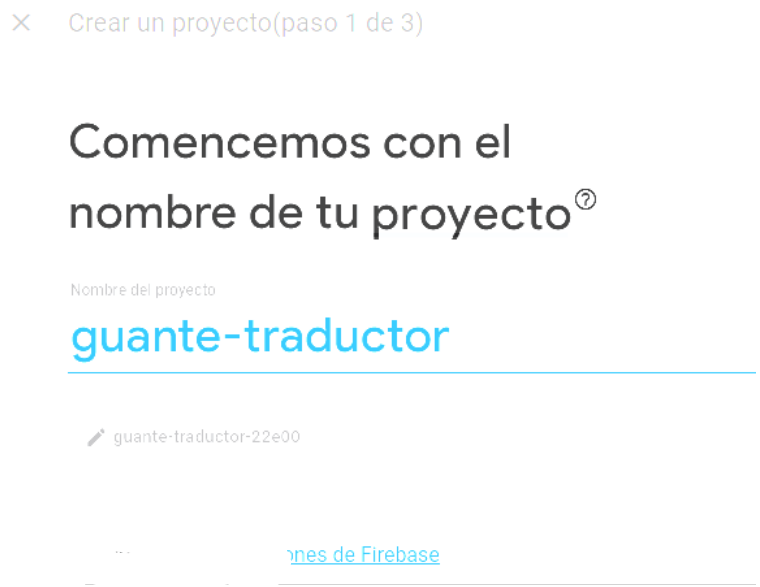


Figura 3.24 Nombre del proyecto

- El siguiente paso es elegir si se desea usar "*Google Anaytics".* Esta herramienta sirve para entender el comportamiento de los usuarios para tomar decisiones fundamentadas en cómo comercializar el proyecto [14]. En este caso, como es un dispositivo que será donado, no es necesario hacer uso de este instrumento. En la Figura 3.25 se observa que se deshabilita esta opción y se da en el botón de "Crear proyecto".

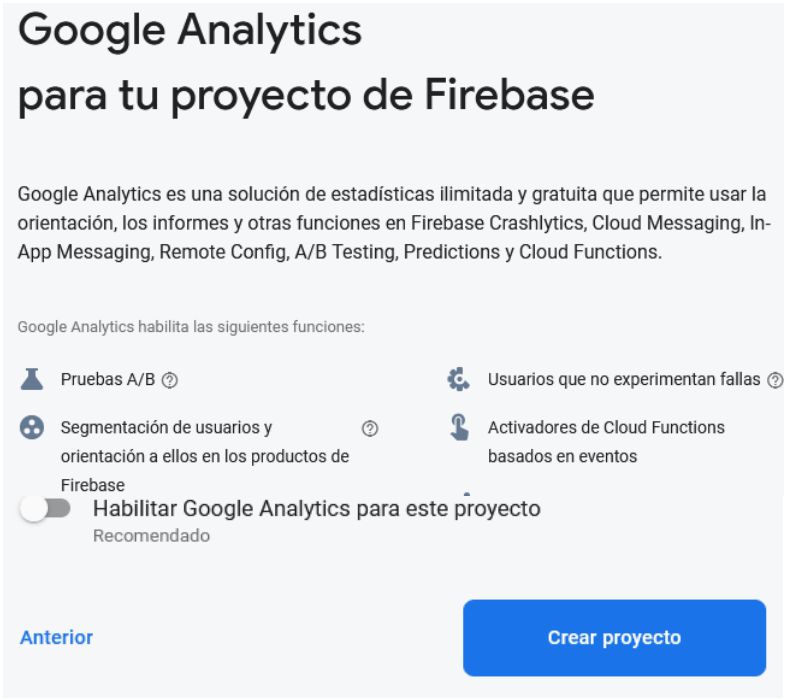


Figura 3.25 *Google* Analytics

- Se espera a que cargue el nuevo proyecto y se despliega la pantalla inicial donde se encuentra un menú con muchas opciones para agregar al proyecto. Aquí se debe buscar el apartado de “configuración” y darle clic, como se observa en la Figura 3.26.

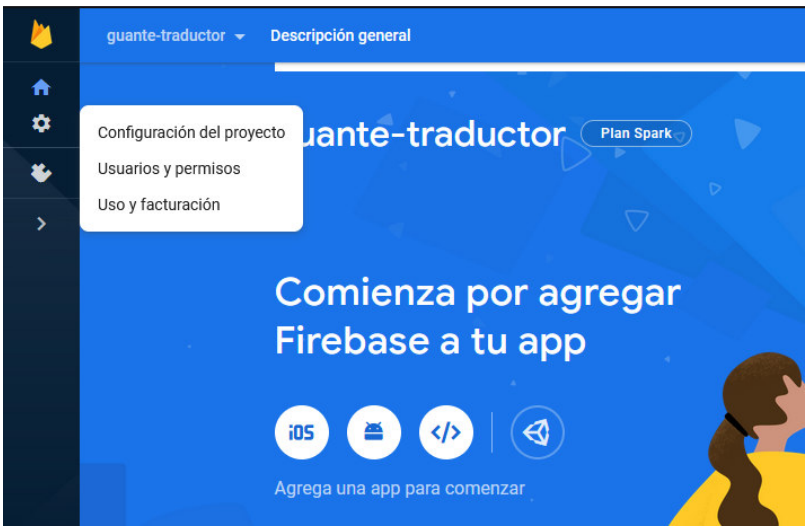


Figura 3.26 Entorno principal del proyecto

- En la Figura 3.27 se observan los parámetros asignados por el servidor al proyecto. Estos parámetros son con los que se va a trabajar.

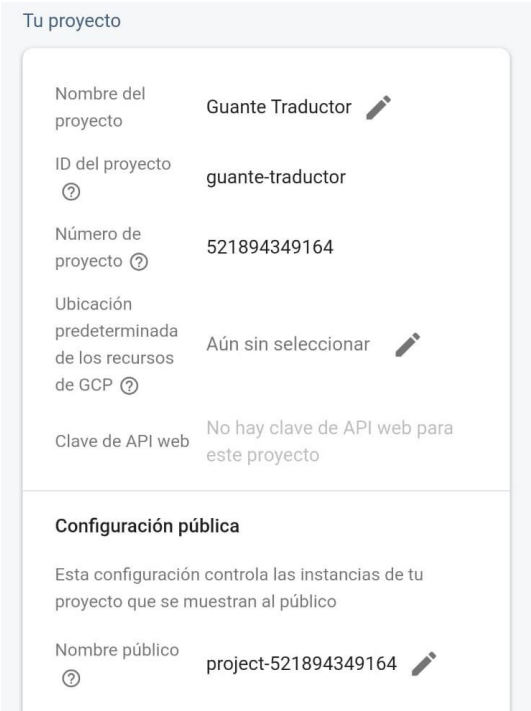


Figura 3.27 Parámetros proyecto *firebase*

En la Figura 3.28 se aprecia el diagrama de flujo del NodeMCU para la recepción y el envío de datos hacia el servidor de *firebase* y para así posteriormente ser enviado hacia la aplicación de Android.

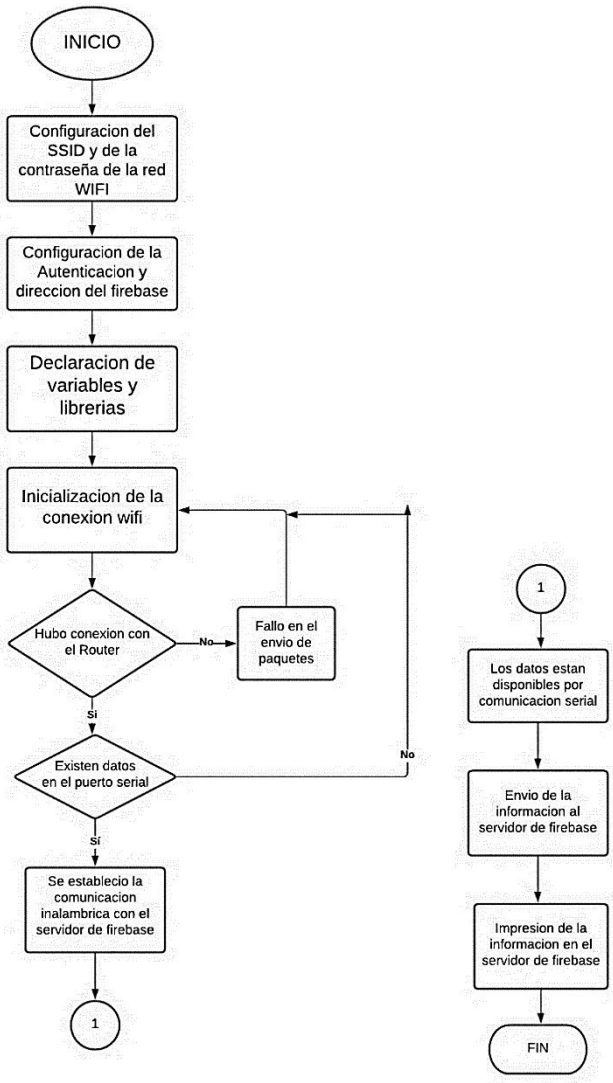


Figura 3.28 Diagrama de flujo del NodeMCU

En el Anexo 5 se muestran los respectivos códigos de programación.

3.5 Aplicación móvil en la plataforma MIT *App Inventor*

La aplicación para teléfonos móviles fue desarrollada en la plataforma *Web* MIT *App Inventor*. La principal función de la aplicación es la toma de información (traducción del lenguaje dactilológico) de la base de datos de la plataforma de *firebase*, y mostrarla de manera visual en cualquier dispositivo *Android*.

La creación de la aplicación se divide en dos partes, en la primera ventana se aprecia la pantalla de ingreso, ver en la Figura 3.29, con unos botones que permiten inicializar la aplicación o a su vez finalizar y salir de la aplicación. Además, se aprecia su respectivo código en bloques en la Figura 3.30.

Figura 3.29 Pantalla de inicio de la aplicación móvil

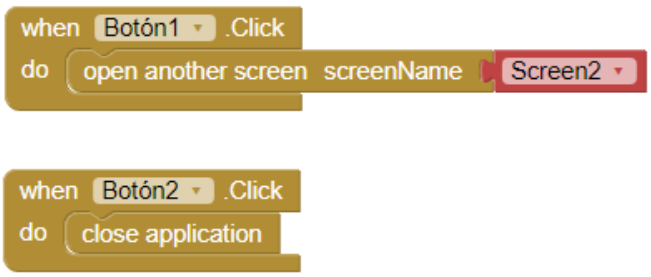


Figura 3.30 Bloques de código aplicación “pantalla inicio”

En la segunda ventana (Figura 3.31) de la aplicación se visualiza la traducción directa desde el *firebase*, en donde se aprecia el alfabeto en el idioma castellano, además esta ventana consta de dos botones que permiten borrar la información recibida en la

aplicación, y salir de la aplicación por completo. Se observa el código en bloques en la Figura 3.32. Se aprecia también que en la segunda ventana se encuentra un mensaje predeterminado, el cual recuerda que, para que la aplicación se encuentre funcionando correctamente, se debe revisar la conexión a la base de datos.

Figura 3.31 Segunda pantalla de la app móvil

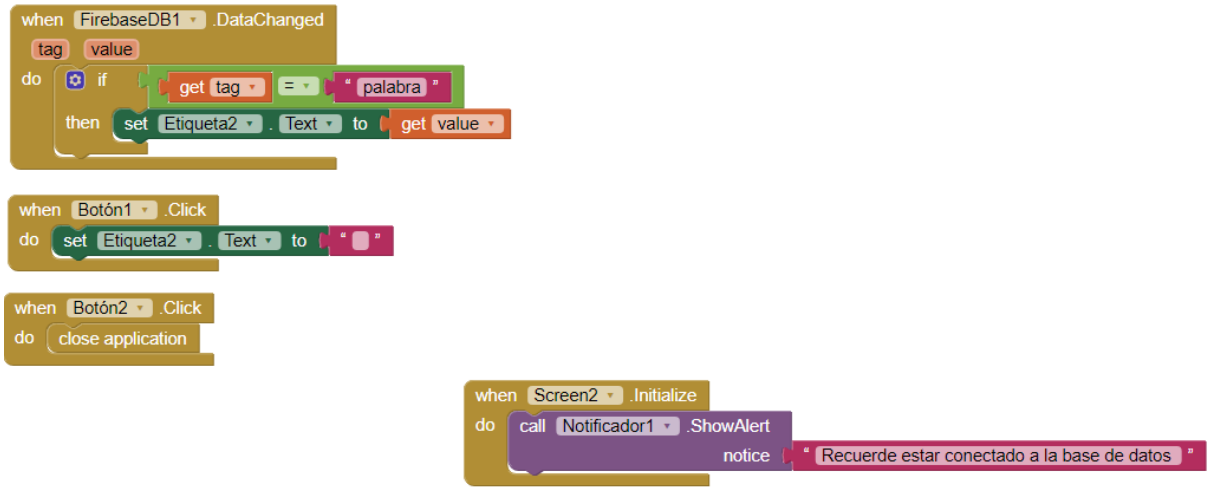


Figura 3.32 Bloques de código aplicación “segunda pantalla”

En la Figura 3.33 se aprecia el diagrama de flujo utilizado en el código para la realización de la aplicación en *App Inventor*.

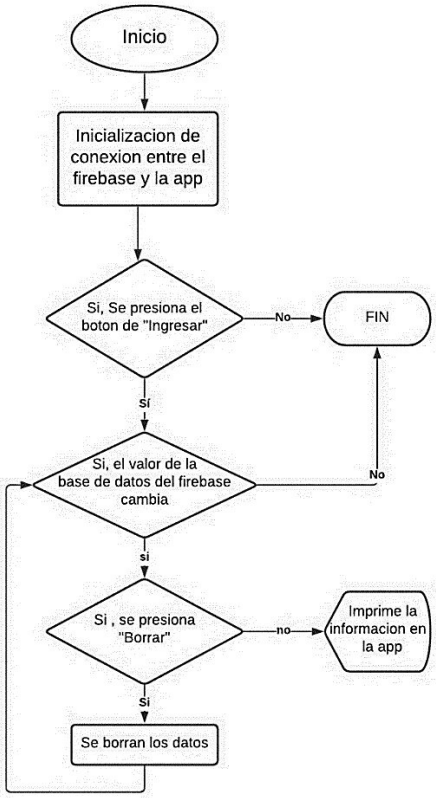


Figura 3.33 Diagrama de flujo de la aplicación

3.6 Implementación

La lista de materiales utilizados para crear este proyecto se observa en la Tabla 3.5.

Tabla 3.5 Lista de elementos

CANTIDAD	ELEMENTO
1	Placa Arduino UNO
1	Módulo NodeMCU ESP8266
5	Sensores flex
1	LCD 2x16
1	Cable UTP
5	Resistencias 4.7 (kΩ)

El guante electrónico fue desarrollado para que el usuario se sienta cómodo mientras lo utiliza, sin miedo a generar posibles daños por lo que las conexiones entre los diferentes

elementos fueron realizadas de manera fija y estable, reforzando con silicona o suelda en varios puntos. El guante con las respectivas conexiones se observa en la Figura 3.34.

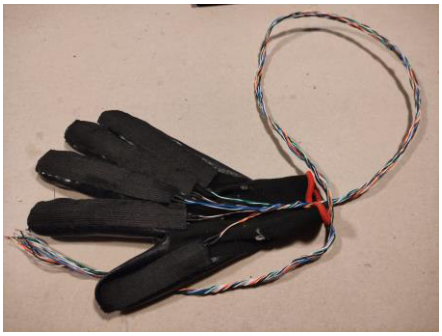


Figura 3.34 Guante con conexiones

Con el diseño realizado en Proteus 8.12, se procedió a grabar la baquelita con sus respectivas perforaciones. El resultado de este proceso se aprecia en la Figura 3.35.

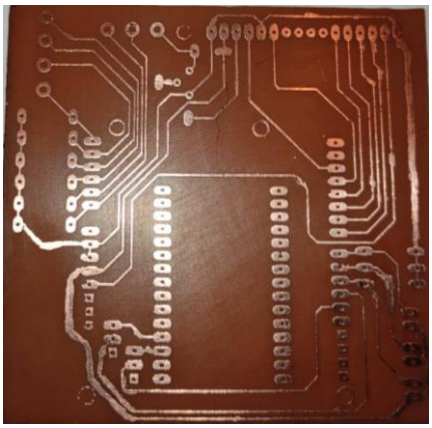


Figura 3.35 Baquelita construida

Una vez quemada la baquelita se soldaron todos los componentes (resistencias, dipswitch, espadines macho y hembra y un par de bloques terminales) como se aprecia en la Figura 3.36.

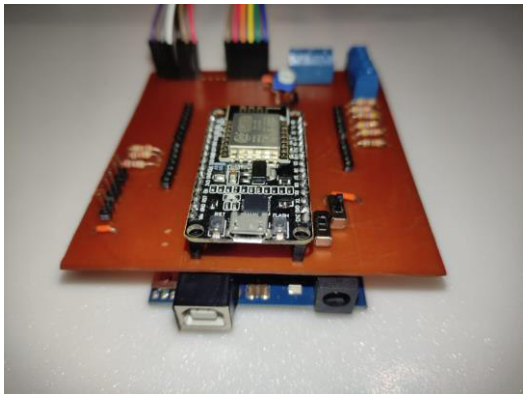


Figura 3.36 Elementos soldados

Después de soldar los elementos se realizó la construcción de la caja con los orificios respectivos. Luego se cortó la madera mediante láser para que el acabado sea más preciso. Con las partes de la caja listas se procedió a su ensamble para colocar el circuito dentro de ella y se aseguraron los elementos con silicona para evitar que se muevan. La caja con los elementos respectivos se observa en la Figura 3.37.

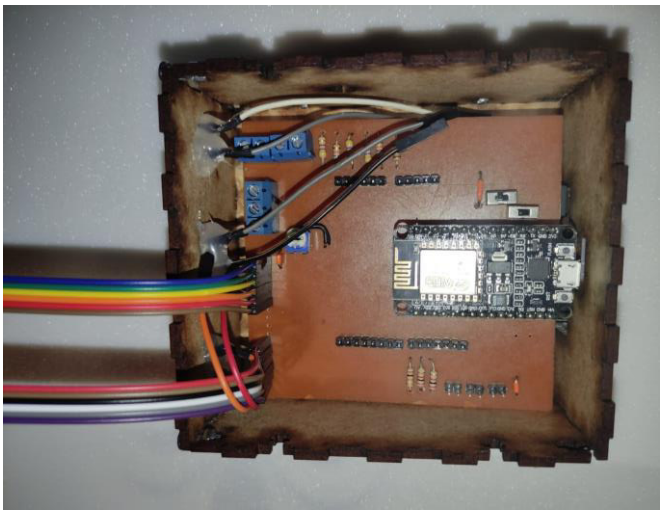


Figura 3.37 Caja con elementos del sistema

3.7 Pruebas y análisis de resultados

Una vez listo todo el circuito, con las conexiones adecuadas, con cada elemento en su posición y el servidor activo; se realizaron las pruebas de funcionamiento del equipo.

Prueba de escritura

Se verificó que el guante sea fácil de manipular y que no exista ningún inconveniente al momento de realizar los movimientos correspondientes de los dedos.

Al inicio se tuvieron problemas de sujeción de los cables a la placa, por lo que se procedió a fortalecer la soldadura y a pegar los elementos en lugares específicos con el fin de que no generen movimiento al utilizar el guante.

Se comprobó que las letras aparecieran en la pantalla del LCD y que los botones de “espacio” y “borrar” cumplan su propósito. Se tuvo que comprobar letra por letra para descartar errores, porque es primordial que todas sean mostradas, caso contrario no se cumpliría con el objetivo del proyecto. A continuación, se muestra un ejemplo escribiendo la palabra “HOLA”.

- Se realiza con el guante la posición de la letra “H”, como se muestra en la Figura 3.38.

Figura 3.38 Letra “H”

- Para evitar letras erróneas, el usuario puede observar en la segunda fila del LCD la letra que esté formando en ese instante, y, solo si está seguro de que es la correcta, deberá presionar el botón para imprimirla. Esto se visualiza en la Figura 3.39.

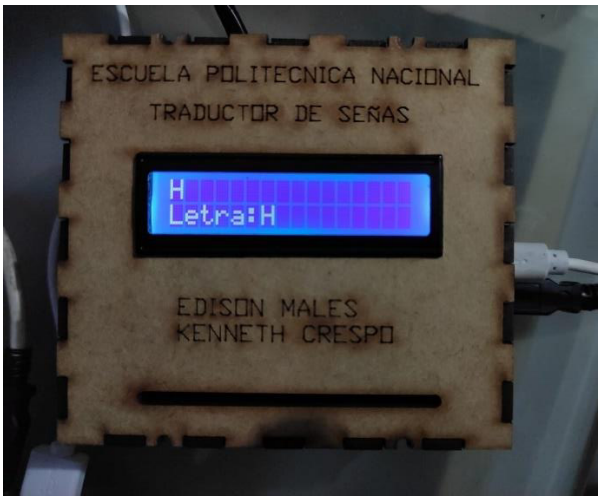


Figura 3.39 Imprimir letra “H”

- Se debe seguir el mismo procedimiento para escribir el resto de las letras. En las Figura 3.40, Figura 3.41, Figura 3.42, Figura 3.43, Figura 3.44, Figura 3.45, se puede verificar que se escribió la palabra completa sin ningún problema.

Figura 3.40 Letra “O”

Figura 3.41 Imprimir letra “O”

Figura 3.42 Letra “L”

Figura 3.43 Imprimir letra “L”

Figura 3.44 Letra “A”

Figura 3.45 Imprimir letra “A”

- También se comprobó que tanto el botón de espacio, como el de borrar funcionen normalmente. En la Figura 3.46 se observa que al presionar el botón de espacio y que luego de imprimir otra letra, efectivamente exista una separación entre ellas.

Figura 3.46 Funcionamiento del botón “espacio”

- Finalmente se comprobó el botón “borrar”, el cual debe eliminar todo el mensaje escrito en la pantalla, tal cual se verifica en la Figura 3.47.

Figura 3.47 Funcionamiento del botón “borrar”

Prueba de comunicación con el servidor

Se comprobó el envío de datos vía wifi con el ESP8266 hacia el servidor de *firebase*. Al igual que en la prueba anterior, se verificó que todas las letras aparecieran en la interfaz del servidor, se va a tomar el mismo ejemplo de la palabra “*HOLA*”.

- Se verificó que exista comunicación entre el servidor y el guante. En la Figura 3.48 se observa que la página *Web* está esperando recibir el mensaje.

Figura 3.48 Inicio *firebase*

- Se procede a realizar la letra “H” e imprimirla para subirla a *firebase* tal como se observa en la Figura 3.49.

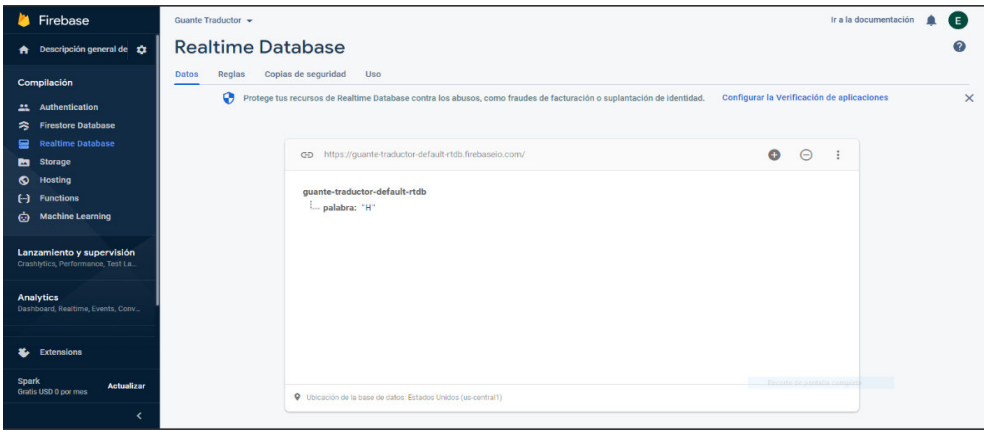


Figura 3.49 Letra “H” en *firebase*

- Se sigue el mismo procedimiento anterior con el resto de las letras, tal cual se observa en la Figura 3.50, Figura 3.51 y Figura 3.52.

Figura 3.50 Letra “O” en *firebase*

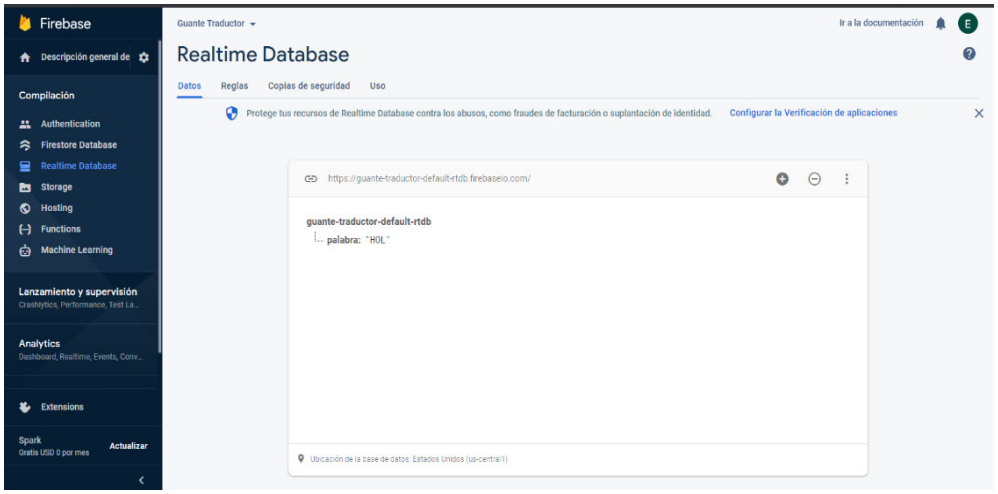


Figura 3.51 Letra “L” en *firebase*

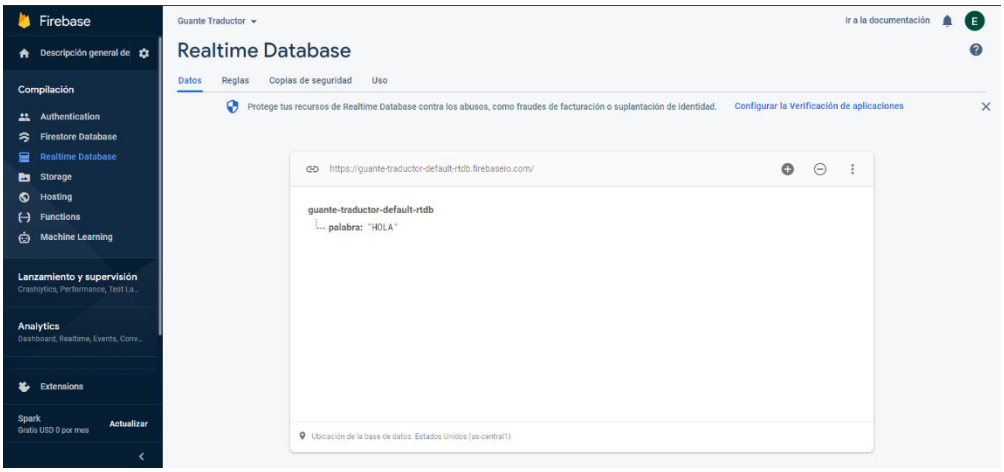


Figura 3.52 Letra “A” en *firebase*

- Finalmente se comprueba que también funcione el botón de espacio. Esto se visualiza en la Figura 3.53.

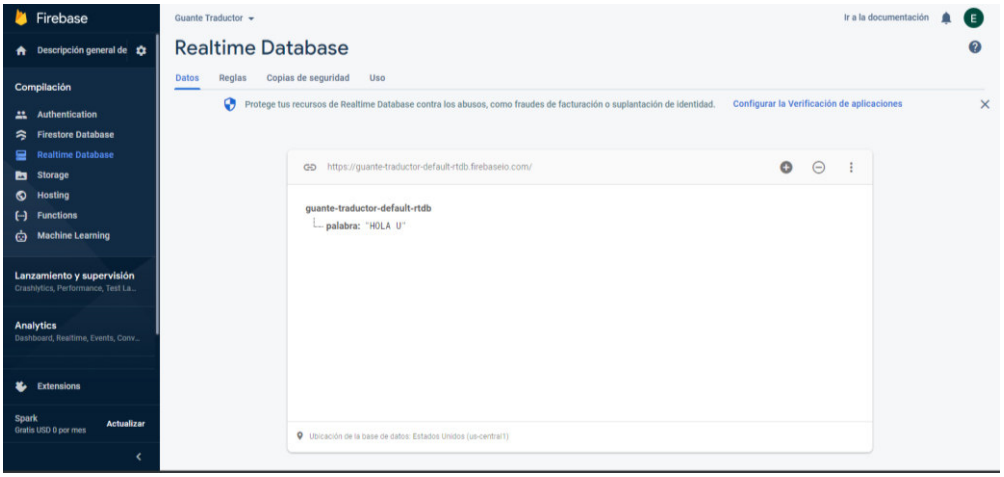


Figura 3.53 Prueba de botón espacio

Prueba con persona con pérdida de audición

Finalmente se comprobó que el prototipo esté listo para ser usado por las personas con pérdidas de audición a las que está dedicado este proyecto. La verificación de esta prueba se observa a través de un video al cual se puede acceder mediante el código QR de la Figura 3.54.

Figura 3.54 Codigo QR prueba con voluntario

Finalmente, en la Tabla 3.6 se registra el cumplimiento de funcionamiento del dispositivo.

Tabla 3.6 Cumplimiento de pruebas del dispositivo

PRUEBA	LCD	*FIREBASE*
Aparecen todas las letras y números	✓	✓
Botón “Imprimir”	✓	✓
Botón “Espacio”	✓	✓
Botón “Borrar”	✓	✓

3.8 Manual de usuario y mantenimiento

Para un correcto uso del guante se realizó un video explicativo de su funcionamiento y mantenimiento. Este video está dedicado tanto para las personas que no poseen la capacidad de escuchar, como para las personas que si pueden hacerlo.

Para acceder al video se debe utilizar el código QR de la Figura 3.55.

Figura 3.55 Código QR manual

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se consiguió construir un dispositivo práctico, manipulable y de fácil uso con un propósito muy importante que logrará generar un cambio en las comunicaciones entre personas con y sin discapacidad auditiva, y personas que desconocen el lenguaje dactilológico.
- Se logró aplicar todo el conocimiento adquirido en la carrera trabajando por un bien común creando un proyecto que se preste al servicio de la comunidad, el cual busca eliminar diferencias con las personas con discapacidad auditiva.
- La investigación realizada durante el proyecto permitió encontrar la solución más adecuada hacia el problema presentado, analizando métodos diferentes y aprendiendo algo nuevo como es el lenguaje dactilológico.
- El mensaje final se presenta de una manera amigable y entendible hacia el usuario gracias a todos los elementos usados, además permite una comunicación dinámica e interactiva entre las personas.
- El desarrollo personal de las personas que van a hacer uso de este dispositivo puede mejorar notablemente, ya que se reduce la limitante principal para comunicarse con otras personas.
- Con este prototipo se crea un mundo de oportunidades nuevas que pueden ser aprovechadas para crear mejoras como llevar la traducción del lenguaje de señas a voz o crear una comunicación bidireccional, para que las conversaciones sean mucho más fluidas.
- Con este proyecto se da pie a una idea que puede crecer a otro nivel de interacción. Se espera que a futuro el mensaje no solo aparezca en forma de texto escrito, sino que también se pueda escuchar a través de audio.
- Gracias a los resultados obtenidos se pudo verificar que Arduino es un dispositivo sencillo de usar y es suficiente para implementar proyectos complejos. Sus características permitieron almacenar el código realizado y procesarlo de tal manera que ayude a cumplir los objetivos del proyecto.
- Las cualidades del módulo ESP6682 permitieron cumplir con el levantamiento de la red inalámbrica que comunica al circuito principal con la red.
- La comunicación wifi abrió una nueva posibilidad de interacción entre las personas con discapacidad auditiva y el resto del mundo, debido a que cualquier

persona que tenga acceso a internet logrará recibir el mensaje enviado por el guante.

- Con la creación de este prototipo, se puede incentivar de una manera más didáctica el aprendizaje e intercambio de información entre el usuario con discapacidad y el resto de la población.
- El prototipo puede ser utilizado por personas de todas las edades, debido a que es un dispositivo fácil de usar, no requiere una instalación o configuración de alto nivel, además se utilizó una medida estándar para que el guante pueda acoplarse a cada usuario que desee utilizarlo.
- Con las pruebas realizadas se evidencia que el guante es cómodo de usar y no resulta complicado aprender a manejarlo.
- Este prototipo puede ser utilizado tanto por personas con discapacidad auditiva, ya sea para comunicarse, como por personas sin la discapacidad para aprender el lenguaje dactilológico, esto se logra gracias a que el prototipo no muestra la letra si no se realizó la seña correctamente, así se ayuda a todo tipo de personas a practicar y mejorar el uso de este lenguaje.

4.2 Recomendaciones

- Antes de utilizar el guante por primera vez, se debe acudir al video explicativo del manual de usuario y mantenimiento para una correcta utilización.
- Para cargar las credenciales de la red wifi a la que se vaya a conectar el dispositivo es necesario utilizar programación en Arduino. Hay que tomar en cuenta que el programa no cargará si los pines de Rx y Tx están en funcionamiento, por lo que antes de subir el código se debe apagar el dipswitch que se encuentra en el interior del prototipo, para deshabilitar la comunicación entre los módulos, ya que esto genera un error que no permite que se cargue el nuevo código.
- Mientras el guante no sea utilizado, es recomendable colocarlo sobre alguna superficie y no dejarlo colgado, esto con el fin de que las conexiones no tiendan a desconectarse ni las pistas se dañen.
- En caso de presentar problemas con alguna letra, pueda que el tamaño de la mano del usuario no se ajuste correctamente a la primera calibración del guante.
- Evitar realizar movimientos bruscos con el guante puesto para no dañar las conexiones o provocar algún problema en los sensores flex.

- Se recomienda, para futuras actualizaciones, cambiar el LCD 16x2, por un LCD de mayor tamaño, esto ayudaría en gran medida a tener una mejor visualización de la escritura y el envío del abecedario dactilológico al servidor.
- Basándose en las necesidades de las personas que padecen este tipo de discapacidad, se recomendaría aumentar un botón o una opción para enviar mensajes de alerta o auxilio, en caso de que el usuario necesite alguna clase de asistencia.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] J. Vilela, “Dactitología,” Córdoba, 2005. [Online]. Available: http://www.uco.es/~fe1vivim/alfabeto_dactilologico.pdf.

[2] F. Madero, “Sensor Flex,” *Rambal Automatización y Robótica*, 2014. https://rambal.com/presion-peso-nivel-flex/250-sensor-flex.html (accessed Aug. 01, 2021).

[3] A. Syed, Z. T. H. Agasbal, T. Melligeri, and B. Gudur, “Flex Sensor Based Robotic Arm Controller Using Micro Controller,” *J. Softw. Eng. Appl.*, vol. 05, no. 05, pp. 364–366, 2012, doi: 10.4236/jsea.2012.55042.

[4] A. Alvarado, “NodeMCU v2 ESP8266 WiFi,” *Naylampmechatronics*, 2017. https://naylampmechatronics.com/espressif-esp/153-nodemcu-v2-esp8266-wifi.html (accessed Sep. 09, 2021).

[5] P. Velasteguí, “Firebase.” https://firebase.google.com/?hl=es-419&gclid=Cj0KCQjw4eaJBhDMARIsANhrQACaMgd51NpT1H59VUQISvz9OjZVwZ9krfkfUsOhIkmHRFbu8pHGBCAaAqnPEALw_wcB&gclsrc=aw.ds (accessed Sep. 09, 2021).

[6] M. Poveda, “TOMi Inc.,” *Aplicaciones Android con APPInventor*, 2020. https://tomi.digital/es/43829/aplicaciones-android-con-appinventor-aaca?utm_source=google&utm_medium=se (accessed Sep. 15, 2021).

[7] M. Á. Rodriguez, “Lenguaje de signos,” Confederación Ncional de Sordos De España, 1991.

[8] M. J. Duron, “Abecedario en Señas - Minders - YouTube,” *Abecedario en Señas*, 2020. https://www.youtube.com/watch?v=g1Yxx1PzSjg (accessed Aug. 07, 2021).

[9] P. Aguilar and H. Pogo, “Diseño y construcción de un guante prototipo electrónico capaz de traducir el lenguaje de señas de una persona sordomuda al lenguaje de letras.,” 2013.

[10] Y. A. Badamasi, “The working principle of an Arduino,” *Proc. 11th Int. Conf. Electron. Comput. Comput. ICECCO 2014*, 2014, doi:

10.1109/ICECCO.2014.6997578.

[11] M. C. Maldonado, “Arduino Uno Rev3 — Arduino Online Shop,” *Arduino online shop*, 2014. https://store-usa.arduino.cc/products/arduino-uno-rev3 (accessed Sep. 09, 2021).

[12] A. Sánchez, “NodeMCU v2 ESP8266 WiFi,” *All electronics*, 2018. https://naylampmechatronics.com/espressif-esp/153-nodemcu-v2-esp8266-wifi.html (accessed Sep. 11, 2021).

[13] J. V. Edison Hernandez, “Tutorial: Aprender a usar un sensor Flex con Arduino ~ LuneGate,” *Lunegate*, 2015. http://www.lunegate.net/2016/07/tutorial-aprender-usar-un-sensor-flex_31.html (accessed Sep. 12, 2021).

[14] “Google Analytics | Firebase.” https://firebase.google.com/docs/analytics?hl=es-419 (accessed Sep. 12, 2021).

ANEXOS

ANEXO 1: CERTIFICADO DE FUNCIONAMIENTO

ESCUELA POLITECNICA NACIONAL

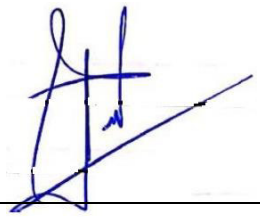
Campus Politécnico "J. Rubén Orellana R

Quito, 14 de diciembre de 2021

CERTIFICADO DE FUNCIONAMIENTO DE PROYECTO DE TITULACIÓN

Yo, *Alan Daniel Cuenca Sánchez* docente a tiempo completo de la Escuela Politécnica Nacional y como director de este trabajo de titulación, certifico que he constatado el correcto funcionamiento del guante electrónico traductor de señas mediante wifi, el cual fue implementado por los estudiantes Kenneth Crespo y Edison Males.

El proyecto cumple con los requerimientos de diseño y parámetros necesarios para que los usuarios de la ESFOT puedan usar el guante con seguridad.

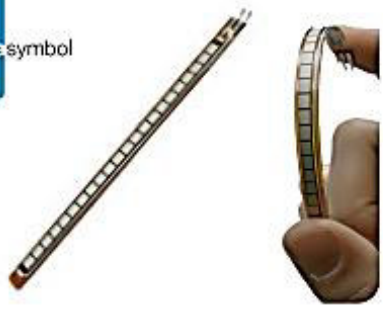


DIRECTOR

Ing. Alan Cuenca., Msc.

ANEXO 2: DATOS TÉCNICOS

spectrasymbol



FLEX SENSOR FS

Special Edition Length

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

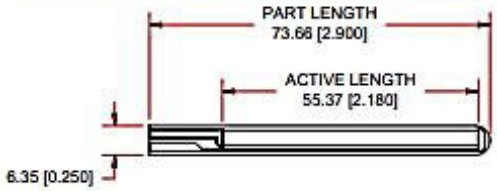
Mechanical Specifications

-Life Cycle: >1 million
-Height: ≤0.43mm (0.017")
-Temperature Range: -35°C to +80°C

Electrical Specifications

-Flat Resistance: 25K Ohms
-Resistance Tolerance: ±30%
-Bend Resistance Range: 45K to 125K Ohms (depending on bend radius)
-Power Rating : 0.50 Watts continuous. 1 Watt Peak

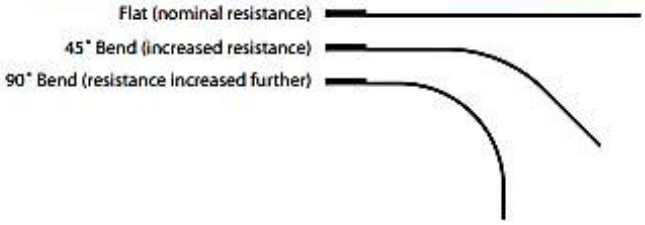
Dimensional Diagram - Stock Flex Sensor



How to Order - Stock Flex Sensor

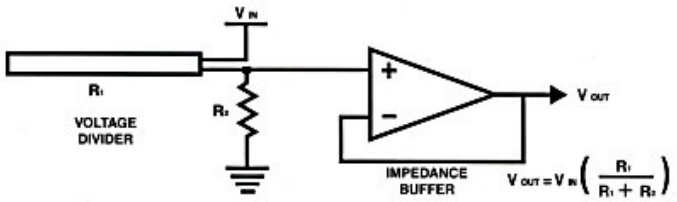
FS	—	L	—	0055	—	253	—	ST
Series		Model		Active Length		Resistance		Connectors
FS = Flex Sensor		L = Linear		0055 = 55.37mm		253 = 25K Ohms		ST = Solder Tab

How It Works



Schematics

BASIC FLEX SENSOR CIRCUIT:

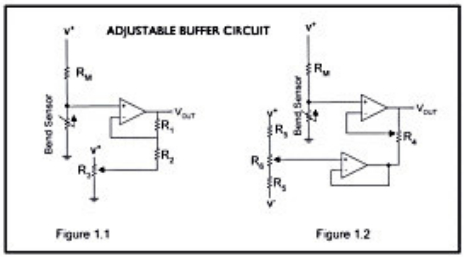


Following are notes from the ITP Flex Sensor Workshop

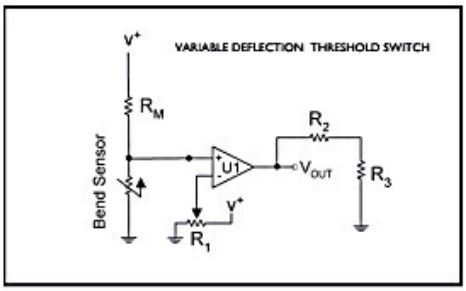
"The impedance buffer in the [Basic Flex Sensor Circuit] (above) is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces error due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324."

"You can also test your flex sensor using the simplest circut, and skip the op amp."

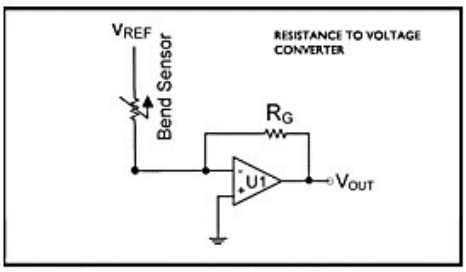
"Adjustable Buffer - a potentiometer can be added to the circuit to adjust the sensitivity range."



"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."

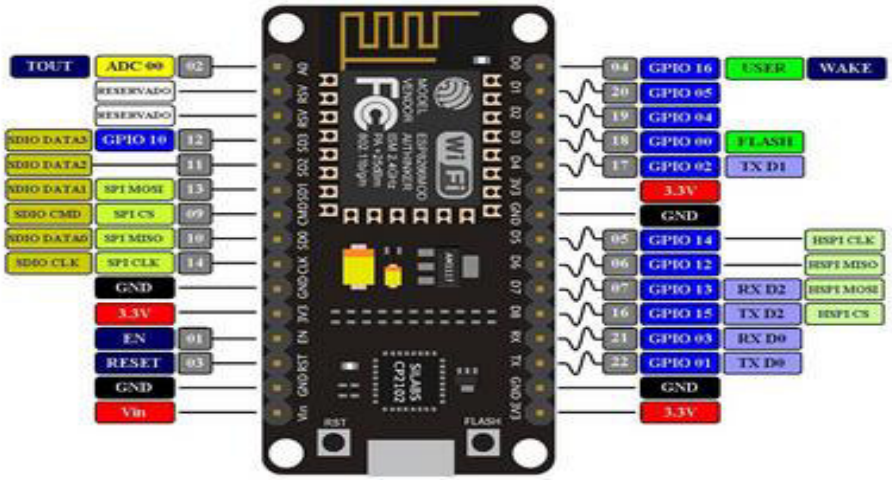


NodeMCU Development Board Pinout Configuration

Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

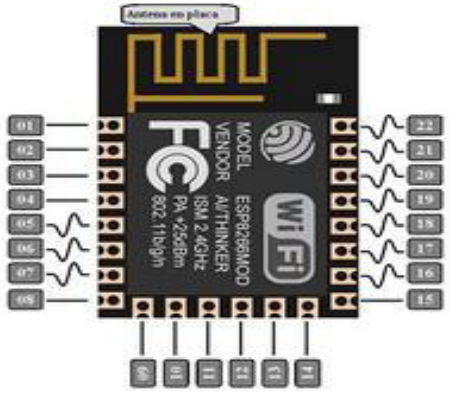
PLACA NodeMCU 1.0 (V2)

PINOUT



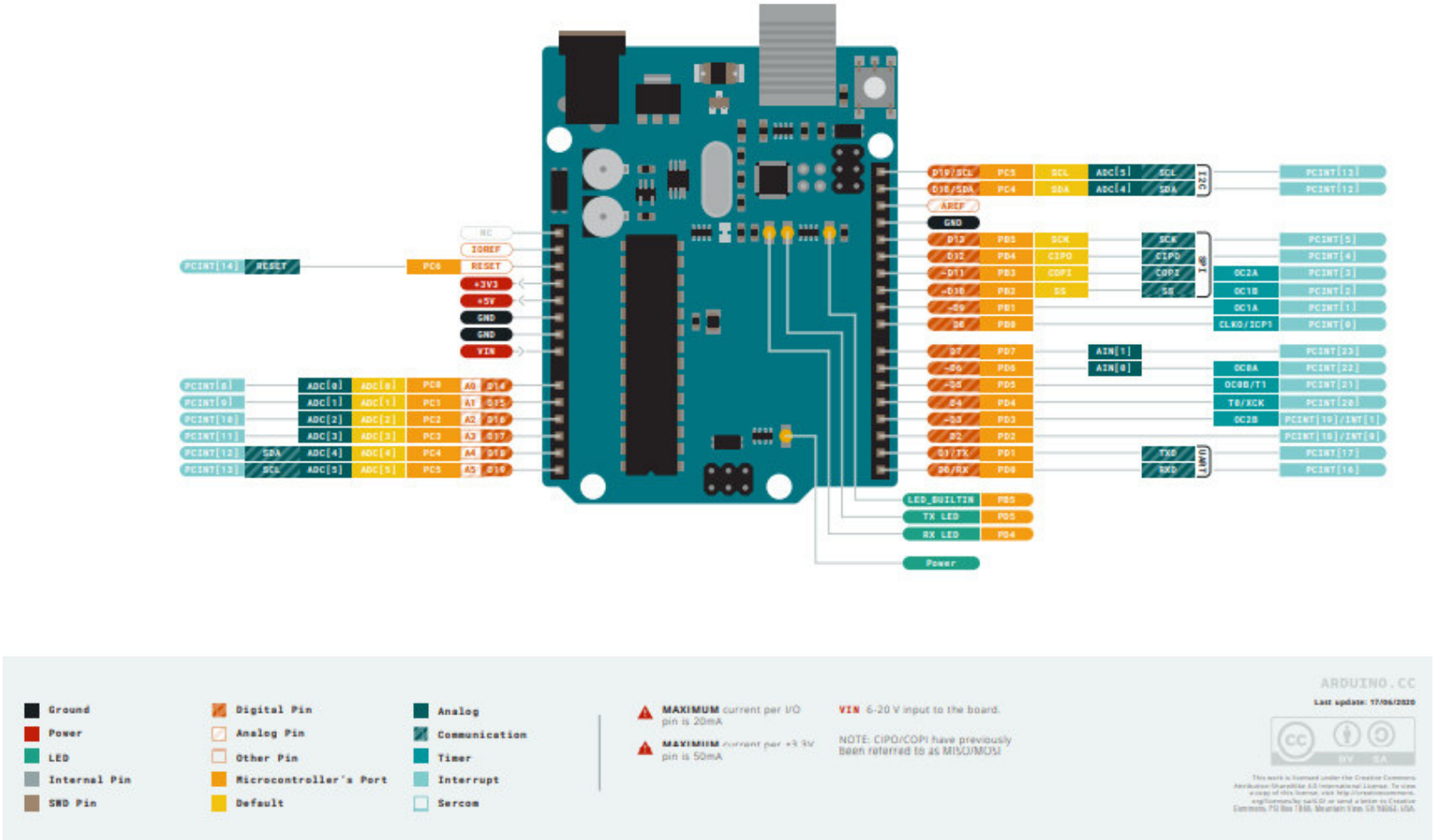
Vin	ALIMENTACIÓN EXTERNA (de 5V a 10V).
3.3V	ALIMENTACIÓN INTERNA (desde la placa a dispositivos).
GND	TIERRA (GND *Ground*).
GPIO	PIN DE ENTRADA/SALIDA +3,3V (GPIO *General Purpose Input/Output*). Entrada digital ——. Entrada analógica ∿. (Todas las salidas son digitales).
ADC	PIN DE SALIDA ANALÓGICA (el rango es entre +0V y +1V dividido en 1023 intervalos).
SPI	BUS SPI (*Serial Peripheral Interface*)
HSPI	BUS HSPI (*Hardware-Serial Peripheral Interface*).
SDIO	PINES PARA INICIO DEL ESP8266 DESDE UNA TARJETA SD. Para activar el modo SDIO el pin GPIO 15 debe estar en tensión cuando se enciende la placa.
TX/RX	COMUNICACIÓN SERIE TX/RX. Los pines GPIO01 y GPIO02 están conectados al puerto MicroUSB a través del conversor UART.

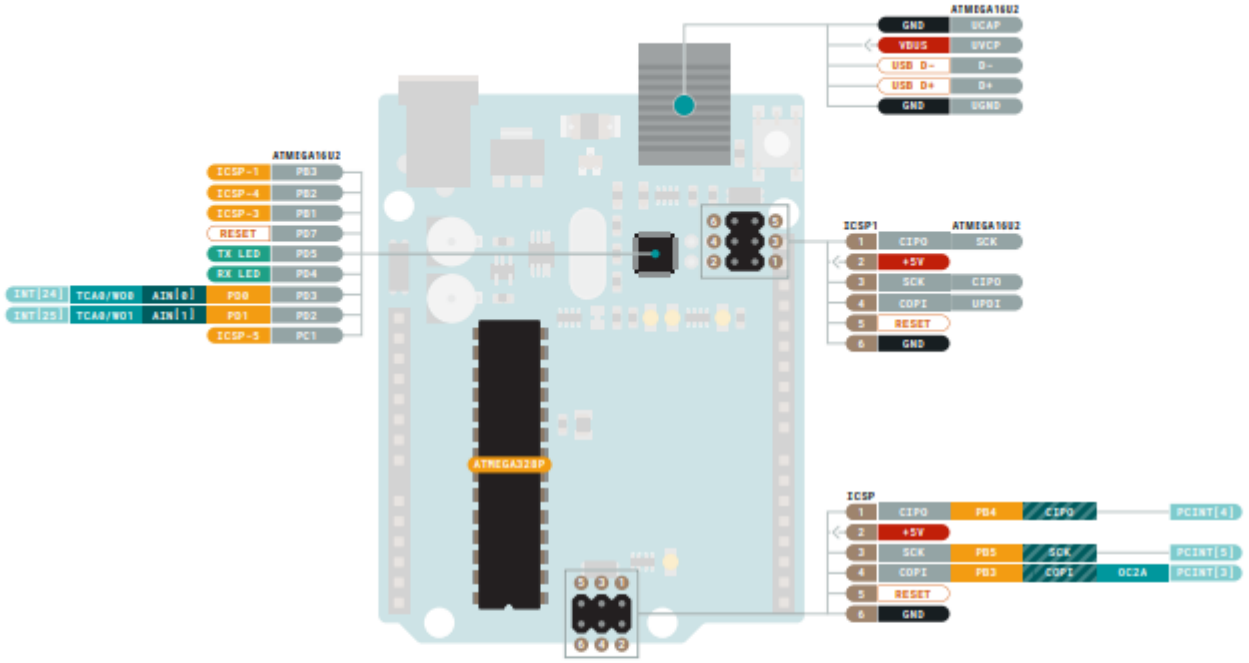
ESP8266 12E



NOTAS:

- El voltaje de alimentación (Vin) debe estar comprendido entre 5 V y 10 V.
- La intensidad de **máxima** de salida a un pin es de 12 mA. No se debe demandar mas intensidad para no quemar el procesador. La intensidad de salida normal será de 6 mA.
- Para activar el modo de reposo (*sleep mode*), unir los pines GPIO16 (D0) y RESET y poner el pin GPIO16 en tensión (*HIGH*). Para reactivar (*wakeup*), quitar la tensión en el pin GPIO16 (*LOW*). El sistema se reiniciará.
- En *boot/reset/wakeup* (inicio/reseteo/reactivado), los pines GPIO00 (D3) o GPIO15 (D8) **no** deben estar con tensión (*+3,3V*). **Tampoco** el pin GPIO2 (D4) debe estar conectado a tierra (*+0.0V*).
- Los pines GPIO01 (TX) y GPIO03 (RX) se utilizan en el puerto MicroUSB, por lo que no se deben utilizar simultáneamente con otro dispositivo ya que la conexión se interferiría.
- Los pines GPIO00 y GPIO02 **no** debe utilizarse para lectura (*Input*). El pin GPIO09 **no** debe utilizarse ni para lectura ni para escritura (*Input/output*).
- El pin GPIO02 (D4) controla el LED azul del ESP8266. Se enciende cuando no tiene tensión (*+0.0V*).
- El pin GPIO16 (D0) controla el LED azul de la placa. Se enciende cuando no tiene tensión (*+0.0V*). *(En la placa LoLin este LED no está disponible).*
- Para flashear, en el caso de que la placa quede bloqueada, se debe conectar el pin GPIO00 (D3) a tierra, el MicroUSB con el ordenador y ejecutar el flasher.


Ground
Power
LED
Internal Pin
SWD Pin
Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default
Analog
Communication
Timer
Interrupt
Sercom
MAXIMUM current per I/O pin is 20mA
MAXIMUM current per +3.3v pin is 50mA
VIN 6-20 V input to the board.
NOTE: CIPO/COPI have previously been referred to as MISO/MOSI
ARDUINO.CC
Last update: 17/06/2020



Ground
Power
LED
Internal Pin
SND Pin

Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default

Analog
Communication
Timer
Interrupt
Sercom

MAXIMUM current per I/O pin is 20mA

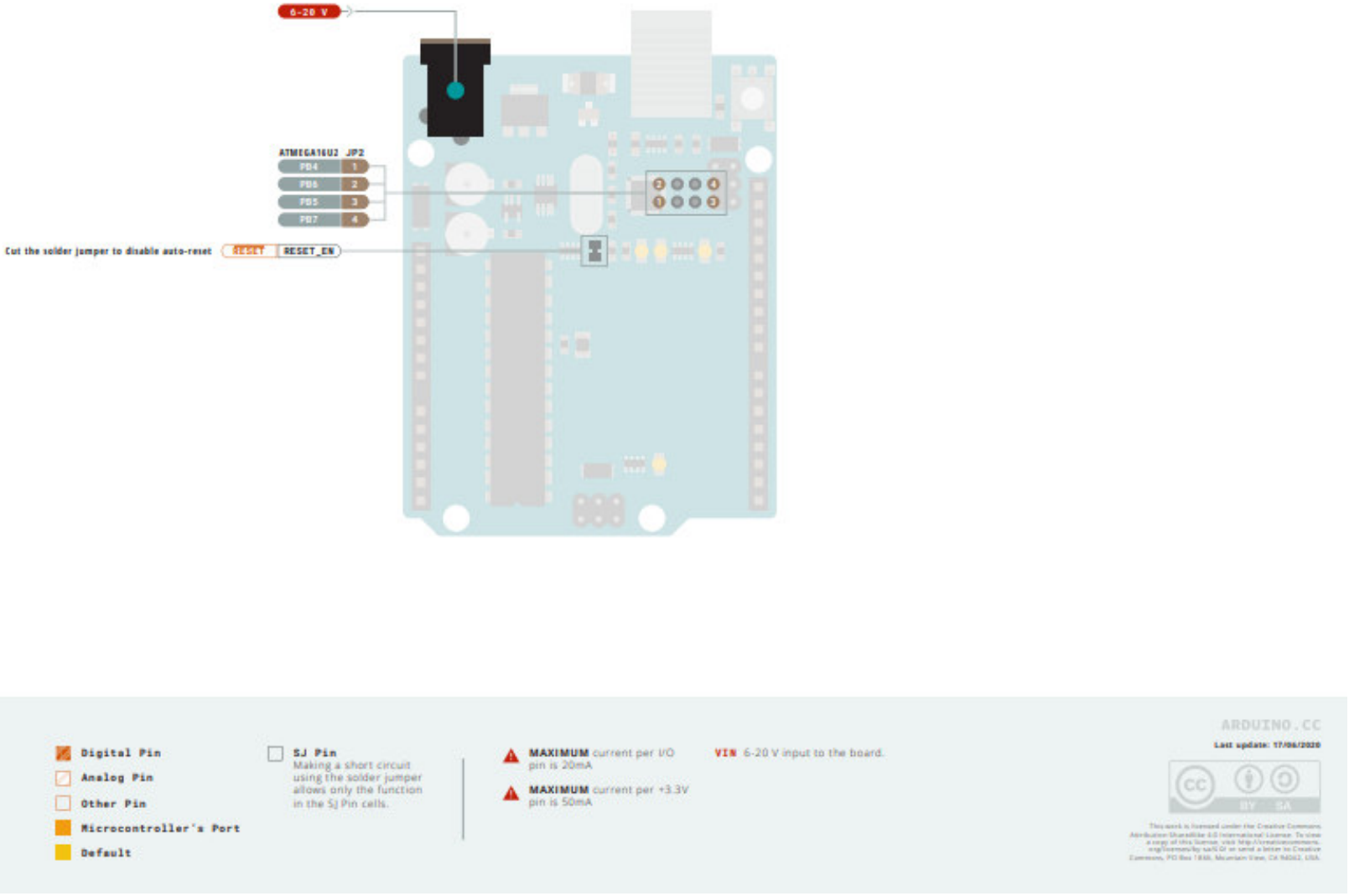
MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

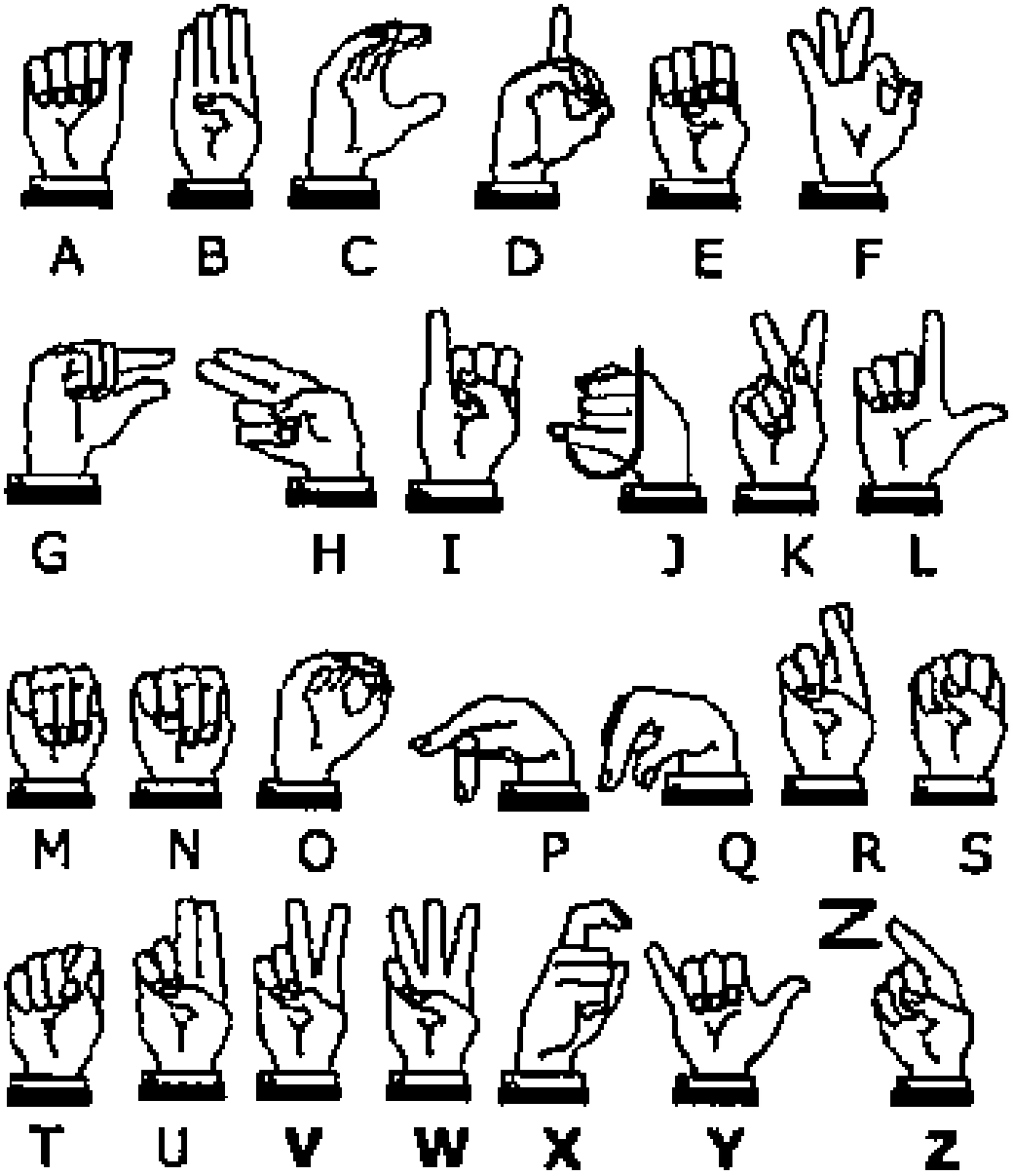
NOTE: CIPO/COPI have previously been referred to as MISO/MOSI

ARDUINO.CC

Last update: 17/06/2020


6-20 V
ATMEGA16U2
JP2
PD4
PD6
PD5
PD7
Cut the solder jumper to disable auto-reset
RESET
RESET_EN
Ground
Power
LED
Internal Pin
SWD Pin
Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default
SJ Pin
Making a short circuit using the solder jumper allows only the function in the SJ Pin cells.
MAXIMUM current per I/O pin is 20mA
MAXIMUM current per +3.3V pin is 50mA
VIN 6-20 V input to the board.
ARDUINO.CC
Last update: 17/06/2020

ANEXO 3: LENGUAJE DACTILOLÓGICO



ANEXO 4: PROGRAMA PRINCIPAL

```
#include <LiquidCrystal.h>  //incluir librerias del LCD
#include <SoftwareSerial.h>  //incluir librerias para la comunicacion
serial
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);  //rs ,en, d4, d5, d6 y d7
SoftwareSerial mySerial(12, 11); // Colocacion de nombre y definicion
los pines de RX y TX

short int pulsadorEnviar=8;                                    //
almacenamiento corto de la variable en el pin8
short int pulsadorBorrar=10;                                   //
almacenamiento corto de la variable en el pin10
short int pulsadorEspacio=9;                                   //
almacenamiento corto de la variable en el pin9
short int estado1,estado2,estado3;                             //
almacenamiento corto de las variables
short int contaPulsos1=0,contaPulsos2=0,contaPulsos3=0;  //
almacenamiento de las varibales
short int i=-1;                                                //
almacenamos un valor en la variable "i"
char letra = ' ';                                              // Variable
para almacenar caracteres

const int FLEX_PIN0 = A0;                  //variable tipo lectura
const int FLEX_PIN1 = A1;                  //variable tipo lectura
const int FLEX_PIN2 = A2;                  //variable tipo lectura
const int FLEX_PIN3 = A3;                  //variable tipo lectura
const int FLEX_PIN4 = A4;                  //variable tipo lectura
int P,I,M,A,Q;                        // almacenamiento de las variables

int conta=0,retorno;                  // almacenamiento variables
boolean LEDState = false;            // variable booleana ( true - false)
long buttonTimer = 0;                        // variable tipo extendida
long longPressTime = 250;                    // variable tipo extendida
boolean buttonActive = false; // variable booleana ( true - false
boolean longPressActive = false; // variable booleana ( true - false
char n;                              // Variable para almacenar caracteres



void setup() {
Serial.begin(9600);   // Inicializamos la comunicacion Serial de los
pines nativos del arduino
mySerial.begin(9600);   // Inicializamos la comunicacion serial de los
pines secundarios del arduino
lcd.begin(16, 2);   // Inicializamos el LCD
lcd.setCursor(2, 0); lcd.print("TRADUCTOR DE ");    //Coloccamos el
cursor en la primera fila  la segunda columna
lcd.setCursor(4, 1); lcd.print("LENGUAJE");   // Colocamos el cursor
en la segunda fila y cuarta columna
delay(2000);    //Agregamos un delay de 2000ms
lcd.clear();    //Limpiamos el LCD
lcd.setCursor(0, 1); lcd.print("Letra/Numero: ");    //Colocamos el
cursor en la segunda fila y primera columna e imprimimos el texo
"letra"
```

```
pinMode(pulsadorEnviar, INPUT);   //Definimos a "pulsadorEnviar" como
entrada
pinMode(pulsadorBorrar, INPUT);   //Definimos a "pulsadorBorrar" como
entrada
pinMode(pulsadorEspacio, INPUT);    //Definimos a "pulsadorEspacio"
como entrada
  pinMode(FLEX_PIN0, INPUT);    //Definimos A0 como entrada
  pinMode(FLEX_PIN1, INPUT);    //Definimos A1 como entrada
  pinMode(FLEX_PIN2, INPUT);    //Definimos A2 como entrada
  pinMode(FLEX_PIN3, INPUT);    //Definimos A3 como entrada
  pinMode(FLEX_PIN4, INPUT);    //Definimos A4 como entrada
}

void loop(){
P = analogRead(FLEX_PIN0);   //Lectura del pin analogico 0 se almacena
en la variable P
I = analogRead(FLEX_PIN1);   //Lectura del pin analogico 1 se almacena
en la variable I
M = analogRead(FLEX_PIN2);   //Lectura del pin analogico 2 se almacena
en la variable M
A = analogRead(FLEX_PIN3);   //Lectura del pin analogico 3 se almacena
en la variable A
Q = analogRead(FLEX_PIN4);   //Lectura del pin analogico 4 se almacena
en la variable Q


    /////////CALIBRACION  DE LAS LETRAS//////////
   /*  Serial.print("A0: ");   Serial.print(P);  Serial.print(" ");
     Serial.print("A1: ");   Serial.print(I);  Serial.print(" ");
     Serial.print("A2: ");   Serial.print(M);  Serial.print(" ");
     Serial.print("A3: ");   Serial.print(A);  Serial.print(" ");
     Serial.print("A4: ");   Serial.print(Q);  Serial.print(" ");
     Serial.println(); */


////LETRAS Y NUMEROS DEL ADC
////Se procedio a calibrar cada una de las letras y numeros
dependiendo del valor de ADC, para ello
// se coloco un rango definido dependiendo de la posicion de la mano
derecha en el lenguaje dactilologico

if((P>215 && P<255) && (I>480 && I<510) && (M>620 && M<660) && (A>660
&& A<695) && (Q>620 && Q<660))
{letra='A'; lcd.setCursor(14, 1); lcd.print(letra);}

if((P>165 && P<195) && (I>610 && I<640) && (M>830 && M<860) && (A>815
&& A<845) && (Q>805 && Q<835))
{letra='B'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>205 && P<235) && (I>605 && I<635) && (M>745 && M<775) && (A>755
&& A<785) && (Q>770 && Q<800))
{letra='C'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>210 && P<240) && (I>540 && I<570) && (M>690 && M<720) && (A>700
&& A<730) && (Q>805 && Q<835))
{letra='D'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>160 && P<205) && (I>510 && I<530) && (M>615 && M<645) && (A>660
&& A<685) && (Q>645 && Q<675))
{letra='E'; lcd.setCursor(6, 1); lcd.print(letra);}
```

```
if((P>215 && P<245) && (I>625 && I<655) && (M>830 && M<860) && (A>815
&& A<845) && (Q>710 && Q<740))
{letra='F'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>220 && P<250) && (I>470 && I<500) && (M>655 && M<685) && (A>665
&& A<700) && (Q>790 && Q<810))
{letra='G'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>175 && P<205) && (I>500 && I<530) && (M>835 && M<865) && (A>685
&& A<715) && (Q>805 && Q<835))
{letra='H'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>190 && P<220) && (I>595 && I<625) && (M>660 && M<695) && (A>700
&& A<730) && (Q>645 && Q<675))
{letra='I'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>215 && P<235) && (I>595 && I<625) && (M>675 && M<705) && (A>715
&& A<745) && (Q>680 && Q<710))
{letra='J'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>200 && P<225) && (I>450 && I<475) && (M>820 && M<850) && (A>690
&& A<720) && (Q>800 && Q<835))
{letra='K'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>210 && P<240) && (I>515 && I<545) && (M>635 && M<665) && (A>680
&& A<710) && (Q>800 && Q<830))
{letra='L'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>160 && P<190) && (I>520 && I<550) && (M>710 && M<735) && (A>725
&& A<755) && (Q>705 && Q<735))
{letra='M'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>175 && P<185) && (I>510 && I<540) && (M>715 && M<755) && (A>675
&& A<705) && (Q>715 && Q<745))
{letra='N'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>215 && P<245) && (I>565 && I<595) && (M>720 && M<750) && (A>730
&& A<760) && (Q>715 && Q<745))
{letra='O'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>230 && P<260) && (I>430 && I<485) && (M>730 && M<776) && (A>690
&& A<720) && (Q>800 && Q<830))
{letra='P'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>225 && P<255) && (I>485 && I<515) && (M>660 && M<690) && (A>685
&& A<715) && (Q>790 && Q<820))
{letra='Q'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>180 && P<205) && (I>535 && I<565) && (M>815 && M<855) && (A>675
&& A<705) && (Q>800 && Q<830))
{letra='R'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>180 && P<210) && (I>520 && I<540) && (M>645 && M<675) && (A>685
&& A<715) && (Q>660 && Q<685))
{letra='S'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>175 && P<205) && (I>520 && I<550) && (M>650 && M<680) && (A>685
&& A<715) && (Q>725 && Q<755))
{letra='T'; lcd.setCursor(6, 1); lcd.print(letra);}
```

```
if((P>180 && P<210) && (I>510 && I<530) && (M>825 && M<855) && (A>690
&& A<720) && (Q>805 && Q<835))
{letra='U'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>190 && P<220) && (I>525 && I<555) && (M>835 && M<865) && (A>700
&& A<730) && (Q>805 && Q<845))
{letra='V'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>175 && P<205) && (I>550 && I<580) && (M>830 && M<860) && (A>815
&& A<845) && (Q>805 && Q<835))
{letra='W'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>190 && P<215) && (I>510 && I<540) && (M>660 && M<690) && (A>685
&& A<715) && (Q>745 && Q<775))
{letra='X'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>200 && P<235) && (I>595 && I<615) && (M>670 && M<700) && (A>700
&& A<730) && (Q>670 && Q<705))
{letra='Y'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>185 && P<215) && (I>525 && I<555) && (M>665 && M<695) && (A>685
&& A<715) && (Q>800 && Q<830))
{letra='Z'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>135 && P<165) && (I>465 && I<495) && (M>670 && M<700) && (A>680
&& A<710) && (Q>790 && Q<820))
{letra='1'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>140 && P<170) && (I>455 && I<490) && (M>835 && M<865) && (A>700
&& A<730) && (Q>795 && Q<825))
{letra='2'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>210 && P<240) && (I>535 && I<565) && (M>845 && M<875) && (A>810
&& A<840) && (Q>795 && Q<825))
{letra='3'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>180 && P<210) && (I>625 && I<655) && (M>835 && M<865) && (A>825
&& A<855) && (Q>800 && Q<830))
{letra='4'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>210 && P<240) && (I>625 && I<660) && (M>810 && M<840) && (A>810
&& A<840) && (Q>800 && Q<830))
{letra='5'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>210 && P<240) && (I>465 && I<495) && (M>645 && M<675) && (A>665
&& A<695) && (Q>645 && Q<675))
{letra='6'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>230 && P<260) && (I>485 && I<515) && (M>700 && M<730) && (A>690
&& A<720) && (Q>805 && Q<835))
{letra='7'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>220 && P<250) && (I>450 && I<480) && (M>845 && M<875) && (A>705
&& A<735) && (Q>805 && Q<835))
{letra='8'; lcd.setCursor(6, 1); lcd.print(letra);}

if((P>155 && P<190) && (I>500 && I<530) && (M>690 && M<720) && (A>690
&& A<720) && (Q>685 && Q<715))
{letra='9'; lcd.setCursor(6, 1); lcd.print(letra);}
```

```
/////PULSADOR ENVIA LA LETRA A LA LCD////////
if(digitalRead(pulsadorEnviar)==HIGH){estado1=1;}
 // Condicion si fue pulsado el boton, se escribe "1" en la variable
estado1

if(estado1==HIGH && digitalRead(pulsadorEnviar)==LOW){contaPulsos1++;
estado1=0;}  //condicion si estado1 esta en alto y ademas se ha dejado
de pulsar el mismo boton se incrementa en 1 la variable contapulsos


if(contaPulsos1==1){i++; lcd.setCursor(i, 0); lcd.print(letra);
mySerial.print(letra); if(i>=15){i=-1;} contaPulsos1=0;} //si
contrapulsos es igual a 1 , incrementamos la variable i , colocamos el
cursor e imprimimos letra
// se tiene un rango maximo de 15 posiciones para las traduccion

retorno=presionarBoton();       // Funcion retorno

if(digitalRead(pulsadorEspacio)==HIGH){estado3=1;}      // condicion
si fue pulsado el boton, se escribe 1 en estado3
if(estado3==HIGH && digitalRead(pulsadorEspacio)==LOW){contaPulsos3++;
estado3=0;}    //condicion si estado3 esta en 1 , se aumenta en 1
contapulsos 3

if(contaPulsos3==1){i++; lcd.setCursor(i, 0); lcd.print(" ");
mySerial.print(' '); contaPulsos3=0;} //si contapulsos es igual a 1 ,
se incrementa la variable i y se imprime un espacio
}

int presionarBoton(){       // empezamos una funcion int sin retorno
if (digitalRead(pulsadorBorrar) == HIGH) { //condicion si se presiono
el boton Borrar
if (buttonActive == false) {           // colocamos la variable en
falso buttonActive = true;               // colocamos la variable en
verdadero
buttonTimer = millis();}           // igualamos la funcion millis con
la variable buttontimer
if ((millis() - buttonTimer > longPressTime) && (longPressActive ==
false)) {longPressActive = true;} // Si la diferencia entre el
contador millis y el tiempo de presionado del botontimer es menor a
250 , ademas el boton longpressactive esta eb falso, colocamos en
verdadero la variable longPressActive
}else{
if (buttonActive == true) {    // si la variable buttonActive esta en
verdadero
if (longPressActive == true) {   //y si la variable longPresActive
esta en verdadero
mySerial.print('#'); // imprimimos el simbolo (#) en el puerto serial
lcd.clear();   // limpiamos el lcd
lcd.setCursor(0, 1); lcd.print("Letra/Numero: ");    // imprimimos en
el lcd
i=-1;    // restamos 1 a la posicion del lcd
longPressActive = false;     // volvemos a colocar la variable en
falso
}else{

lcd.setCursor(i, 0); lcd.print(" ");  i--;     // imprimimos en
espacio
         }
```

```
buttonActive = false; // volvemos la variable a el estado falso
    }
return conta;   //retornamos
  }
}
```

ANEXO 5: PROGRAMACIÓN NODEMCU

```
#include <ESP8266wifi.h> // incluimos libreria del modulo esp 8266
#include <FirebaseESP8266.h>  // incluimos librerias de vinculacion
del esp8266 con firebase

#define wifi_SSID "Repetidor_Mbytesoluciones"  //Nombre dela RED wifi
#define wifi_PASSWORD "Mbytesol2020"   //Contraseña de la RED wifi
#define FIREBASE_HOST "guante-traductor-default-
rtdb.firebaseio.com"  //Database firebase
#define FIREBASE_AUTH
"RFJIbzgqsqC17QHARPaYENbaPvNc9YdaU4Nb6qrK"    //Contraseña de
seguridad firebase

FirebaseData fbdo; //

char recibir; // variable tipo cracter
String palabra="";    // variable tipo cadena de caracteres

void setup(){
     Serial.begin(9600);   delay(50);          //Inicializa puerto
serial
     wifi.begin(wifi_SSID, wifi_PASSWORD);     //Busca conectarse a la
red wifi
     //Serial.print("Connecting to Wi-Fi");      //Impresion texto
CONECTANDOSE
     while (wifi.status() != WL_CONNECTED){    //Conectando
     //Serial.print(".");
     delay(300);
     }
     //Serial.println();
     //Serial.print("Connected with IP: ");
     //Serial.println(wifi.localIP());           //Impresion
direccion ip de la red wifi
     //Serial.println();

     Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); //Autenticacion de
firebase de host y contraseña
     Firebase.reconnectwifi(true);

     //wifi RX/TX Data
     fbdo.setBSSLBufferSize(1024, 1024);
     //HTTP
     fbdo.setResponseSize(1024);
     //Tiempo maximo de lectura 15 minutos
     Firebase.setReadTimeout(fbdo, 1000 * 60);
     Firebase.setwriteSizeLimit(fbdo, "tiny");
     Firebase.setFloatDigits(2);
     Firebase.setDoubleDigits(6);
}

void loop() {
if(Serial.available()){    // verificamos si existe algun dato en el
puerto serial
recibir=(char)Serial.read();     // almacenamos en la variable el
caracter recibido en el puerto serial
palabra+=recibir;   //aumentamos un caracter a la variable
```

```
Firebase.setString(fbdo, "palabra", "'"+palabra+"'"); // enviamos por
el puerto serial el caracter
//Serial.println(palabra);
if(recibir=='#'){palabra=""; Firebase.setString(fbdo, "palabra",
palabra); } //Condicion si recibimos el simbolo # por el puerto serial
, imprimimos un espacio(vaciamos el firebase)
     }
}
```