Guante Traductor de Señas

Mauricio Orlando Gómez. Instituto de Investigación e Innovación en electrónica de la Universidad Don Bosco

Mauricio.gomez@udb.edu.sv

***Abstract*— One of the bets of the Research at the Don Bosco University, is precisely the social projection, in this framework a small investigation is presented on what is the use of technology for the implementation of a sign language translator. Specifically, the results of the implementation of a resistive glove are presented, which will translate the movements of the hand into language understandable to a normal person and that this information is processed in an Arduino, sending this data to a serial monitor and that you can see the translation of what the person who has in the glove wants to express.**

***Abstracto*— Una de las apuestas de la Investigación en la Universidad Don Bosco, es justamente la proyección social, en ese marco se presenta una pequeña investigación sobre lo que es la utilización de la tecnología para la implementación de un traductor del lenguaje de señas. Concretamente, se presentan los resultados de la implementación de un guante resistivo, el cual traducirá los movimientos de la mano al lenguaje entendible para una persona normal y que dicha información sea procesada en un Arduino, mandando este los datos a un monitor serial y que se pueda ver la traducción de lo que la persona que tiene en guante quiere expresar.**

***Index terms* — sign language, translator, Arduino, variable resistance, flexible resistance, glove, alphabet, data, variables, conditions, circuit, divisor, voltage, microcontroller.**

***Índice de términos* — Señas, lenguaje, traductor, Arduino, Resistencia variable, Resistencia flexible, guante, alfabeto, datos, variables, condiciones, circuito, divisor, voltaje, microcontrolador.**

I. INTRODUCCIÓN

En el mundo luego del covid que nos encerró, las formas de comunicación están cambiando y es necesario hacer las integraciones entre los diferentes grupos poblacionales; en ese sentido, se presentan los resultados de un estudio desarrollado en la Universidad Don Bosco, donde se trabajó un esquema que permitirá entrenar a personas en el uso del lenguaje de señas para que se pueda comunicar con personas que poseen discapacidad auditiva, durante el proceso de realización se han considerados los trabajos que hay en este campo y se han tomado como referencias los trabajos de Duque Áreas e Ibarra Caicedo [1] ya que considera elementos de bajo costo para el proceso de trabajo, adicional a esto y en el apartado II se presentan los materiales que se han contemplado para este desarrollo. En el apartado III veremos las pruebas y ensayos que se realizaron, En el apartado IV quedan las conclusiones y las recomendaciones que se han logrado de la presente; de hecho, hay algunas cosas que se pueden retomar para mejorar lo que es el trabajo y llevarlo a su siguiente estadio.

Objetivos

- Investigar la realización de un guante traductor guante de señas utilizando materiales de bajo costo que se encuentren en el Mercado nacional actual.

- Utilizar nuevas tecnologías de uso común, Smartphones, para facilitar la integración de las personas con discapacidad auditiva pero que pueden ver en la sociedad, por medio de una app para la traducción del lenguaje de señas.

A. ¿Qué es el lenguaje de señas?

Al menos 88 mil personas en El Salvador sufren de alguna discapacidad auditiva esto de acuerdo a los datos estadísticos del Consejo Nacional de Atención Integral a la persona con Discapacidad (CONAIPD) del año 2015, por lo que utilizan el lenguaje de señas para comunicarse con otros, sin necesidad de usar el recurso verbal logran expresarse a través de sus manos.

La mayoría de la población desconoce que la lengua de señas como todo proceso de comunicación cuenta con códigos, sintaxis y un conjunto de reglas que hacen de esta expresión corporal todo un arte comunicacional, no solo para la comunidad sorda, sino también para sus familiares y amigos.

La lengua de señas es la lengua natural de las personas sordas. Se basa en movimientos y expresiones a través de las manos, los ojos, el rostro, la boca y el cuerpo. Muchos sordos se comunican con esta lengua y requieren de un intérprete o persona que la maneje para relacionarse con oyentes que no la conocen.

Fig. 1. Representación de la lengua de señas en uso.

II. MATERIALES

En esta sección se presentan la descripción y uso de cada

uno de los elementos utilizados para la realización del proyecto, junto con las diversas etapas que se tuvieron del mismo, los errores y cambios involucrados para las pruebas en las diferentes etapas.

A. Resistencia flexible

El Sensor Flex produce una resistencia variable en función del grado al que esté doblada.
Convierte la curvatura en distintos valores de resistencia eléctrica. Son por lo general en la forma de una delgada tira de 5 cm de largo que varía en resistencia de aproximadamente 10 a 50 K ohm.

Características

- Tolerancia de la Resistencia: ± 30%.
- Potencia nominal: 0,50 Volts continuos.
- La resistencia al no estar doblada es: 25K Ohms.
- Rango de la curva de la resistencia: 45K a 125K Ohms (dependiendo del radio de curvatura).
- Altura: 0.43 mm (0.017 ").
- Rango de temperatura: -35 ° C a +80 ° C.
- Voltaje: 5 a 12 Volts.

Fig. 2. Sensor Flex de la compañía Rambal Ltda.

¿Cómo funciona?

Los Sensores Flex son resistencias analógicas. Trabajan como divisores de tensión analógica variable.

Dentro de la flexión del sensor son elementos resistivos de carbono dentro de un sustrato flexible y delgado. (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 40K ohmios.

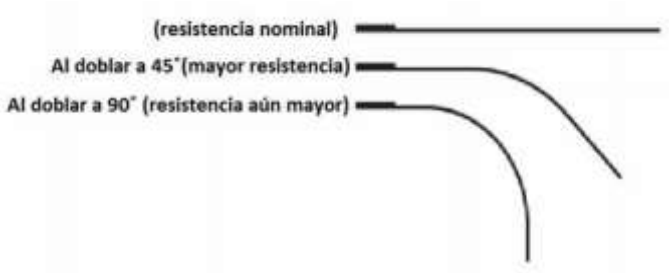


Fig. 3. Funcionamiento general del sensor flex de Rambal Ltda.

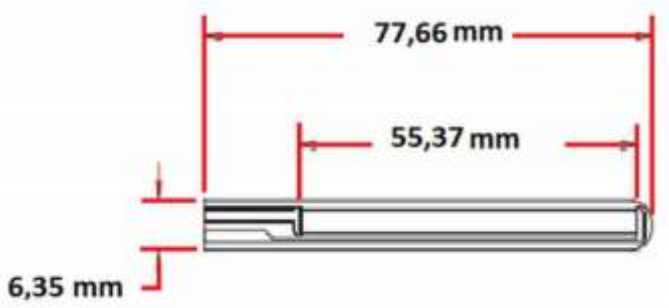


Fig. 4. Dimensiones promedio de un sensor flex de la marca Rambal Ltda.

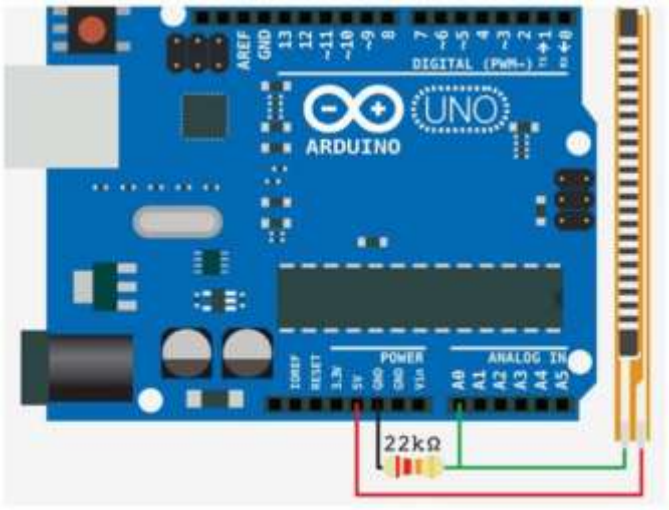


Fig. 5. Ejemplo de conexión de sensor con Arduino uno

El sensor flex cambia su resistencia cuando se flexiona para que podamos medir ese cambio usando uno de los pines analógicos de algún microcontrolador como Arduino. Pero para hacer eso necesitamos una resistencia fija de 22 k Ohm. Esto se llama divisor de tensión ya que al estar en serie divide la tensión de la fuente de voltaje de 5v, entre el sensor de flexión y la resistencia. El Pin análogo del microcontrolador puede leer el voltaje, se trata entonces de un medidor de voltaje. Así podemos medir la cantidad de tensión en el sensor.

Cabe destacar que no todas las resistencias se comportan de la misma manera y se tienen cambios en el valor inicial al estar estiradas, además que la posición inicial en la mano estás dan

un valor diferente para cada uno de los sensores utilizados.

Otras formas de conexiones no directas al microcontrolador:

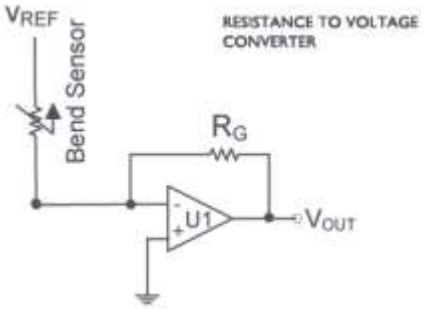


Fig. 6. Conversor de resistencia a voltaje con amp op.

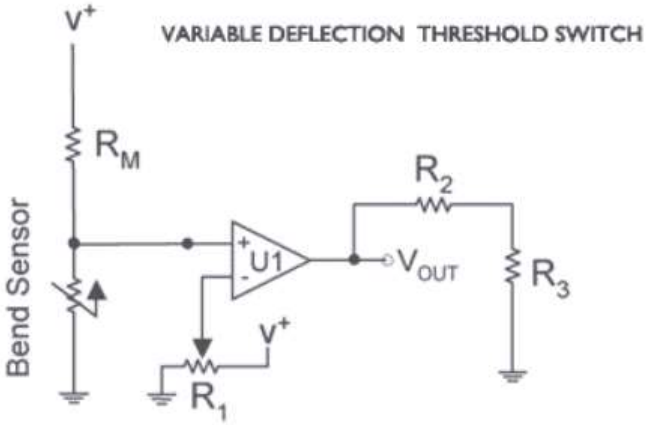


Fig. 7. Detector de estado por medio de divisor de voltaje y amp op.

Otra variación del sensor de flexibilidad es el de la compañía spectra symbol:

Fig 8. Flex sensor Spectra Symbol de 55.37 mm

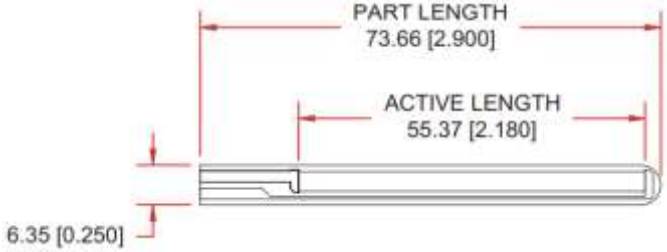


Fig. 9. Dimensiones Flex Sesor de Spectra Symbol.

Características:

- Resistencia sin doblar: 25kOhms
- Tolerancia de la Resistencia: ±30%
- Rango de Resistencia al doblarse: 45K a 125K Ohm (dependiendo del radio)
- Rango de consumo: 0.5 Watts continuos, 1Wmax.
- Rango de temperatura: -35° a 80°C
- Ciclo de vida: >1 millón

B. Arduino

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable y una serie de pines de conexión, los que permiten establecer el dialogo entre el microcontrolador y los diferentes sensores y actuadores de una manera muy práctica.

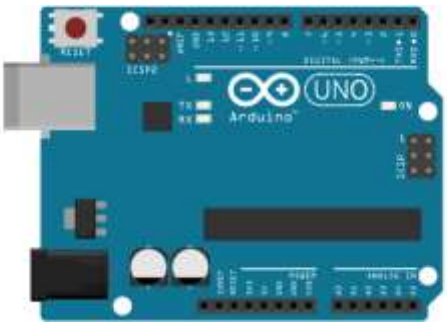


Fig. 10. Representación de vector de placa Arduino Uno.

Una PCB es la forma más compacta y estable de construir un circuito electrónico. Así que la placa Arduino no es más que una PCB que implementa un determinado diseño de circuitería interna, de esta forma el usuario final no se debe preocupar por las conexiones eléctricas que necesita el microcontrolador para funcionar, y puede empezar directamente a desarrollar las diferentes aplicaciones electrónicas que necesite.

Cuando hablamos de “Arduino” deberíamos especificar el modelo concreto, ya que se han fabricado diferentes modelos de placas Arduino oficiales, cada una pensada con un propósito diferente y características variadas (como el tamaño físico, número de pines E/S, modelo del microcontrolador, etc.). A pesar de las varias placas que existen todas pertenecen a la misma familia (microcontroladores AVR marca Atmel), esto significa que comparten la mayoría de sus características de software, como arquitectura, librerías y documentación.

Fig. 11. Diversos modelos de la familia Arduino Atmel.

¿Por qué usar Arduino?

Arduino es libre y extensible: esto quiere decir que cualquiera que desee ampliar y mejorar el diseño hardware de las placas como el entorno de desarrollo, puede hacerlo sin problemas. Esto permite que exista un rico ecosistema de placas electrónicas no oficiales para distintos propósitos y de librerías de software de tercero, que pueden adaptarse mejor a nuestras necesidades.

- Su entorno de programación es multiplataforma: Se puede instalar y ejecutar en sistemas operativos Windows, Mac OS y Linux.

- Lenguaje de programación de fácil compresión: Su lenguaje de programación basado en C++ es de fácil compresión que permite una entrada sencilla a los nuevos programadores y a la vez con una capacidad tan grande, que los programadores más avanzados pueden exprimir todo el potencial de su lenguaje y adaptarlo a cualquier situación.

- De bajo costo: La placa Arduino estándar (Arduino UNO) tiene un valor aproximado de $12.00 dólares, incluso uno mismo la podría construir (una gran ventaja del hardware libre), con lo que el precio de la placa seria incluso menor.

- Re-usabilidad y versatilidad: Es re-utilizable porque una vez terminado el proyecto es muy fácil poder desmontar los componentes externos a la placa y empezar con un nuevo proyecto.

Modelo del microcontrolador

El microcontrolador que lleva la placa Arduino UNO es el modelo ATmega328P de la marca Atmel. La «P» del final significa que este chip incorpora la tecnología «Picopower» (propietaria de Atmel), la cual permite un consumo eléctrico ligeramente menor comparándolo con el modelo equivalente sin «Picopower», ATmega328 (sin la «P»). Aunque el ATmega328P pueda trabajar a un voltaje menor y consumir menos corriente que el ATmega328, ambos modelos son funcionalmente idénticos, es decir, pueden ser remplazados el uno por el otro.

Fig. 12. Modelo de microcontrolador de la placa Arduino Uno.

Al igual que ocurre con el resto de microcontroladores usados en otras placas Arduino, el ATmega328P tiene una arquitectura de tipo AVR, arquitectura desarrollada por Atmel y en cierta medida «competencia» de otras arquitecturas como por ejemplo el PIC del fabricante Microchip. Más concretamente, el ATmega328P pertenece a la subfamilia de microcontroladores megaAVR. Otras subfamilias de la arquitectura AVR son la tinyAVR (cuyos microcontroladores son más limitados y se identifica con el nombre ATtiny) y la XMEGA (cuyos microcontroladores son más capaces y se identifican con el nombre de ATxmega).

El software de Arduino es un IDE, entorno de desarrollo integrado (siglas en inglés de Integrated Development Environment). Es un programa informático compuesto por un conjunto de herramientas de programación.

El IDE de Arduino es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Además, incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware.

Fig. 13. Entorno de programa de programación, compilación y visualización de Arduino.

C. Circuito Divisor de voltaje

Un divisor de voltaje requiere que se conecte una fuente de voltaje a través de dos resistencias en serie. Es posible que el divisor de voltaje sea dibujado de distintas maneras, pero siempre debe ser esencialmente el mismo circuito.

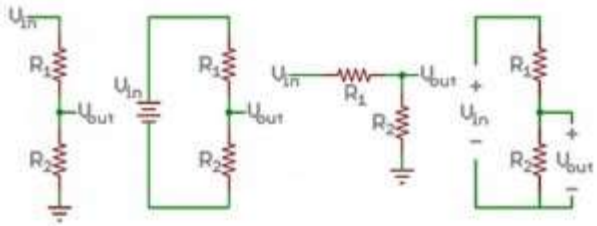


Fig 14. Representaciones del circuito divisor de voltaje.

Llamamos a la resistencia más cercana al voltaje de entrada (Vin) R1 y a la resistencia más cercana a tierra R2. La caída de voltaje en R2 es nuestro voltaje de salida (Vout), este es el voltaje resultante de nuestro circuito, que como ya se mencionó es una fracción de nuestro voltaje de entrada.

La ecuación del divisor de voltaje supone que se conocen tres valores del circuito anterior: el voltaje de entrada (Vin), y ambos valores de resistencia (R1 y R2). Teniendo en cuenta estos valores, podemos usar esta ecuación para encontrar el voltaje de salida (Vout):

$$Vout = Vin * \frac{R2}{R1 + R2} \quad (1)$$

Esta ecuación establece que el voltaje de salida es directamente proporcional al voltaje de entrada conforme a la relación de R1 y R2.

D. Modulo Bluetooth HC-06

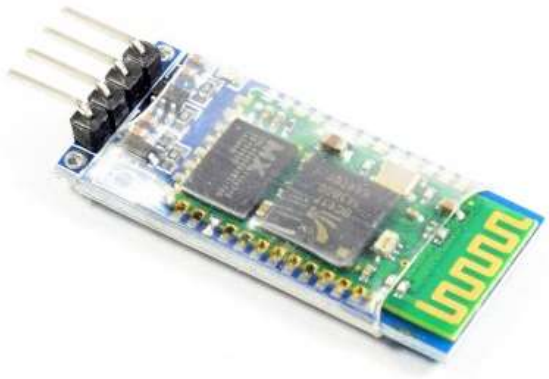


Fig 15. Modulo bluetooth HC-06

Parámetros que vienen por defecto se muestran a continuación:

- Nombre por defecto: “linvor” o “HC-06”
- Código de emparejamiento por defecto: 1234
- La velocidad por defecto (baud rate): 9600

El Módulo HC-06 viene configurado de fábrica como "Esclavo" (Slave) y no puede ser cambiado a "Maestro".

- Vcc, Voltaje positivo de alimentación, trabaja en el rango de 3.3V a 6V.

- GND, Voltaje negativo de alimentación, se debe que conectar al GND del Arduino o al GND de la placa que se esté usando.

- TX, Pin de Transmisión de datos, por este pin el HC-06 transmite los datos que le llegan desde la PC o Móvil mediante bluetooth, este pin debe ir conectado al pin RX del Arduino

- RX, pin de Recepción, a través de este pin el HC-06 recibirá los datos del Arduino los cuales se transmitirán por Bluetooth, este pin va conectado al Pin TX del Arduino.

El Módulo Bluetooth HC-06 tiene dos estados de funcionamiento los cuales es importante conocer:

E1. Modo AT (Desconectado):

- Entra a este modo tan pronto alimentamos el modulo, y cuando no se ha establecido una conexión bluetooth con ningún otro dispositivo…
- EL LED del módulo está parpadeando (frecuencia de parpadeo del LED es de 102ms).
- En este modo es cuando se debe enviar los comandos AT en caso se quiera configurar algún parámetro, si se envían otros datos diferentes a los comandos AT, el HC-06 los ignorará.

```
String letra;
void leer_entradas(void);
void enviar_letra(void);
int sensor[8]={0,0,0,0,0,0,0,0};

void setup() {
  // put your setup code here, to run once:
  pinMode(2,INPUT);
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(5,INPUT);
  pinMode(6,INPUT);
  pinMode(7,INPUT);
  pinMode(8,INPUT);
  pinMode(9,INPUT);
  Serial.begin(9600);//115200 para BT
}
void loop() {
  // put your main code here, to run repeatedly:
  leer_entradas();
  enviar_letra();
}
void leer_entradas(){
  sensor[0]=digitalRead(2);
  sensor[1]=digitalRead(3);
  sensor[2]=digitalRead(4);
  sensor[3]=digitalRead(5);
  sensor[4]=digitalRead(6);
  sensor[5]=digitalRead(7);
  sensor[6]=digitalRead(8);
  sensor[7]=digitalRead(9);

if
((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==0)&&(
sensor[3]==0)&&(sensor[4]==0)&&(sensor[5]==0)&&(s
ensor[6]==0)&&(sensor[7]==0))
  letra=" ";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==1)&
&(sensor[3]==1)&&(sensor[4]==0)&&(sensor[5]==0)&
&(sensor[6]==0)&&(sensor[7]==0))
  letra="a";
  else
if((sensor[0]==1)&&(sensor[1]==0)&&(sensor[2]==0)&
&(sensor[3]==0)&&(sensor[4]==0)&&(sensor[5]==0)&
&(sensor[6]==0)&&(sensor[7]==0))
  letra="b";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==1)&
&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5]==0)&
&(sensor[6]==0)&&(sensor[7]==0))
  letra="c";
  else
if((sensor[0]==1)&&(sensor[1]==0)&&(sensor[2]==1)&
&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5]==0)&
&(sensor[6]==0)&&(sensor[7]==0))
  letra="d";
  else
```

E2. Modo Conectado:

- Entra a este modo cunado se establece una conexión con otro dispositivo bluetooth.
- El LED permanece encendido sin parpadear.
- Todos los datos que se ingresen al HC-06 por el Pin RX se trasmiten por bluetooth al dispositivo conectado, y los datos recibidos se devuelven por el pin TX. La comunicación es transparente para el programador.
- En este Modo el HC-06 no puede interpretar los comandos AT.

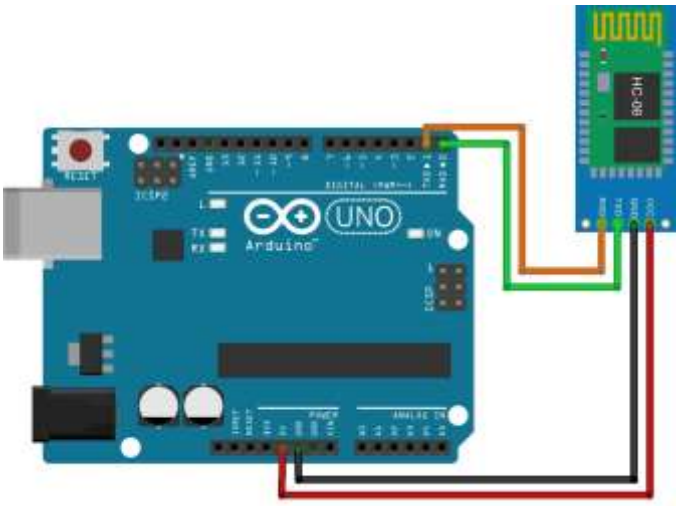


Fig. 16. Conexión del módulo con Arduino Uno.

Para cargar el programa al Arduino, desconectaremos los pines RX0 y TX0 del Arduino, pues internamente el Arduino trabaja con los mismos pines para cargar el programa y si están conectados al módulo Bluetooth, no nos dejara cargar (para evitar este inconveniente se puede usar el software serial y usar otros pines).

Las velocidades de conexión del módulo son: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 Baudios.

III. PROCESO Y PRUEBAS

A. Selección del alfabeto

Debido a las variaciones que sufre el alfabeto de señas según regiones y expresiones que se dan en estas, se llevó a cabo la selección de uno que tuviera las expresiones más generalizadas para las letras en nuestra región, y presentara un menor grado de complejidad.

Fig. 17. Alfabeto de señas seleccionado para las pruebas.

B. Prueba de lógica binaria

Las primeras pruebas se realizaron conforme la utilización de una lógica binaria, es decir no se tenían posiciones intermedias de flexión de los dedos.

Para empezar, se hizo una tabla con todos los valores posibles para las letras en función de la flexión siendo un 1 y la no flexión un cero.

	MEÑIQUE	ANULAR	MEDIO	INDICE	PULGAR
A	1	1	1	1	0
B	0	0	0	0	1
C	1	1	1	1	1
D	1	1	1	0	1
E	1	1	1	1	1
F	0	0	0	1	0
G	1	1	1	0	0
H	1	1	0	0	0
I	0	1	1	1	1
J	0	1	1	1	1
K	1	1	0	0	0
L	1	1	1	0	0
M	1	1	1	1	1
N	1	1	1	1	1
Ñ	1	1	1	1	1
O	1	1	1	1	1
P	1	1	0	0	0
Q	1	1	1	0	0
R	1	1	0	0	1
S	1	1	1	1	1
T	1	1	1	1	0
U	1	1	0	0	1
V	1	1	0	0	1
W	1	0	0	0	1
X	1	1	1	1	0
Y	0	1	1	1	0
Z	1	1	1	0	1

Tabla1. Códigos binarios para una resistencia por dedo utilizando el alfabeto de la Fig.17

Se hizo una comparación de todas las letras con códigos iguales para establecer las que darían problemas al momento de realizarse en el guante junto con el código.

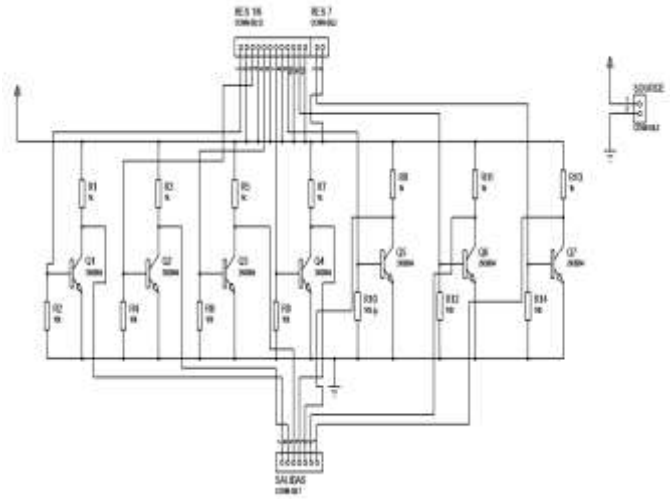


Fig. 18. Esquemático de switch de voltaje para prueba binaria.

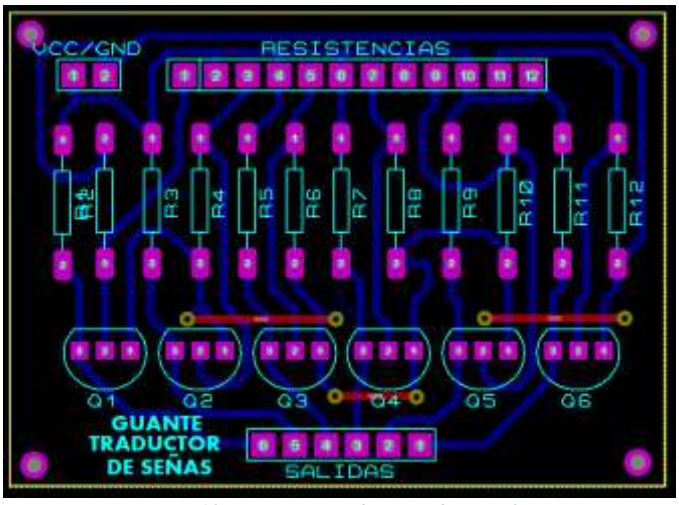


Fig. 19. Generación de PCB de switch.

Fig 20. Circuito realizado para prueba de binario.

Debido al parecido de las letras estas no dejaban declarar bien la variable de tipo carácter en Arduino por lo tanto se estableció poner dos datos más para poder generar diferentes códigos así también como uno para la muñeca.

Se probaron las formas de realizar los códigos y se obtuvo la siguiente tabla de comparación de datos:

Letra	**B2**	**B1**	**Muñeca**	**Pulgar**	**Índice**	**Medio**	**Anular**	**Meñique**
A	0	0	0	0	1	1	1	1
B	0	0	0	0	0	0	0	1
C	0	0	0	1	1	1	1	1
D	0	0	0	1	1	1	0	1
E	1	1	0	1	1	1	1	1
F	0	0	0	0	0	0	1	0
G	0	1	0	0	0	1	1	1
H	1	0	0	1	1	0	0	0
I	0	0	0	1	1	1	1	0
J	0	0	1	1	1	1	1	0
K	0	0	0	0	0	0	1	1
L	0	0	0	0	0	1	1	1
M	0	1	0	1	1	1	1	1
N	1	0	0	1	1	1	1	1
Ñ	1	0	1	1	1	1	1	1
O	0	0	1	1	1	1	1	1
P	0	0	1	0	0	0	1	1
Q	0	0	1	0	0	1	1	1
R	0	0	0	1	0	0	1	1
S	1	1	1	1	1	1	1	1
T	1	0	0	0	1	1	1	1
U	0	1	0	1	0	0	1	1
V	1	0	0	1	0	0	1	1
W	0	0	0	1	0	0	0	1
X	0	1	0	0	1	1	1	1
Y	0	0	0	0	1	1	1	0
Z	0	0	1	1	0	1	1	1

Tabla2. Corrección para forma binaria de traducción de letras.

Finalmente se descartó esta manera de realizar el proyecto debido al gran cambio que se realizaba en la forma de realizar la letra y le quitaba naturalidad al movimiento, así como también de hacerlo más difícil y por ende más lento, ya que se debía agregar dos botones que debían ser presionados según la letra a expresar.

```
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==0)&&(sensor[6]==1)&&(sensor[7]==1))
  letra="e";
  else
if((sensor[0]==0)&&(sensor[1]==1)&&(sensor[2]==
0)&&(sensor[3]==0)&&(sensor[4]==0)&&(sensor[5
]==0)&&(sensor[6]==0)&&(sensor[7]==0))
  letra="f";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==0)&&(sensor[4]==0)&&(sensor[5
]==0)&&(sensor[6]==1)&&(sensor[7]==0))
  letra="g";
  else
if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==
0)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==0)&&(sensor[6]==0)&&(sensor[7]==1))
  letra="h";
  else
if((sensor[0]==0)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==0)&&(sensor[6]==0)&&(sensor[7]==0))
  letra="i";
  else
if((sensor[0]==0)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==1)&&(sensor[6]==0)&&(sensor[7]==0))
  letra="j";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
0)&&(sensor[3]==0)&&(sensor[4]==0)&&(sensor[5
]==0)&&(sensor[6]==0)&&(sensor[7]==0))
  letra="k";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==0)&&(sensor[4]==0)&&(sensor[5
]==0)&&(sensor[6]==0)&&(sensor[7]==0))
  letra="l";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==0)&&(sensor[6]==1)&&(sensor[7]==0))
  letra="m";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==0)&&(sensor[6]==0)&&(sensor[7]==1))
  letra="n";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==1)&&(sensor[6]==0)&&(sensor[7]==1))
  letra="ñ";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
1)&&(sensor[3]==1)&&(sensor[4]==1)&&(sensor[5
]==1)&&(sensor[6]==0)&&(sensor[7]==0))
  letra="o";
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==
0)&&(sensor[3]==0)&&(sensor[4]==0)&&(sensor[5
]==1)&&(sensor[6]==0)&&(sensor[7]==0))
  letra="p";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==1)&&(sensor[3]==0)&&(sensor[4]==0)&
&(sensor[5]==1)&&(sensor[6]==0)&&(sensor
[7]==0))
  letra="q";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==0)&&(sensor[3]==0)&&(sensor[4]==1)&
&(sensor[5]==0)&&(sensor[6]==0)&&(sensor
[7]==0))
  letra="r";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==1)&&(sensor[3]==1)&&(sensor[4]==1)&
&(sensor[5]==1)&&(sensor[6]==1)&&(sensor
[7]==1))
  letra="s";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==1)&&(sensor[3]==1)&&(sensor[4]==0)&
&(sensor[5]==0)&&(sensor[6]==0)&&(sensor
[7]==1))
  letra="t";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==0)&&(sensor[3]==0)&&(sensor[4]==1)&
&(sensor[5]==0)&&(sensor[6]==1)&&(sensor
[7]==0))
  letra="u";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==0)&&(sensor[3]==0)&&(sensor[4]==1)&
&(sensor[5]==0)&&(sensor[6]==0)&&(sensor
[7]==1))
  letra="v";
  else
if((sensor[0]==1)&&(sensor[1]==0)&&(sensor
[2]==0)&&(sensor[3]==0)&&(sensor[4]==1)&
&(sensor[5]==0)&&(sensor[6]==0)&&(sensor
[7]==0))
  letra="w";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==1)&&(sensor[3]==1)&&(sensor[4]==0)&
&(sensor[5]==0)&&(sensor[6]==1)&&(sensor
[7]==0))
  letra="x";
  else
if((sensor[0]==0)&&(sensor[1]==1)&&(sensor
[2]==1)&&(sensor[3]==1)&&(sensor[4]==0)&
&(sensor[5]==0)&&(sensor[6]==0)&&(sensor
[7]==0))
  letra="y";
  else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor
[2]==1)&&(sensor[3]==0)&&(sensor[4]==1)&
&(sensor[5]==1)&&(sensor[6]==0)&&(sensor
[7]==0))
  letra="z";
}
void enviar_letra(){
  Serial.print(letra);
  delay (2000);
```

C. Prueba de lectura de voltaje

Para esta prueba se verificaron los voltajes que generaba cada dedo según el divisor de voltaje establecido por la resistencia flexible y una resistencia de 10kOhms, para un voltaje de alimentación de 5v.

Se leyó por medio del siguiente código de Arduino el valor de cada dedo:

```
int val=0;
int val1=0;
int val2=0;
int val3=0;
int val4=0;
int analogPin = A0;
Int analogPin1 = A1;
int analogPin2 = A2;
int analogPin3 = A3;
int analogPin4 = A4;
int number = 0;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
val = analogRead(analogPin);
val1 = analogRead(analogPin1);
val2 = analogRead(analogPin2);
val3 = analogRead(analogPin3);
val4 = analogRead(analogPin4);
Serial.println(val);
delay(2000);
}
```

También se realizó una prueba con las resistencias totales como conjunto para junto con una resistencia variable para ajustar el valor de voltaje obtenido.

Ambas pruebas fueron infructuosas debido a los bajos cambios de voltaje que se presentaban las cuales no superaban ni el 1.2v entre letras, y que para letras con formas similares de expresión era demasiado bajo del rango de 0.1V de cambio por lo que se abandonó esta idea.

Si bien se realizaron circuitos que luego fueron desechados para el reciclaje de piezas en circuitos posteriores.

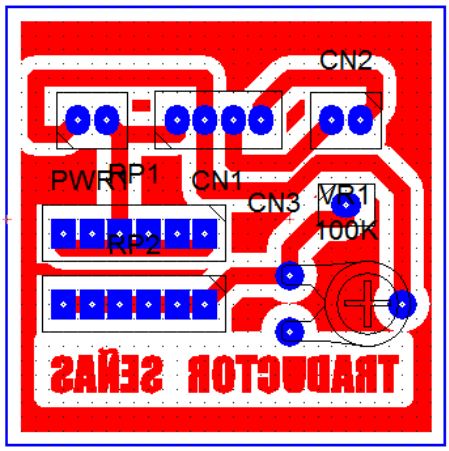


Fig 21. PCB para prueba de voltaje conjunto.

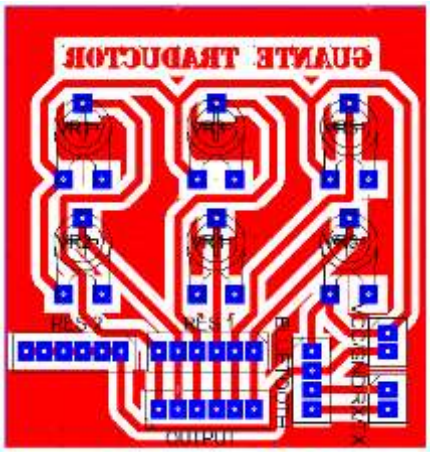


Fig. 22. PCB para prueba de voltajes individuales con resistencias variables para calibración.

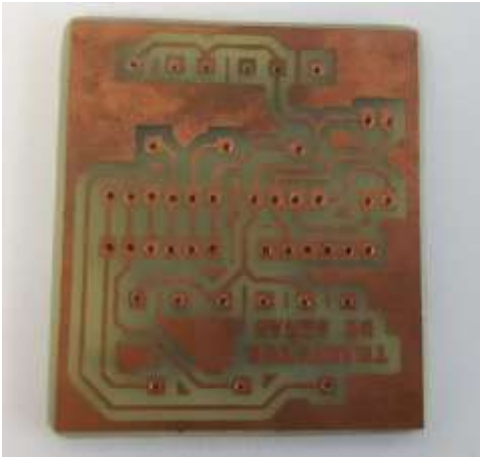


Fig. 23. Realización de pistas para circuito de pruebas de voltaje.

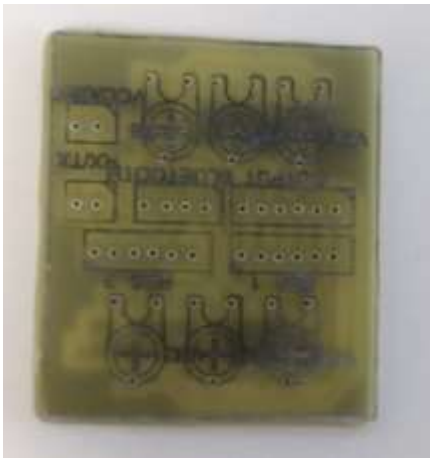


Fig. 24. Serigrafía de circuito para prueba de voltaje.

D. Prueba de mapeo de datos de ADC de arduino

Se establecieron datos de mapeo y restricción de los límites que se podían obtener por cada uno de los dedos, utilizando el ADC integrado de arduino.

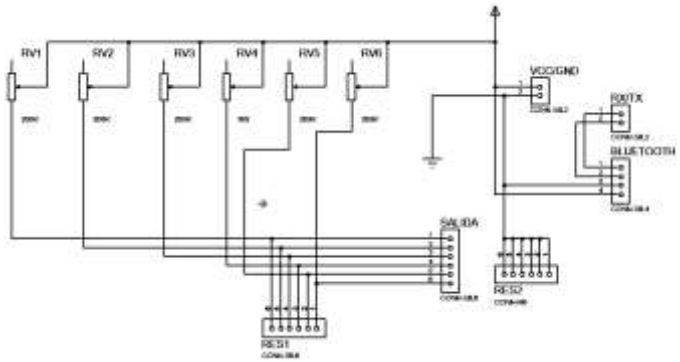


Fig. 25. Circuito de divisor para mapeo con trimmer.

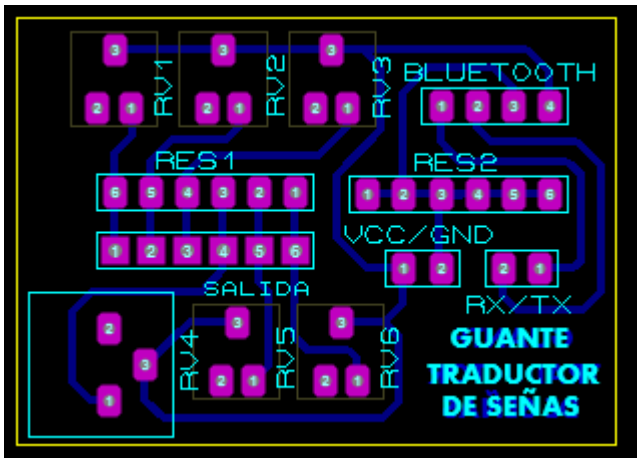


Fig 26. PCB de circuito divisor para mapeo.

Fig. 27. Circuito terminado para pruebas de mapeo.

Se establecieron los rangos a los que se llegaba cada dedo en una prueba de flexión total, con lo que se restringió en el mapeo el valor de 1 a 3, donde estos intervalos estarían ligados al valor con el que se flexionaban en cada letra, modificando los rangos según necesidad en la prueba de realización de cada letra.

```
String letra;
int pulgar = 0;
int indice = 0;
int medio = 0;
int anular = 0;
int menique = 0;
int muneca = 0;
int mpulgar = 0;
int mindice = 0;
int mmedio = 0;
int manular = 0;
int mmenique = 0;
int mmuneca = 0;
void setup () {
  // put your setup code here, to run once:
  Serial.begin(9600);
}
void loop () {
  // put your main code here, to run repeatedly:
  pulgar = analogRead(A0);
  pulgar = constrain(pulgar, 100, 141);
//restringimiento de los valores mínimos y máximos
de lectura de voltaje
  mpulgar = map(pulgar, 100, 141, 0, 2); //95-165
//lectura dedo indice
  indice = analogRead(A1);
  indice = constrain(indice,210, 423);
  mindice = map(indice, 210, 423, 0, 2); //225 -465
//lectura dedo medio
  medio = analogRead(A2);
  medio = constrain(medio, 224, 437);
  mmedio = map(medio, 224, 437, 0, 2);//225-
470/490
```

```
//lectura dedo anular
  anular = analogRead(A3);
  anular = constrain(anular, 240, 366);
  manular = map(anular, 240, 366, 0, 2); //260-
445/460
  //lectura dedo meñique
  menique = analogRead(A4);
  menique = constrain(menique, 264, 305);
  mmenique = map(menique, 264, 305, 0, 2); //270-
420/440
  //lectura muñeca
  muneca = analogRead(A5);
  muneca = constrain(muneca, 220, 315); //bueno
  mmuneca = map(muneca, 220, 315, 0, 2);
  Serial.print("pulgar:");
  Serial.println(mpulgar);
  Serial.println(pulgar);
  Serial.print("indice:");
  Serial.println(mindice);
  Serial.println(indice);
  Serial.print("medio:");
  Serial.println(mmedio);
  Serial.println(medio);
  Serial.print("anular:");
  Serial.println(manular);
  Serial.println(anular);
  Serial.print("menique:");
  Serial.println(mmenique);
  Serial.println(menique);
  Serial.println(" ");
  Delay (2500);
 }
```

Se trataron de establecer los rangos para cada letra, pero se llegó a un punto donde todas las calibraciones realizadas para las letras se debían de realizar con cada vez que se ponían, debido a que estas cambiaban los límites y quedaba ya fuera un límite inferior o superior del mapeo (1 o 3) anclado en la lectura.

Se intentaron hacer otras re-calibraciones siguiendo este mismo método, pero se llegó al mismo punto dentro de

```
//definición de las variables de valor entero para cada
dedo, para captura del valor del voltaje real y el valor de
mapeo
String letra;
void leer_entradas(void);
void establecer_letra(void);
void enviar_letra(void);
int pulgar = 0;
int indice = 0;
int medio = 0;
int anular = 0;
int menique = 0;
int muneca = 0;
int mpulgar = 0;
int mindice = 0;
int mmedio = 0;
int manular = 0;
```

```
int mmenique = 0;
int mmuneca = 0;
//Arranque y establecimiento de velocidad de transmisión
de la comunicación serial
void setup () {
Serial.begin(9600);
}
void loop () {
  leer_entradas();
  establecer_letra();
  enviar_letra();
   }
void leer_entradas() {
//lectura dedo pulgar
pulgar = analogRead(A0);
pulgar = constrain (pulgar, 210,245); //restringimiento de
los valores mínimos y máximos de lectura de voltaje
mpulgar = map (pulgar, 209,243,1,3);
//lectura dedo índice
indice = analogRead(A1);
indice = constrain (índice, 300,465);
mindice = map (índice, 300,465,1,3);
//lectura dedo medio
medio = analogRead(A2);
medio = constrain (medio, 270,490);
mmedio = map (medio, 270,490,1,3);
//lectura dedo anular
anular = analogRead(A3);
anular = constrain(anular,310,510);
manular = map(anular,305,505,1,3);
//lectura dedo meñique
menique = analogRead(A4);
menique = constrain(menique,300,500);
mmenique = map(menique,360,460,1,3);
//lectura muñeca
muneca = analogRead(A5);
muneca = constrain (muneca,220,315);//bueno
mmuneca = map (muneca,220,315,1,3);
}
void establecer_letra(){
 if
((mpulgar==1)&&(mindice==3)&&(mmedio==3)&&(ma
nular==3)&&(mmenique==3)&&(mmuneca==1))
 letra="A";
   }
void enviar_letra(){
Serial.println(mpulgar);
Serial.println(pulgar);
  //Serial.print(letra);
  Delay (2000);
}
```

inconsistencia.

E. Creación de app

El desarrollo de la aplicación que nos permitirá traducir el abecedario del lenguaje de señas para smartphone fue MIT App Inventor 2, con la cual podemos desarrollar aplicaciones compatibles con el sistema Android.

Fig. 28. Entorno de diseño de la aplicación MIT App Inventor 2.

La aplicación cuenta con dos interfaces importantes para el desarrollo se encuentra en la esquina superior derecha, y con esos se ingresa al modo diseño y al modo bloques la cual nos permite hacer el código de las acciones que realizará todo lo del interfaz diseño.

Fig. 29.Botones para selección la interfaz a trabajar.

Fig 30. Entorno de la interfaz bloque de MIT App Inventor 2.

Para el desarrollo se utilizaron tres botones, para la conexión bluetooth, y un label donde mostrará las letras, según lo que se envié del Arduino. Se necesita BluetoothClient para poder utilizar los códigos de este mismo y un clock, estos dos últimos no se ven en la pantalla, pero si en la parte inferior de esta interfaz.

Fig. 31. Interfaces de usuarios utilizadas en el modo Diseño.

Luego de tener todo en el modo diseño, procedemos a hacer el código en la interfaz bloques.

Fig. 32. Bloques para programación.

Los bloques de programación varían según la interfaz de usuarios que se han agregado.

El primer bloque realiza la siguiente acción, al dar clic en el botón 1, se desconectará el bluetooth y en el label dirá “Estado: Desconectado”.

Fig. 33.Código para desconectarse del bluetooth.

En el siguiente bloque, si uno tiene el Bluetooth del celular desactivado, al abrir la aplicación le pedirá que encienda el bluetooth para poder utilizar la aplicación.

Fig. 34.Bloque 2, acción de pedir activar el Bluetooth.

El siguiente bloque, al dar clic en dispositivos nos cambiara de pantalla y nos mostrara todos los dispositivos que tienen el bluetooth activado y solo debemos escoger al que nos queremos conectar, si el bluetooth sigue apagado nos preguntara de iniciarlo.

Fig. 35.Bloque 3, selección del bluetooth a conectarse.

En este bloque, luego de haber seleccionado al dispositivo que nos emparejaremos dentro de la lista, entonces entablará la conexión de ser exitosa veremos que el módulo bluetooth cambiará a una luz roja sin cambiar de estado, y en un Label dirá "Estado: Conectado" de no ser así dirá "Error de Conexión".

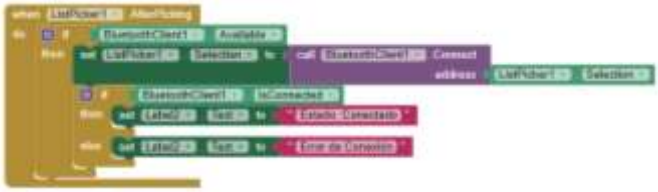


Fig. 36. Bloque 4, conexión al dispositivo escogido.

Este bloque tendrá la acción de salir por completo de la aplicación y cerrando el emparejamiento con el dispositivo conectado al dar click en el botón salir.

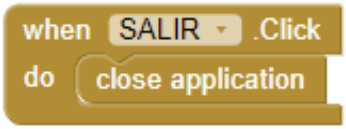


Fig. 37.Bloque 5, salir de la aplicación.

La acción del bloque 6 indica que se tendrá una variable "dato" que no tendrá ningún valor

Fig. 38.Bloque 6, variable dato

El bloque 7, contiene las acciones que realizará al tener establecida la conexión con un dispositivo y al recibir un dato entonces en el Label designado y según la lectura que registre la variable dato escribirá este dicho valor.

Para tener el valor deseado, la variable dato deberá recibir un valor igual al establecido en el código hecho en el Arduino, según la respectiva letra del abecedario.

Fig. 39.Bloque 7, datos para escribir en el label según variable recibida.

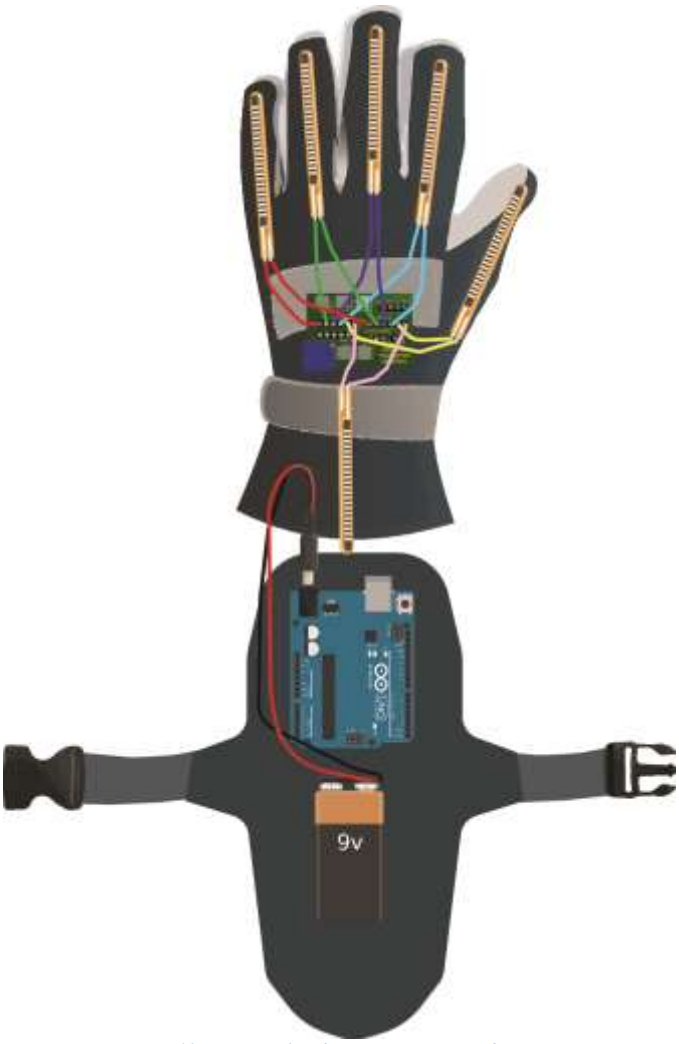


Fig. 40. Diseño final para guante traductor.

IV. CONCLUSIONES Y RECOMENDACIONES

- El proyecto es viable, pero se debe manejar muy cuidadosamente debido a las variables involucradas, en las cuales estaría el tamaño de la mano del usuario, la variación del lenguaje establecido, la resistencia real de variación del flex sensor, las interferencias que se tengan debido a la resistencia remanente del flex sensor, el rango de variaciones de voltaje que se logran con cada dedo individual y el peso en general de toda la estructura.

- Para obtener mejores resultados es conveniente utilizar otro tipo de guante o una estructura que permita mantener las resistencias en un solo sitio, debido a que se originaba el problema de des calibración ya que las resistencias cambiaban de posición cada vez que se quitaba y ponía el guante, aunque este fuera un cambio que se cree insignificante como los cambios de voltaje eran muy pequeños estos cambiaban los rangos de calibración ya establecidos. Se presentan dos alternativas de estructura de guante encontradas que ayudan con este problema.

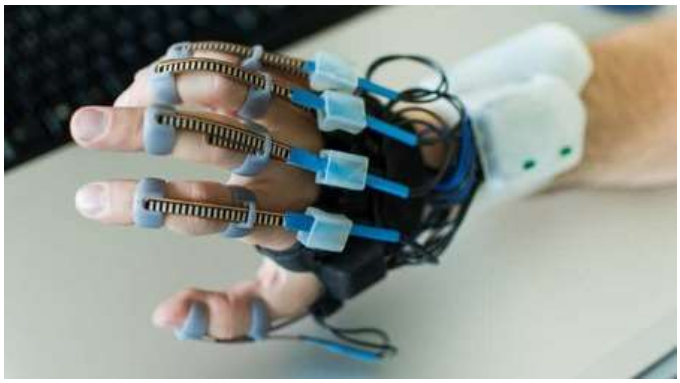


Fig. 41. Estructura de plástico hecha a partir de impresión digital, para mejor sostenibilidad y ajustabilidad de los Flex sensor. Fuente: xataka móvil

Fig. 42. Estructura de malla con resistencias cocidas en el interior para mantener estables la posición de los Flex sensor. Fuente: TheTechy.

- En cuanto a tarjeta de Arduino se podría recomendar cambiar a la tarjeta nando o a la tarjeta lilypad, estas no difieren mucho en cuanto a las capacidades de procesamiento, pero tienen una ventaja en cuanto al Arduino; sus dimensiones. Ambas tarjetas antes mencionadas son de una dimensión extremadamente menor a las del Arduino uno, y en el caso de la Arduino lilypad esta está diseñada para ser utilizada directamente en proyectos de textiles, puede ser montada en telas y unida por medio de hilos conductores especiales que facilitan todas las conexiones, además de que su forma da una ventaja de maniobrabilidad y conectividad sobre las otras.

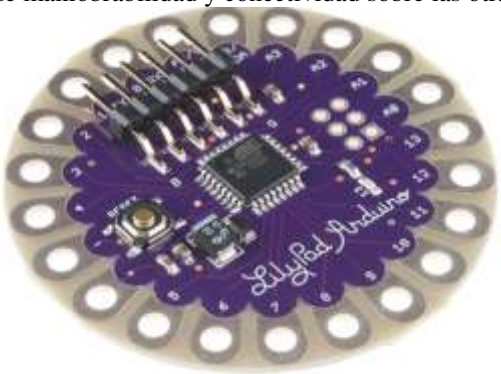
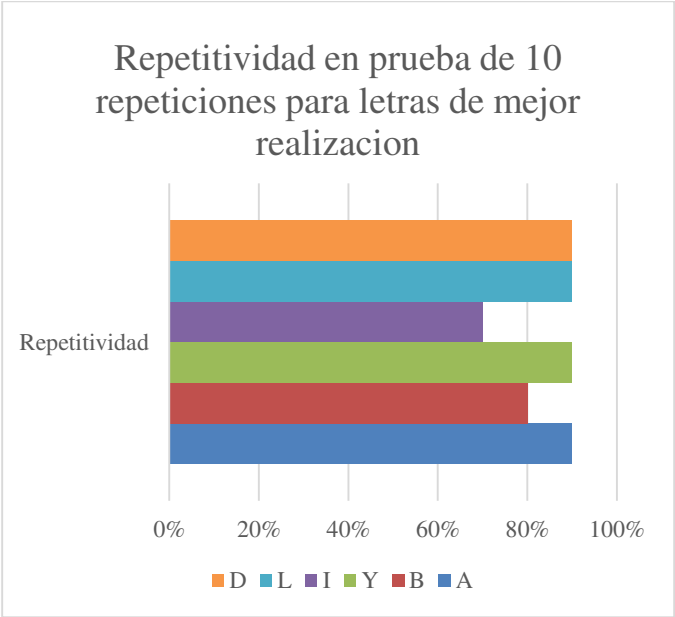


Fig. 43. Tarjeta lilypad Arduino de diseño para e-textiles. Fuente: Arduino.

- Se puede utilizar dos acelerómetros y giroscopio para mejorar la captación de variables de movimiento de la mano. Estableciendo uno en la punta de los dedos y otro en la base de la muñeca, para obtener registro de la dirección del movimiento de la mano, así como también establecer la intensidad con la que se realizó para tener una variable más para establecer las letras.

Muestras de letras con mayor repetitividad y fácil realización en las pruebas en el monitor serial de Arduino y grafica de comparación de repetitividad.



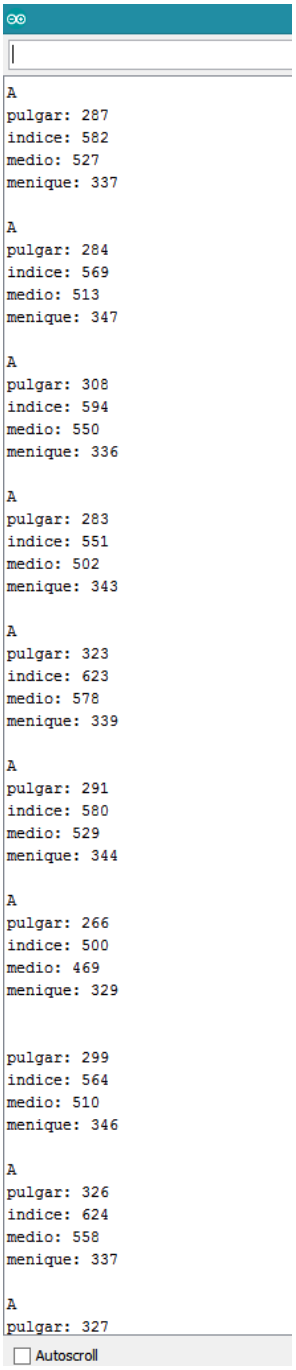


Fig. 44. Letra A en prueba en monitor serial de Arduino.

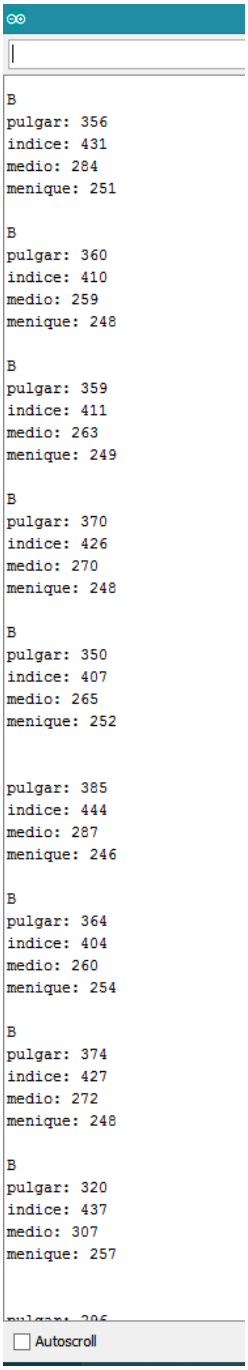


Fig. 45. Letra B en prueba en monitor serial de Arduino.

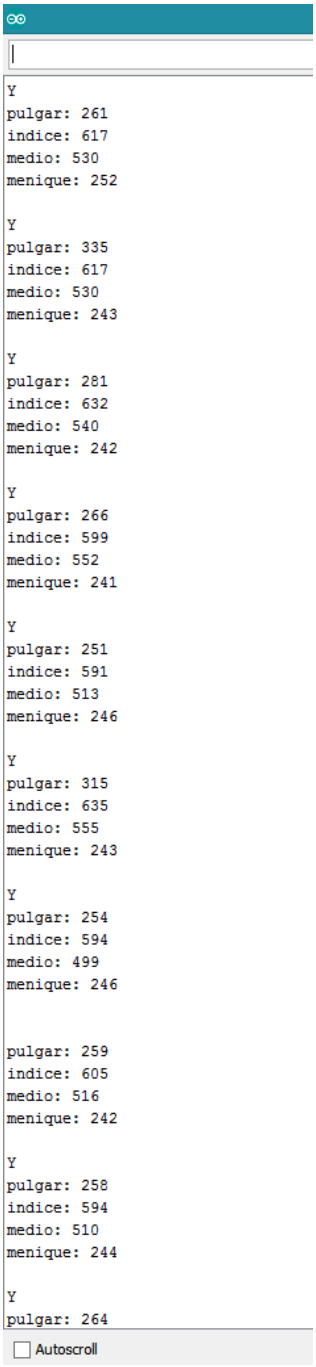


Fig. 46. Letra Y en prueba en monitor serial de Arduino

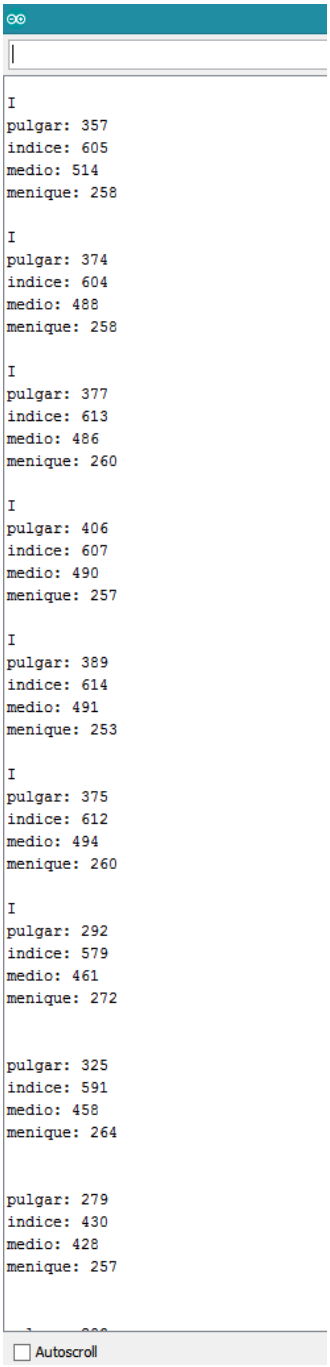


Fig. 47. Letra I en prueba en monitor serial de Arduino

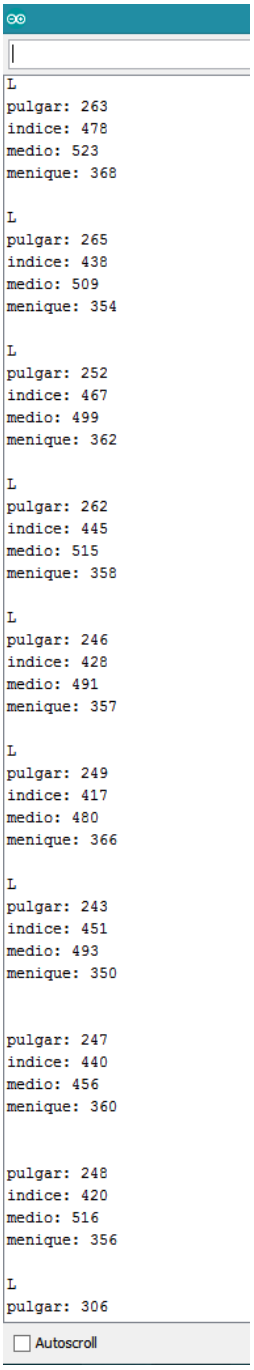


Fig. 48. Letra L en prueba con monitor serial de Arduino.

REFERENCIAS

- [1] Duque Arias, Diego Fernando e Ibarra Caicedo Diseño e implementación de un guante electrónico que permite transformar el lenguaje de señas, http://dspace.ups.edu.ec/handle/123456789/6329
- No.EAD-FLEX-SENSOR-45, Flex Sensor 4.5", Rambal Automatización y robótica , Providencia, Santiago https://www.sparkfun.com/datasheets/Sensors/Flex/flex22.pdf
- [2] No. EAI-FLEX-SENSOR-22, Flex Sensor, Rambal Automatización y robótica, Providencia, Santiago https://rambal.com/presion-peso-nivel-flex/250-sensor-flex.html
- [3] P. Espinosa, H. Pogo, Diseño y construcción de guante prototipo electrónico capaz de traducir lenguaje de señas, Cuenca, 2013 https://dspace.ups.edu.ec/bitstream/123456789/4211/1/UPS-CT002598.pdf
- [4] D.Duque, M,Ibarra, Diseño e implementación de un guante electrónico que permite transformar el lenguaje de señas en caracteres, Universidad politécnica salesiana, Quito, febrero 2014 https://dspace.ups.edu.ec/bitstream/123456789/6329/1/UPS-ST001078.pdf
- [5] L. Estrada, Diseño e implementación de un prototipo de traducción de lenguaje de señas, Escuela politécnica nacional, Quito, septiembre 2016 https://bibdigital.epn.edu.ec/bitstream/15000/16712/1/CD-7309.pdf
- [6] S. Fernández, "Guantes que "traducirán" de lengua de signos a lengua escrita a través del móvil" https://www.xatakamovil.com/movil-y-sociedad/ya-existen-los-guantes-que-haran-que-los-sordomudos-puedan-hablar-a-traves-del-movil