Instituto Tecnológico de Tuxtla Gutiérrez

Reporte Final De Residencia Profesional

Proyecto:

Guante Didáctico Con Fotosensores.

Realizado a:

Instituto Tecnológico De Tuxtla Gutiérrez.

Asesor:

Dr. Héctor Hernández de León.

Residentes:

Esquinca Espinosa Enrique

Mentado González Leonardo

Tuxtla Gutiérrez, Chiapas, Diciembre de 2012.

CONTENIDO

1. INTRODUCCION

La persona con discapacidad auditiva, ha descubierto y desarrollado ot formas de expresión del lenguaje en su vida cotidiana, como lo es el caso particular del lenguaje de señas propio para cada nación. “Lenguaje” es un concepto más amplio que la palabra hablada; abarca distintos canales para comunicarnos (los sentidos, el cuerpo, los gestos, el movimiento, la mirada, las expresiones artísticas, la escritura, las señales, los signos, etc.). La persona oyente, reduce estas otras posibilidades de lenguaje y no utiliza en todo su potencial estos canales, a excepción de ambientes particulares como en el deporte, uso militar, navegación, etc.

El lenguaje de signos es una respuesta creativa de los sordos ante la experiencia de una **sordera profunda (hipoacusia profunda)**, es independiente de cualquier lenguaje oral (lengua hispana). Brinda a las personas sordas la oportunidad de expresarse o comunicarse, desarrollando totalmente su potencial en una forma en que el lenguaje oral no se lo permite.

Basado en lo anterior diseñaremos y construiremos un dispositivo que permitirá al usuario con problemas auditivos principalmente, interactuar con la computadora de manera que pueda imitar la posición de la seña mostrada. Con esto logrará de manera paulatina colocar su mano en la posición correcta y de esta forma aprenderá el lenguaje de señas mexicanas.

Para conseguir los objetivos de esta investigación seguiremos unos pasos como son la obtención de información sobre el lenguaje de señas mexicano, la implementación del guante con los foto sensores, posteriormente la interfaz con el software, obteniendo pequeños resultados que ayudarán a obtener el dispositivo final.

1.1 Justificación

El elemento principal de comunicación para un sordo y para defender sus derechos culturales y lingüísticos es el uso de la LSM, por tal motivo, se consideró necesario plantear el desarrollo de la primera etapa (que convierta en señales eléctricas la signación de queiremas) de un instrumento que funja como traductor de las señas por las que se comunica un sordo respetando los parámetros formativos quinésicos de la LSM.

Mediante este dispositivo se lograra el aprendizaje de una manera interactiva. A través de este dispositivo se pretende contribuir al avance en los sensores fotoeléctricos de movimiento.

Con esto mismo el dispositivo de guante didáctico se lograra que las personas sordomudas y normoyentes puedan aprender el lenguaje de señas de forma más fácil y rápida y con esto se logrará una mayor integración de las personas sordomudas en el aspecto social y sobretodo laboral.

Con el uso de tecnología fotoeléctrica se aprovecha el avance en el campo de las tecnologías ópticas combinando foto sensores con microprocesadores para realizar una tarea específica y con esto lograremos desarrollar un método didáctico innovador, revolucionando así todo lo establecido por la tecnología en el campo de la educación hasta el momento.

Resulta viable realizar un guante didáctico puesto que su tecnología no resulta muy costosa y su interfaz es a través de una computadora que evitara la necesidad de una persona para enseñar el lenguaje de señas y los altos costos de contrato de esta persona o bien el transporte hacia las instituciones educativas, evitando así también la compra de libros o los gastos hechos por asistir a cursos.

1.2 Objetivo

Diseñar y construir un dispositivo didáctico el cual permita interactuar directamente con el usuario a través de una interfaz de computadora usando dispositivos foto eléctricos.

1.2.1 Objetivos específicos

- Diseño y construcción del guante que permitirá la recepción de movimientos de la mano y enviará los datos obtenidos hacia la interfaz
- Diseño y construcción de la interfaz con la cual se podrán recibir los datos analógico y se enviaran datos digitales a la PC.
- Desarrollo del software que funcionará como una interfaz amigable entre el usuario y la PC.

1.3 Planteamiento Del Problema

Los métodos de enseñanza actuales no contribuyen al buen aprendizaje debido a que la forma de aprender no es autodidacta ya que en todos los métodos actuales se depende directamente de una persona que funja como guía o tutor ya sea dentro de un curso o bien en la ayuda de la compresión de un libro.

La principal causa de que los métodos actuales no sean tan efectivos y que no haya desarrollo o evolución de ellos es la falta de investigación para desarrollar prototipos directamente orientados a la enseñanza de lenguaje de señas autodidactas, por eso mismo las personas tienen que experimentar cual es el método adecuado para su aprendizaje, y el aprendizaje es tardío.

Otra causa importante de este problema es que existen muy pocas instituciones y por lo mismo pocos profesores enfocados a la enseñanza del lenguaje de señas además de que de esta forma la enseñanza requiere de mucha paciencia tanto del profesor como de la persona aprendiz, lo cual acarrea que las personas en muchas ocasiones tengan que viajar para poder llegar a un centro de ayuda o bien los profesores tengan que viajar hasta el lugar de origen de las personas sordomudas para impartir un curso.

1.4 Hipótesis

Realizar un guante didáctico de uso práctico usando sensores fotoeléctricos con la finalidad de simular los movimientos de la mano portadora. A su vez establecer una interfaz amigable usando MatLab que desplegara las formas de cada letra y calificara si la posición de la mano portadora es la correcta o no para esto se usara el lenguaje de señas mexicano usando las letras del abecedario y las palabras que solo necesitan de una mano para poder expresarla.

La interfaz entre guante y computadora se realizara con una tarjeta de adquisición de datos que recibirá las señales analógicas de los sensores fotoeléctricos y enviara a la computadora señales digitales para el uso de MatLab.

1.5 Delimitación

Los dispositivos desarrollados solo pueden detectar el pliegue de los dedos, pero no su movimiento de aducción/abducción. Por lo que se hará necesario incluir otro tipo de dispositivos que permitan completar esta característica de los movimientos. Por el momento, se puede desarrollar la siguiente etapa que permitirá utilizar este juego de señales eléctricas en combinación con nuevos dispositivos que permitan diferenciar la forma o configuración de la mano el lugar de articulación o espacio donde se articula el signo o bien movimiento de la mano.

1.5.1 Dimensiones físicas:

El dispositivo constará de un solo guante y una interfaz hacia una computadora en la cual se implantará una interfaz gráfica que permitirá al usuario imitar con el guante la letra mostrada, el guante constara de fotosensores que enviaran señales eléctricas a la tarjeta de adquisición de datos el cual a su vez convertirá dichas señales analógicas y las enviara a la PC.

1.5.2 Funcionalidad:

La finalidad del dispositivo es detectar la posición de la mano del usuario e indicar si la posición del guante es correcta, esto se conseguirá mediante imágenes que indicaran que letra imitar e indicar cuando este imitando correctamente. Y así continuar con la siguiente letra de la lección o bien se podrá elegir si se sigue repasando la misma letra.

2. FUNDAMENTO TEÓRICO

De acuerdo con la consulta hecha al sitio WEB del INEGI en junio del 2006, aparecen publicados los resultados del XII Censo general de población y vivienda del año 2000, en los cuales se incluye la estadística del total de la población mexicana con discapacidades (1.84%), dentro de las que se contempla la auditiva con 0.29% de la población; es decir, 281,793 personas con sordera.

En la actualidad los métodos de enseñanza de señas son poco prácticos, como por ejemplo los libros de enseñanza y los cuadernos de práctica, además de que las instituciones de ayuda o enseñanza para sordomudos muchas veces se encuentran alejadas de las personas con este problema, y por otra parte el costo de algunos métodos son elevados ejemplo de ello los cursos de capacitación.

Es por ello que se busca la forma de solucionar dicho problema mediante los materiales didácticos que son distintos elementos que pueden agruparse en un conjunto, reunidos de acuerdo a su utilización en algún fin específico. Los elementos del conjunto pueden ser reales (físicos), virtuales o abstractos.

El material didáctico es aquel que reúne medios y recursos que facilitan la enseñanza y el aprendizaje. Suelen utilizarse dentro del ambiente educativo para facilitar la adquisición de conceptos, habilidades, actitudes y destrezas.

Es importante tener en cuenta que el material didáctico debe contar con los elementos que posibiliten un cierto aprendizaje específico. Por eso, un libro no siempre es un material didáctico. Por ejemplo, leer una novela sin realizar ningún tipo de análisis o trabajo al respecto, no supone que el libro actúe como material didáctico, aun cuando puede aportar datos de la cultura general y ampliar la cultura literaria del lector.

En cambio, si esa misma novela es analizada con ayuda de un docente y estudiada de acuerdo a ciertas pautas, se convierte en un material didáctico que permite el aprendizaje.

Los especialistas afirman que, para resultar didáctica, una obra debe ser comunicativa (tiene que resultar de fácil comprensión para el público al cual se dirige), tener una estructura (es decir, ser coherente en sus partes y en su desarrollo) y ser pragmática (para ofrecer los recursos suficientes que permitan al estudiante verificar y ejercitar los conocimientos adquiridos).

Cabe destacar que no sólo los libros pueden constituir un material didáctico: las películas, los discos, los programas de computación y los juegos, por ejemplo, también pueden serlo.

Teniendo en cuenta la biomecánica de la mano y la capacidad de contar con una amplia configuración por el movimiento de los dedos al comunicarse con la LSM (queiremas), se desarrollará un dispositivo foto-eléctrico cuyo principio de funcionamiento se asemeja al de la fibra óptica (Alonso y Finn, 1998), y se basa en la percepción de un haz de luz emitido por una fuente luminosa constante.

Esta luz se transmite dentro de un volumen de superficie cilíndrico elástico cerrado que responderá a los movimientos de flexión de los dedos; por lo que las variaciones en longitud, diámetro y posición del transmisor/receptor modularán la intensidad de la luz percibida en el elemento.

Dicha información servirá para implementar un método didáctico dentro de una pc, mediante el dispositivo con una interfaz hacia la computadora en la cual se desplegara en la pantalla un menú con las diferentes opciones ya sea en modo maestro o modo repaso en las cuales la persona con discapacidad podrá elegir entre comenzar un nuevo curso o bien repasar lo aprendido en el día, se podrá visualizar las diferentes lecciones con las cuales el alumno aprenderá la perfecta colocación de su mano para poder expresar las diferentes letras del abecedario.

2.1 Lenguaje De Señas

Como seres sociables por naturaleza, normalmente buscamos la manera de interactuar y comunicarnos con nuestro semejante. Sin embargo, a diferencia del lenguaje oral, que aprendemos a hablar por medio de repetir lo que escuchamos, y que posteriormente aprendemos a escribir y leer, el lenguaje del sordo se basa en signos o señas, por medio del cual puede expresar sus ideas y sentimientos. Se puede calificar este lenguaje como *gesto-visual* ya que está basado en el uso de las manos, expresiones faciales y el cuerpo en general y como *espacio-visual* al ubicar lo que se dice en un "espacio".

Aunque la comunicación por medio de señas constituye un lenguaje natural para las personas sordas, en México no todas las escuelas dedicadas a la enseñanza de los sordos promueven su uso. Algunos sordos aprenden a pronunciar, leer los labios e incluso leer y escribir español, en un esfuerzo por "integrarlos" a la sociedad de oyentes. En el mejor de los casos, además de aprender lo anterior, también aprenderán a comunicarse entre sordos por el uso del **Lenguaje de signos mexicano** (**LSM**), el lenguaje usado por la población de sordos en México.

Sin embargo, otros sordos han sido marginados. Nunca han ido a la escuela y no conocen el LSM, sino que se comunican sólo con su familia y allegados por medio del uso de "señas familiares" (creadas por ellos o sus parientes), mímica o dibujos.

Aún otra manera de comunicarse con los sordos consiste en el "español de señas exactas" o "españolización", el cual consiste "transliterar" palabra por palabra el idioma Español usando las señas del LSM, y "deletreando" con el *abecedario en LSM* los términos que en este último no se utilizan, (como los artículos y muchos de los pronombres). Esto sin embargo, puede resultar confuso para los sordos, pues en realidad el LSM es muy diferente del español. De hecho no existe una seña para cada palabra en español.

Por otro lado, el que exista en nuestro país un lenguaje "oficial" no quiere decir que éste sea completamente uniforme. Muchas veces varía según la ciudad o región, y se diferencia particularmente en lo relativo a terminología religiosa.

Si esto aplica al LSM, podemos imaginarnos la gran variedad de lenguajes de señas que existen alrededor del mundo, en donde prácticamente cada país, aunque comparta el mismo idioma hablado, tendrá un conjunto de signos diferenciados para la comunicación entre sordos, los cuales parecen muy poco entre sí. Por ejemplo en México, aunque el LSM se deriva del sistema de signos francés (traído a finales del siglo XIX), se diferencia del lenguaje de señas francés y americano (ASL) pues utiliza muchas de las señas que ya se utilizaban antes de esto en el país. Asimismo, una característica del LSM es la "inicialización", es decir, tomar del alfabeto del lenguaje de signos la seña que corresponde a la primera letra de la palabra en español que se está explicando.

2.1.1 Diferencias básicas entre el idioma Español y el LSM

- **Cantidad de palabras:** Si se le compara con el español, el LSM utiliza un léxico más básico. No existe una seña para cada palabra en español. Muchas veces se puede utilizar una misma seña para las diferentes grados o niveles de una palabra en Español, en donde la intensidad de dicho nivel lo da la manera en que se signa (velocidad, fuerza y sobre todo la expresión facial).
- **Uso de verbos:** En LSM el verbo aparece sin conjugarse, en infinitivo. Para indicar el tiempo en que ocurre la acción, se utiliza una seña aparte (véase el apartado *"Tiempo"* de la sección *Reglas gramaticales del LSM)*. Asimismo, para aplicarle la acción a alguien se ubica previamente a este en un espacio para después hacer la seña cerca de dicho espacio.
    - En LSM rara vez se utilizan los verbos *ser* o *estar.*
- En LSM el número generalmente va después del sustantivo. Ejemplo: *Hijo 2* (LSM) en lugar de *2 hijos* (Español).
- Se omiten los artículos (el, la, los, etc.) y la mayoría de los pronombres en LSM. La manera de aplicar una idea o acción a alguien es mediante la ubicación de este en un espacio, para posteriormente señalarlo, o bien, hacer la seña cerca del espacio donde se posicionó.

2.1.2 Abecedario en Lenguaje de señas mexicanas

Se representan la mayoría de las letras desde el punto de vista del *observador,* con excepción de las letras G, H, K, Q, X y Y, las cuales se aparecen como las vería quien las está configurando.

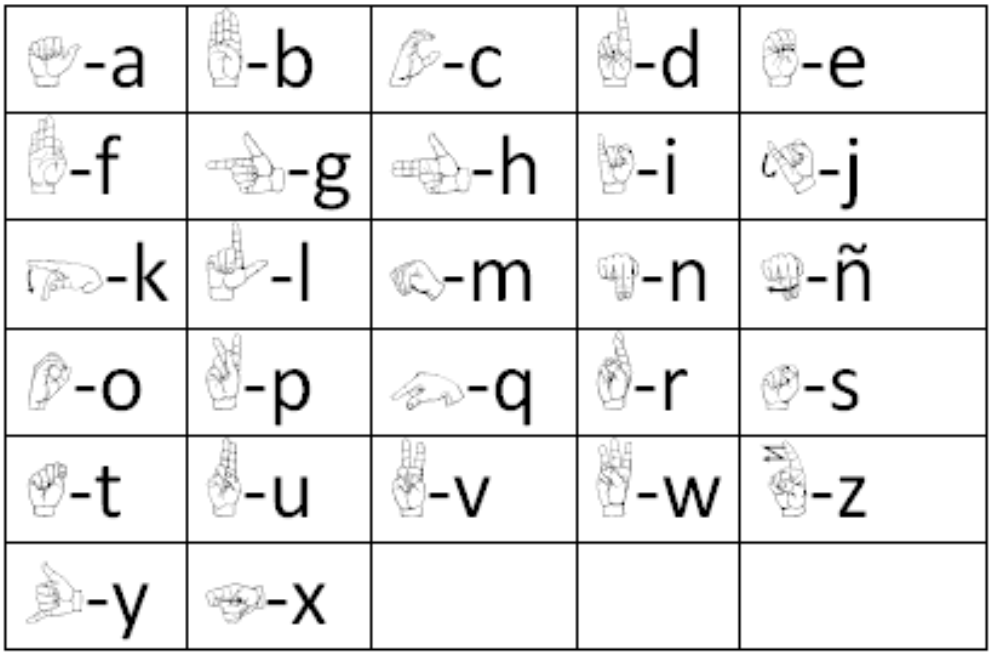


Figura 1 Abecedario LSM

2.2 Sensor de Estado Lógico

2.2.1. Fotorresistencia

Fotocelda o fotorresistencia cambia su valor resistivo conforme la intensidad de luz que incide en su superficie sensitiva.

Una fotorresistencia es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. Puede también ser llamado fotorresistor, fotoconductor, célula fotoeléctrica o resistor dependiente de la luz, cuyas siglas, LDR, se originan de su nombre en inglés *light-dependent resistor*. Su cuerpo está formado por una célula o celda y dos patillas. En la siguiente imagen se muestra su símbolo eléctrico.

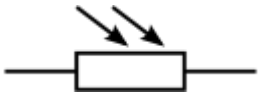


Figura 2 Símbolo eléctrico LDR

El valor de resistencia eléctrica de un LDR es bajo cuando hay luz incidiendo en él (puede descender hasta 50 ohms) y muy alto cuando está a oscuras (varios megaohmios).

2.2.1.1. Características De La LDR

Su funcionamiento se basa en el efecto fotoeléctrico. Un fotorresistor está hecho de un semiconductor de alta resistencia como el sulfuro de cadmio, CdS. Si la luz que incide en el dispositivo es de alta frecuencia, los fotones son absorbidos por las elasticidades del semiconductor dando a los electrones la suficiente energía para saltar la banda de conducción. El electrón libre que resulta, y su hueco asociado, conducen la electricidad, de tal modo que disminuye la resistencia. Los valores típicos varían entre 1 MΩ, o más, en la oscuridad y 100 Ω con luz brillante.

Las células de sulfuro del cadmio se basan en la capacidad del cadmio de variar su resistencia según la cantidad de luz que incide la célula. Cuanta más luz incide, más baja es la resistencia. Las células son también capaces de reaccionar a una amplia gama de frecuencias, incluyendo infrarrojo (IR), luz visible, y ultravioleta (UV).

La variación del valor de la resistencia tiene cierto retardo, diferente si se pasa de oscuro a iluminado o de iluminado a oscuro. Esto limita a no usar los LDR en aplicaciones en las que la señal luminosa varía con rapidez. El tiempo de respuesta típico de un LDR está en el orden de una décima de segundo. Esta lentitud da ventaja en algunas aplicaciones, ya que se filtran variaciones rápidas de iluminación que podrían hacer inestable un sensor (ej. tubo fluorescente alimentado por corriente alterna). En otras aplicaciones (saber si es de día o es de noche) la lentitud de la detección no es importante.

Se fabrican en diversos tipos y pueden encontrarse en muchos artículos de consumo, como por ejemplo en cámaras, medidores de luz, relojes con radio, alarmas de seguridad o sistemas de encendido y apagado del alumbrado de calles.

2.2.1.2 Valores En Uso De La LDR

Los valores de una fotoresistencia cuando está totalmente iluminada y cuando está totalmente a oscuras varía.

Puede medir de 50 ohmios a 1000 ohmios (1K) en iluminación total y puede ser de 50K (50,000 Ohms) a varios megaohmios cuando está a oscuras.

El valor de la fotorresistencia (en Ohmios) no varía de forma instantánea cuando se pasa de luz a oscuridad o al contrario, y el tiempo que se dura en este proceso no siempre es igual si se pasa de oscuro a iluminado o si se pasa de iluminado a oscuro.

Esto hace que el LDR no se pueda utilizar en muchas aplicaciones, especialmente aquellas que necesitan de mucha exactitud en cuanto a tiempo para cambiar de estado (oscuridad a iluminación o iluminación a oscuridad) y a exactitud de los valores de la fotorresistencia al estar en los mismos estados anteriores. Su tiempo de respuesta típico es de aproximadamente 0.1 segundos.

2.2.2 LED

Un led (de la sigla inglesa *LED*: *Light-Emitting Diode:* 'diodo emisor de luz', también 'diodo luminoso') es un diodo semiconductor que emite luz. Se usan como indicadores en muchos dispositivos, y cada vez con mucha más frecuencia, en iluminación.

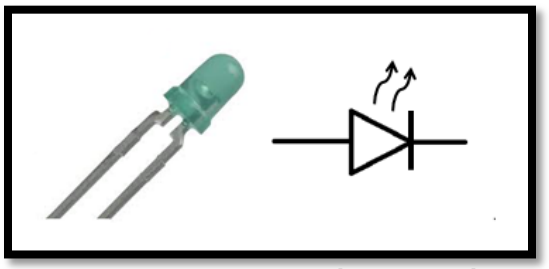


Figura 3 LED verde y su símbolo eléctrico

Cuando un led se encuentra en polarización directa, los electrones pueden recombinarse con los huecos en el dispositivo, liberando energía en forma de fotones. Este efecto es llamado electroluminiscencia y el color de la luz (correspondiente a la energía del fotón) se determina a partir de la banda de energía del semiconductor. Por lo general, el área de un led es muy pequeña (menor a 1 mm^2), y se pueden usar componentes ópticos integrados para formar su patrón de radiación.

En corriente continua (CC), todos los diodos emiten cierta cantidad de radiación cuando los pares electrón-hueco se recombinan; es decir, cuando los electrones *caen* desde la banda de conducción (de mayor energía) a la banda de valencia (de menor energía) emitiendo fotones en el proceso. Indudablemente, por ende, su color dependerá de la altura de la banda prohibida (diferencias de energía entre las bandas de conducción y valencia), es decir, de los materiales empleados.

2.2.2.1 LED Verde (Nitruro De Galio)

El nitruro de galio (GaN) es un compuesto emisor de luz que ya se usa en los flashes de las cámaras, los faros de las bicicletas, los teléfonos móviles y en la iluminación del interior de autobuses, trenes y aviones, pero un equipo de investigadores británicos prevé que sus posibilidades van mucho más allá.

El **nitruro de Galio** (**Galio Nitruro**, **GaN**) es una aleación binaria de semiconductores del III/V con una banda prohibida directa que se ha venido usando en diodos emisores de luz (LEDs) desde los años noventas. Este compuesto químico es un material muy duro que tiene una estructura cristalina Wurtzita. Su amplia banda prohibida de 3.4 eV[2] le dan propiedades especiales para aplicaciones en optoelectrónica,[3] [4] dispositivos de alta potencia y dispositivos de alta frecuencia.

2.2.3 Termofit

Los Tubos Termocontráctiles son tubos aislantes que se contraen al 50% de su diámetro original al aplicarles calor. Son la mejor opción para aislar cables, uniones, tubos y varillas, etc. tanto eléctrica como mecánicamente o solo para mejorar apariencia.

Son fabricados de poliolefina reticulada, libres de plomo y sustancias nocivas, cuentan con aprobaciones de UL, CSA, RoHS y Sony.

Figura 4. Thermofit 5 mm

2.2.3.1 CARACTERÍSTICAS TÉCNICAS

- Temperatura de contracción: 70° C
- Relación de contracción: 2:1
- Soporta: 600 V
- Retardante de flama
- Resistencia a materiales abrasivos, humedad, solventes, etc.

2.3 C++ Builder

C++ Builder es un entorno de desarrollo rápido de aplicaciones en lenguaje C++ para Windows, combina la biblioteca Visual Component Library y el IDE escrito en Delphi con un compilador de C++. El ciclo de lanzamiento es anual. Incluye herramientas que permiten desarrollo visual de arrastrar-y-soltar componentes sobre la aplicación e incorpora constructor de interfaz gráfica WYSIWYG en su IDE. IDE es el acrónimo de **I**ntegrated **D**evelopment **E**nvironment o entorno de desarrollo integrado.

C++ Builder es una aplicación Windows que proporciona un entorno de trabajo visual para construir aplicaciones Windows que integra distintos aspectos de la programación en un entorno unificado o integrado. La integración y facilidad de manejo hace que sea una herramienta indispensable para el desarrollo rápido de aplicaciones o RAD (**R**apid **A**pplication **D**evelopment). Guarda una gran similitud con el IDE de Visual Basic, aunque existen ciertas diferencias que veremos.

El IDE de C++ Builder es una aplicación Windows 95 y como tal cumple con los estándares de aspecto, diseño y comportamiento que aconseja Microsoft a los desarrolladores de aplicaciones. En consecuencia, cualquiera que esté familiarizado con el manejo a nivel de usuario de Windows 95 no le supondrá ningún esfuerzo manejarlo con soltura.

2.3.1 Una visión general del IDE de C++ Builder.

El entorno de desarrollo se divide, básicamente, en tres partes. Una serie de ventanas, que pueden estar visibles u ocultas, constituyen la base de C++ Builder.

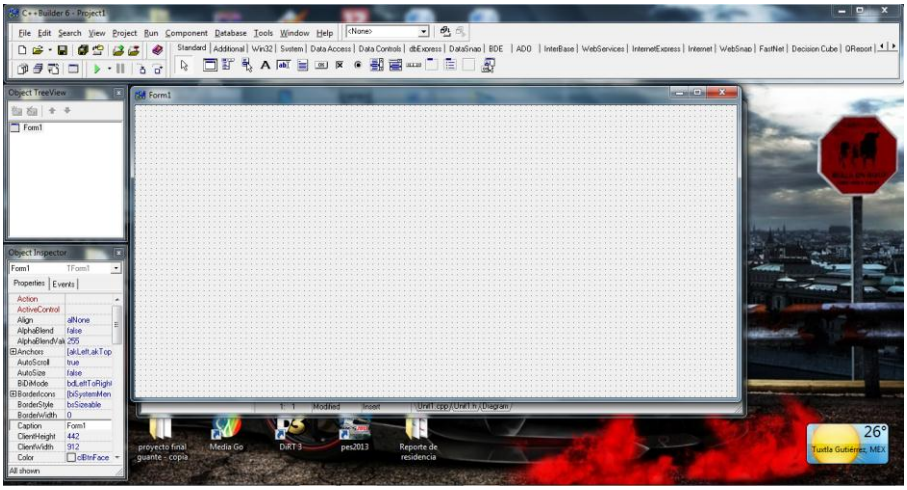


Figura 5 Aspecto del *C++ Builder* al inicio de una sesión.

En la parte superior se coloca la ventana principal, que contiene el menú principal, la barra de herramientas (a la izquierda) y la paleta de componentes (a la derecha). Debajo de la ventana principal, y a la izquierda se coloca el inspector de objetos. A

la derecha del inspector de objetos está el área de trabajo de C++ Builder, que inicialmente muestra el diseñador de formularios, y escondido u oculto parcialmente tras éste aparece el editor de código. Veamos a grandes rasgos la misión de cada uno de ellos.

2.3.2 Ventana principal.

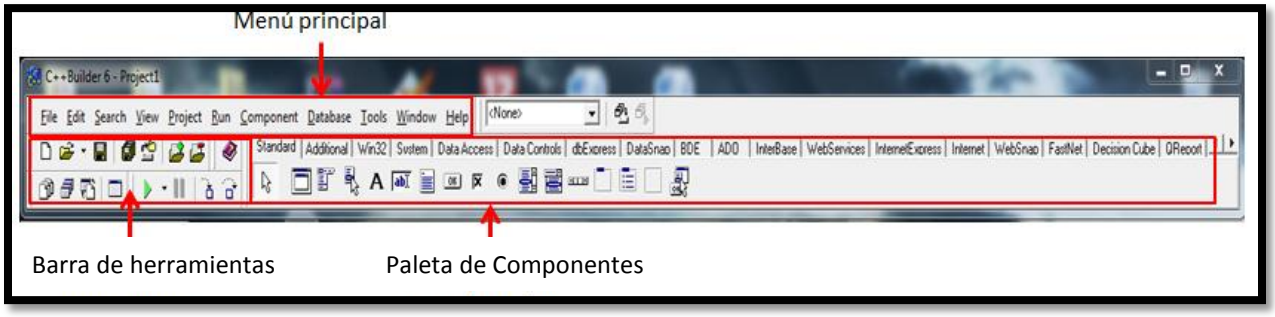


Figura 6. Ventana principal de *C++ Builder*

2.3.2.1 Menú principal.
Permite el acceso a todas las operaciones y posibilita la configuración del programa.

2.3.2.2 Barra de herramientas.
Permite un acceso rápido a las operaciones que se realizan más frecuentemente.

2.3.2.3 Paleta de componentes.
Agrupa a los componentes que pueden incluirse en las aplicaciones.

2.3.3 Inspector de objetos.

Para cambiar las *propiedades* de los objetos que forman la aplicación y seleccionar los *eventos* a los que debe responder la aplicación.

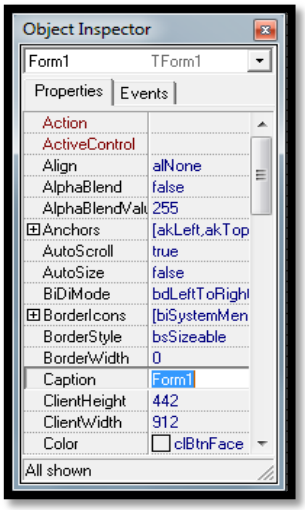


Figura 7. Inspector de Objetos

2.3.4 Diseñador de formularios.

Es una ventana cuadriculada sobre el que se disponen los componentes para diseñar las ventanas que formarán la aplicación.

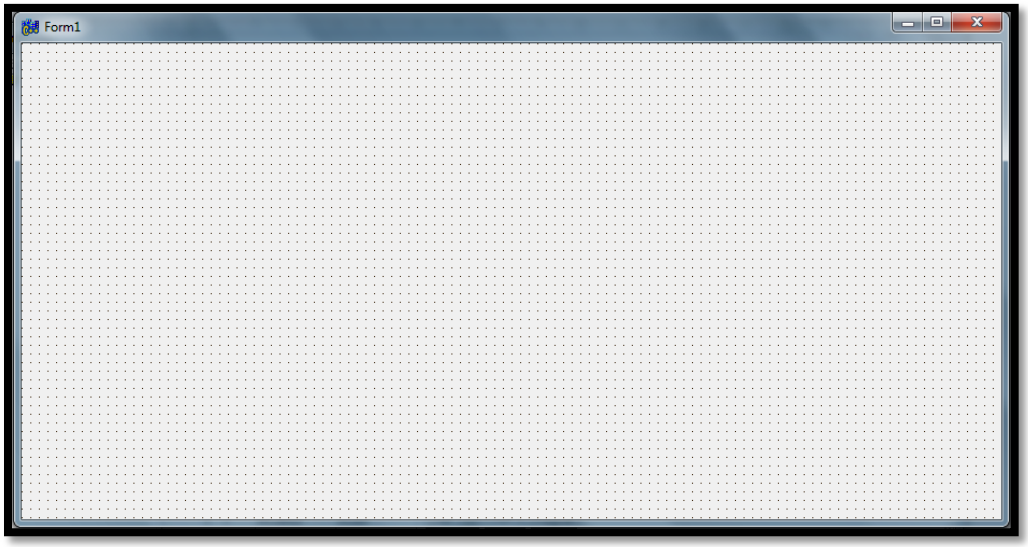


Figura 8. Diseñador de formularios.

2.3.5 Editor de código.

Un típico editor de texto multiventana para ver y editar el código de la aplicación. Está perfectamente integrado con el inspector de objetos y el diseñador de formularios.

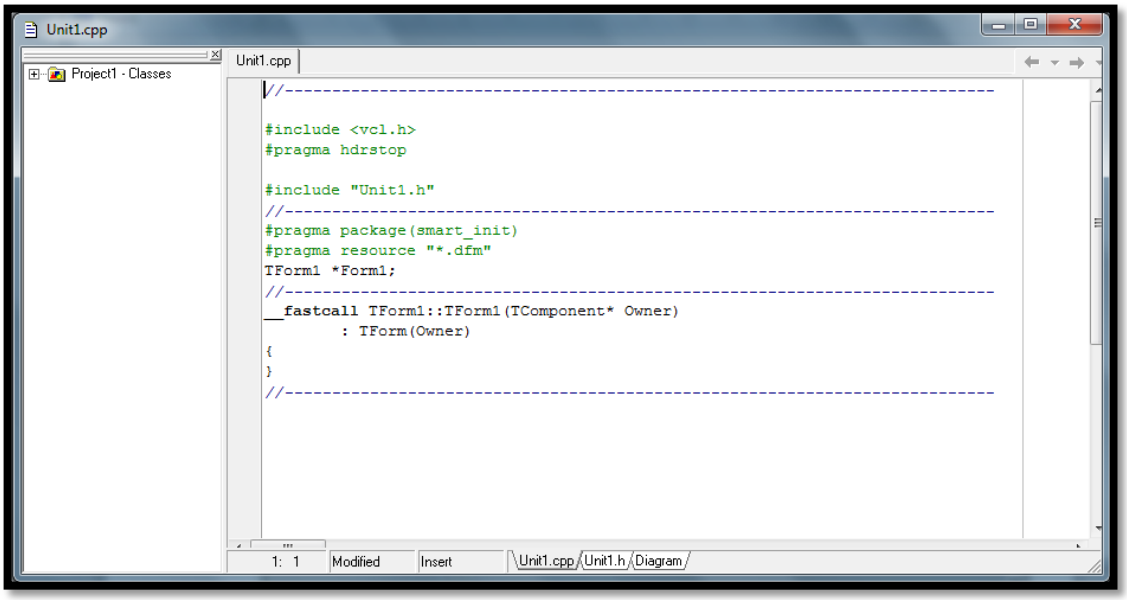


Figura 9. Editor de código.

2.3.6 Componentes De Tcomport

Figura 10. Paleta de componente TComport

El paquete de diseño habrá instalado un nuevo conjunto de componentes que incluye:

- **ComPort**: Componente de acceso al puerto serie en sí.
- **ComDataPacket**: Componente para realizar operaciones de lectura de paquetes.
- **ComComboBox**: Componente visual tipo lista de texto desplegable para crear interface de configuración del puerto.
- **ComRadioGroup**: Componente visual tipo radio para crear interfaces de configuración del puerto.
- **ComLed**: Componente visual tipo LED para mostrar la actividad del puerto.
- **ComTerminal**: Componente que muestra datos serie en pantalla.

A continuación se detalla el uso de cada uno de estos componentes.

2.3.6.1 ComPort

Este es el principal componente de este paquete. Nos permitirá configurar y administrar el puerto serie. Contiene todas las propiedades de configuración de la conexión serie. Sin entrar en todos los detalles lo mas destacado es:

- Propiedad **Port**: Esta propiedad define el puerto a usar en el PC. En la lista desplegable de la propiedad aparecen sólo los puertos existentes (virtuales o físicos).
- Propiedad **Baudrate**: Establece la velocidad serie del enlace de transmisión.
- Propiedad **DataBits**: Establece el número de bits de cada carácter transmitido o recibido en el enlace serie.
- Propiedad **Parity**: Define el tipo de bit de paridad. Aquí es posible definir si la comprobación de paridad debe ser realizado en la lectura, y si se lleva a cabo para sustituir los carácteres con la falta de paridad en la trama recibida por un carácter específico. Para realizar pruebas es aconsejable no permitir la verificación de paridad, no es esencial para el buen funcionamiento y se puede añadir fácilmente más adelante si se desea reforzar el control. De

todos modos, en caso de transmisión/recepción de tramas es mejor manejar una suma de comprobación de tipo CRC.

- Propiedad **StopsBits**: Número de bits de parada de cada carácter.
- Propiedad **FlowControl**: Este conjunto de propiedades se utilizan para definir la gestión del flujo de caracteres en el enlace de serie. Normalmente es preferible no utilizar control de flujo, ya que esto sólo complica la gestión y el cableado del enlace. Además una gestión incorrecta puede hacer inestable al sistema operativo e incluso bloquearlo. Hay excepciones que obligan a su uso, como la gestión de un sistema multipunto con convertidores RS232-RS485.
- Propiedad **Timeouts**: Este conjunto de propiedades se utilizan para definir los tiempos de espera en la transmisión y recepción. Su uso depende de las aplicaciones y en principio se pueden dejar como están.
- Propiedad **Events**: Se utiliza para definir los eventos relacionados con la actividad de la conexión serie. De forma predeterminada están activadas, pero se pueden eliminar todos aquellos que no ayudan a evitar la sobrecarga del diálogo con el TComport.
- Propiedad **EventChar**: Esta propiedad establece el carácter de activación de OnRxChar. Por supuesto si el evento está habilitado en la propiedad **Events**.
- Propiedad **Connected**: Esta característica permite abrir/cerrar la conexión serie y también para comprobar su estado.

2.3.6.2 Comdatapacket

Este control permite gestionar de una manera muy sencilla la recepción de tramas de caracteres, o una cierta cantidad de caracteres.

- Propiedad **Port**: Selecciona a qué componente TComPort debe estar asociado. Esta propiedad siempre debe estar definida.
- Propiedad **Size**: Longitud de la estructura a recibir, si la duración no es fija, establecer esta propiedad en cero.
- Propiedad **StartString**: Cadena que define el inicio de la trama.
- Propiedad **StopString**: Cadena que define el final de la trama.
- Propiedad **CaseInsensitive**: Permite la selección de mayúsculas y minúsculas en los caracteres recibidos.
- Propiedad **IncludeStrings**: Establece si las cadenas StartString y StopString deben incluirse en la trama o no.

2.3.6.3 Tcomcombobox Y Tcomradiogroup

Estos componentes permiten configurar visualmente el enlace serie. Se utilizan como interfaz para la configuración del puerto COM.

La lista de propiedades es la siguiente:

- Propiedad **ComPort**: Selecciona a qué componente TComPort debe estar asociado. Esta propiedad siempre debe estar definida.

- Propiedad **ComProperty**: Se utiliza para definir los parámetros de la conexión que mostrará el componente. Tales como la velocidad, paridad, etc. Por ejemplo, en la imagen siguiente se muestra un ComRadioGroup con el valor de la propiedad ComProperty en ***pBaudRate***.

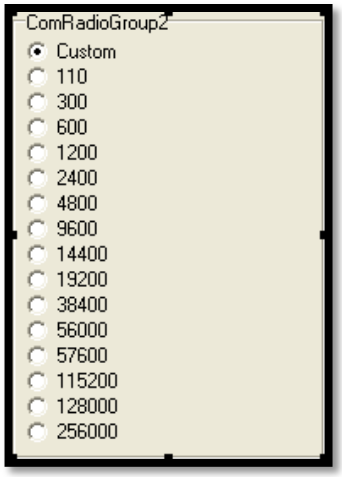


Figura 11. ComRadioGroup

- Propiedad **AutoApply**: Establece si los cambios en la selección se introducen automáticamente en la configuración del enlace. Si esta propiedad es True, los cambios se aplicarán directamente a la configuración del puerto serie. Si esta propiedad es False, se debe llamar mediante código al método ApplySettings de este componente para que los cambios surtan efecto.

2.3.6.4 TComLed

TComLed es un indicador LED para las señales del puerto serie. Es un componente que permite mostrar de manera sencilla si el puerto esta conectado y el estado de las señales CTS, DSR, RLSD, Ring, Tx y Rx . Podemos utilizar mapas de bits personalizados para mostrar el estado de las señales. TComLed se actualiza automáticamente ante los cambios de la señal de control, pero para que se muestre hay que refrescar la ventana de la aplicación. Está por determinar cual es el problema de fondo y como resolverlo.

La lista de propiedades es la siguiente:

- Propiedad **ComPort**: Selecciona a qué componente TComPort debe estar asociado. Esta propiedad siempre debe estar definida.
- Propiedad **Kind**: Establece el tipo de diseño del componente. En el caso de que queramos personalizarlo usaremos el valor ***lkCustom*** y además debemos proporcionar imágenes de ON y OFF en las propiedades ***GlyphON*** y ***GlyphOFF***.

- Propiedad **LedSignal**: Puede asociar el componente a un estado particular de la conexión serie.
- Propiedad **State**: Establece el estado ON/OFF del componente. Cuando el enlace está activo, esta propiedad es de sólo lectura.

2.3.6.5 TComTerminal

Este componente visual permite definir un área de visualización de caracteres enviados / recibidos en el enlace serie. Tiene un interés limitado en la aplicación final pero puede ser de gran ayuda en el desarrollo de un enlace serie. Puede utilizarse como consola ASCII tipo VT100 o VT52.

La lista de propiedades es la siguiente:

- Propiedad **ComPort**: Selecciona a que componente TComPort debe estar asociado. Esta propiedad siempre debe estar definida.
- Propiedad **WrapLines**: Define si la envoltura es automática al adelantar a la derecha.
- Propiedad **LocalEcho**: Especifica si los caracteres escritos en el teclado también se muestran en la consola.
- Propiedad **SendLF**: Establece la posibilidad de enviar un retorno de carro (CR, caráctecter ASCII 13) debe ir seguido de un carácter de nueva línea (LF, Line-Feed, ASCII 10).
- Propiedad **AppendLF**: Define si la recepción de un retorno de carro CR debe interpretarse como una combinación de caracteres CR LF.
- Propiedad **Force7bit**: Fuerza de la transmisión / recepción de caracteres de 7 bits. Todos los caracteres extendidos (8 bits) se reducirán a 7.
- Propiedades de **Columns** y **Rows**: Definen el tamaño de los caracteres de la consola.
- Propiedad **Connected**: Es equivalente a la propiedad Connected de los componentes asociados.
- Propiedad **Caret**: Define la forma del cursor de la consola.
- Propiedad **Emulation**: Define el tipo de emulación de la consola en el caso de recibir una secuencia de escape. Esta propiedad es especialmente útil cuando se utiliza el componente en lugar de una consola ASCII.
- Propiedad **ArrowsKeys**: Establece si las teclas de dirección debe ser enviada en carácter (akTerminal) o se utiliza para mover el cursor (akWindows).

2.4. Arduino

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.

Arduino puede sentir el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el Arduino Programming Language (basado en Wiring) y el ArduinoevelopmentEnvironment (basado en Processing). Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador (Por ejemplo con Flash, Processing, MaxMSP, C++ Builder, etc).

Las placas se pueden ensamblar a mano o encargarlas preensambladas; el software se puede descargar gratuitamente. Los diseños de referencia del hardware (archivos CAD) están disponibles bajo licencia open-source, por lo que eres libre de adaptarlas a tus necesidades.

2.4.1 ¿Por Qué Arduino?

Hay muchos otros microcontroladores y plataformas microcontroladoras disponibles para computación física. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT'sHandyboard, y muchas otras ofertas de funcionalidad similar.

Todas estas herramientas toman los desordenados detalles de la programación de microcontrolador y la encierran en un paquete fácil de usar. Arduino también simplifica el proceso de trabajo con microcontroladores, pero ofrece algunas ventajas para profesores, estudiantes interesados sobre otros sistemas.

2.4.2 La placa Arduino

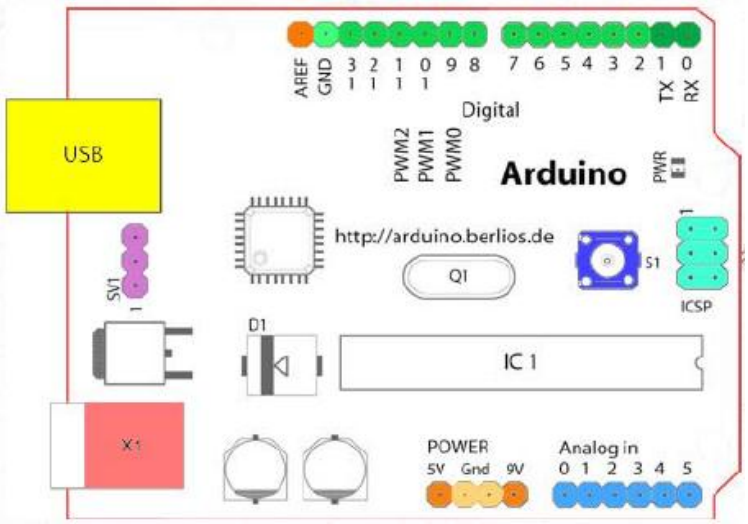


Figura 12. Placa Arduino

2.4.3 Alimentación

Dos alternativas:

- Puerto USB
- Regulador de Tensión: 5 a 15V

2.4.4 La placa Arduino – E/S

- 14 (hasta 20) pines de E/S digitales
- 6 entradas analógicas
- 6 salidas analógicas (PWM)

2.4.5 La placa Arduino - Comunicaciones

- Puerto serie: RX/TX
- Puerto USB (FTDI)
- ICSP

2.4.6 La placa Arduino – Otros

- Botón de reset
- Reloj a 16/20 Mhz

- Microcontrolador Atmega8/168 -> 8/16 Kb
- Bootloader
- Software

2.4.7Asignación de pines:

	Arduino 168/328		
Digital pin	Port	Analog pin	Usage
0	PD 0		USART RX
1	PD 1		USART TX
2	PD 2		Ext Int 0
3	PD 3		**PWM** T2B, Ext Int 1
4	PD 4		
5	PD 5		**PWM** T0B
6	PD 6		**PWM** T0A
7	PD 7		
8	PB 0		Input capture
9	PB 1		**PWM** T1A
10	PB 2		**PWM** T1B, SS
11	PB 3		**PWM** T2A, MOSI
12	PB 4		SPI MISO
13	PB 5		SPI SCK
14	PC 0	0	
15	PC 1	1	
16	PC 2	2	
17	PC 3	3	
18	PC 4	4	I2C SDA
19	PC 5	5	I2C SCL

Figura 13. Tabla de conexión

3. METODOLOGÍA

Para la elaboración de un dispositivo de enseñanza partimos de la idea de solucionar un problema, como lo es la comunicación de las personas sordomudas con las personas normoyentes. A partir del uso de fotosensores ubicados en las articulaciones de la mano, mediante la ayuda de un guante.

Las señales obtenidas están dirigidas hacia una tarjeta de adquisición de datos, en este caso usamos la tarjeta Arduino One debido a que ya es un microcontrolador con entorno desarrollado, es decir no necesitábamos agregar componentes extras a la placa y esto se vería reflejado en el ahorro de tiempo y presupuesto. Las señales capturadas por la tarjeta arduino, serán recibidas por la PC mediante el lenguaje de programación C++ Builder y con esto se lograra cerrar el lazo de trabajo que inicia en la misma PC.

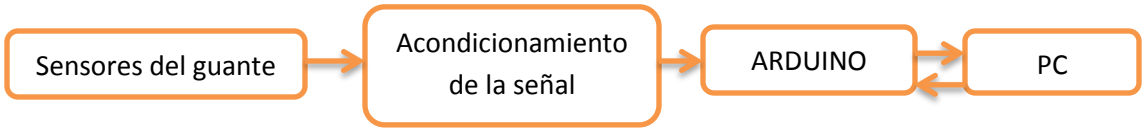


Figura 14. Diagrama a bloques del dispositivo

En la figura 13 en primer lugar tenemos los sensores ubicados en el guante en la posición de la unión de los huesos metacarpianos y falanges proximales (nudillos), y la unión de los huesos falanges proximal y media. Figura 14. Los sensores dejan pasar o no la señal, hacia el Arduino dependiendo si esta flexionado o no el sensor, la señal que proviene de la LDR en configuración de divisor de voltaje en configuración superior, la cual hace que llegue un nivel alto si existe voltaje o nivel bajo si no existe.

El arduino hace una interpretación de señal de los sensores y dependiendo de los niveles de lectura altos y bajos, este hace una pequeña sumatoria de acuerdo a los valores asignados a cada sensor, el valor obtenido se compara con el valor que se envía desde la PC y de ser correcto se repite la acción para cada letra.

En este caso no se usa ningún tipo conector para realizar la comunicación serial entre el arduino y la PC, debido a que la tarjeta arduino ya cuenta con una comunicación serial programable y la PC se configura desde el C++ Builder para realizar esta conexión, de esta forma se ahorra una etapa que tendría que usarse con cualquier otra tarjeta de adquisición de datos.

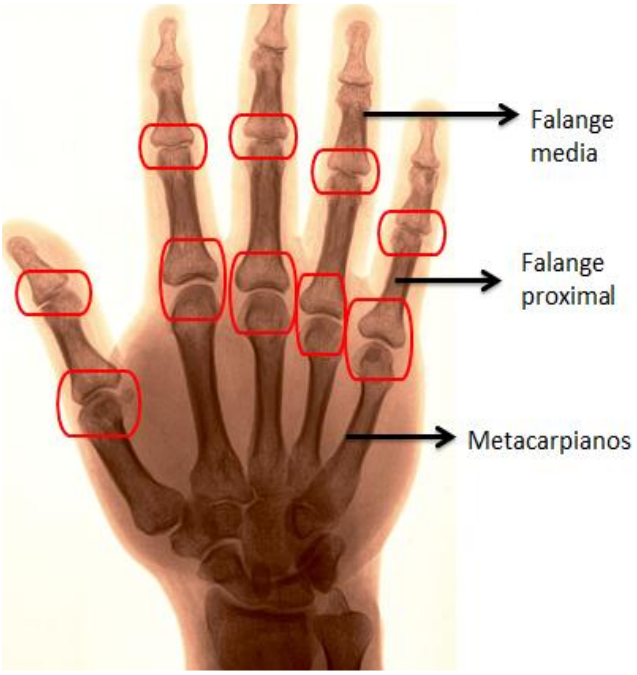


Figura 15. ubicacion de los sensores

3.1 Diseño del guante

Primero que nada tuvimos que buscar un guante adecuado para la creación de nuestro dispositivo, necesitábamos un guante con las siguientes características

- Precio accesible
- Que cubriera en su totalidad la mano hasta la muñeca
- Que tuviera doble capa (Guante y forro extra)

Conseguimos un guante marca Pirma el cual cumplió con los requisitos requeridos al cual se le abrió a través de su costura el forro superior, dejando expuesto el guante o forro interior para trabajar sobre él.

Se tomaron medidas de referencia en la posición de los nudillos donde nos interesaba poner los sensores.

Creación de los sensores.

Para los sensores usamos:

- 10 LDRs
- 10 LEDs de 3mm color verde
- 2 metros de cable para teléfono

- 1 metro de thermofit de 5mm
- 1.5 metros de thermofit de 6mm
- 1 resistencia de 100ohms a 1W

Se le cortaron las patas a las LDRs y a los LEDs procurado dejar aproximadamente 4 mm de largo para poder soldar los cables que alimentaran a dichos elementos. Utilizamos cable de teléfono por su flexibilidad (varios hilos), además que se compone de cuenta con 4 líneas de colores distintos (amarillo, café, rojo y verde)

Dicho cable se cortó de la siguiente manera

- 10 líneas de 1.2 metros de color amarillo
- 1 línea de 1.2 metros de color verde
- 1 línea de 1.2 metros de color café
- 1 línea de 1.2 metros de color rojo
- 10 líneas de 15 centímetros de color verde
- 10 líneas de 15 centímetros de color café
- 10 líneas de 15 centímetros de color rojo
- Conector DV9

Los cables rojos conducirán 5V para las LDR
Los cables amarillos conducirán la señal de las LDRs 5V o 0V
Los cables cafés serán la tierra de los LEDs
Los cables verdes conducirán 3.3 V para los LEDs

Después procedimos a soldar los cables los 10 cables d 15 cm fueron soldados en los ánodos de los LEDs, los 10 cables de 15 cm de color verde se usaron para los cátodos de los LEDs. Los 10 cables de 1.2 metros de color amarillo fueron soldados a una de las patas de las LDRs y los 10 cables de 15 cm de color rojo se usaron para la pata restante de las LDRs, como se muestra en la figura 16.

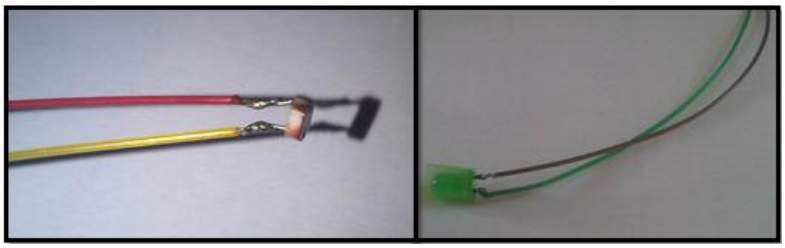


Figura 16. Conexión de la LDR y LED

Una vez con los elementos con sus propios cables continuamos con el corte del thermofit de 5mm en pedazos de 4 cm de largo cada uno, se pegó un LED y una LDR a los extremos de cada uno de los cortes de thermofit y aprovechando la cualidad del thermofit de contraerse cuando se calienta el silicón provocó su contracción en los elementos (LED y LDR) dejándolos asegurados e inmóviles.

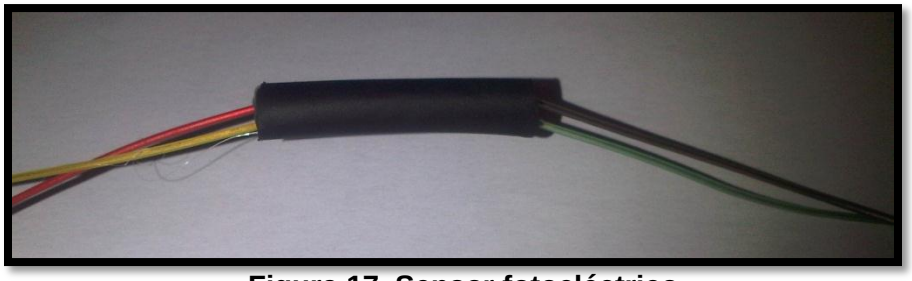


Figura 17. Sensor fotoeléctrico

Hecho lo anterior pasamos a costurar cada uno de los sensores en sus posiciones correspondientes en el guante previamente preparado para este punto como se ve en la figura 18.

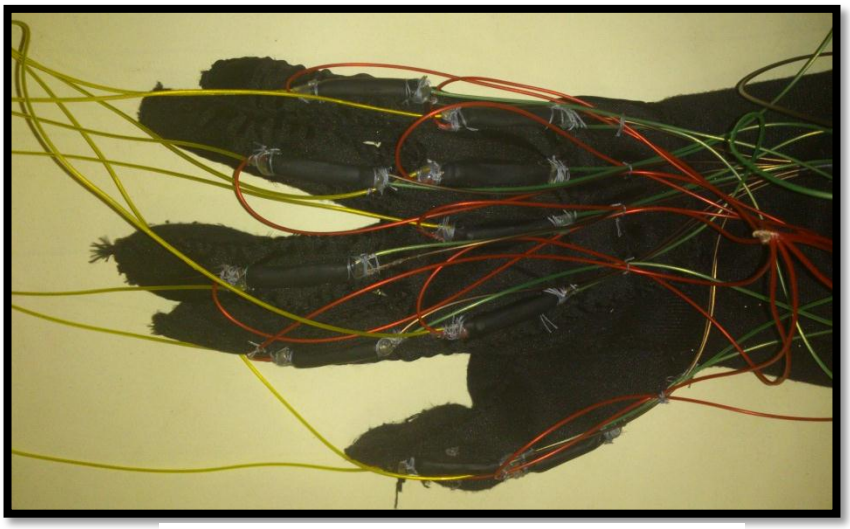


Figura 18. Colocación de los sensores

Después se procedió a soldar todos los cables verdes de 15 cm en un punto común y este punto común al cable de color verde de 1.2 metros de color verde previamente cortado. (El mismo proceso se repite con los cables de color café y rojo). Continuando se hizo una abertura a el guante en la parte inferior para la salida de todos los cables y se costuro el forro exterior del guante para que los sensores quedaran protegidos e invisibles para el usuario, una vez hecho esto se colocaron todos los cables exteriores dentro del thermofit de 6mm el cual se calentó para que se contrajera y quedara como un cable único dejando solo las puntas por fuera para soldarlas a un conector DB15.

3.2 Acondicionamiento De La Señal

Para el acondicionamiento de la señal debemos hacer el ya conocido divisor de tensión, de donde sacaremos la señal para conectar a nuestra entrada analógica.

Podemos conectarlo de dos maneras diferentes:

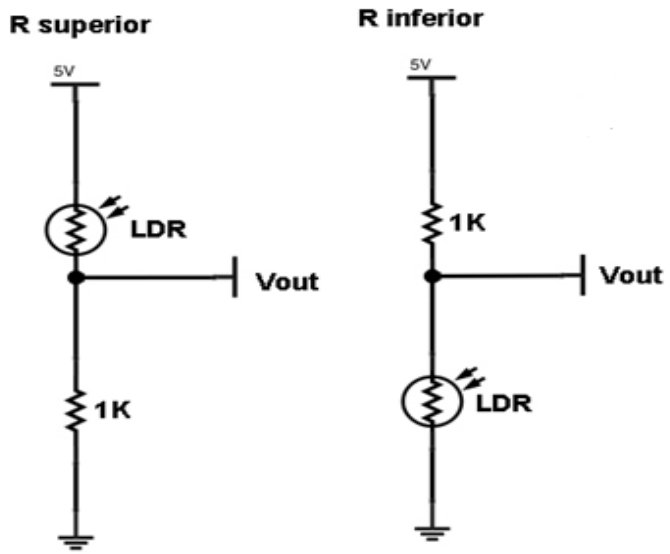


Figura 19. Divisor De Tension Inferior Y Superior

Si utilizamos el LDR como resistencia inferior del divisor de tensión, nos dará la tensión máxima cuando tengamos el LDR en plena oscuridad, ya que estará oponiendo el máximo de su resistencia al paso de la corriente derivándose esta por Vout al completo, si lo utilizamos como resistencia superior, el resultado será el inverso, tendremos la tensión máxima cuando esté completamente iluminado, ya que se comportará prácticamente como un circuito abierto, con una resistencia de 50Ω o 100Ω.

Para el caso de nuestro circuito decidimos usar la configuración superior del divisor de voltaje ya que necesitábamos que en su estado inicial del sensor nos arrojara un valor para ser leído e interpretado por el arduino y cada vez que este cambiara al estado de interrupción se acercara lo más posible a un cero lógico.

La conexión real de los sensores con el arduino se muestran el la figura 19. Algunos sensores necesitaban una resistencia más para incrementar la sensibilidad.

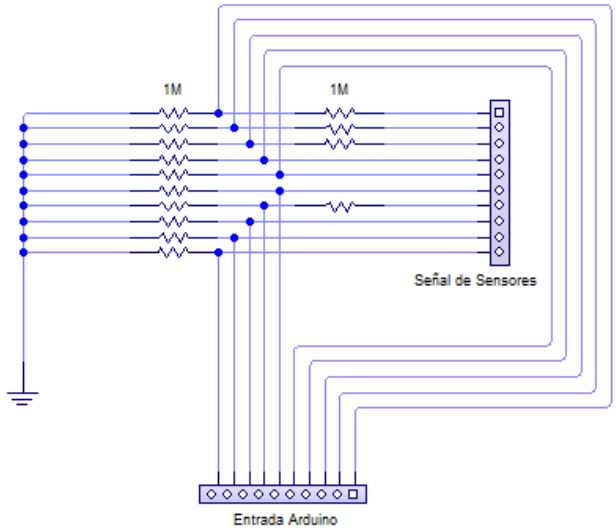


Figura 20. Conexion de los sensores y Arduino

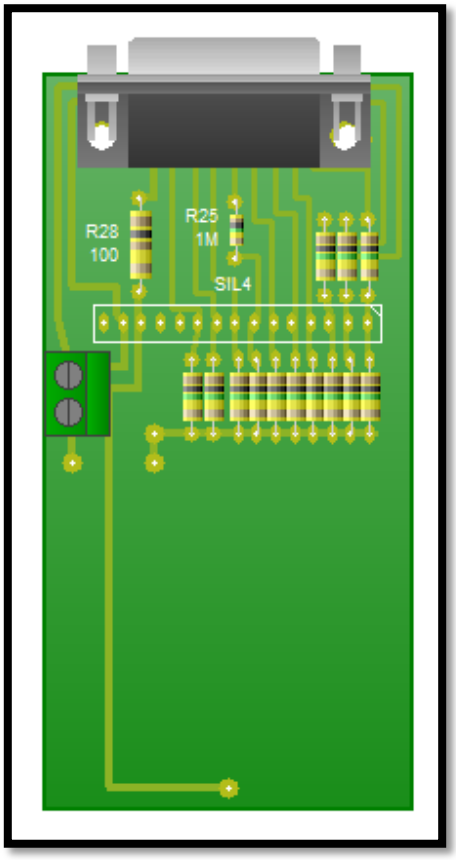


Figura 20.a. Vista del diseño PCB

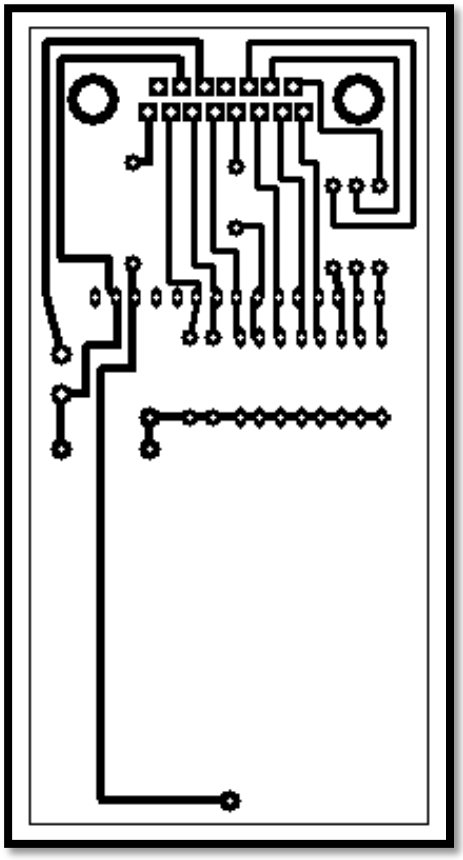


Figura 20.b. Vista del diseño (pistas)

3.3 Diseño De Fuente

Para el funcionamiento se tuvo que implementar una fuente variable, en este caso se decidió por una de este tipo debido a que tenemos contemplado una variación del voltaje para los sensores, es decir que como los sensores son elaborados a mano no tenemos el control total sobre la sensibilidad de los mismos, por ello se tenia que regular dependiendo de las pruebas que se realizaban, ajustando un voltaje en el cual la tarjeta arduino recibiría el “1” o “0” lógico dependiendo de la posición de los sensores. Al final el voltaje ideal es de 7.6 volts, a continuación se muestra el diseño de nuestra fuente.

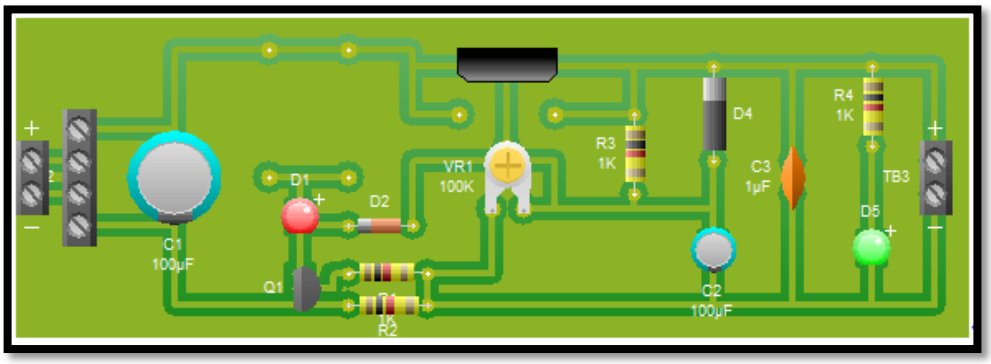
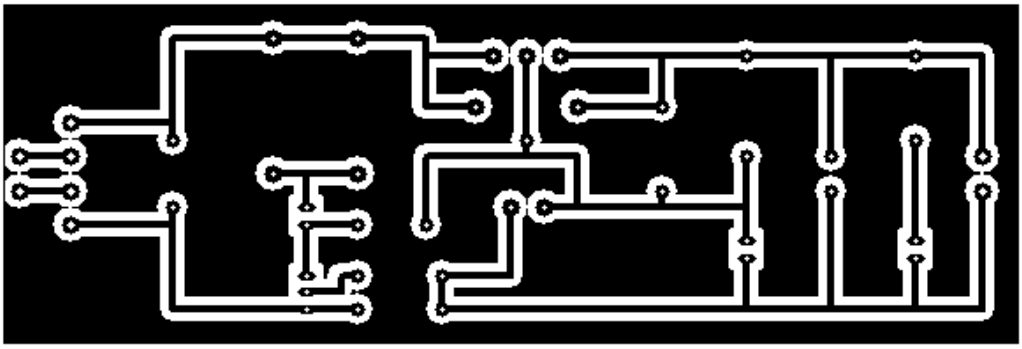


Figura 21.a. Diseño de la fuente PCB



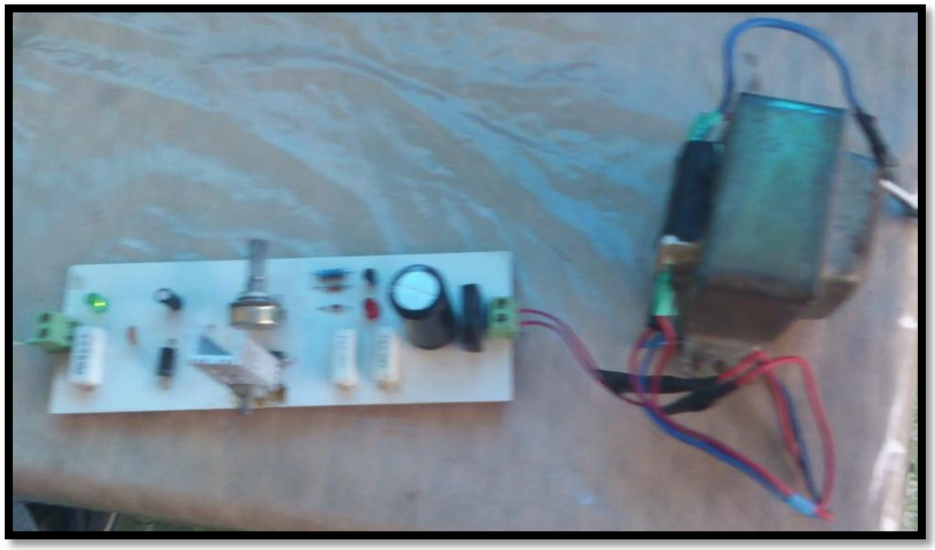
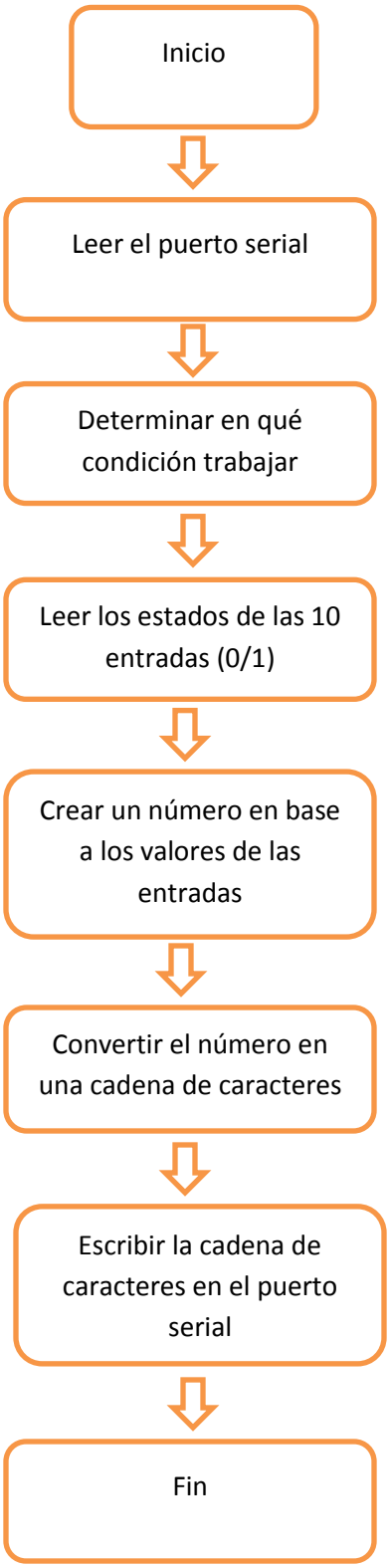


Figura 22. Vista final de la fuente

3.4 Programación de la tarjeta de adquisición de datos



El programa se diseñó para que hiciera las siguientes funciones:

Leer el puerto serial:

Son las letras enviadas por el programa de c++ que van desde la “a” hasta la “z”. Según la letra recibida el programa entrara a una serie de condiciones en las cuales se compara la letra y en la indicada entrara a seguir las instrucciones de dicha condición

Dentro de la condición:

Aquí se lee las 10 entradas que están en uso del arduino de la número 2 hasta la 11 y se obtiene un valor con los estados de las entradas el cual se crea de una suma de multiplicaciones por cada bit y se guarda en una variable. A continuación la ecuación para obtener el valor:

Valor= (bit1*1) + (bit2*2) + (bit3*4) + (bit4*6) + (bit5*8) + (bit6*10) + (bit7*12) + (bit8*14) + (bit9l*16) + (bit10*18);

Ese valor se compara con uno prestablecido para la posición de la mano y los bits que esta posición entrega al arduino, si el valor resulta igual que el de la condición el arduino escribe en el puerto serial una cadena de caracteres que puede ser xa, xb, xc, xd,…,xx, xy, xz según sea el caso

El valor prestablecido para cada condición se obtuvo de las posiciones de los sensores en la mano del portador de nuestro guante, tomando como bit 1 el nudillo inferior del dedo meñique, bit2 el nudillo superior del dedo meñique y de la misma forma hasta llegar al dedo pulgar; para lo cual se realizó una tabla para capturar esos valores que se muestran en la figura 21.

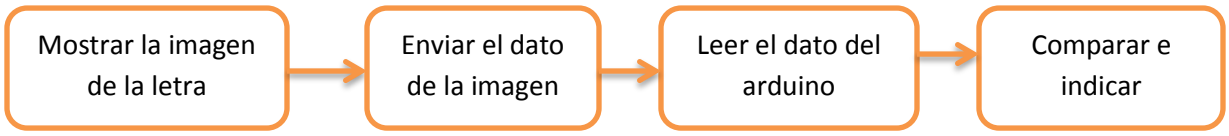
Figura 22. Tabla de valores

Columna1	1	2	3	4	5	6	7	8	9	10	Columna2
a	0	0	0	0	0	0	0	0	1	1	34
b	1	1	1	1	1	1	1	1	0	1	75
c	1	0	1	0	1	0	1	0	1	0	41
d	1	0	1	0	1	0	1	1	1	0	55
e	1	0	1	0	1	0	1	0	0	0	25
f	1	1	1	1	1	1	1	0	1	0	59
g	0	0	0	0	0	0	1	1	1	1	60
h	0	0	0	0	1	1	1	1	1	1	78
i	1	1	1	0	0	0	1	0	1	0	35
j	1	1	1	0	0	0	1	0	1	0	35
k	0	0	0	0	0	1	1	1	0	1	54
l	0	0	0	0	0	0	1	1	1	1	60
m	1	1	0	1	0	1	1	1	0	1	63

n	0	0	0	0	0	1	0	1	0	1	42
ñ	0	0	0	0	0	1	0	1	0	1	42
o	1	0	1	0	0	0	1	1	1	0	47
p	0	0	0	0	0	1	1	1	1	1	70
q	0	0	0	0	0	0	0	1	1	1	48
r	1	0	1	0	1	1	1	1	0	1	67
s	0	0	0	0	0	0	0	0	1	0	16
t	0	0	0	0	0	0	1	1	0	1	44
u	1	0	1	0	1	1	1	1	0	1	67
v	1	0	1	0	1	1	1	1	0	1	67
w	1	0	1	1	1	1	1	1	0	1	73
x	0	0	0	0	0	0	1	0	0	1	30
y	1	1	0	0	0	0	0	0	1	1	37
z	0	0	0	0	0	0	1	1	0	1	44

3.5 Programación de la interfaz grafica

Desarrollo de la interfaz grafica



La función del programa desarrollado en C++ Builder es mostrar al usuario las lecciones y repasos que incluyen las diferentes letras del abecedario, de una manera amigable y sencilla.

La primera etapa del programa esta diseñada para que dependiendo de la lección o repaso seleccionado se muestre una imagen relacionada con una letra del lenguaje de señas mexicano (LSM) figura 21, en caso de seleccionar una lección se muestran la imagen de las letras en forma secuencial excepto en la lección aleatoria, en caso de seleccionar un repaso se muestra una letra del alfabeto español.

Después el usuario comprobará si la posición de su mano es la correcta mediante el botón comprobar, es aquí cuando se ejerce la función de enviar el dato incluido en el componente ComPort, el dato es enviado al Arduino mediante la comunicación serial y hay mismo también es ejercida la función leer en la cual el programa de la PC recibe la información generada en el arduino.

La ultima etapa es donde se realiza la comparación del dato enviado y el dato recibido, de estar recibiendo la información correcta la interfaz mostrara un mensaje en el cual se indica la correcta posición de la mano y dejara al usuario continuar con

la lección como se muestra en la figura 21, de ser incorrecta no se podrá continuar con la lección o repaso hasta que el obtenga la posición adecuada de la mano.

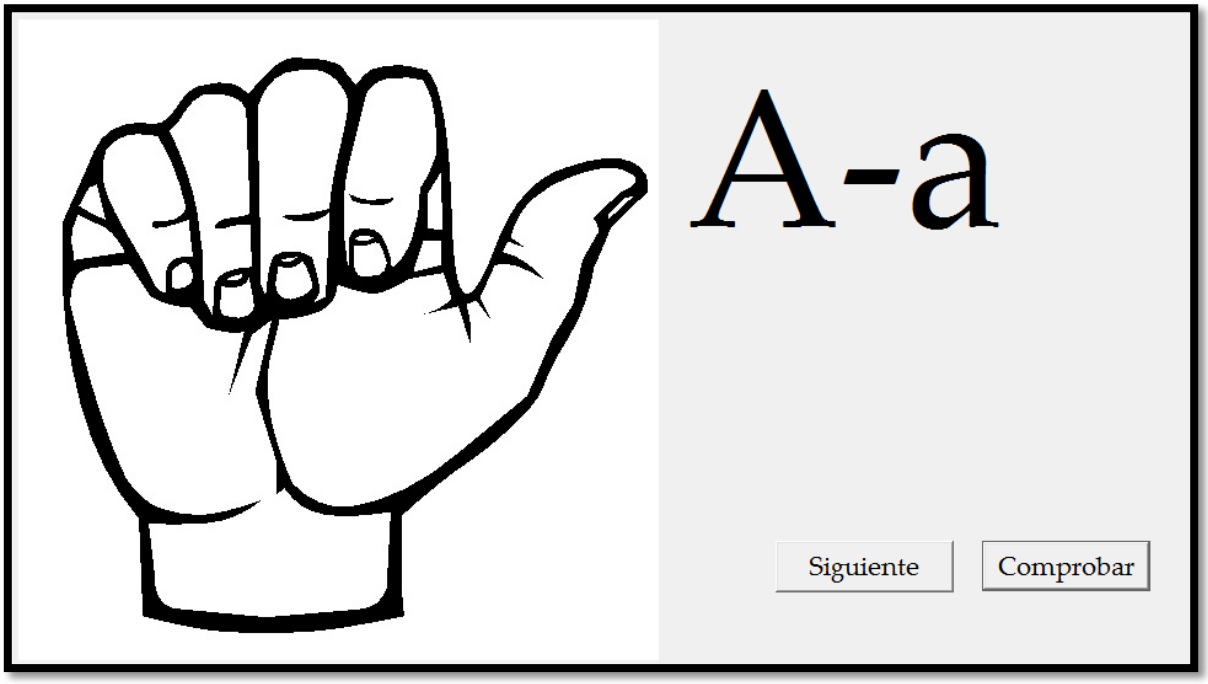


Figura 23. Presentación de una letra y botones

4. PRUEBAS

Para verificar el correcto funcionamiento del dispositivo se realizaron varias pruebas que detallaremos a continuación, las primeras pruebas consintieron en verificar el funcionamiento del programa en la tarjeta arduino después se realizo una preliminar que se realizo de forma informal, es decir con el dispositivo desensamblado y con el guante aun sin costurar, esto con el objeto de que se pudiese visualizar el funcionamiento del mismo. La prueba final se realizo con todo el dispositivo ensamblado de la forma en que se tenía contemplado para su término.

La primera prueba constaba en conectar el guante y verificar el programa en la tarjeta arduino y la comunicación con el mismo, en esta prueba verificamos que los sensores respondían a la forma de la mano y arrojaban los datos correspondientes según la seña, para cada letra se asigno una variable, en el caso de la “a” si estaba el guante en la posición correcta el programa mostraba la variable “xa”, en el caso de la “b” mostraba la variable “xb” y así sucesivamente agregando la variable “x” a cada letra del abecedario.

A continuación se muestran las imágenes de las primeras pruebas en donde observaremos la comunicación y a variable dependiendo de la letra indicada.

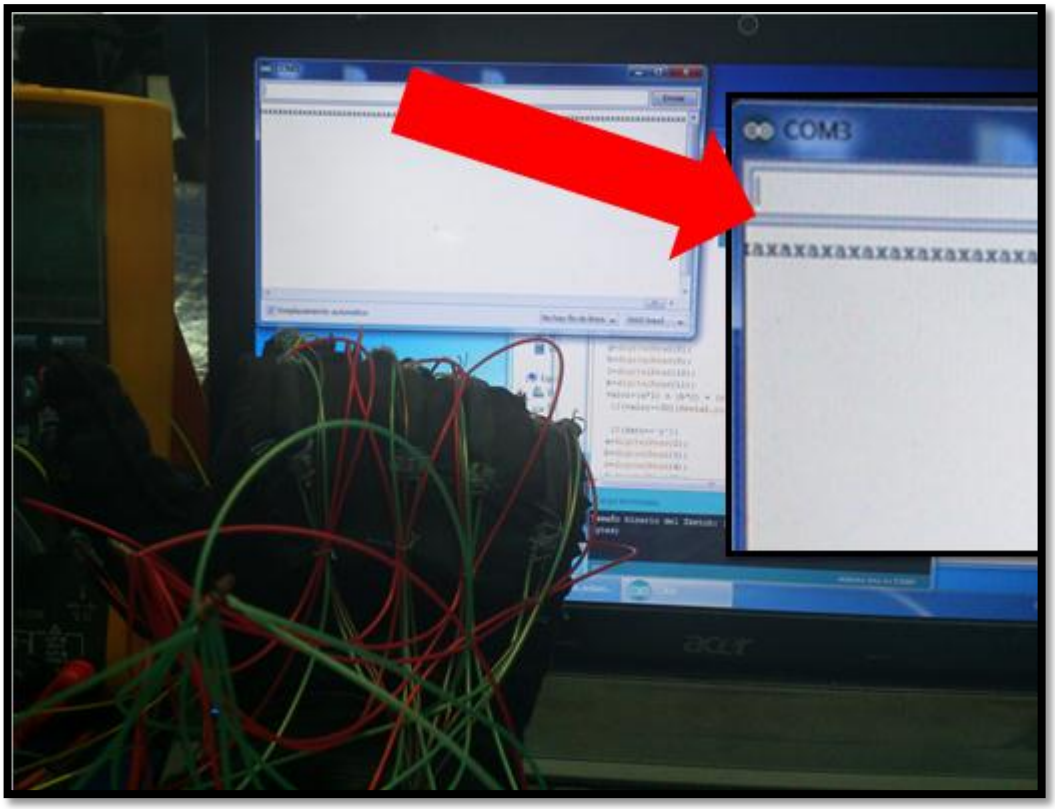


Figura 24. Prueba de la imagen a

La prueba preliminar se realizo como se menciono antes de manera informal, es decir con el dispositivo desconectado y el guante sin costurar, a continuación se muestra una imagen de dicha prueba.

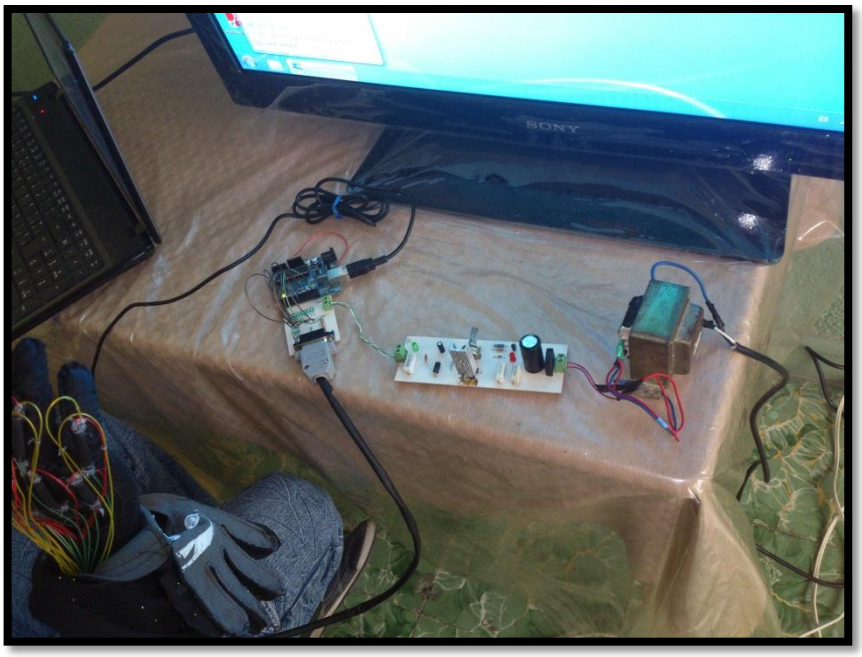


Figura 25. Prueba preliminar

La ultima prueba se realizo con todo el dispositivo completo y ya con la forma en que estaba diseñado, es decir con el modulo principal armado y con el guante terminado.

Figura 26. Prueba final

5. RESULTADOS

Los resultados obtenidos son los que se plantearon al inicio del proyecto, con todo el trabajo pudimos concluir la programación de la interfaz grafica de la PC, la programación de la tarjeta de adquisición de datos y comunicación serial, y el guante con los sensores.

Para la interfaz grafica se utilizo el programa C++ Builder y no Matlab, debido a la familiaridad que teníamos con este, y además cubre al 100 % las funciones que nosotros utilizamos para la realización del dispositivo, es decir en concreto cuenta con la conexión serial que en este caso era de suma importancia. La figura que se muestra a continuación es la pantalla principal del programa final.

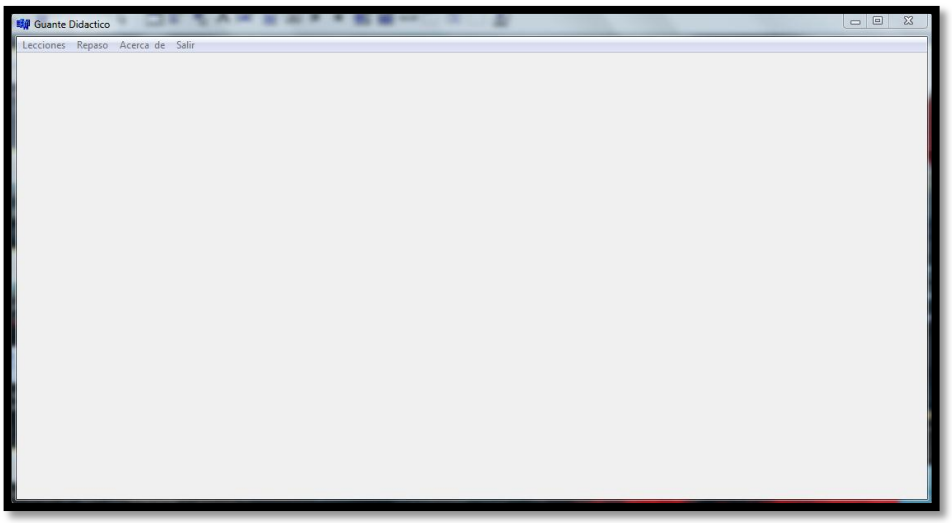


Figura 27. Ventana principal

La tarjeta de adquisición de datos que utilizamos es la tarjeta arduino one, la elección en este caso es que, ya es un dispositivo desarrollado y que además contaba con todas las especificaciones que necesitamos, cuenta con las entradas digitales y con la alimentación deseadas, además cuenta con la comunicación mediante USB. Para la conexión de los sensores con la tarjeta arduino diseñamos un circuito de protección basado en resistencia en configuración de divisor de voltaje como se muestra a continuación.

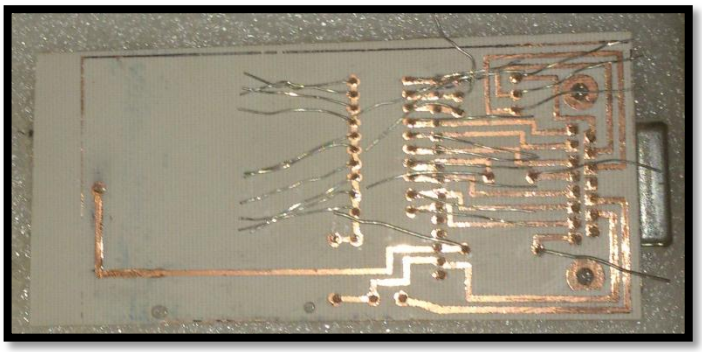


Figura 28. Circuito de protección (pistas)

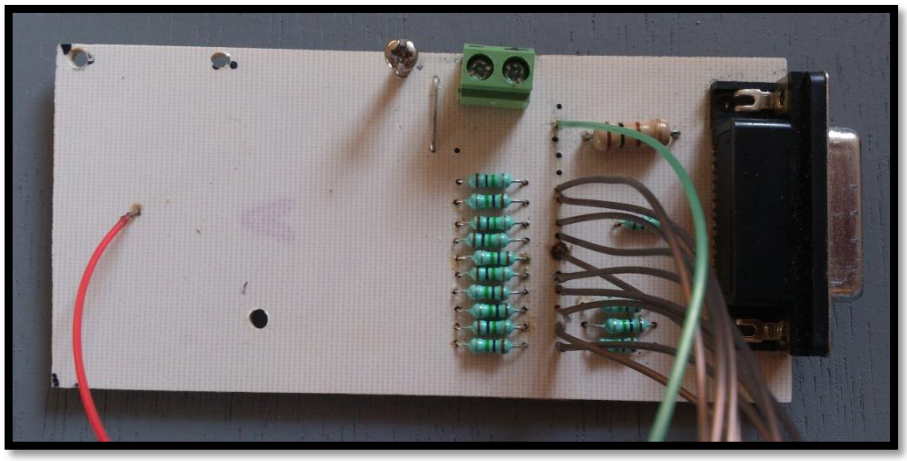


Figura 29. Circuito de protección (elementos)

El funcionamiento final del guante fue el ideal, aunque se realizaron modificaciones al voltaje que conducían los sensores, con el fin de que la tarjeta arduino recibiera perfectamente tanto el nivel bajo “0” (cero volts) así como también el nivel alto “1” (5 volts). A continuación se muestra la costura del guante para su estado final.

Figura 30. Costura del guante.

Figura 31. Guante terminado

El ensamble final del dispositivo se realizo en una caja de madera con dos niveles en el primero se coloco la fuente del dispositivo con su conexión hacia los 110 V y con un Switch de encendido.

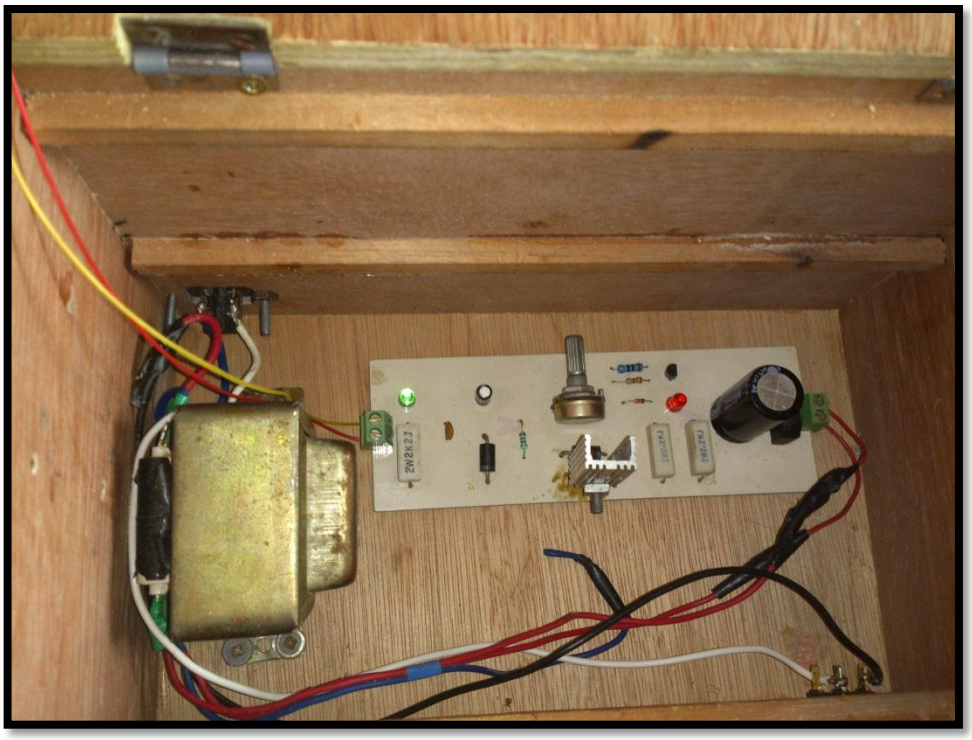


Figura 32. Instalación de la fuente.

En el segundo nivel se monto el arduino junto con el circuito de protección con la conexión DB15 hembra para la conexión de los sensores.

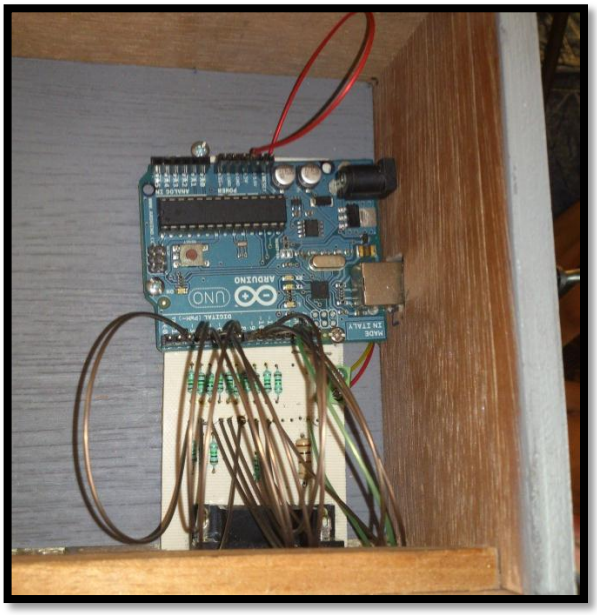


Figura 33. Instalación de la tarjeta arduino y circuito de protección

Como resultado final del dispositivo se muestran las siguientes imágenes con el modulo principal terminado.

Figura 34. Vista frontal (conexión del guante y botón de encendido)

Figura 35. Vista lateral (conector USB)

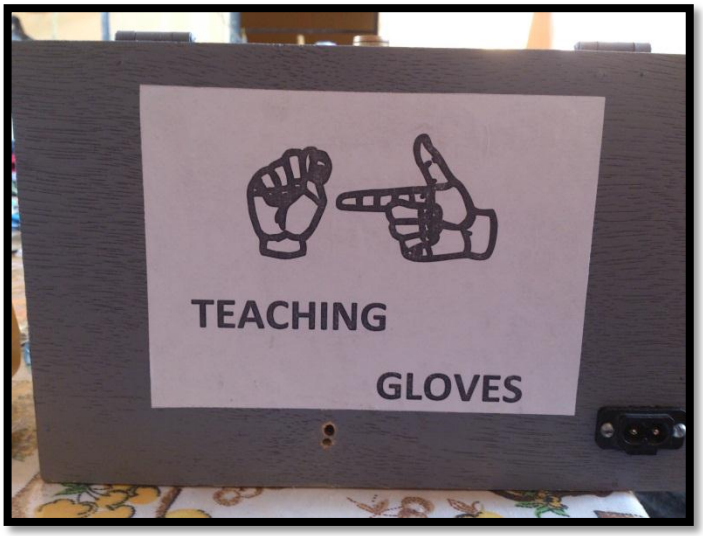


Figura 36. Vista trasera (entrada de alimentación de la fuente)

6. CONCLUSIÓN

Es importante el avance en la enseñanza de lenguas sobre todo cuando se nota el nivel de discriminación que se aplica a las personas sordomudas principalmente al momento de solicitar un empleo, por lo cual tratamos de fomentar que el lenguaje de señas sea tan universal como el hablado ya que así se disminuiría la poca comprensión que tenemos los norma oyentes ante una persona sordomuda. Y de esta misma forma ir reduciendo la discriminación al entender lo que realmente nos expresan y contestar de forma correcta a sus palabras.

La enseñanza de forma didáctica a través de una computadora es también relevante debido que hoy en día es más accesible este método que cualquier curso especializado en el tema, evitando gastos de transporte y contratación de algún maestro. Actualmente los cursos didácticos han sido muy exitosos ya que demuestran la sencillez y flexibilidad con la que se puede aprender cualquier cosa desde la comodidad de su hogar

La implementación de sensores a base de LEDs y LDRs constituye un avance en la rama de la optoelectrónica debido a que las LDRs son un componente complicado de controlar más no de manejar. Se torna difícil de controlar debido a que presenta una comportamiento exponencial que se incrementa o decrementa muy rápidamente con la presencia o ausencia de luz debido a esto se tienen que hacer configuraciones como divisores de voltaje con una determinada resistencia para cada una diferente situaciones de uso que se puedan presentar, convirtiendo nuestra LDR más sensible o menos sensible según se requiera.

El uso de componentes de bajo precio y regidos ante las normas internacionales ISO como lo es el arduino se torna factible y viable de utilizar ya que nos permite usarlo como tarjeta de adquisición de datos y como un lenguaje de programación donde se es posible manipular las señales que estamos recibiendo además de poder escribir por el puerto serial datos y todo a través de una conexión USB que actualmente es la mas utilizada.

REFERENCIAS BIBLIOGRAFICAS

- http://elvex.ugr.es/decsai/builder/
- http://es.wikipedia.org/wiki/C%2B%2BBuilder
- http://www.disca.upv.es/aperles/comport/GuiaComport.html
- http:/ /arduino-1.0.2-windows/arduino-1.0.2/reference/Serial.html
- http://www.programnation.com/arduino-logger/
- http://es.wikipedia.org/wiki/Fotorresistencia
- http://es.wikipedia.org/wiki/Led

8. ANEXOS

8.1 Programación C++ BUILDER

Esta es una parte de la programación de la lección 1, las otras lecciones son similares, lo único que cambia es la letra que se muestra y oculta.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Imagena->Visible=false;
Labela->Visible=false;
Button1->Visible=false;
Button2->Visible=true;
Imagenb->Visible=true;
Labelb->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Imagenb->Visible=false;
Labelb->Visible=false;
Button2->Visible=false;
Button3->Visible=true;
Imagenc->Visible=true;
Labelc->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
Imagenc->Visible=false;
Labelc->Visible=false;
Button3->Visible=false;
Button4->Visible=true;
Imagend->Visible=true;
Labeld->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
Imagend->Visible=false;
Labeld->Visible=false;
Button4->Visible=false;
Button5->Visible=true;
Imagene->Visible=true;
Labele->Visible=true;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
Imagene->Visible=false;
Labele->Visible=false;
Button5->Visible=false;
Button6->Visible=true;
Imagenf->Visible=true;
Labelf->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
Imagenf->Visible=false;
Labelf->Visible=false;
Button6->Visible=false;
Button7->Visible=true;
Imageng->Visible=true;
Labelg->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{
Imageng->Visible=false;
Labelg->Visible=false;
Button7->Visible=false;
Button8->Visible=true;
Imagenh->Visible=true;
Labelh->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button8Click(TObject *Sender)
{
Imagenh->Visible=false;
Labelh->Visible=false;
Button8->Visible=false;
Button9->Visible=true;
Imageni->Visible=true;
Labeli->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button9Click(TObject *Sender)
{
Imageni->Visible=false;
Labeli->Visible=false;
Button9->Visible=false;
Button1->Visible=true;
Imagena->Visible=true;
Labela->Visible=true;
}
```

Esta es la programación de la lección de repaso 1, las otras lecciones de repaso son similares, lo único que cambia son las letras que se muestran.

```
void __fastcall TForm1::BR1Click(TObject *Sender)
{
BR1->Caption="Siguiente";
int x;
int variaciones[] = {1,2,3,4,5,6,7,8,9};
x = variaciones [random(sizeof (variaciones) / sizeof(int))];
        if (x==1)
        {
        Imagenv->Visible=false;
         LeccionAleatorioAZ1->Click();
         Bale->Visible=false;
         BR1->Visible=true;
         Imagena->Visible=true;
         Labela->Visible=true;

        }
        else
         {
        if (x==2) {
        Imagenv->Visible=false;
         LeccionAleatorioAZ1->Click();
         Bale->Visible=false;
         BR1->Visible=true;
         Imagenb->Visible=true;
         Labelb->Visible=true;
        }
         else
         {
        if (x==3) {
        Imagenv->Visible=false;
         LeccionAleatorioAZ1->Click();
         Bale->Visible=false;
         BR1->Visible=true;
         Imagenc->Visible=true;
         Labelc->Visible=true;
        }
         else
         {
        if (x==4) {
        Imagenv->Visible=false;
         LeccionAleatorioAZ1->Click();
        Bale->Visible=false;
         BR1->Visible=true;
         Imagend->Visible=true;
         Labeld->Visible=true;
        }
         else
```

```
        {
       if (x==5) {
       Imagenv->Visible=false;
        LeccionAleatorioAZ1->Click();
      Bale->Visible=false;
        BR1->Visible=true;
        Imagene->Visible=true;
        Labele->Visible=true;
       }
        else
        {
       if (x==6) {
       Imagenv->Visible=false;
        LeccionAleatorioAZ1->Click();
        Bale->Visible=false;
        BR1->Visible=true;
        Imagenf->Visible=true;
        Labelf->Visible=true;
       }
        else
        {
       if (x==7) {
       Imagenv->Visible=false;
        LeccionAleatorioAZ1->Click();
        Bale->Visible=false;
        BR1->Visible=true;
        Imageng->Visible=true;
        Labelg->Visible=true;
       }
        else
        {
       if (x==8)  {
        LeccionAleatorioAZ1->Click();
        Bale->Visible=false;
        BR1->Visible=true;
        Imagenh->Visible=true;
        Labelh->Visible=true;
       }
        else
        {
       if (x==9)  {
       Imagenv->Visible=false;
        LeccionAleatorioAZ1->Click();
     Bale->Visible=false;
        BR1->Visible=true;
        Imageni->Visible=true;
        Labeli->Visible=true;
       }
       }}}}}}}}
}
```

8.2 Programación Tarjeta Arduino One

Esta es la programación de la función de la comunicación serial y de las 5 primeras letras del abecedario LSM, la programación de las demás letras es repetitiva con la única diferencia

```
char dato;
int a,b,c,d,e,f,g,h,l,k,A,B,C,D,E,F,G,H,L,K;
int valor=0;
void setup(){
 Serial.begin(9600);
 pinMode(2,INPUT);
 pinMode(3,INPUT);
 pinMode(4,INPUT);
 pinMode(5,INPUT);
 pinMode(6,INPUT);
 pinMode(7,INPUT);
 pinMode(8,INPUT);
 pinMode(9,INPUT);
 pinMode(10,INPUT);
 pinMode(11,INPUT);
}

void loop()
{
    interrupts();
   if(Serial.available())
   {dato=Serial.read();}

   if(dato=='a'){
  a=digitalRead(2);
  b=digitalRead(3);
  c=digitalRead(4);
  d=digitalRead(5);
  e=digitalRead(6);
  f=digitalRead(7);
  g=digitalRead(8);
  h=digitalRead(9);
  l=digitalRead(10);
  k=digitalRead(11);
  valor=(a*1) + (b*2) + (c*4) + (d*6) + (e*8) + (f*10) + (g*12) + (h*14) + (l*16) + (k*18);
   if (valor==34){Serial.print("xa");delay (50);}}

   if(dato=='b'){
  a=digitalRead(2);
  b=digitalRead(3);
  c=digitalRead(4);
  d=digitalRead(5);
  e=digitalRead(6);
  f=digitalRead(7);
```

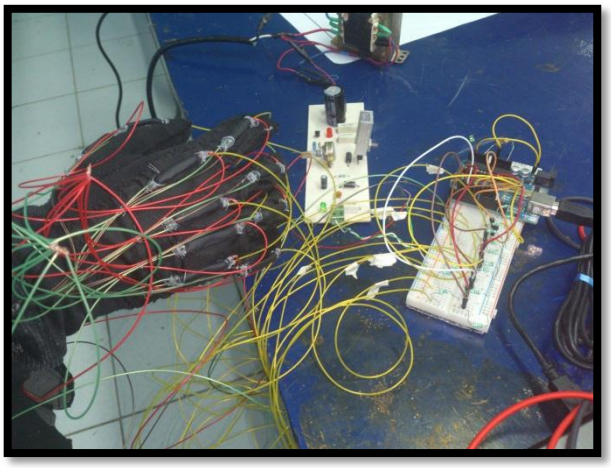
```
g=digitalRead(8);
h=digitalRead(9);
l=digitalRead(10);
k=digitalRead(11);
valor=(a*1) + (b*2) + (c*4) + (d*6) + (e*8) + (f*10) + (g*12) + (h*14) + (l*16) + (k*18);
if (valor==75){Serial.print("xb");delay (50);} }

 if(dato=='c'){
a=digitalRead(2);
b=digitalRead(3);
c=digitalRead(4);
d=digitalRead(5);
e=digitalRead(6);
f=digitalRead(7);
g=digitalRead(8);
h=digitalRead(9);
l=digitalRead(10);
k=digitalRead(11);
valor=(a*1) + (b*2) + (c*4) + (d*6) + (e*8) + (f*10) + (g*12) + (h*14) + (l*16) + (k*18);
 if (valor==41){Serial.print("xc");delay (50);}} //igual o

 if(dato=='d'){
a=digitalRead(2);
b=digitalRead(3);
c=digitalRead(4);
d=digitalRead(5);
e=digitalRead(6);
f=digitalRead(7);
g=digitalRead(8);
h=digitalRead(9);
l=digitalRead(10);
k=digitalRead(11);
valor=(a*1) + (b*2) + (c*4) + (d*6) + (e*8) + (f*10) + (g*12) + (h*14) + (l*16) + (k*18);
 if (valor==26){Serial.print("xd");delay (50);}} //igual z

 if(dato=='e'){
a=digitalRead(2);
b=digitalRead(3);
c=digitalRead(4);
d=digitalRead(5);
e=digitalRead(6);
f=digitalRead(7);
g=digitalRead(8);
h=digitalRead(9);
l=digitalRead(10);
k=digitalRead(11);
valor=(a*1) + (b*2) + (c*4) + (d*6) + (e*8) + (f*10) + (g*12) + (h*14) + (l*16) + (k*18);
 if(valor==25){Serial.print("xe");delay (50);}}
```

8.3 Fotografías.

Figura 37 y Figura 38. Elaboración de los sensores

Figura 39. Conector DB15 del guante

Figura 40. Perforación de la placa



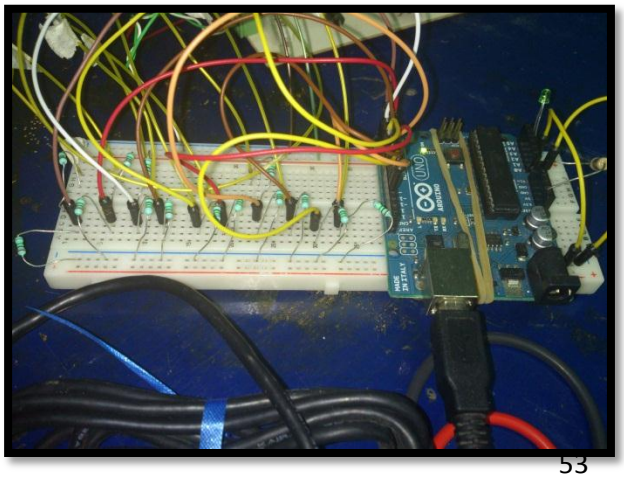


Figura 41 y 42 . Prueba con Protoboard