UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA ELECTRÓNICA

Diseño programación e implementación de una impresora braille

Trabajo de graduación previo a la obtención del título de:
INGENIERO ELECTRÓNICO.

Autor:

MIGUEL ÁNGEL MALDONADO GUTIÉRREZ

Director:

SIMÓN BOLÍVAR MÉNDEZ RENGEL

CUENCA-ECUADOR
2015

DEDICATORIA

Este trabajo de graduación se lo dedico a mi madre, abuelos y hermanos, quienes han sido el pilar de mi formación brindándome su apoyo incondicional. A mi familia en general, que ha estado pendiente de mi progreso desde la escuela hasta la universidad, a todos ellos muchas gracias.

Miguel Ángel.

AGRADECIMIENTO

Fundamentalmente a nuestro señor Dios que nos permite desarrollarnos y progresar en este mundo cambiante.

De manera muy especial a mi director de tesis, Ingeniero Bolívar Méndez Rengel, docente de la Universidad del Azuay quién demostró su paciencia y optimismo para conmigo en el desarrollo de este trabajo. También agradecer al Ingeniero Francisco Vásquez y Licenciado Leopoldo Vásquez, quienes conformaron el tribunal y aportaron con sus conocimientos.

Finalmente agradezco a mi familia y amigos que con su apoyo me dieron impulso para seguir adelante y conseguir esta importante meta en mi vida.

Miguel Ángel.

DISEÑO PROGRAMACIÓN E IMPLEMENTACIÓN DE UNA IMPRESORA BRAILLE

RESUMEN

En el desarrollo de este trabajo de graduación se presenta el procedimiento para el diseño, programación y construcción de un sistema de impresión braille, para beneficio directo de las personas que tengan un alto grado de ceguera. Utilizando la plataforma LabVIEW, se diseña la interface en la cual, se podrán ingresar los caracteres que se marcarán en celdas braille sobre una hoja de papel de polietileno. Arduino es la plataforma en donde se desarrolla el programa para el control de los *drivers*. Éstos a su vez comandarán los motores paso a paso, cuyos propósitos son: desplazar el solenoide y arrastrar la hoja. Dicho arrastre será comandado a través de un sensor de luz. Las distancias de arrastre serán ingresadas en el código Arduino y podrán ser modificadas únicamente usando el código fuente.

Palabras Clave: braille, LabVIEW, Arduino, drivers, interface.

Ing. Hugo Marcelo Torres Salamea
Director de Escuela

Ing. Simón Bolívar Méndez Rengel
Director de Tesis

Miguel Ángel Maldonado Gutiérrez.
Autor

DESIGN, PROGRAMMING AND IMPLEMENTATION OF A BRAILLE PRINTER

ABSTRACT

This graduation work presents the procedure for the design, planning and construction of a Braille printing system, for the direct benefit of people with a high degree of blindness. By means of the LabVIEW platform, we designed the interface, where the characters will be marked in Braille cells over a polyethylene sheet. Arduino is the platform where the program for monitoring the drivers is developed. These in turn, will command the stepper motors, whose purposes are: move the solenoid and drag the sheet. Such drag must be operated by a light sensor. Dragging distances will be entered into the Arduino code, and may be modified only by using the source code.

Keywords: Braille, Labview, Arduino, Drivers, Interface.

Ing. Hugo Marcelo Torres Salamea
School Director

Ing. Simón Bolívar Méndez Rengel
Thesis Director

Miguel Angel Maldonado Gutiérrez.
Author

UNIVERSIDAD DEL AZUAY
Dpto. Idiomas

Translated by,
Lic. Lourdes Crespo

ÍNDICE DE CONTENIDOS

CAPÍTULO 2: HARDWARE PARA LA ELABORACIÓN DEL SISTEMA DE IMPRESIÓN BRAILLE

CAPÍTULO 3: ANÁLISIS DEL SOFTWARE APLICADO

CAPÍTULO 4: DESARROLLO DEL PROYECTO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

Maldonado Gutiérrez, Miguel Ángel
Trabajo de Graduación
Ing. Simón Bolívar Méndez Rengel
Julio 2015

"DISEÑO PROGRAMACIÓN E IMPLEMENTACIÓN DE UNA IMPRESORA BRAILLE"

INTRODUCCIÓN

En la actualidad las personas con capacidades diferentes, no todas pueden acceder a la información de los libros que se imprimen normalmente a través de sistemas de tinta, cinta, o láser, entre otros, limitando de esta forma a las personas que no poseen el grado necesario de visión para una lectura normal.

Gran parte de los medios escritos de comunicación, tales como: revistas, periódicos, editoriales, entre otros, no han atendido de una forma satisfactoria, la necesidad de acceder a la información para las personas que poseen un alto grado de ceguera.

Este trabajo de graduación tiene por objeto, superar esta adversidad de las personas con discapacidad visual, para que puedan acceder a la información en textos a la cual no pueden hacerlo normalmente. El valor del sistema braille en el mundo de las personas con un alto grado de déficit visual es infinito, pues, éste les permite acceder a la información a su manera, contribuyendo a la adquisición del conocimiento.

Considerando la fácil adquisición de los elementos eléctricos y componentes electrónicos, dentro de la ciudad y el país, sí es viable el diseño y elaboración del sistema de control para la impresora.

CAPÍTULO I

INTRODUCCIÓN Y CONCEPTOS GENERALES

El sistema braille, que se desarrollará conjuntamente con el presente trabajo de graduación, tiene como objetivo principal el de brindar una alternativa a las personas con un alto grado de déficit de visión, para que puedan acceder al conocimiento a través su sentido del tacto, para con esto elevar su calidad de vida.

Para el desarrollo del sistema se partirá desde el estudio previo de las características y por menores del código braille para entender su forma de representar cada carácter. Posterior a esto, se necesitará diseñar la maqueta que abarcará el prototipo. Serán necesarios estudios sobre interfaces Arduino, programación para que la interface sea de fácil acceso y manejo de cualquier persona a la que está dedicada este trabajo. El usuario podrá manipular el sistema desde una computadora, la cual poseerá la interface y estará conectada al sistema de impresión braille.

Figura 1.1. Esquema del Sistema Braille
Fuente: Autor

1.1. Definiciones básicas

Las definiciones básicas son importantes para familiarizarse con la terminología apropiada y no cometer errores, puesto que las exigencias sociales imponen una comunicación apropiada y precisa. Los primeros términos a definir son:

- ✓ Deficiencia
- ✓ Discapacidad
- ✓ Minusvalía

1.1.1. Deficiencia

"Dentro de la experiencia de salud, una deficiencia es toda pérdida o anormalidad de una estructura o función psicológica, fisiológica o anatómica. La deficiencia se caracteriza por pérdidas o anormalidades que pueden ser temporales o permanentes, entre las que se incluye la existencia o aparición de una anomalía, defecto o pérdida producida en un miembro, órgano, tejido u otra estructura del cuerpo, incluidos los sistemas propios de la función mental. La deficiencia representa la exteriorización de un estado patológico y, en principio, refleja perturbaciones a nivel del órgano" (Sambrano, 2007).

1.1.2. Discapacidad

"Dentro de la experiencia de la salud, una discapacidad es toda restricción o ausencia (debida a una deficiencia) de la capacidad de realizar una actividad en la forma o dentro del margen que se considera normal para un ser humano. La discapacidad se caracteriza por excesos o insuficiencias en el desempeño del comportamiento en una actividad rutinaria normal, los cuales pueden ser temporales o permanentes, reversibles o irreversibles y progresivos o regresivos. Las discapacidades pueden surgir como consecuencia directa de la deficiencia o como una respuesta del propio individuo, sobre todo la psicológica, a deficiencias físicas, sensoriales o de otro tipo. La discapacidad representa la objetivación de una deficiencia y, en cual tal, refleja alteraciones a nivel de la persona" (Sambrano, 2007).

1.1.3. Minusvalía

"La minusvalía está en relación con el valor atribuido a la situación o experiencia de un individuo cuando se aparta de la norma. Se caracteriza por la discordancia entre el rendimiento o estatus del individuo y las expectativas del individuo mismo y del grupo en concreto al que pertenece. La minusvalía representa, pues, la socialización de una deficiencia o discapacidad, y en cuanto tal refleja las consecuencias –culturales, sociales, económicas y ambientales- que para el individuo se derivan de la presencia de la deficiencia o discapacidad" (Sambrano, 2007).

Deficiencia	Discapacidad	Minusvalía
Del órgano de la audición	para escuchar	de orientación
Del órgano de la visión	para ver	de orientación
músculo-esquelética	para arreglarse	de independencia física
músculo-esquelética	para alimentarse	de independencia física
músculo-esquelética	de ambulación	de movilidad
psicológica de	la conducta	de integración social

Figura.1.2. Diagrama explicativo de algunos ejemplos correlacionados con su categoría.
Fuente: http://www.elsevier.es/imatges/146/146v27n05/146v27n05-13080110tab04.gif

Se debe ser cauteloso al tratar a una persona como deficiente, discapacitada o minusválida, puesto que se podrían provocar malos entendidos. Para erradicar la confusión en el uso de estos términos y para evitar el mal sentir de las personas a las que se los aplicaban se creó el término "capacidades diferentes", puesto que no es lo mismo decirle a una persona que es minusválida, a decirle que tiene capacidades diferentes.

1.2. Aspectos legales de la discapacidad

"La discapacidad ha sido tomada desde siempre como un problema para el desarrollo de los pueblos. Más de mil millones de personas viven en todo el mundo con alguna forma de discapacidad; de ellas, casi 200 millones experimentan dificultades considerables en su funcionamiento. En los años futuros, la discapacidad será un motivo de preocupación aún mayor, pues su prevalencia está aumentando. Ello se debe a que la población está envejeciendo y el riesgo de discapacidad es superior entre los adultos mayores, y también al aumento mundial de enfermedades crónicas tales como la diabetes, las enfermedades cardiovasculares, el cáncer y los trastornos de la salud mental" (Mundial, 2001).

Con estas cifras alarmantes era necesario realizar programas a nivel mundial para poder incorporar a las personas con algún tipo de discapacidad a la sociedad, programas tales como el Programa de Acción Mundial para las Personas con Discapacidad. Esta resolución figura en el documento A/37/51, Documentos Oficiales de la Asamblea General de las Naciones Unidas, trigésimo séptimo período

de sesiones, Suplemento No. 51 aprobado el 2 de diciembre de 1982 en su resolución 37/52.

"El Programa de Acción Mundial es una estrategia global para mejorar la prevención de la discapacidad, la rehabilitación y la igualdad de oportunidades, que busca la plena participación de las personas con discapacidad en la vida social y el desarrollo nacional. En el Programa también se subraya la necesidad de abordar la discapacidad desde una perspectiva de derechos humanos. En sus tres capítulos se analizan las definiciones, los conceptos y los principios relativos a la discapacidad; se examina la situación mundial de las personas con discapacidad; y se formulan recomendaciones para la adopción de medidas a nivel nacional, regional e internacional" (s.a., 2012).

1.2.1. ¿Qué dice la constitución política del Ecuador sobre la discapacidad?

Ecuador no podía ser la excepción en buscar la forma de insertar al mundo socio-económico a las personas con algún tipo de discapacidad, por ello implantó políticas de apoyo para las personas con capacidades diferentes en la Constitución de 2008, las cuales entre otras cosas, garantizan la toma de medidas para prevenir las discapacidades y cierto tipo de beneficios económicos debido a su condición. Algunos de las políticas más sobresalientes son las siguientes:

- ✓ Toda persona con discapacidad tendrá atención especializada en toda institución pública o privada que brinden a la sociedad servicios de salud.
- ✓ Descuento para movilización y espectáculos.
- ✓ Exoneraciones dentro de lo que es régimen tributario.
- ✓ Contarán con el apoyo de políticas que les permita a las personas con discapacidad insertarse en el mundo laboral dentro de empresas públicas o privadas.
- ✓ Viviendas, calles, vías de paso, entre otras, adecuadas para su cómodo traslado. Un ejemplo de esto es las rampas que existen para las personas que usan sillas de ruedas.
- ✓ Trato diferenciado dentro del sistema educativo fiscal o particular.
- ✓ Centros educativos y programas de enseñanza específicos sobre todo para personas con discapacidad mental.

El estado, como ente regulador que establece políticas de desarrollo y mejoramiento del buen vivir, creó algunas instituciones para apoyar a los discapacitados y asegurarse aún más de que se les ayude a formar parte de una sociedad como lo hace cualquier ciudadano sin limitación alguna. Entre las más destacadas se tienen:

- ✓ CONADIS.
- ✓ Misión Solidaria Manuela Espejo

1.2.2. CONADIS

"Consejo Nacional de Discapacidades del Ecuador, organismo autónomo de carácter público, que trabaja a nivel nacional, dicta políticas, coordina acciones y ejecuta e impulsa investigaciones sobre el área de discapacidades" (Márquez & Miranda, 2012).

1.2.3. Misión Solidaria Manuela Espejo

"La Misión Solidaria Manuela Espejo es una cruzada sin precedentes en la historia del Ecuador; que en un primer momento fue un estudio científico – médico para determinar las causas de las discapacidades y conocer la realidad bio-psico-social de esta población desde los puntos de vista biológico, psicológico, social, clínico y genético, con el fin de delinear políticas de Estado reales, que abarquen múltiples áreas como salud, educación y bienestar social. Fue en Noviembre del 2009 que 14 ministerios e instituciones firmaron un acuerdo con la vicepresidencia de la República del Ecuador con el fin de mejor el modus vivendi de las personas con discapacidad que se encuentran dentro del territorio nacional" (s.a., 2012).

1.2.4. Tabla estadística de personas con ceguera en la provincia del Azuay

La siguiente Tabla muestra en números la problemática que enfrenta la provincia del Azuay en cuanto a lo que tiene que ver con las personas que tienen un alto grado de ceguera. Los datos fueron obtenidos a través de un software cuyos datos son exclusivos del Censo de población y vivienda 2010.

Tabla 1.1: Datos de personas con ceguera en la provincia del Azuay en el 2010

AREA # 01 **AZUAY**

Discapacidad Visual (Ceguera)	**Sexo**		
	Hombre	Mujer	**Total**
Si	4.240	4.601	8.841
Se ignora	2.793	2.444	5.237
Total	7.033	7.045	14.078

NSA : 698.049

Fuente: INEC, ECUADOR

1.3. La ceguera como discapacidad

Al considerarse un fenómeno aleatorio, la ceguera frecuentemente no es sólo un trastorno médico, sino que produce en las personas que la sufren un descontento emocional consigo mismos y por ende, su sicología y su relación con el mundo que lo rodea también se ven afectados. Como toda discapacidad, existen niveles de afectación, los cuales generan un cierto tipo de variación en sus acepciones, tales como ceguera total, deficiencia visual o baja visión.

1.3.1. Terminología de la ceguera

Dos de los términos que se deben tener en cuenta para entender para cuantificar o dar un grado de ceguera en los pacientes que la poseen son los siguientes:

1.3.1.1. Agudeza visual

Es la capacidad que poseen las personas para definir el mundo a través de sus ojos, dándoles forma a cada una de las cosas que perciben a través de ellos. Esta capacidad también resalta la posibilidad de detectar detalles y distinguirlos entre otros que sean similares. Los optotipos son los instrumentos más idóneos para la medición de esta característica visual (ver figura 1.3).

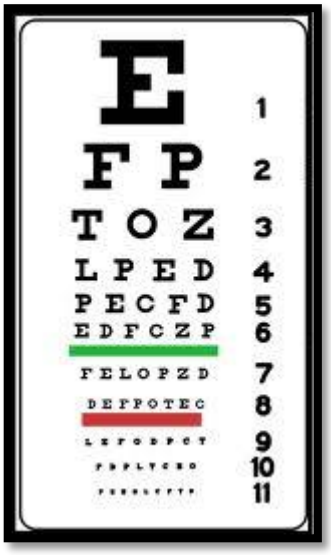


Figura.1.3. Optotipo

Fuente:http://www.qvision.es/blogs/javier-sebastian/files/2011/05/SNELLEN.jpg

Como se puede apreciar en la figura, un optotipo no es más que un conjunto de letras predispuestas en filas. Éstas van decreciendo en tamaño conforme su ubicación baja. El paciente debe ir leyendo desde las letras más grandes hasta las más pequeñas, para que el optómetra pueda hacer el diagnóstico del nivel de agudeza visual que posee.

1.3.1.2. Campo visual

Es la capacidad visual de percibir los objetos que fuera del foco central de visión, teniendo en cuenta que lo que se ve de frente está dentro del rango de nitidez más alto. Para medir esta capacidad es necesario utilizar campímetros como el que se muestra en la figura 1.4.

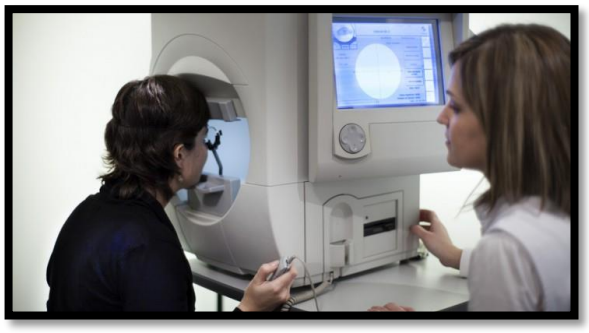


Figura.1.4. Campímetro

Fuente: http://www.institutmacularetina.com/wp-content/uploads/Campimetria_visual4-718x384.jpg

1.3.2. Tipos de ceguera

Existen algunos términos que ayudan a clasificar mejor a las personas según su grado de déficit de visión.

- ✓ **Ceguera total:** Es la ausencia total de la percepción del entorno a través de los ojos, lo que conlleva a no poder realizar tareas habituales como normalmente lo haría una persona que no padezca de esta enfermedad.
- ✓ **Discapacidad visual profunda:** Este tipo de ceguera impide que las personas que la poseen realicen tareas que impliquen ser detallistas en su realización, tales como trabajo en fotografía, confección, diseño, etc.
- ✓ **Discapacidad visual severa:** Con esta anomalía visual se pueden realizar tareas que impliquen ser minuciosos en los detalles pero con la ayuda de una persona que no padezca de esta discapacidad.
- ✓ **Discapacidad visual moderada:** Para combatirla, simplemente se debe tener una muy buena iluminación. Con esto se pueden realizar tareas con total normalidad.

1.4. Sistema braille

Una vez definidos algunos términos como deficiencia, discapacidad y minusvalía, así como también términos relacionados con la ceguera, es necesario definir la forma en la que las personas con algún problema visual absorben el mundo a través de la lectura.

1.4.1. Historia y definición

"El sistema Braille fue una invención de Louis Braille, quien a los 15 años creó un sistema de lectura para personas con limitaciones visuales, aprovechando la capacidad de captar por medio de las terminales nerviosas de las yemas de los dedos, el símbolo generador o como se le conoce más comúnmente celda Braille. Este símbolo está formado por seis puntos distribuidos en dos columnas de tres puntos cada una, dependiendo de la presencia o ausencia de esos puntos, se obtienen 64 combinaciones suficientes para representar varias letras del alfabeto" (Visuales, Salas & Rivera, 2005).

El sistema braille es uno de los recursos más utilizados para que los estudiantes puedan acceder a la información en caso de no disponer de un interlocutor para ello.

Su simpleza hace que sea un sistema práctico y fácil de aprender mediante la repetición y memorización.

1.4.2. Representación de las letras del alfabeto según el código braille

Para representar el alfabeto a través del sistema braille, se utilizan las celdas del mismo nombre, teniendo 64 posibles combinaciones, de las cuales se pueden utilizar todas excepto una. El alfabeto en el sistema braille se representa como se muestra en la figura 1.5.

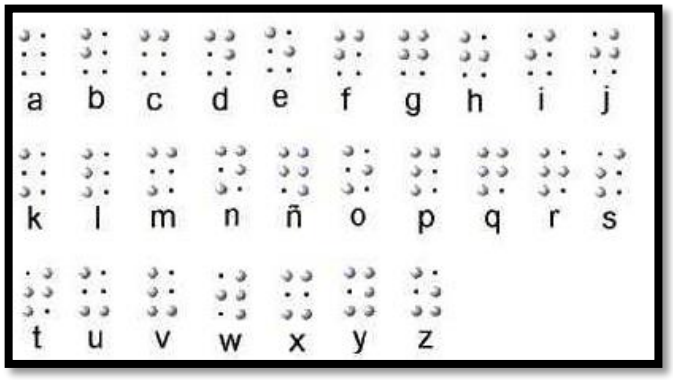


Figura1.5. Alfabeto representado en braille

Fuente: usuarios.discapnet.es/ojo_oido/esquemas_cuerpo_humano/sistema_braille_clip_image001_0002.jpg

Como se puede apreciar, se tienen 27 letras que conforman nuestro alfabeto (no se toman en cuenta la “ch”, “ll” y “rr” puesto que son caracteres que conformados por dos letras) representados a través del código braille. Algunos caracteres especiales ocupan las sobrantes posibles combinaciones del sistema.

1.4.3. Caracteres especiales

Dentro de los caracteres especiales se pueden encontrar los signos de puntuación, letras mayúsculas y números.

1.4.3.1. Signos de puntuación

Dentro de ellos se pueden encontrar la coma, el punto, punto y coma, dos puntos, signos de admiración, signos de interrogación, comillas y paréntesis. La figura 1.6 muestra cómo se representan este tipo de caracteres en el sistema braille.

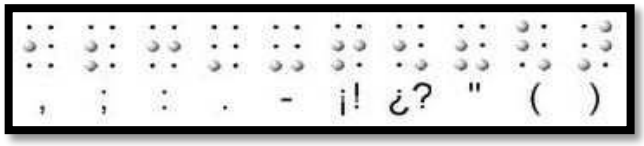


Figura1.6. Caracteres especiales en el sistema braille

Fuente: http://usuarios.discapnet.es/ojo_oido/esquemas_cuerpo_humano/sistema_braille_clip_image003.jpg

1.4.3.2. Letras mayúsculas

Para representar los caracteres de la figura 1.5 en mayúscula, simplemente se debe anteponer el caracter que se muestra en la figura 1.7. Con esto queda sobre entendido que por cada letra mayúscula que se quiera plasmar en braille, se deben utilizar dos celdas braille.

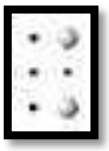


Figura 1.7. Celda braille que ayuda a distinguir las mayúsculas

Fuente: http://usuarios.discapnet.es/ojo_oido/esquemas_cuerpo_humano/sistema_braille_clip_image004.jpg

1.4.3.3. Números

Al igual que las letras mayúsculas, los números son caracteres de doble celda o símbolos dobles, pues necesitan de ello para poder ser representados (ver figura 1.7). Los números están formados por las letras a, b, c, d, e, f, g, h, i y j precedidos por el signo de la figura 1.8 para formar los números 1, 2, 3, 4, 5, 6, 7, 8, 9 y 0 respectivamente.

Figura 1.8. Celda braille que ayuda a distinguir los números

Fuente: http://usuarios.discapnet.es/ojo_oido/esquemas_cuerpo_humano/sistema_braille_clip_image007.jpg

1.5. Uso de las TIC's para el alivio de las personas con un alto grado de ceguera

"El acceso universal a la educación básica y las condiciones para su calidad son imprescindibles, pero también lo es lograr que todos los alumnos y alumnas alcancen las competencias básicas para proseguir estudios posteriores, para incorporarse a la sociedad de forma activa y para ejercer sus derechos y deberes como ciudadanos

libres y responsables […]. Universalizar la alfabetización, la educación básica y otras oportunidades de capacitación para jóvenes y adultos a lo largo de toda la vida, con diferentes metodologías y, en especial, con las nuevas tecnologías, es una estrategia fundamental en la lucha contra la pobreza, a favor de la inclusión..." (Zappalá, Köppel & Suchodolski, 2011).

El uso de las TIC's en el proceso educativo son de vital importancia, por cuanto ayudan a llegar a estudiantes que no asimilan igual que el estudiante promedio por tener algún tipo de discapacidad, en este caso un alto grado de ceguera. Las *tablets*, computadoras y celulares han ayudado mucho en este campo a través de programas que potencian el proceso de enseñanza-aprendizaje. La selección de las ayudas tecnológicas en cuanto a *software* se refiere debe hacerse de acuerdo a las necesidades de cada estudiante.

CAPÍTULO II

HARDWARE PARA LA ELABORACIÓN DEL SISTEMA DE IMPRESIÓN BRAILLE

2.1 Introducción

En este capítulo se hace referencia a los elementos físicos (materiales de construcción y elementos eléctricos-electrónicos) que la constituyen para poner de manifiesto su importancia y comportamiento para tener una previa de lo que se va utilizar y se pueda entender de mejor manera al momento de recibir la explicación del funcionamiento de todo el prototipo de impresión.

2.2 Elementos físicos para la construcción de la impresora braille

Los principales elementos físicos que utilizarán para el desarrollo del sistema se describen a continuación:

2.2.1 Madera MDF

Es el material del que se elaborará la maqueta sobre la que se dispondrán todos los elementos físicos del sistema de impresión. Este tipo de material posee mayor densidad en las capas exteriores y mayor homogeneidad en las interiores, lo que permite tener mayor fijación al momento de realizar contactos a presión para que sean más firmes y seguros.

“Maloney (1996), define al MDF como aquellos tableros fabricados en seco, con fibras lignocelulósicas combinadas con una resina sintética u otro agente de aleación, compactados a una densidad entre 0,50 y 0,80 g/cm3 por prensado en caliente, en un proceso en que la totalidad de la adhesión entre las fibras depende del adhesivo adicionado. Según Youngquist (1998), el MDF es un producto homogéneo, uniforme, estable, de superficie plana y lisa, que ofrece buena trabajabilidad y maquinado para encajar, tallar, cortar, atornillar, perforar y moldurar. Incluso, produce economía en cuanto a la reducción del uso de tintas, pinturas y lacas, economía en el consumo de adhesivo por metro cuadrado, además de presentar óptima aceptación para recibir revestimientos con diversos acabados. El Brasil

presenta condiciones favorables para convertirse en un importante productor mundial de tableros de madera. Sin embargo, para que haya desarrollo es necesario invertir en tecnologías destinadas a mejorar la producción de tableros derivados de madera" (Scielo, 2014).

Figura 2.1: Madera MDF

Fuente: http://images.fordaq.com/p-17840000-17839938-D0/Fibra-de-madera-de-densidad-media---(MDF).jpg

2.2.2 Arduino MEGA 2560

Con un microcontrolador Atmega 2560 y 54 pines que pueden configurarse para que trabajen como entrada o salida (14 de ellos pueden generar PWM), 16 entradas analógicas, 4 puertos de hardware en serie, un oscilador de cristal de 16 MHz, un puerto USB de conexión, entre otros atributos físicos, esta placa se ha convertido en la mejor alternativa para los programadores de microcontroladores a nivel mundial.

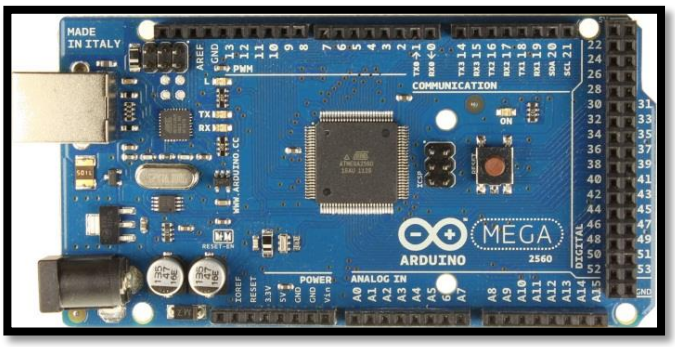


Figura 2.2: ArduinoMega 2560

Fuente: https://paruro.pe/sites/default/files/ArduinoMega2560_R3_Front.jpg

2.2.3 DVR 8825

Es un controlador par motores paso a paso (ver figura 2.3) que soporta una corriente de 1,5 amperios sin que exista la necesidad de refrigerarlo y soporta una de 2,2 con enfriamiento. Su función en el sistema de impresión braille será el de controlar a los

motores paso a paso. Por su continuo uso en el prototipo, se le dotará de un ventilador de computador para que pueda mantener baja su temperatura.

Figura 2.3: DRV8825

Fuente: Autor

2.2.3 Motor paso a paso

Es un motor bipolar cuyo uso, generalmente, se da en la construcción de impresoras (ver figura 2.4). Por su principio de funcionamiento se vuelve un sistema muy preciso y confiable. Las especificaciones eléctricas se muestran en la Tabla 2.1.

Figura 2.4: Motor paso a paso

Fuente: http://ultra-lab.net/sites/default/files/imagecache/Portada-derecha/motor1.png

Las especificaciones eléctricas se muestran en la Tabla 2.1.

Tabla 2.1: Especificaciones eléctricas del motor paso a paso

ESPECIFICACIONES ELÉCTRICAS										
Modelo N°	Ángulo de paso	Longitud del motor	Voltaje	Corriente	Resistencia de fase	Inductancia de fase	Torque	# de cables	Inercia del rotor	Peso
	°	mm	V	A	Ω	mH	g-cm		g-cm²	kg
42BYGHW811	1,8	48	3,1	2,5	1,25	1,8	4800	4	68	0,34

Fuente: Autor

2.2.4 Solenoide

Este dispositivo electrónico sirve para dar movimiento lineal activado por un pulso que dependiendo de su magnitud lo hará a mayor o menor velocidad. Al momento de recibir un pulso de voltaje, el solenoide se activa y desplaza su cabeza hacia abajo seguido por un efecto rebote invocado por el resorte que hace presión desde la base hacia arriba.

Figura 2.5: Solenoide
Fuente: http://www.olimex.cl/product_info.php?products_id=1186

El propósito de este artefacto es el de puntear, pero, al no tener el forma de hacerlo por sí solo, se le adaptará una aguja, la cual está ubicada al otro lado de la cabeza del solenoide, de tal forma que al momento de recibir el pulso de accionamiento, ésta se desplace de forma vertical para golpear el papel lo suficientemente fuerte como para dejar la marca requerida (ver figura 2.6). Se debe tener cuidado de que al momento de la calibración del pulso no se exceda, puesto que el papel podría ser perforado.

Figura 2.6: Solenoide con la punta adaptada
Fuente: http://www.olimex.cl/product_info.php?products_id=1186

2.2.5 Tornillo sinfín

Es un rueda dentada que sirve para transmitir movimiento giratorio a o desde otro engrane que se encuentran en dispuestos en forma perpendicular (ver figura 2.7).

Figura 2.7: Tornillo sinfín
Fuente: autor

2.2.6 TCRT5000

Es un sensor de luz de alta frecuencia (no es perceptible por el ojo humano) que posee un emisor infrarrojo y un foto transistor que cumple la función de receptor reflexivo con un separador entre ambos, lo cual es de mucha utilidad para evitar que el receptor se active por una luz lateral. En la figura 2.8 se observa la forma física de un TRCT5000 y en la figura 2.8 se puede observar el esquema electrónico que lleva en su interior.

Figura 2.8: Sensor TCRT5000
Fuente: http://media.digikey.com/photos/Vishay%20Photos/TCRT5000.JPG

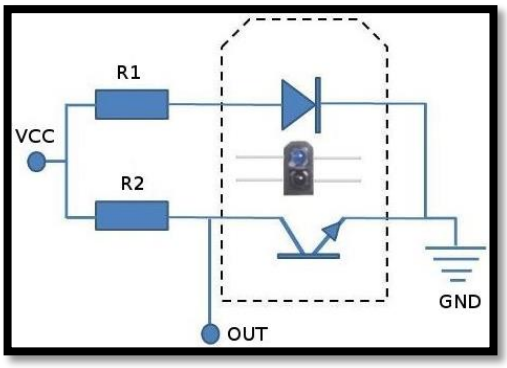


Figura 2.9: Circuito eléctrico del TRCT5000
Fuente: http://zerokol.com/post/5216613ce84c556437000016/1/es

Las características técnicas se muestras a continuación en la Tabla 2.2.

Tabla 2.2: Características técnicas de un TCRT5000

Hojas De Datos	TCRT5000 (L)
Fotos del producto	TCRT5000
Dibujos Catálogo	TCRT5000 (L)
Paquete estándar	4500
Categoría	Sensores y transductores
Familia	Sensores ópticos - Reflexivo - Salida analógica
Serie	-
Embalaje	Tubo
Distancia de detección	0.591 "(15 mm)
Método de detección	Reflexivo
Voltaje - Desglose de colector emisor (Max)	70V
Corriente - Collector (Ic) (Max)	100mA
Corriente continua - Directa (If) (Max)	60mA
Tipo de salida	Fototransistor
Tiempo De Respuesta	-
Tipo de montaje	A Través Del Agujero
Paquete / Cubierta	Para PCB
Temperatura de funcionamiento	-25 ° C ~ 85 ° C
Página del Catálogo	2733 (CA2011 PDF)
Otros Nombres	751-1033-5

Fuente: http://www.digikey.ca/product-detail/en/TCRT5000/751-1033-5-ND/1681167

CAPÍTULO III

ANÁLISIS DEL SOFTWARE APLICADO

3.1 Introducción

Como ya es de conocimiento público, una buena educación no puede sobrevivir sin que ella vaya de la mano con la aplicación de las tecnologías de la información y de la comunicación. Por ello, en este capítulo, se dará una breve introducción a lo que es el software que se ha elegido para la elaboración de la interface y adquisición de datos del prototipo de impresión braille.

3.2 LabVIEW

"LabVIEW es una herramienta de programación gráfica. Originalmente este programa estaba orientado para aplicaciones de control de equipos electrónicos usados en el desarrollo de sistemas de instrumentación, lo que se conoce como instrumentación virtual. Por este motivo los programas creados en LabVIEW se guardarán en ficheros llamados VI (*Virtual Instrument*), y con la misma extensión" (Viscaíno & Sebastiá, 2011).

Dada la previa definición lo que se puede deducir es que el entorno de aplicación natural de LabVIEW no es otro que el de adquisición y tratamiento de datos. Hay que recalcar que la versión de LabVIEW que se utilizará para la realización del proyecto será la 2012, la cual se muestra en la figura 3.1. Las dos caras de la moneda, por así decirlo, de LabVIEW son las mostradas en las figuras 3.2 y 3.3.

Figura3.1. LabVIEW 2012

Fuente: Autor

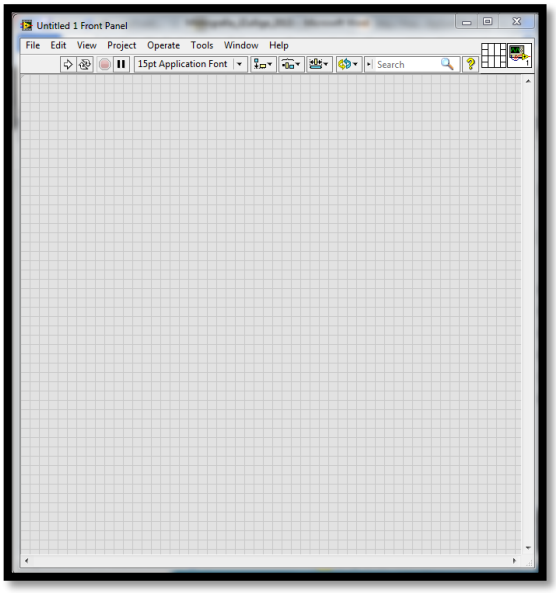


Figura3.2. Panel frontal

Fuente: Autor

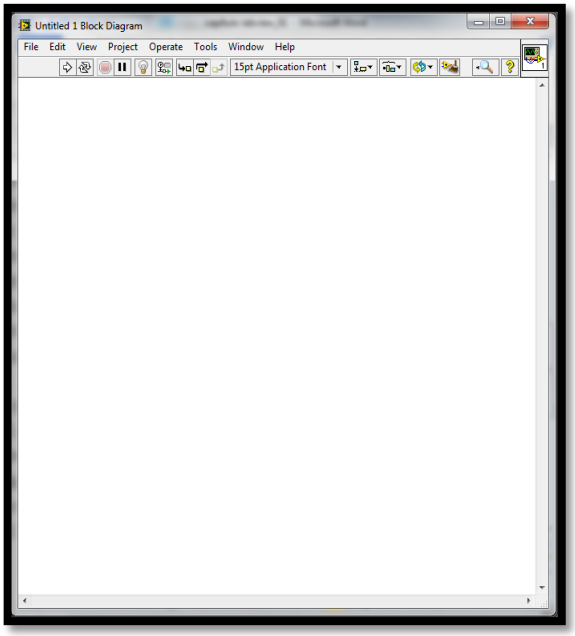


Figura3.3. Diagrama de bloques

Fuente: Autor

3.2.1 Panel frontal

Es de color plomo y cuadriculado. En él se encontrarán todos los controladores e indicadores de un VI. Se podría decir sin miedo a equivocarse de que esta es la pantalla de interacción entre el usuario y el sistema de control. Se puede observar al dar clic derecho dentro del panel de control una paleta flotante llamada *Controls*

como la que se muestra en la figura 3.4, en ella se encuentran todas las herramientas de interface de las que se disponen en LabVIEW.

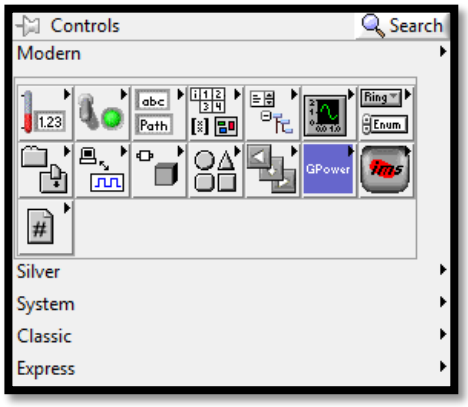


Figura3.4. Paleta flotante del panel de control

Fuente: Autor

3.2.1.1 Controles

Son elementos configurables tanto en tamaño, forma y color a disposición del usuario que se encuentran en la paleta flotante del panel frontal, los cuales permiten manejar al antojo la interface dependiendo de lo que se quiera controlar o ver. La figura 3.5 muestra algunos de los controladores más básicos pero a la vez utilizados que se encuentran en LabVIEW.

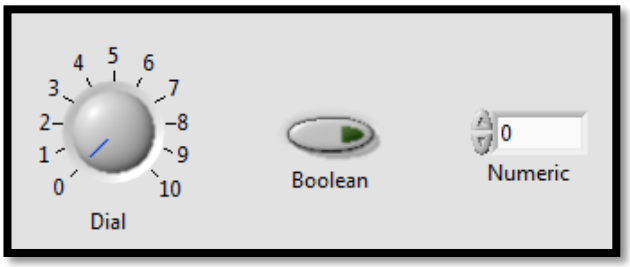


Figura3.5. Controladores en el panel frontal

Fuente: Autor

En LabVIEW los controles tienen una característica especial en el diagrama de bloques, pues poseen unas flechas al costado derecho en dirección a la izquierda, es decir que entregan datos (ver figura 3.6).

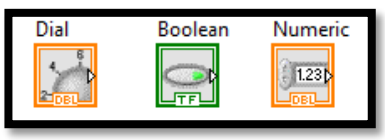


Figura3.6. Controladores en el diagrama de bloques

Fuente: Autor

3.2.1.2 Indicadores

Como su nombre lo indican son elementos que permiten la visualización al usuario de los datos y la forma en cómo se están procesando al momento de la ejecución. Algunos de los indicadores más usados son los tanques, booleanos y numéricos. Estos ejemplos se muestran en la figura 3.7.

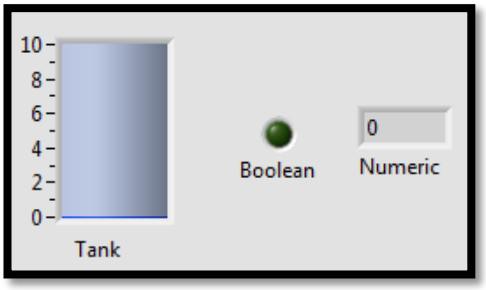


Figura3.7. Indicadores del panel frontal
Fuente: Autor

Los indicadores en el diagrama de bloques poseen la característica de tener una flecha en la izquierda con dirección al mismo lado en su ícono representativo tal y como se muestra en la figura 3.8.

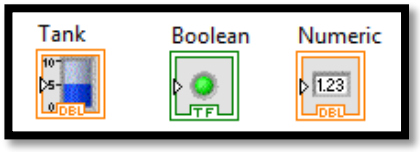


Figura3.8. Indicadores en el diagrama de bloques
Fuente: Autor

La figura numérica (*numeric* por su traducción en inglés), puede crear confusión por el motivo de que al parecer se encuentra tanto dentro de los controles como de los indicadores, pero si se pone atención, el controlador numérico difiere del indicador numérico por la barra lateral izquierda que sirve para aumentar o disminuir de valor, mientras que el otro no la posee, puesto que ya es conocido que su función únicamente es la de mostrar.

3.2.1.3 Elementos de ejecución de un VI

Se encuentran tanto dentro del panel de control como del diagrama de bloques. Presentan funciones bien definidas y se distribuyen en forma horizontal en la parte alta e izquierda de las pantallas de LabVIEW. Existen muchas, pero se detallarán las

4 principales, puesto que éstas se encuentran en el panel frontal y son las que manipulará el usuario final.

3.2.1.3.1 *Run*

Ejecutar por su traducción al español, permite correr el programa para que se ejecute su diagramación gráfica. De no existir estructuras de repetición dentro del código gráfico el VI se ejecutará una sola vez (ver figura 3.9).

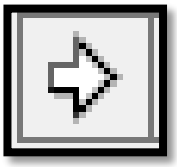


Figura3.9. *Run*
Fuente: Autor

3.2.1.3.2 *Run continuously*

Ejecución continua por su traducción al español, permite ejecutar continuamente y sin parar al VI, sin importar que existan o no estructuras de repetición (ver figura 3.10).

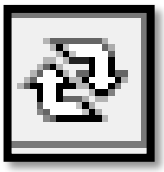


Figura3.10. *Run continuously*
Fuente: Autor

3.2.1.3.3 *Abort execution*

Abortar ejecución por su traducción al español, es la botón que permite detener totalmente la ejecución (ver figura 3.11).

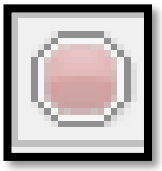


Figura3.11. *Abort execution*
Fuente: Autor

3.2.1.3.4 *Pause*

Pausa por su traducción al español, permite detener momentáneamente la ejecución del código gráfico (ver figura 3.12).

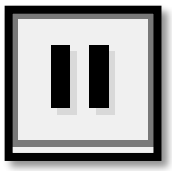


Figura3.12. *Pause*
Fuente: Autor

3.2.2 Diagrama de bloques

El diagrama de bloque (o *block diagram* por su traducción en inglés*)* es la parte de LabVIEW en la que se programa de forma gráfica. En ella se desarrolla y aplica toda la lógica del sistema y al momento de realizar un ejecutable, el usuario no tendrá acceso a este diagrama. Al igual que en el panel de control, al momento de dar clic derecho sobre la parte vacía de la pantalla aparecerá una paleta flotante llamada *Functions*, la cual brindará al programador las herramientas necesarias para desarrollar el programa de forma gráfica, así como se muestra en la figura 3.13.

Figura3.13. Paleta flotante del diagrama de bloques
Fuente: Autor

3.2.3 Tipos de terminales de datos

Al igual que en todos los lenguajes de programación, LabVIEW maneja varios tipos de datos para la estructuración de códigos gráficos, lo cual brinda gran flexibilidad y versatilidad al momento de diseñar un VI. Dependiendo de las necesidades del programador estos terminales pueden ser configurados para mejorar su precisión y

forma de utilización. A continuación se detallarán algunos de estos tipos de terminales para tener una idea de su funcionamiento específico dentro de LabVIEW.

3.2.3.1 Numéricos

Existen de varios tipos dentro de LabVIEW, los cuales se diferencian por su extensión en *bits*, con o sin signo, con decimales y en de tipo complejo simple. A continuación se detallan algunos de los tipos de terminales de datos numéricos que se tienen a disposición en LabVIEW.

- ✓ Los datos numéricos enteros son terminales de color azul que tienen en su parte baja una "I" que representa su característica de entero puesto que esta letra representa la inicial de *Integer* que en español significa entero en español y pueden ser de 8, 16, 32 o 64 *bits* dependiendo de la precisión con la que se quiera trabajar al momento de manipular los datos. Dicha extensión viene marcada a través de un número al lado de la "I" antes mencionada, tal y como se muestra en la figura 3.14.

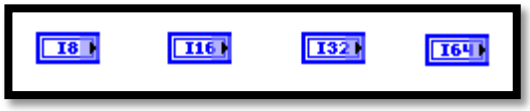


Figura3.14. Tipos de terminales de datos numéricos enteros
Fuente: Autor

- ✓ Los datos numéricos sin signo (*Unsigned*) son terminales casi idénticas a los numéricos enteros excepto que en llega su característica en la letra "U" (ver figura 3.15).

Figura3.15. Tipos de terminales de datos numéricos sin signo
Fuente: Autor

- ✓ Los datos numéricos de punto flotante son de color naranja y tienen 3 caracteres que los diferencian como se muestra en la figura 3.16.
  - *Single* (SGL) de 32 *bits*
  - *Double*(DBL) de 64 *bits*
  - *Extended* (EXT) de 80 *bits*

Figura3.16. Tipos de terminales de datos numéricos de punto flotante

Fuente: Autor

3.2.3.2 Booleanos

Son terminales de color verde de 16 bits (desde el bit 0 hasta el bit 15), poseen únicamente dos estados: verdadero (*true*) o falso (*false*). La determinación de este estado se da simplemente cuando el programa verifica el *bit* más significativo del terminal booleano, si éste tiene un valor de 1, el terminal estará en *true* de lo contrario está en 0 representado un valor *false* (ver figura 3.17).

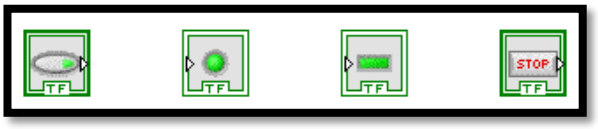


Figura3.17. Tipos de terminales booleanos

Fuente: Autor

3.2.3.3 Cadena

Más conocido como *String* son de color fucsia, representan una serie de caracteres concatenados en forma serial. Pueden ser tanto letras como numéricos, es decir, son alfanuméricos, pero hay que tener en cuenta de que al momento de ingresar un número, esta terminal no lo tomará como un valor numérico sino como una figura. En la figura 3.18 se muestran algunos de los terminales *String* y algunas de las herramientas que ayudan a manipular este tipo de datos.

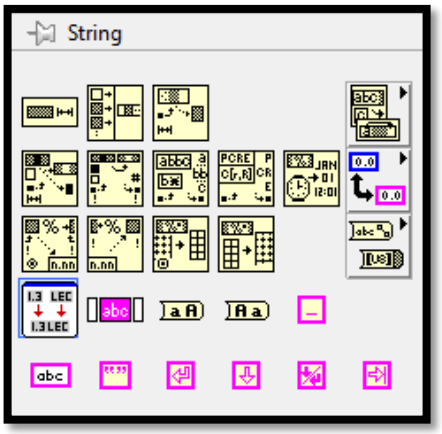


Figura3.18. Terminales y herramientas *String*

Fuente: Autor

3.2.4 Estructuras

También llamadas *Structures* por su traducción al inglés y se las pueden encontrar dando clic derecho dentro del diagrama de bloques, permiten al programador generar ciclos o bucles de repetición de la ejecución finitos o infinitos dentro del código para el tratamiento de datos. A continuación se detallarán algunas de las estructuras que posee LabVIEW para generar bucles (ver figura 3.19).

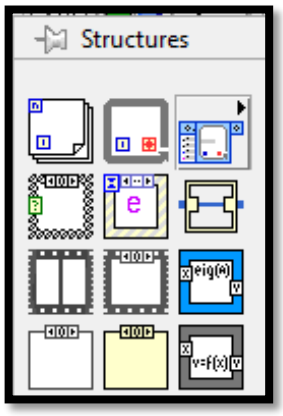


Figura 3.19. Tipos de estructuras en Labview

Fuente: Autor

3.2.4.1 For loop

Es de forma cuadrada y posee dos terminales de color azul, lo cual indica que recibirán o arrojarán datos numéricos enteros (ver figura 3.20). La terminal "N" representa el número de veces que se ejecutará el código que esté dentro de esta estructura. Se puede colocar cualquier número entero, pero si se le coloca un "0", lo que está dentro de la estructura jamás se ejecutará. La letra "i" representa el número de internaciones completadas. Estas dos terminales vienen por de defecto al graficar esta estructura dentro del diagrama de bloques.

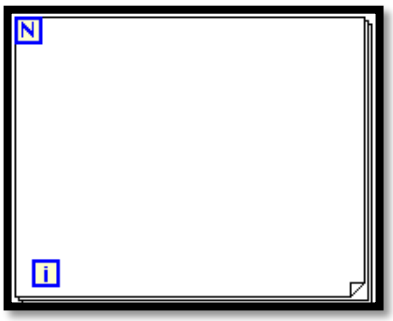


Figura 3.20. For loop

Fuente: Autor

3.2.4.2 While loop

Al igual que toda estructura de repetición, ejecuta constantemente el código que contine adentro. Posee un terminal de iteraciones "i" y un terminal de *Stop* rojo de borde verde que cambia de estado cuando se le coloca un control booleano que al momento de colocarse está en *true* (ver figura 3.21). En esta estructura el código encerrado se ejecutará de forma infinita hasta el terminal booleano cambie su estado a *false*.

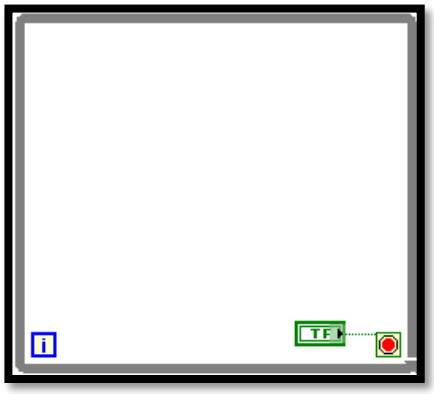


Figura 3.21. While loop

Fuente: Autor

3.2.4.3 Case structure

Posee uno o más subdiagramas dentro de sí, los cuales se ejecutan dependiendo del valor booleano que posea el terminal que se le coloque a la izquierda. Por ejemplo si el valor del terminal booleano es verdadero, se ejecutará el código que se encuentra dentro de la estructura cuando en la cabecera esté la palabra *true*. Caso contrario, si el terminal tiene un valor falso, se ejecutará el código que se encuentra cuando en la cabecera se encuentre la palabra *false* (ver figura 3.22). Para poder ver lo que se encuentra en cada uno de los casos, simplemente se recurre a las flechas negras que se encuentran en la parte alta de la estructura.

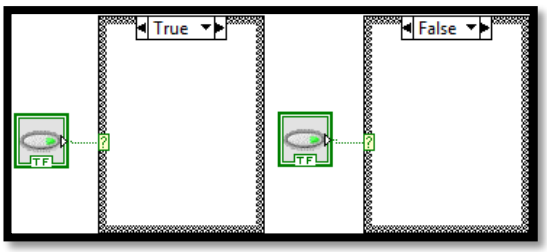


Figura 3.22. Case structure

Fuente: Autor

3.3 Arduino

La tecnología apuntaba a que todo sea controlado a través de un ordenador, así que se desarrolló la plataforma de desarrollo Arduino. De código abierto y provisto de un entorno para desarrollar programas a través de una placa y un microcontrolador, Arduino se ha convertido en uno de los lenguajes más utilizados en el mundo de la electrónica y sus derivados.

"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 Arduino *Development Environment* (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, etc.)" (Herrador, 2009).

Las ventajas de Arduino con otras plataformas son muchas, pero entre las más importantes se pueden destacar las siguientes:

- ✓ El *hardware* es barato comparado con otras familias de microcontroladores.
- ✓ Entorno de programación simple para que sea fácil de entender y aplicar.
- ✓ Brinda la facilidad de que los usuarios pueden acceder al código fuente de Arduino.
- ✓ Puede ejecutar en cualquier sistema operativo.

3.3.1 Estructura de programación en Arduino

Arduino, en su estructura básica de programación, contiene dos partes indispensables para su funcionamiento. El *void setup (),* es la primera parte del programa, se ejecuta por una sola vez cuando el programa empieza, y dentro de él se declaran las variables, así como también se configurar los puertos como entradas o salidas. El *loop setup (),* es un ciclo que se repite constantemente ejecutando el código que en él se haya escrito (ver figura 3.23).

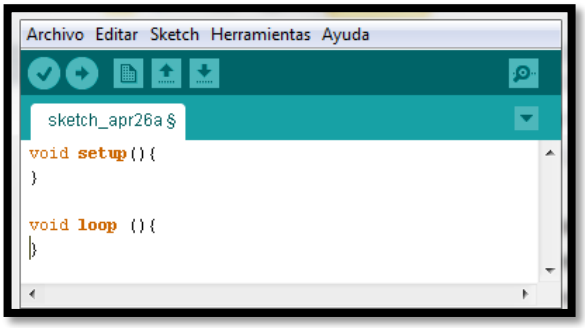


Figura 3.23. Estructura básica de la programación en Arduino

Fuente: Autor

3.3.2 Tipos de datos

Los tipos de datos de los que se dispone en Arduino se detallan en la Tabla 3.1:

Tabla3.1. Estructura básica de la programación en Arduino

Tipos de Datos	Memoria que ocupa	Rango de valores
boolean	1 byte	0 o 1 (True o False)
byte / unsigned char	1 byte	0 – 255
char	1 byte	-128 – 127
int	2 bytes	-32.768 – 32.767
word / unsigned int	2 bytes	0 – 65.535
long	2 bytes	-2.147.483.648 – 2.147.483.647
unsigned long	4 bytes	0 – 4.294.967.295
float / double	4 bytes	-3,4028235E+38 - 3,4028235E+38
string	1 byte + x	Array de caracteres
array	1 byte + x	Colección de variables

Fuente: http://rduinostar.com/wp-content/uploads/2012/10/Tipos-de-Variables-Arduino.jpg

Como se puede apreciar en la Tabla 3.1, Arduino brinda una serie de tipos de datos que sirven para desarrollar programas. Los datos que representan números sin parte decimal son: *byte, unsigned char, char, int, word, unsigned int y unsigned long*. Aquellos que tienen parte decimal son los datos tipo *float* y *double*. Por su parte, un array, es un conjunto de posiciones sucesivas en donde se almacenan datos. Por último un tipo de dato booleano sólo posee dos posibles *estados*: true o *false.*

La mayoría de datos se los define en el *void setup ()*, pero esto no es una regla, puesto que también se pueden declarar dentro del lazo de ejecución (*void loop*) del programa, dependiendo de las necesidades y comodidad del programador.

3.3.3 Operadores

Son instrumentos de programación entre datos del mismo tipo indispensables en cualquier plataforma. Se clasifican en:

- ✓ Operadores aritméticos
- ✓ Operadores de relación
- ✓ Operadores lógicos

3.3.3.1 Operadores aritméticos

Sirven para efectuar procesos matemáticos básicos, tales como: suma, resta, multiplicación, división y módulo. La Tabla 3.2 muestra todos los operadores que brinda Arduino al momento de programar.

Tabla3.2. Operadores aritméticos

Operadores Aritméticos			
Operador	Descripcion	Tipo	Ejemplo
+	signo positivo	unario	+var
-	signo negativo	unario	-var
*	multiplicación	binario	var1 * var2
/	división	binario	var1 / var2
%	módulo (resto de división entera)	binario	var1 % var2
+	suma	binario	var1 + var2
-	resta	binario	var1 - var2

Fuente: http://microembebidos.com/wp-content/gallery/tutorial-ansi-c/operaritmeticos.jpg

3.3.3.2 Operadores lógicos y de relación

Son aquellos cuya respuesta arrojan valores booleanos (verdadero o falso). Su función es establecer comparación entre dos variables o entre una variable y una constante ver Tabla 3.3).

Tabla 3.3. Operadores lógicos y de relación

Operadores de Relación			
Operador	Descripcion	Tipo	Ejemplo
<	Menor que	binario, relacional	var1 < var2, retorna 1 si se cumple, sino 0.
<=	Menor o igual que	binario, relacional	var1 <= var2, retorna 1 si se cumple, sino 0.
>	Mayor que	binario, relacional	var1 > var2, retorna 1 si se cumple, sino 0.
>=	Mayor o igual que	binario, relacional	var1 >= var2, retorna 1 si se cumple, sino 0.
==	Es igual que	binario, relacional	var1 == var2, retorna 1 si se cumple, sino 0.
!=	Es desigual que	binario, relacional	var1 != var2, retorna 1 si se cumple, sino 0.
Operadores Lógicos (para combinar el resultado de los operadores de relación)			
Operador	Descripcion	Tipo	Ejemplo
&&	AND	binario, lógico	X && Z, retorna su tabla de verdad, siendo X y Z resultados de un operador de relación.
||	OR	binario, lógico	X || Z, retorna su tabla de verdad, siendo X y Z resultados de un operador de relación.
!	NOT	unario, lógico	!X, retorna su tabla de verdad, siendo X un resultado de un operador de relación.

Fuente: http://microembebidos.com/wp-content/gallery/tutorial-ansi-c/operrelacionlogicos.jpg

3.3.4 Sentencias condicionales

Arduino le proporciona al programador disponer de sentencias condicionales como *if, if… else, for, while y do… while*. La función de estas sentencias es igual que en las demás plataformas de programación como C, java, LabVIEW, entre otras.

3.3.5 Funciones

Pueden ser matemáticas y de tiempo, tales como la función *delay()*, cuyo trabajo es generar una recarga igual al número en mili segundos que lleva dentro de su paréntesis o min (x,y), que devuelve el valor máximo y mínimo respectivamente de las variables que estén dentro de sus paréntesis. También pueden ser aleatorias como la función *random()*, que genera valores aleatorios. Esta función es de mucha utilidad para simular señales de entrada al momento de programar un control.

3.3.6 Variables

Es una letra que representa un valor que va a puede ser modificado o no dentro de la ejecución de un programa. Puede ser declarada dentro del *void setup ()* o el *void loop ()*. Pueden ser locales o globales, pudiendo las primeras ser utilizadas dentro de un bucle o pedazo de código y las segundas en cualquier parte o función del programa. Sólo estas últimas deben ser declaradas al inicio del programa antes de la función *setup ()*.

CAPÍTULO IV

DESARROLLO DEL PROYECTO

En este capítulo se describen los pasos que se realizaron para la elaboracióndel sistema de impresión braille. En el desarrollo de este proyecto se utilizaron los elementos antes mencionados en el Capítulo II, para la interface se utilizó LabVIEW y en el control de los drivers, sensores y demás implementos que forman la impresora se utilizó la plataforma Arduino.

4.1 Selección de componentes

Los componentes para la construcción de la impresora se resumen en los siguientes:

- ✓ Madera MDF para la construcción de la maqueta.
- ✓ Acrílico negro y transparente para mejorar la estética de la impresora.
- ✓ Tornillo sinfín para el desplazamiento de la base del solenoide.
- ✓ Fuente de alimentación de 12 voltios a 5 amperios.
- ✓ Motores paso a paso que realizan el arrastre de la hoja de papel a imprimirse y para el movimiento del tornillo sinfín.
- ✓ Sensor TCR T5000 para el detectar la posición del papel.

4.2 Diseño de la impresora braille

Como se dijo en el capítulo, el material del cual será fabricado la maqueta para la implementación del sistema braille, será madera MDF, la cual presenta las condiciones necesarias por su consistencia y estabilidad por peso para el fin predispuesto. Los cortes para el armado de la maqueta quedan como se muestra en la figura 4.1 y las dimensiones de la maqueta armada en el ANEXO 1.

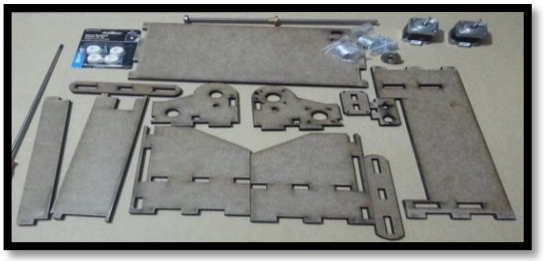


Figura 4.1: Partes en madera MDF para el armado de la maqueta

Fuente: Autor

Una vez con los cortes a disposición se procede al armado utilizando goma como se muestra en la figura 4.2.

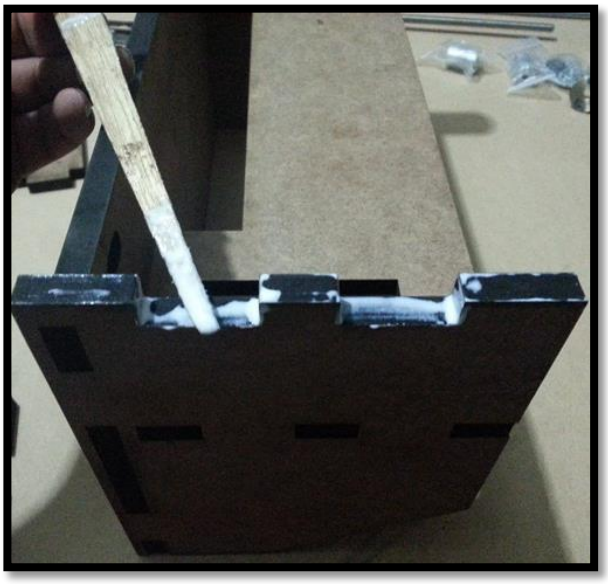


Figura 4.2: Armado de la maqueta usando goma
Fuente: Autor

En la figura 4.3 se puede apreciar la maqueta a medio armar. Dentro de esta base, irá el soporte de los motores, del tornillo sinfín, de la base del solenoide y 2 varillas más: una será la que lleve unos pequeños cauchos que servirán para arrastrar la hoja de papel y la otra irá predispuesta en forma paralela al tornillo sinfín para que sirva de apoyo y así el solenoide no se descuadre durante su acción de golpeo.

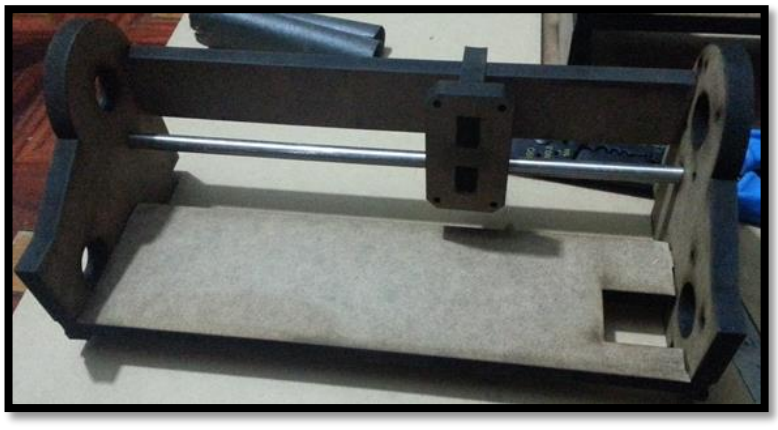


Figura 4.3: Maqueta a medio armar
Fuente: Autor

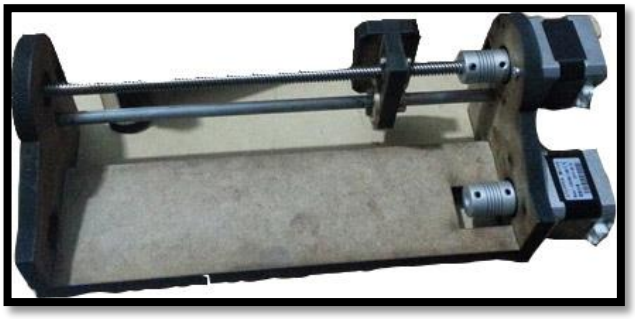


Figura 4.4: Soporte de los motores, tornillo sinfín y varillas
Fuente: Autor

Una vez armadas las partes que muestran gráficamente en las figuras 4.3 y 4.4 la maqueta queda como se muestra en figura 4.5.

Figura 4.5: Ensamblado de las figuras 4.3 y 4.4
Fuente: Autor

Una vez colocadas la base del solenoide y las partes de acrílico que ayudan a mejorar la estética del prototipo, la maqueta final quedará como se muestra en las figuras 4.6, 4.7, 4.8 y 4.9.

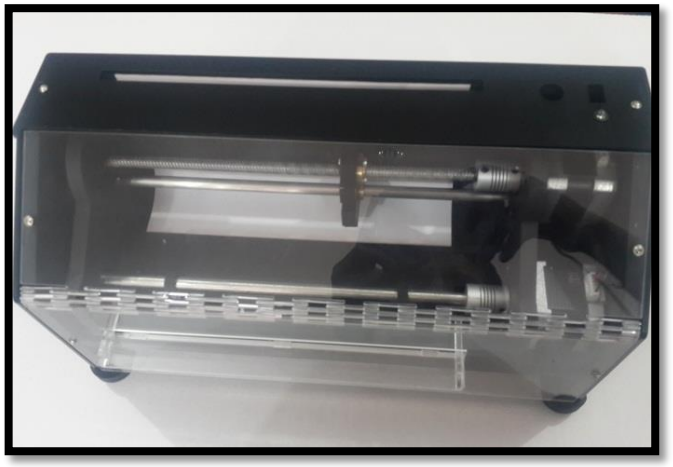


Figura 4.6: Vista superior de la maqueta

Fuente: Autor

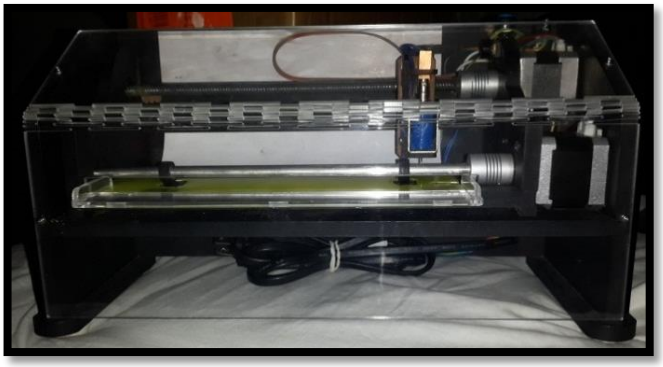


Figura 4.7: Vista frontal de la maqueta

Fuente: Autor

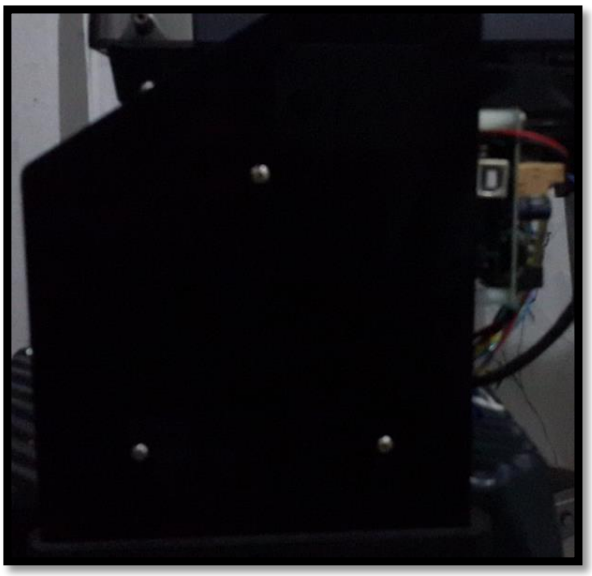


Figura 4.8: Vista lateral de la maqueta

Fuente: Autor

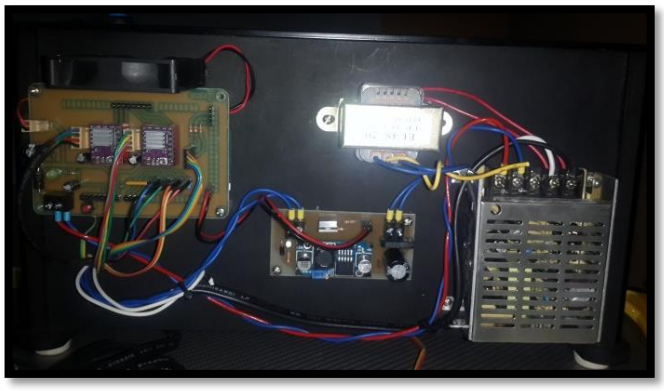


Figura 4.9: Vista posterior de la maqueta
Fuente: Autor

4.3 Interface y código de comunicación del sistema de impresión braille.

Todo lo que tiene que ver en cuanto a la programación, tanto interface, como ingreso de datos y comunicación con el sistema de impresión braille se lo resume en el diagrama de flujo de la figura 4.10.

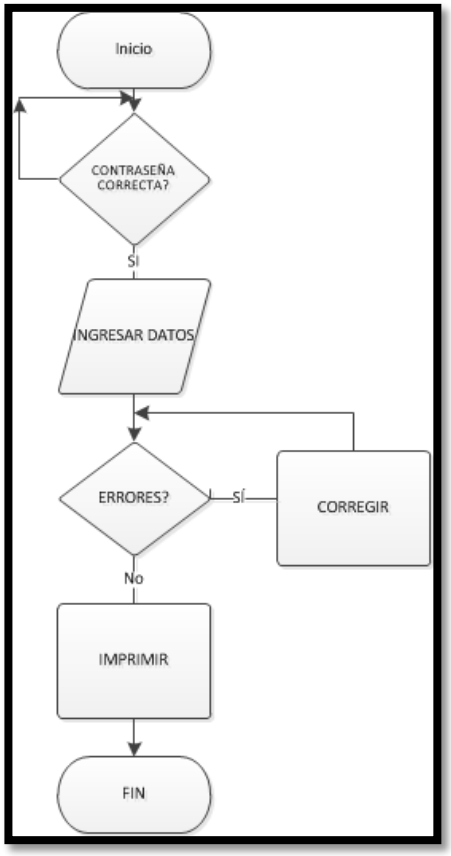


Figura 4.10: Diagrama de flojo general del sistema de impresión braille
Fuente: Autor

4.3.1 Interface de usuario

La interface de presentación en el ordenador del sistema de impresión braille presenta como característica un diseño gráfico atractivo para el usuario (ver figura 4.10). Éste debe ser una persona con capacidades visuales que le permitan observar las características de la interface y que pueda manipular toda la funcionalidad del sistema de impresión braille (ver figura 4.11).

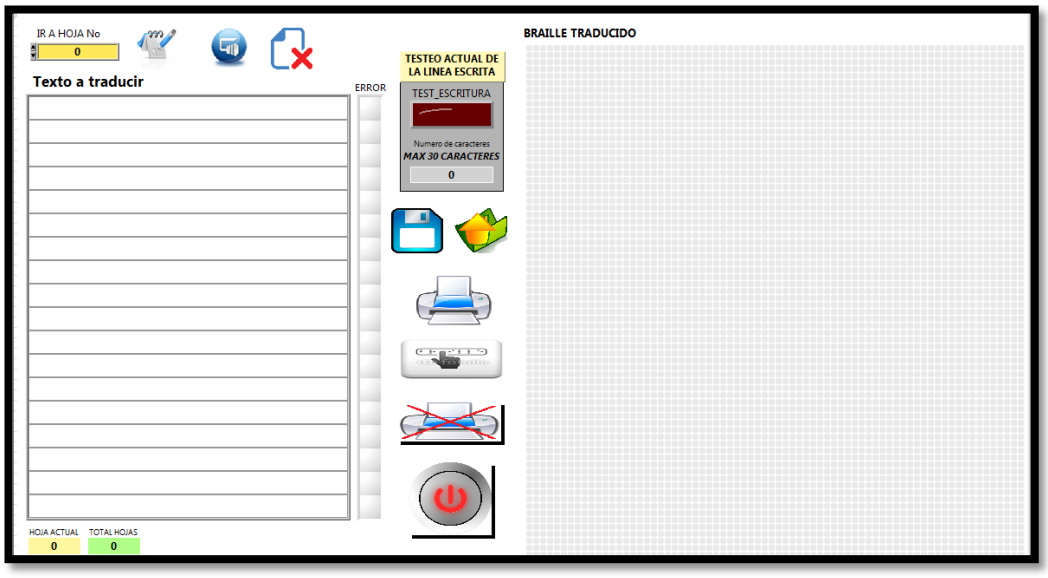


Figura 4.11: Pantalla de ingreso de caracteres

Fuente: Autor

La interfaz gráfica consta de las siguientes partes:

- ✓ Matriz en donde se ingresa el texto que se desea imprimir.
- ✓ Botón de verificación de texto.
- ✓ Botón de borrado de texto.
- ✓ Contador de caracteres por filas.
- ✓ Botón guardar.
- ✓ Botón de lectura de archivo.
- ✓ Botón de impresión.
- ✓ Botón de traducción a braille.
- ✓ Botón de salir del programa.
- ✓ Matriz en donde se ve los puntos que deben quedar grabados en la hoja.

Figura 4.12: Pantalla de inicio de la interfaz del sistema braille
Fuente: Autor

4.3.2 Programación en LabVIEW

La programación gráfica en LabVIEW se realizó utilizando el principio de máquinas de estado, las cuales se fueron diagramando de acuerdo a los requerimientos del sistema de impresión. El diagrama de bloques total se puede apreciar en el ANEXO 2, pero para que el funcionamiento sea entendido de mejor manera el código será explicado por partes.

4.3.2.1 Ingreso de usuario

El ingreso de usuario consta simplemente de un arreglo de caracteres que es ingresado por defecto dentro del diagrama de bloques. En este caso se ha puesto como contraseña “braille_uda”, aunque puede ser modificada al antojo del usuario, basta con cambiar la palabra que va a ser comparada en la programación (ver figura 4.13).

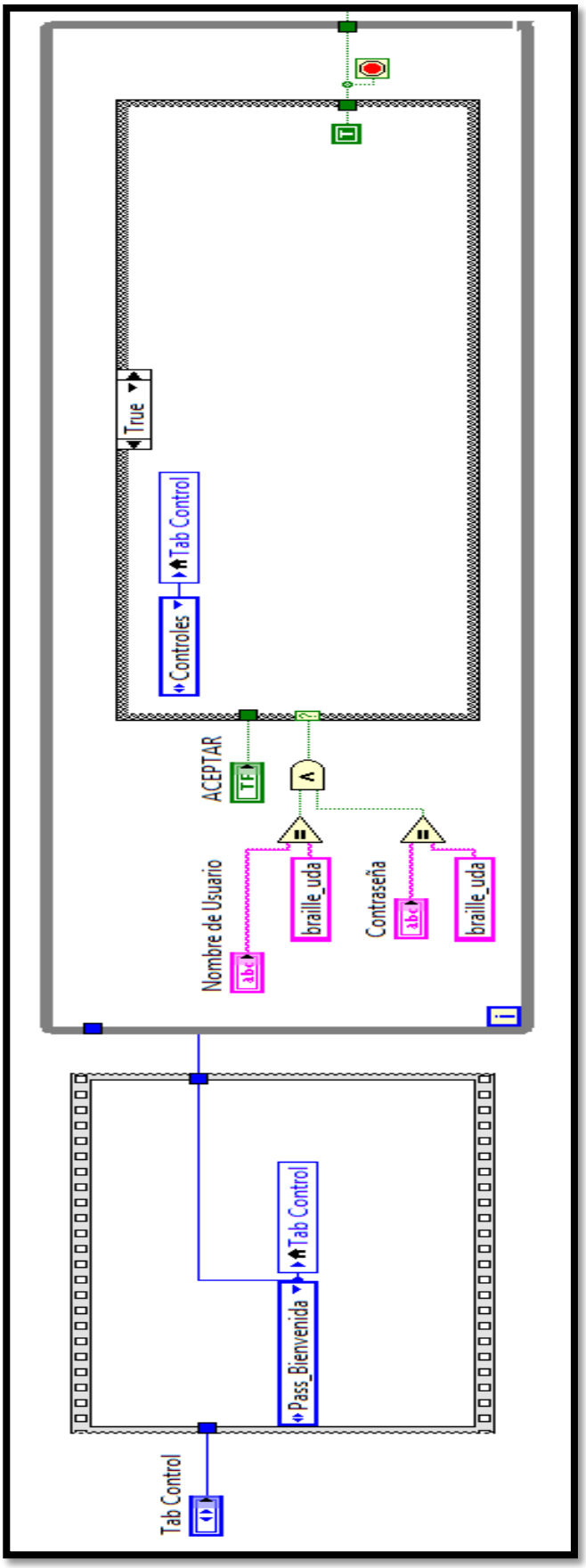


Figura 4.13: Diagrama de bloque para el ingreso de contraseña

Fuente: Autor

4.3.2.2 Máquina de estados

Es un sistema jerárquico, secuencial y cíclico de programación, en donde se realizan los procesos de acuerdo a un orden preestablecido. Los estados que se programarán en el diagrama de bloques se pueden observar en la figura 4.14:

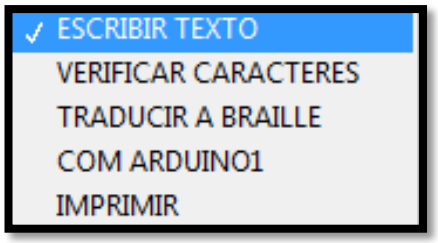


Figura 4.14: Estados que se encuentran el diagrama de bloques

Fuente: Autor

El diagrama de estado se puede observar en la figura 4.15:

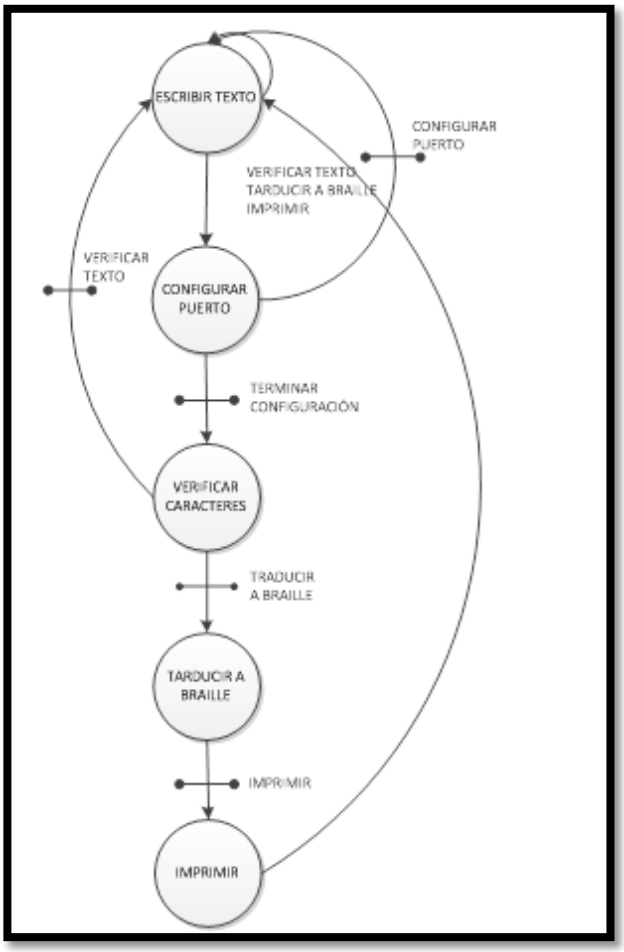


Figura 4.15: Diagrama de estados

Fuente: Autor

4.3.2.2.1 Estado ESCRIBIR TEXTO

Este estado es el inicial (ver figura 4.16), puesto que una vez ejecutado el programa e ingresado la clave, lo primero que va a verificar el software es si se ha ingresado los caracteres en la matriz predispuesta para ello. Dentro de este estado se pueden encontrar algunas de las funciones que muestran en la interfaz

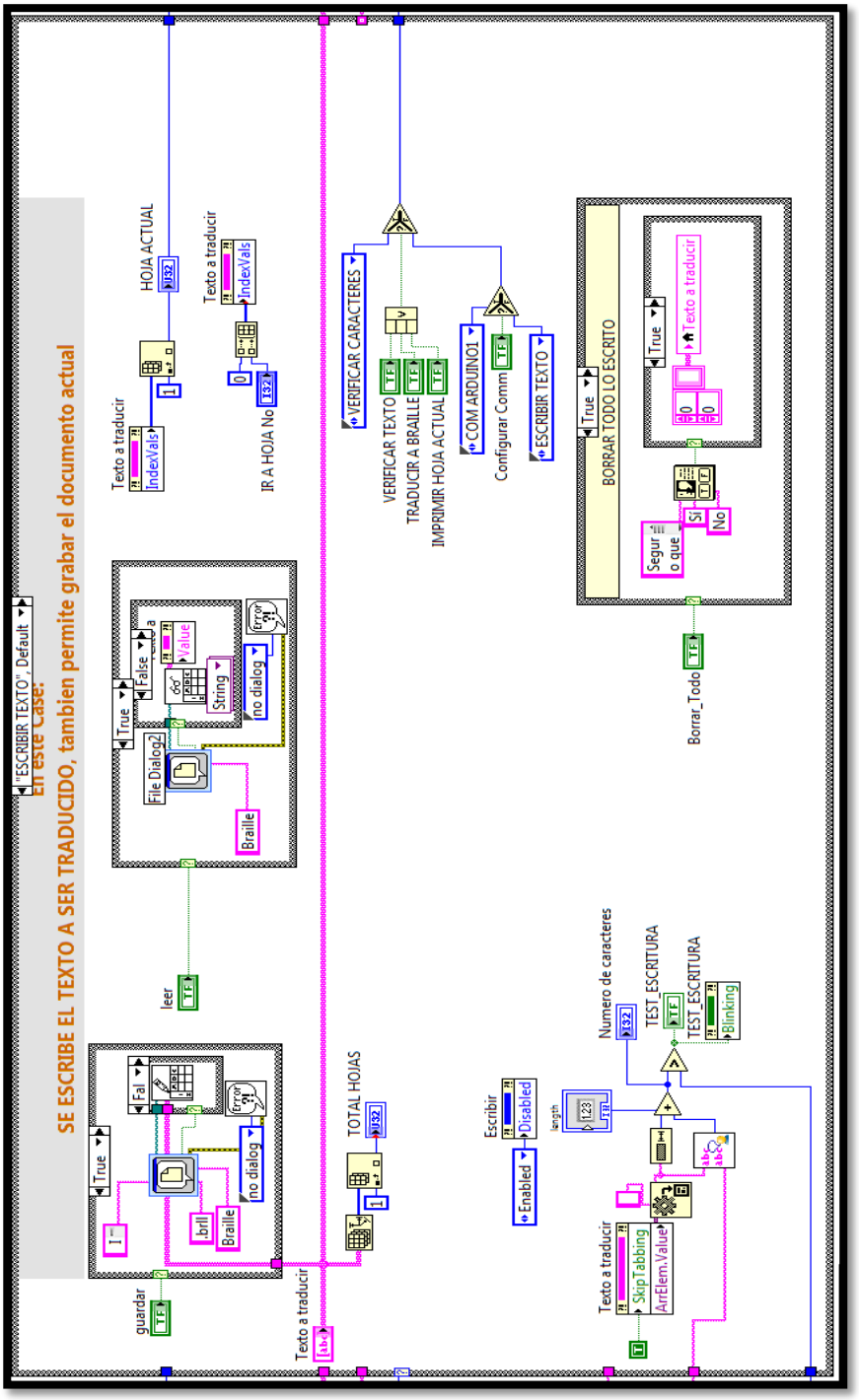


Figura 4.16: Estado "ESCRIBIR TEXTO"

Fuente: Autor

4.3.2.2.1.1 Guardar el documento actual

Permite al usuario guardar el texto escrito en la interfaz para futuros usos (ver figura 4.17). A la extensión del archivo se le asignó el nombre de .brll, y puede ser modificable al gusto del programador en el diagrama de bloque.

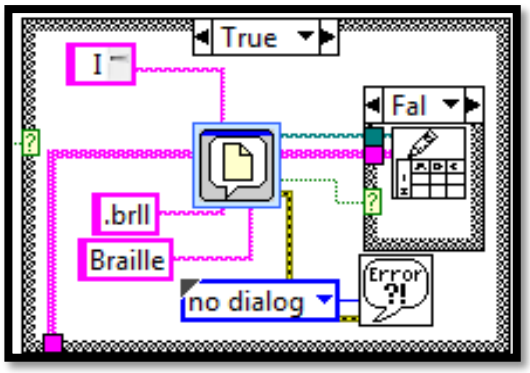


Figura 4.17: Diagrama de bloque para guardar documentos .brll

Fuente: Autor

4.3.2.2.1.2 Abrir un archivo .brll

Permite al usuario abrir algún documento de extensión .brll que haya sido grabado anteriormente (ver figura 4.18).

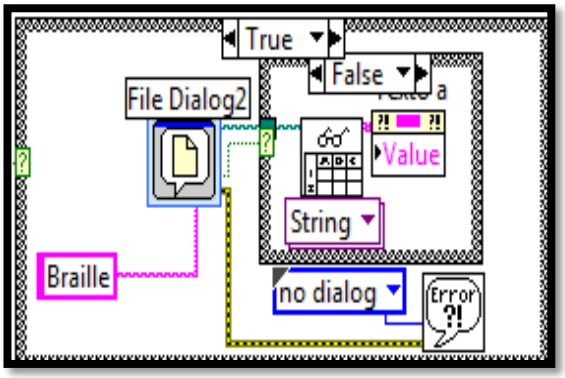


Figura 4.18: Diagrama de bloque para abrir un archivo .brll

Fuente: Autor

4.3.2.2.1.3 Borrar texto

Permite al usuario borrar el texto que se ha ingreso (ver figura 4.19). Antes de borrarlo tiene una pregunta de confirmación para evitar borrados accidentales.

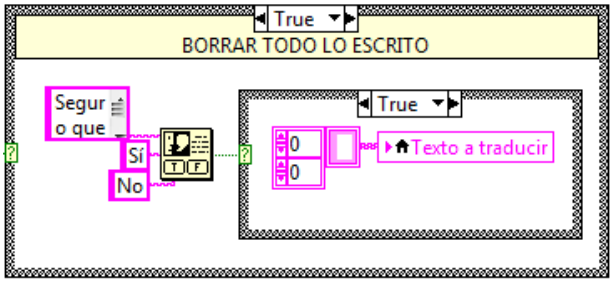


Figura 4.19: Diagrama de bloque para borrar un archivo .brll

Fuente: Autor

4.3.2.2.1.4 Contar caracteres

Para evitar desbordes de impresión se ha predispuesto que cada línea contengo 30 pares de columnas de escritura braille, por ello es necesario ir contando los caracteres para evitar salirse de la hoja (ver figura 4.20). Se debe tener en cuenta que cada letra ocupa dos columnas para su traducción a braille, mientras que los caracteres especiales ocupan 4. Si hubiese un exceso de caracteres en cada línea, se dispone de un LEDque se enciende a rojo cuando esto ocurre.

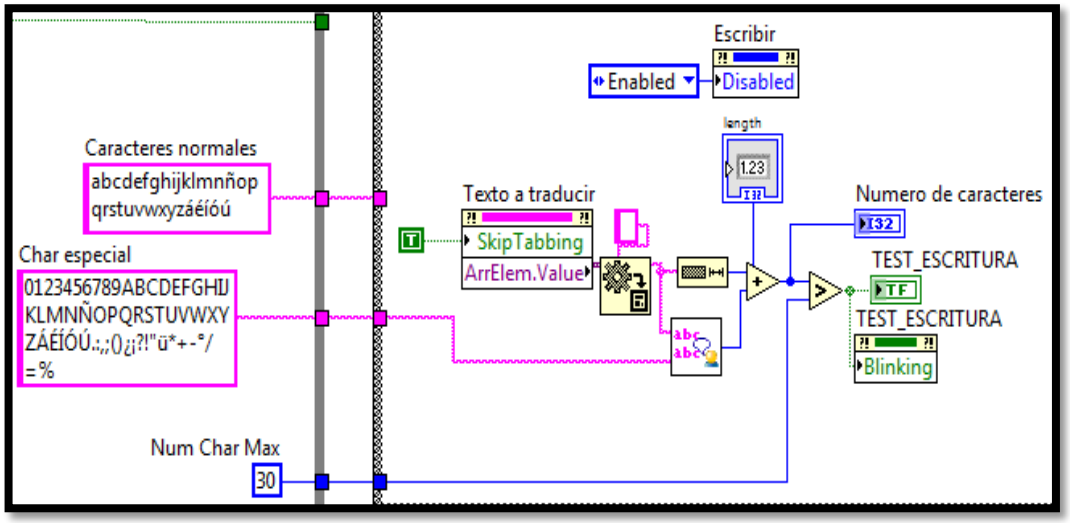


Figura 4.20: Diagrama de bloque para contar caracteres braille

Fuente: Autor

4.3.2.2.2 Estado VERIFICAR CARACTERES

Como su nombre lo indica se encarga de revisar si no existe alguna incongruencia al momento de ingresar el texto que se desea imprimir en braille (ver figura 4.21). Tiene en su estructura un *subVI* que se encarga de contar los caracteres y determinar cuántas columnas se requiere para su impresión en el papel (ver figura 4.22). De

existir un conteo de caracteres mayor a 30, al momento de imprimir aparecerá un mensaje indicando que no puede ejecutar la orden de impresión.

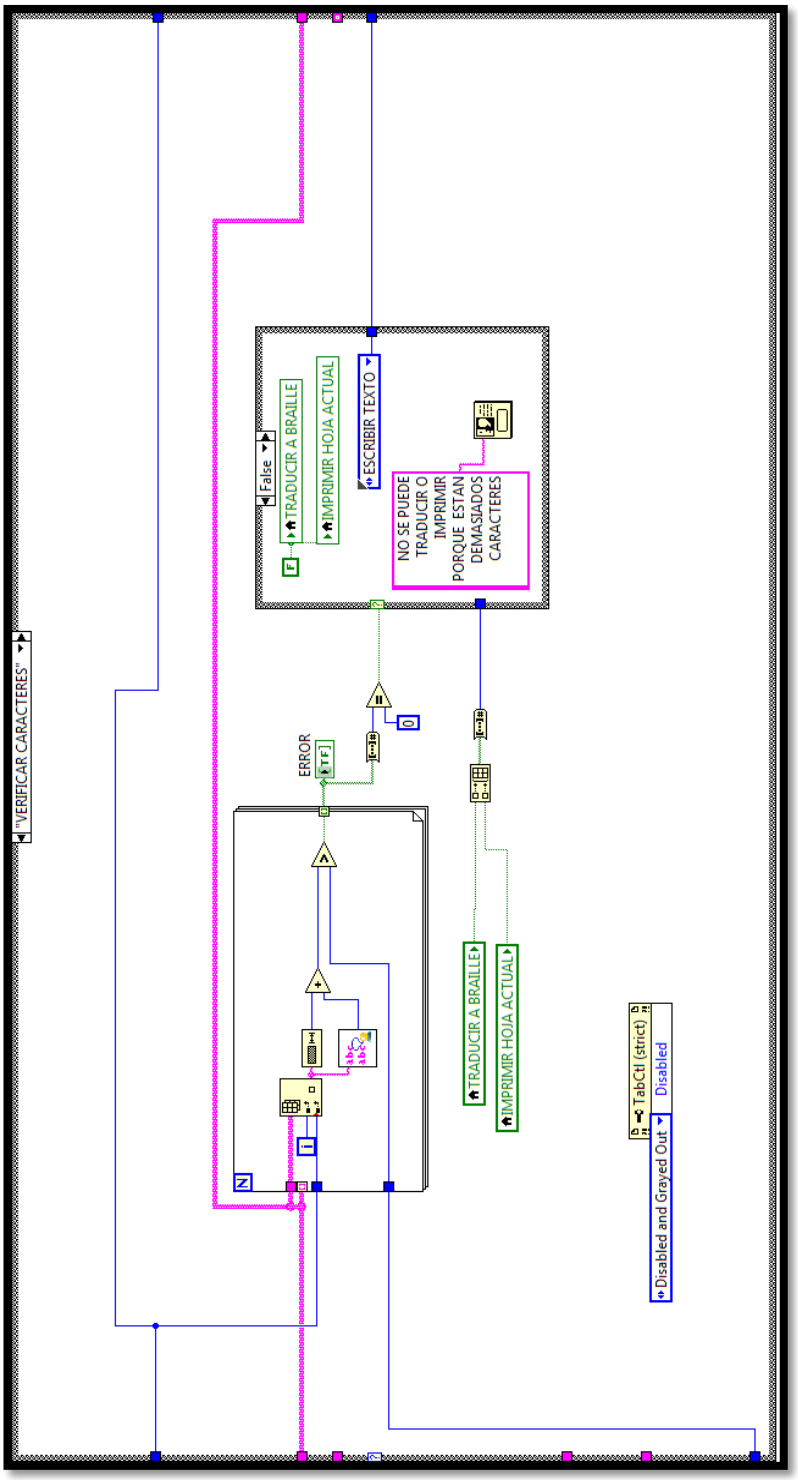


Figura 4.21: Diagrama de bloque para verificar caracteres

Fuente: Autor

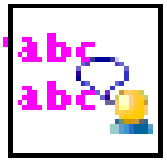


Figura 4.22: SubVI para asignar el número de columnas por caracter

Fuente: Autor

4.3.2.2.3 Estado TRADUCIR A BRAILLE

Se encarga de traducir a braille y mostrar un resultado previo en la pantalla de lo que se va a imprimir en la hoja (ver figura 4.23). Dentro de estado se encuentra el *subVI* que se encargará, además de lo antes mencionado, de traducir los datos booleanos en números que oscilan entre 0 y 7 para que puedan ser enviados al Arduino (ver figura 4.24).

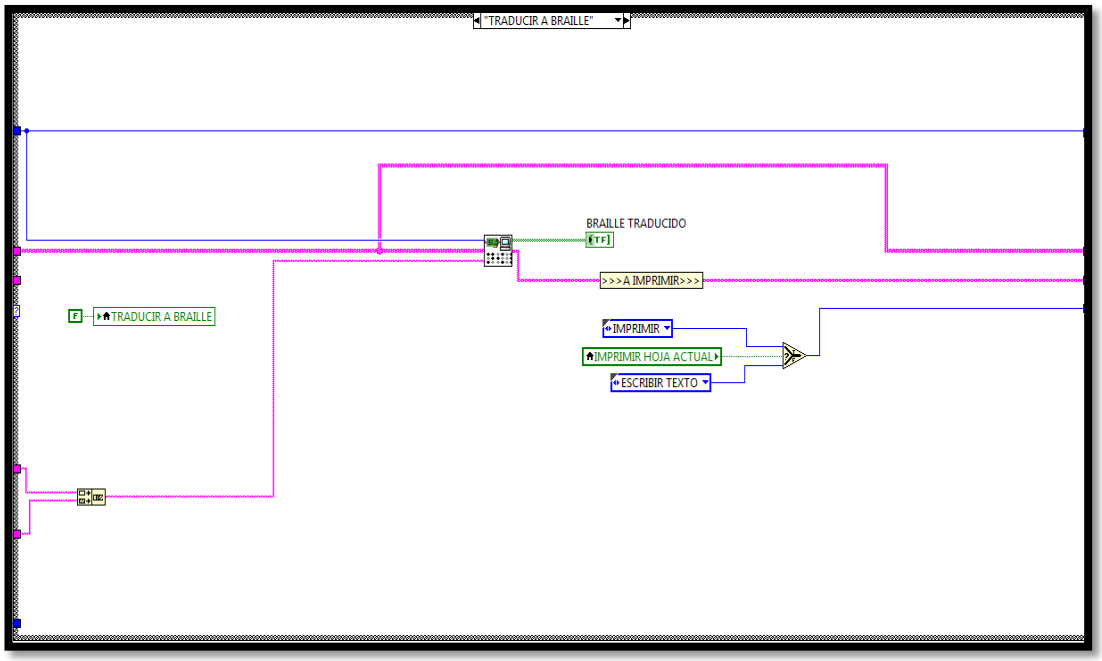


Figura 4.23: Diagrama de bloques para traducir a braille

Fuente: Autor

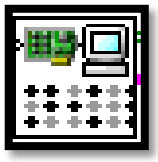


Figura 4.24: SubVI para traducir a braille y convertir arreglos booleanos a números

Fuente: Autor

4.3.2.2.4 Estado COM ARDUINO1

En este estado se encuentra un *subVI* que se encarga de configurar el enlace de comunicación entre el Arduino y LabVIEW (ver figura 4.25)

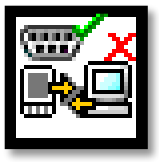


Figura 4.25: SubVI para configuración de enlace entre Arduino y LabVIEW
Fuente: Autor

4.3.2.2.5 Estado IMPRIMIR

Es el estado clave de la impresión braille (ver figura 4.26). Se puede observar el diagrama de estados que se encuentran dentro del estado IMPRIMIR en la figura 4.27. Estos sub estados se encargarán de:

1. Configurar el puerto de transmisión de datos a través de una interfaz gráfica.
2. Enviar los datos que sólo pueden ser caracteres entre el 0 y el 7. Para que el Arduino sepa que los datos van a ser transmitidos, LabVIEW se encargará de enviar una letra “I”.
3. Enviar la letra “F” como señal de que son todos los caracteres que se han enviado por línea.
4. Una vez finalizada la línea y si existe más texto en líneas posteriores LabVIEW envía como dato una “E”, la cual le indica al Arduino que dé un *enter* y se posiciones en la siguiente línea.
5. Este diagrama de bloques limpia el buffer en donde se encuentra la cadena de caracteres que envía por línea.

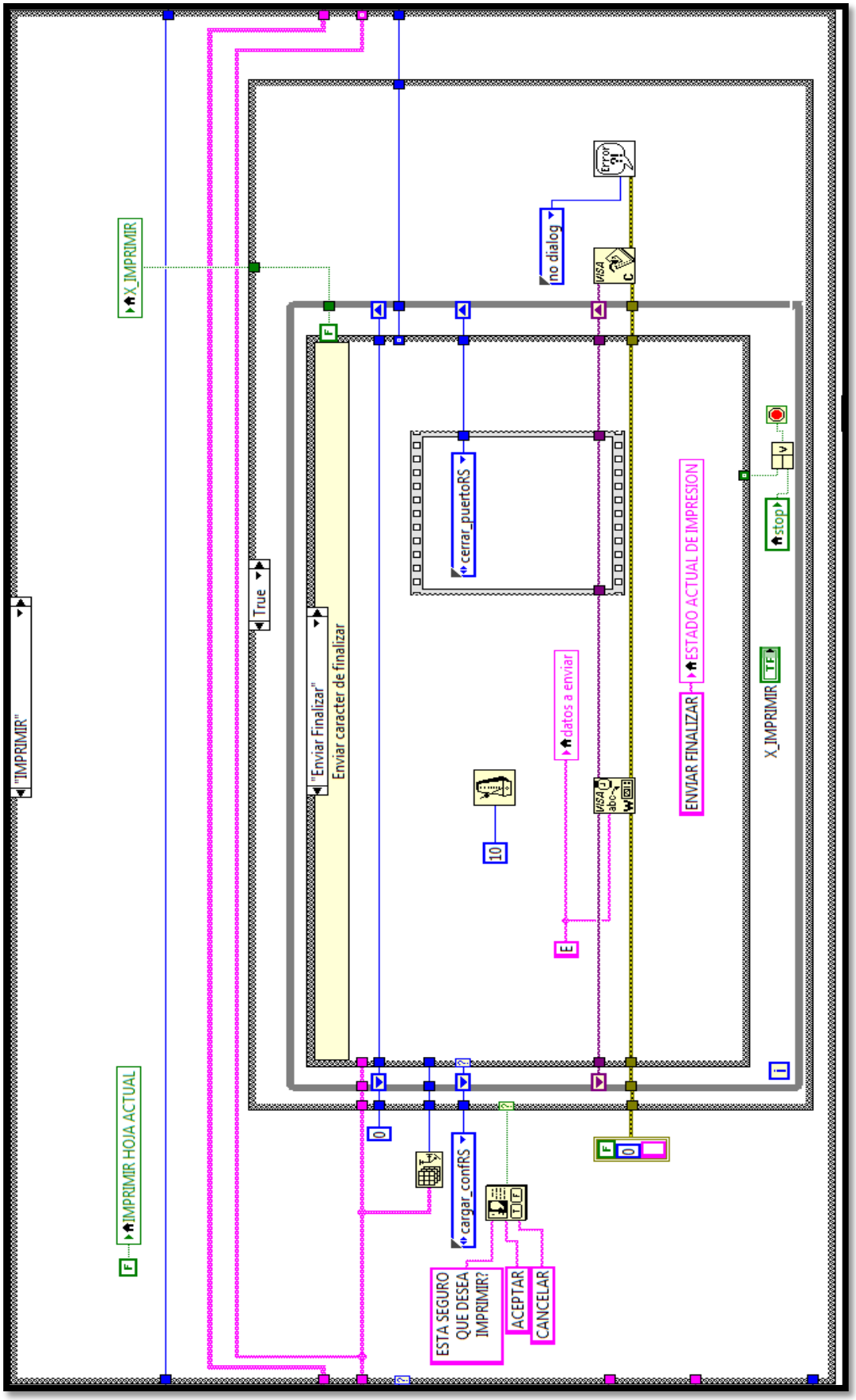


Figura 4.26: Diagrama de bloques para imprimir

Fuente: Autor

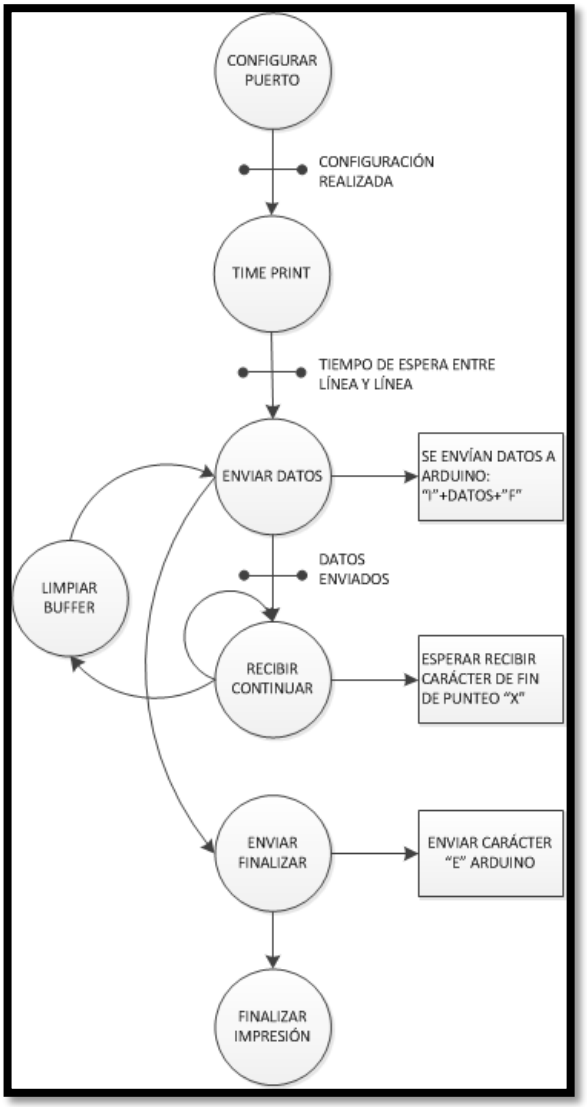


Figura 4.27: Diagrama sub estados dentro del estado IMPRIMIR
Fuente: Autor

4.3.3 Programación en Arduino

El Arduino Mega se encargará de ser el interlocutor entre LabVIEW y los *drivers*, estos últimos se encargarán del manejo de los motores paso a paso, quienes desplazarán la hoja y el solenoide. Para facilitar la comunicación se utilizó una librería denominada "DELduino.h", quedando el código rigiéndose por el diagrama de flujo que se muestra en la figura 4.28.

Figura 4.28: Diagrama de flujo de la programación en Arduino

Fuente: Autor

4.3.3.1 Configuración de puertos del Arduino

En este punto se procede a asignar las salidas digitales del Arduino para el control de los motores paso a paso y del solenoide (ver figura 4.29).

```
PinOut direl(2);
PinOut stepl(3);
PinOut enabl(4);
```

Figura 4.29: Ejemplo de asignación de puertos en el Arduino

Fuente: Autor

4.3.3.2 Asignación de variables

Como en todo lenguaje de programación, se necesitarán de variables para el desarrollo del programa. En este código se utilizarán variables enteras, booleanas y de tipo *char*. La figura 4.30 muestra una sección del código en donde se declaran algunas de las variables a utilizar.

```
char caracter;
char letras[61] = "0000000000000000000000000000000000000000000000000000000000000000";
int letra = 0;
boolean ban = false;
```

Figura 4.30: Ejemplo de declaración de variables

Fuente: Autor

4.3.3.3 Posición inicial del solenoide

El solenoide debe encontrarse en una posición inicial para que pueda empezar la función de punteo, por cuanto se ha colocado una regleta perforada que queda por debajo del papel una vez éste haya ingresado y esté listo para iniciar la impresión. Para que el solenoide quede exactamente sobre el punto de inicio se ha recurrido al método de prueba y error alimentando el motor de desplazamiento en X con pulsos para que pueda girar lo necesario. La figura 4.31 muestra cuáles son estos valores.

```
  while(1)
  {
    L_R.Out1();
    pulsl = analogRead(A1);
    if(pulsl < 500){break;}
    stepl.Out1();
    delayMicroseconds(500);
    stepl.Out0();
    delayMicroseconds(300);
  }
  direl.Out1();
  delay(100);

  for(vec = 0; vec < 330; vec++)
  {
    stepl.Out1();
    delayMicroseconds(500);
    stepl.Out0();
    delayMicroseconds(500);
  }
  enabl.Out0();
  L_R.Out0();
}
```

Figura 4.31: Código de reseteo

Fuente: Autor

En la figura anterior el lazo *While*(1), acciona el motor de desplazamiento horizontal para que llegue al tope del lado derecho, en donde se encuentra un fin carrera. Una vez accionado éste último, el lazo *for* se encargará de llevarlo hacia la izquierda a su posición inicial de impresión.

4.3.3.4 Lectura de sensores de luz

Los sensores de luz, a través de los motores paso a paso, arrastran la hoja para dejarla en la posición inicial de impresión. Para esta función se utiliza el motor de paso posicionado en la parte baja que se encargará de realizar el giro correspondiente para la ubicación exacta de la hoja. En esta parte del código se testean los sensores de luz para, en primera instancia, verificar si hay algún papel. Una vez detectado el mismo, se procede al accionamiento del motor, éste girará hasta que el segundo sensor detecte el papel, agregará unos giros más a través de una subrutina para que el papel quede en la posición inicial de impresión (ver figura 4.32).

```
while(1)
{
pulsl = analogRead(A3);
if(pulsl < 700)
{

 dire2.Outl();
 enab2.Outl();
 while(1)
 {
  pulsl = analogRead(A2);
  if(pulsl < 600)
  {
   for(vec = 0; vec < 50; vec++)
   {
    step2.Outl();
    delayMicroseconds(500);
    step2.Out0();
    delayMicroseconds(500);
   }
   enab2.Out0();
   delay(500);
   break;
  }
```

Figura 4.32: Código de lectura de sensores y colocación del papel en posición inicial de impresión

Fuente: Autor

4.3.3.5 Lectura de datos

Una vez que LabVIEW envíe los datos y la orden de impresión se empezará a verificar el caracter de inicio de envío de datos “I”, fin de envío de datos “F” y salto de línea “E” (ver figura 4.33). Una vez recibido este carácter, Arduino saltará al método imprime () que será descrito más adelante.

```
while(1)
{
  if (Serial.available())
  {
   dato = Serial.read();
   if(dato == 'F')
   {
   ban = false;
   imprime();
   }
     if(ban == true)
   {

       letras[letra] = dato;
       letra++;
   }
   if(dato == 'I')
   {
   ban = true;
   }
   if(dato == 'E')
   {
     enab2.Out1();
     while (1)
     {
       pulsl = analogRead(A3);
       if(pulsl > 700){break;}
       delayMicroseconds(500);
       step2.Out0();
       delayMicroseconds(500);
     }
     for(vec = 0; vec < 100; vec++)
     {
      step2.Out1();
      delayMicroseconds(500);
      step2.Out0();
      delayMicroseconds(500);
     }
       enab2.Out0();
     }

  }

}
```

Figura 4.33: Código de recepción de datos

Fuente: Autor

Dentro del código mostrado en la figura anterior se puede apreciar que Arduino empieza el almacenamiento de datos una vez haya recibido el caracter “I”. Una vez recibido este caracter los siguientes datos serán caracteres tipo *char* representados por números desde el 0 al 7. Esto es porque se utilizarán los 3 últimos bits menos significativos para que el Arduino puede decodificar el mensaje. Los datos recibidos por el Arduino se van acumulando en el arreglo “letras”, mientras que la variable “letra” va variando la posición del dato dentro del arreglo. Para saber si la línea ha terminado, Arduino debe recibir la letra “F”. Una vez se haya terminado de imprimir la línea de caracteres braille, se enviará hacia LabVIEW una caracter “X” que indica que el trabajo ha finalizado.

Cada línea de caracteres braille no es más que una matriz de 3 filas por 60 columnas (60 es por el número de caracteres braille que permite LabVIEW que se ingrese por línea), por ello se programa un doble *for* para que el sistema de punteo se desplace y puntee a través del solenoide horizontalmente para hacerlo de forma más rápida que si se imprimiera en forma vertical.

Como ya se tienen los caracteres almacenados en el arreglo “letras”, el método imprime () lo que hace es comparar el bit menos significativo con un 1 (verdadero en booleano) a través de la función de lectura de bits *bitread* y luego ordena al solenoide si debe puntear o no. La fuerza con la que golpeará el solenoide, así como también el tiempo de espera para que éste regrese a su posición inicial, serán calibradas utilizando como siempre el método de prueba y error (ver figura 4.34).

```
caracter = letras[pos];
if(bitRead(caracter, fila) == 1)
{
bobin.Out1();
delay(100);
bobin.Out0();
delay(300);
}
```

Figura 4.34: Código de comparación de los 3 bits menos significativos
Fuente: Autor

Una vez comparado el primer bit de la primera columna se deberá ordenar al motor de desplazamiento en “X” que desplace el solenoide hacia la izquierda. Este

desplazamiento se logra a través de pulsos de alimentación calibrados utilizando el método de prueba y error (ver figura 4.35). Hecho esto, se procede a revisar el primer bit de la segunda columna y se realiza la misma comparación que se hizo con el primer bit de la segunda columna.

```
for(vec = 0; vec < 58; vec++)
{
 stepl.Out1();
 delayMicroseconds(500);
 stepl.Out0();
 delayMicroseconds(300);
}
```

Figura 4.35: Código de separación entre columnas del mismo caracter
Fuente: Autor

Una vez leídos los dos primeros bits de las dos columnas del carácter braille se procede a calibrar el motor de desplazamiento en "X" para que dé otro salto de caracter en caracter. La separación horizontal entre columna y columna del mismo carácter es menor que la separación entre pares de columnas, para esto se debe programar como se muestra en la figura 4.36.

```
for(vec = 0; vec < 90; vec++)
{
 stepl.Out1();
 delayMicroseconds(500);
 stepl.Out0();
 delayMicroseconds(300);
}
```

Figura 4.36: Código de separación entre pares de columnas
Fuente: Autor

Como se puede apreciar en las figuras 4.35 y 4.36, la única diferencia es el número de veces que se repite el lazo *for*. Con esto se deduce que a mayor sea el número más girará el motor de desplazamiento en "X". El proceso de lectura de bits se repite secuencialmente hasta que se termine de leer la primera fila hasta que el número de lecturas sea igual al número de caracteres que se ingresaron en la interface, para que luego los *drivers* ordenen al motor de desplazamiento en "Y" que gire, y así tragar un poco más la hoja y el sistema esté listo para empezar la segunda fila (ver figura 4.37).

```
dire2.Outl();
enab2.Outl();
for(vec = 0; vec < 10; vec++)
{
 step2.Outl();
  delayMicroseconds(400);
  step2.Out0();
  delayMicroseconds(300);
}
enab2.Out0();
delay(500);
```

Figura 4.37: Código de separación entre filas del mismo caracter

Fuente: Autor

El proceso se repite secuencialmente hasta se haya completado la lectura del último caracter. En este instante, el Arduino se encargará de enviar un carácter "X" que representa el fin de la impresión de la línea, entonces LabVIEW procede a enviar la segunda línea de igual forma como se mencionó al inicio.

Para determinar el espacio entre línea y línea de caracteres se realiza el mismo código de la figura 4.37. Lo único que se varía el número de veces que se realiza el lazo *for*(ver figura 4.38), puesto que, la separación entre filas del mismo caracter, es menor que el espacio vertical entre caracter y caracter.

```
dire2.Outl();
enab2.Outl();

for(vec = 0; vec < 25; vec++)
{
 step2.Outl();
  delayMicroseconds(400);
  step2.Out0();
  delayMicroseconds(300);
}
enab2.Out0();
```

Figura 4.38: Código de separación entre filas de caracteres

Fuente: Autor

4.4 Resultados y pruebas de funcionamiento

Para empezar con las pruebas de funcionamiento se procedió a armar en un *protoboard*, un circuito que permita comprobar el correcto funcionamiento de los

drivers de los motores. Esta prueba detallará si los *drivers* soportan el funcionamiento continuo de los motores paso a paso. También ayudará a verificar si estos últimos giran correctamente hacia la izquierda y a la derecha variando su velocidad de giro a través de Arduino.

Figura 4.39: Montaje de ArduinoMega 2560 y *drivers* en protoboard
Fuente: Autor

Al empezar a comprobar el funcionamiento de los motores paso a paso, se notó que los *drivers* sufrían un pequeño calentamiento, por lo cual se le proveyó de un pequeño ventilador para que ayude a eliminar el calentamiento excesivo (ver figura 4.40).

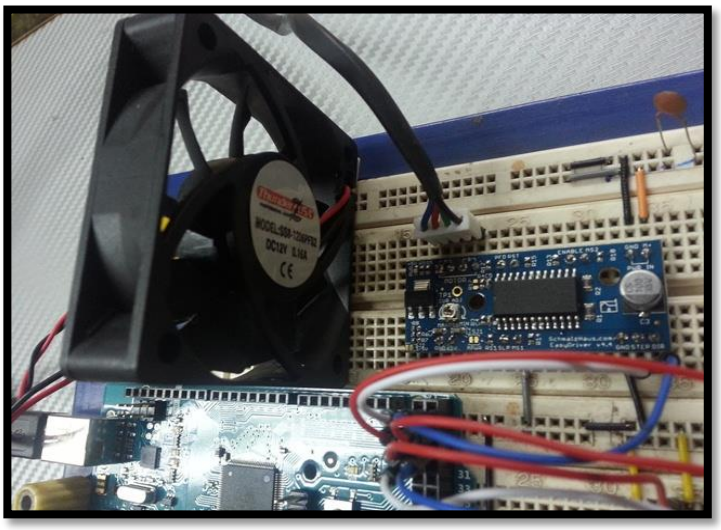


Figura 4.40: Acoplamiento de ventilador para reducir el calentamiento
Fuente: Autor

Para las pruebas de comunicación, se realizó un diagrama de bloques y una interfaz en LabVIEW a fin de verificar la comunicación entre éste y Arduino (ver figura

4.41). Para que la comunicación sea comprobada de una forma visual se predispuso sobre el *protoboard* un par de columnas de 3 filas de LED's, representando los últimos 3 bits menos significativos como se muestra en la figura 4.42.

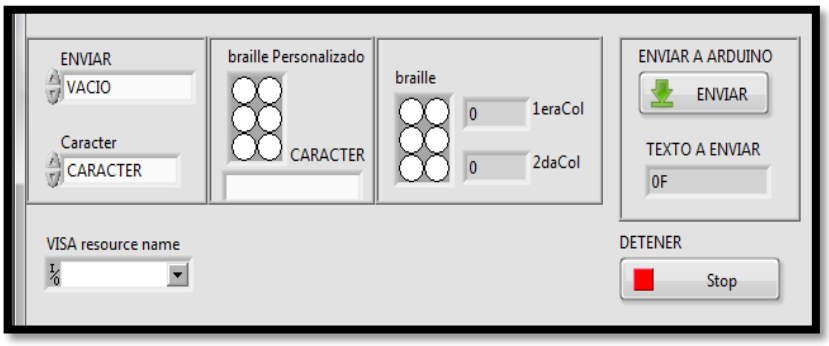


Figura 4.41: Interfaz de prueba para la transferencia de datos entre LabVIEW y Arduino
Fuente: Autor

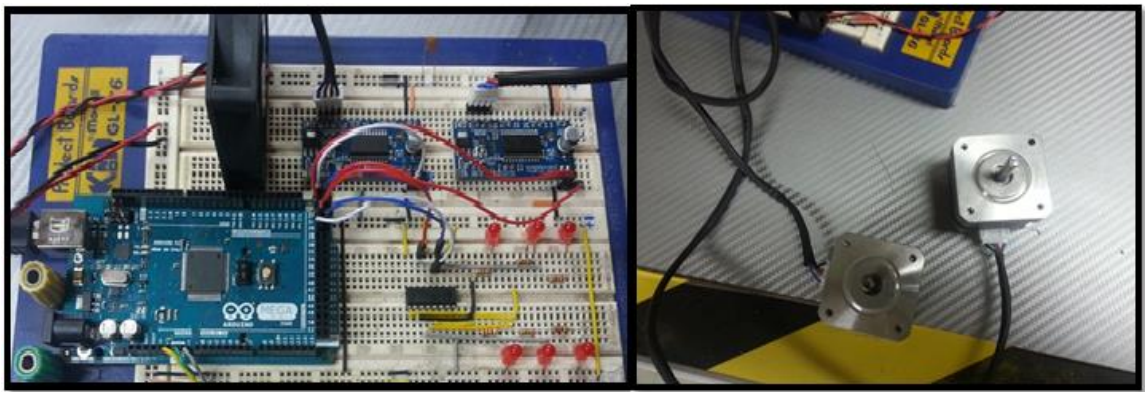


Figura 4.42: Montaje de *hardware* para comprobar la comunicación entre Arduino y LabVIEW
Fuente: Autor

En esta prueba se utilizaron algunas alternativas, tales como la de enviar caracteres desde la "A" hasta la "D", así como también la de enviar las celdas braille vacías, lo cual implica que no representa ningún caracter dentro del sistema, puesto que no se podría ser perceptibles al tacto. También se originó la idea de enviar los datos columna por columna a través de bytes, y utilizando sus 3 bits menos significativos a manera de datos booleanos determinar si son verdaderos o falsos.

Una vez comprobado el funcionamiento se diseñó una placa para que los *drivers* se acoplen físicamente sobre la tarjeta ArduinoMega2560, para así ahorrar espacio al momento del montaje. El diseño esquemático se muestra en la figura 4.43 y en las figuras 4.44 y 4.45 se puede apreciar el diseño en PCB realizado en el programa Eagle.

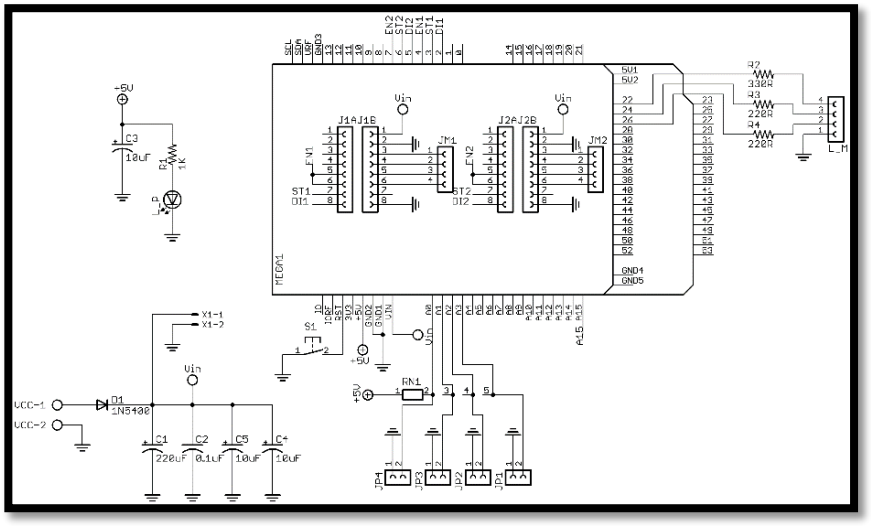


Figura 4.43: Diseño esquemático para montar los *drivers*

Fuente: Autor

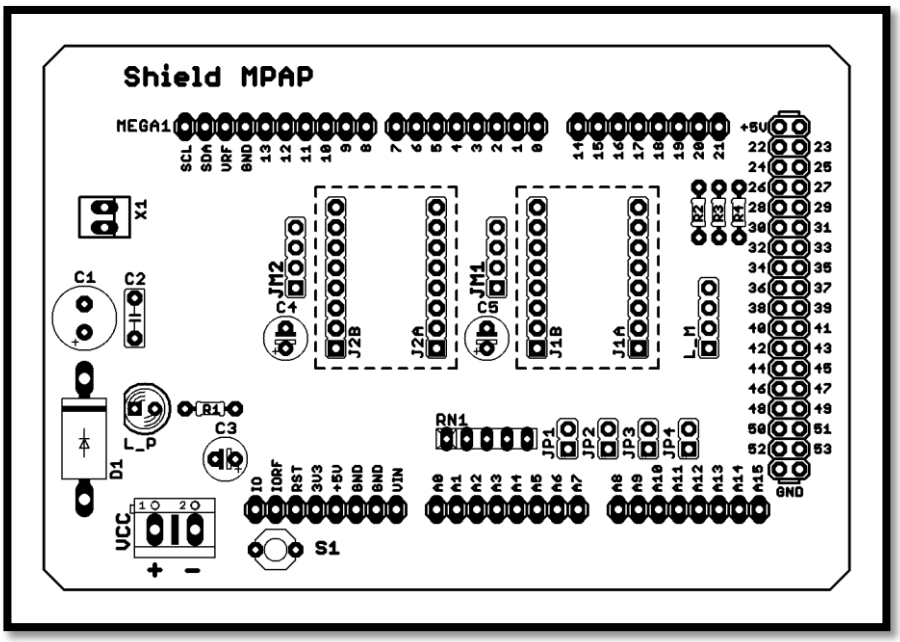


Figura 4.44: Capa inferior del PCB para montaje de *drivers*

Fuente: Autor

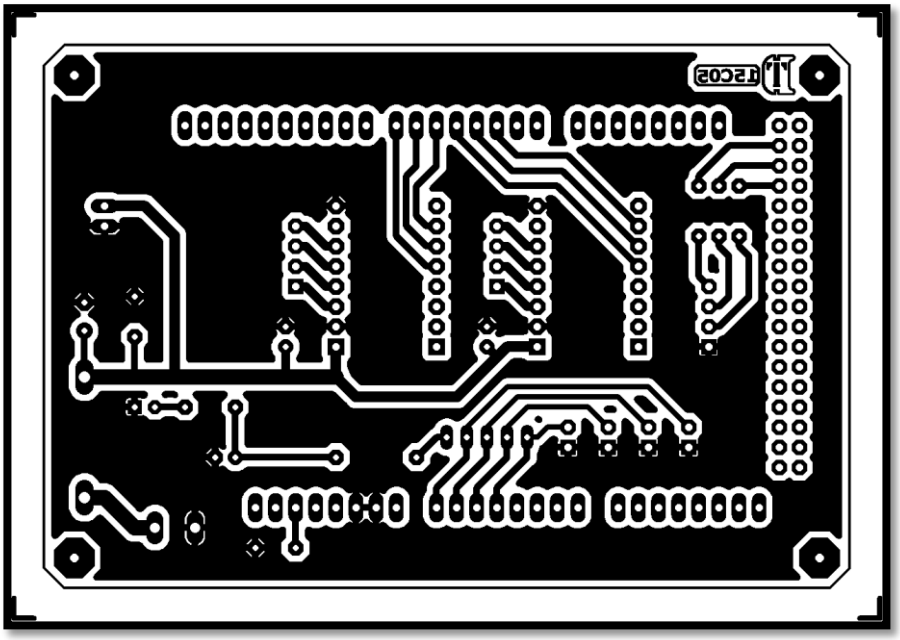


Figura 4.45: Capa superior del PCB para montaje de *drivers*
Fuente: Autor

Una vez obtenida la placa física se procede a soldar los elementos que la componen y a colocarla sobre la tarjeta ArduinoMega2560 (ver figura 4.46). Como se puede apreciar, para beneficiar aún más el proceso de evitar el sobre calentamiento de los *drivers* cuando los motores estén funcionando se adaptó un disipador de calor sobre cada uno de ellos.

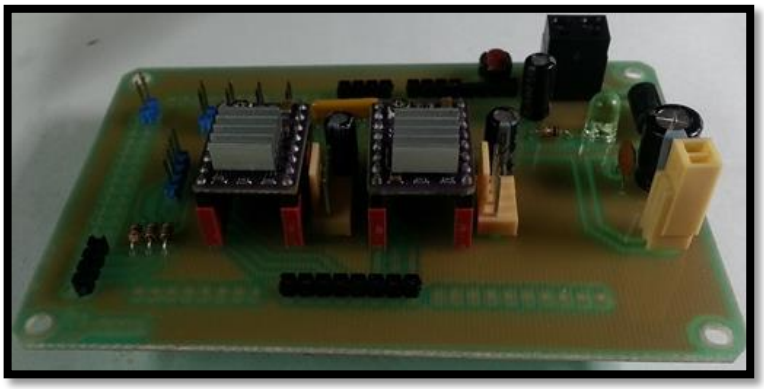


Figura 4.46: Placa para montaje de *drivers*
Fuente: Autor

Los elementos que están soldados en la placa de la figura 4.46 están descritos en la Tabla 4.1.

Tabla 4.1: Elementos y precios de la placa sobre la que están montados los *drivers*

ITEM	CANTIDAD	PRECIO TOTAL($)
Disipadores de calor	2	4
Condensador electrolítico (220 uF-25 V)	1	0.30
Condensador electrolítico (10 uF-25 V)	3	0.30
Condensador electrolítico (0.1 uF-25 V)	1	0.25
Peinetas y sockets	-	1.50
DVR 8825	2	60
Ventilador de 12 V	1	3
Resistencias (220 ohmios)	3	0.15
Resistencias (220 ohmios)	1	0.5
LED	1	0.20
Total		**70.20**

Fuente: Autor

Para el accionamiento y regulación de fuerza de golpe de solenoide se diseñó, en otra placa, un diseño esquemático y PCB para una fuente de voltaje regulable. Su propósito es, a través del método de ensayo y error, calibrar la fuerza de con la que la aguja golpeará la hoja para que quedé marcado en ella un punto que forme parte del carácter braille que se envíe a través de la computadora. Estos diseños se muestran en las figuras 4.47, 4.48 y 4.49 respectivamente.

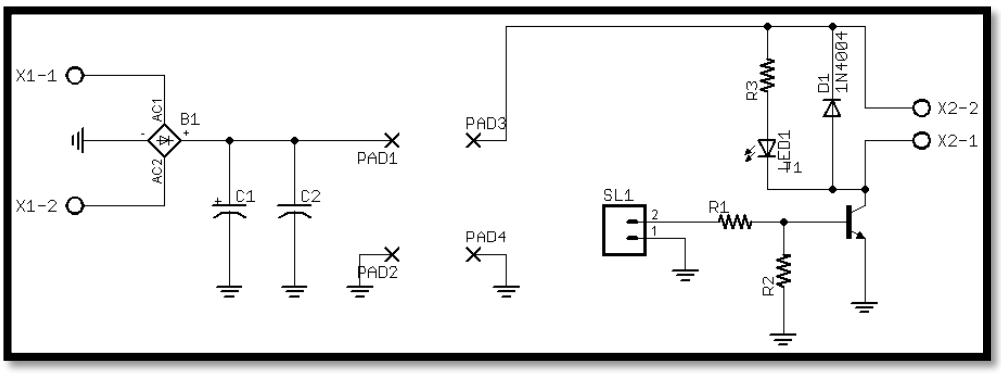


Figura 4.47: Diseño esquemático de la fuente regulable

Fuente: Autor

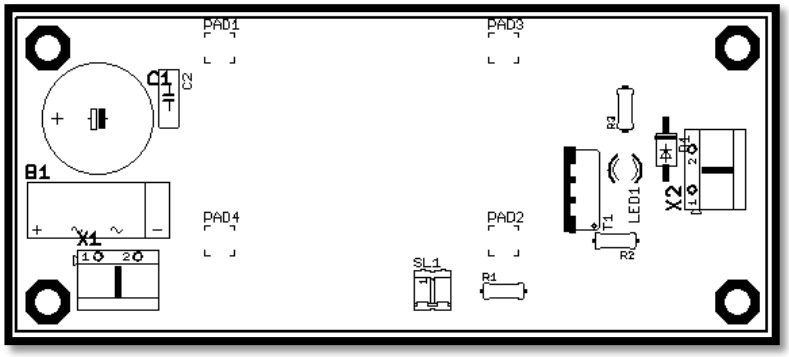


Figura 4.48: Capa inferior del PCB para fuente regulable

Fuente: Autor

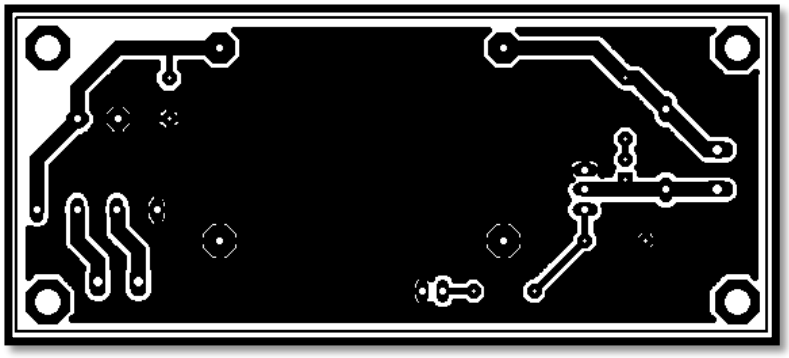


Figura 4.49: Capa superior del PCB para fuente regulable

Fuente: Autor

Una vez obtenidas las placas y los elementos que van sobre ella, se procede a soldar para obtener el resultado que se puede apreciar en la figura 4.50.

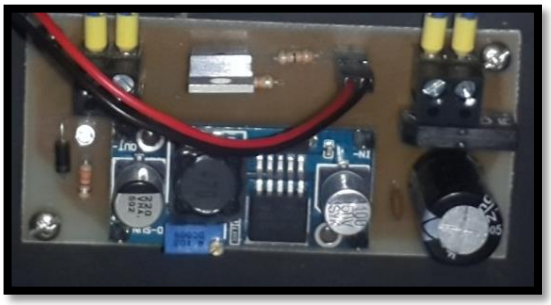


Figura 4.50: Placa de fuente regulable para control de disparo del solenoide

Fuente: Autor

Los elementos que están soldados en la placa de la figura 4.46 están descritos en la Tabla 4.2.

Tabla 4.1: Elementos y precios de la placa de la fuente regulable

ITEM	CANTIDAD	PRECIO TOTAL($)
Transistor irf 540	1	1.50
Condensador electrolítico (2200 uF-35V)	1	0.90
Puente de Graetz	1	0.40
LED RGB	1	1.25
Peinetas y sockets	-	3.00
LED	1	0.20
Total		**7.25**

Fuente: Autor

Una vez comprobada la comunicación entre estas dos plataformasde programación (LabVIEW y Arduino), calibrados los sensores de recepción, golpeo del solenoide y arrastre del papel, a través del control de los motores paso a paso permitiendo que la aguja golpee en los lugares precisos. Luego, se procedió al armado de la maqueta y la colocación de sus elementos eléctricos y montaje de componentes electrónicos dentro y sobre la misma. El funcionamiento y comprobación del sistema de impresión se lo puede evidenciar en elvídeo presentado en el ANEXO 4 y en la figura 4.51.

Figura 4.51: Hoja de prueba en la que se han resaltado los relieves

Fuente: Autor

CONCLUSIONES

- El sistema de control fue diseñado con una interfaz sencilla para que cualquier persona que no posea un alto grado de deficiencia visual pueda acceder a ella y manipularla a su conveniencia.

- Se trató de minimizar costos para que este sistema esté al alcance de cualquier economía.

- Se deben utilizar hojas adecuadas para la impresión de un texto. En este caso se desarrollaron todas las pruebas con hojas de polietileno, las cuales cumplieron las expectativas al momento de la impresión.

- La maqueta sobre la que está montado el sistema de impresión braille es fácilmente desarmable para que cualquiera de sus piezas pueda ser cambiada.

- Se utilizaron motores paso a paso en lugar de servomotores debido a la alta precisión y fácil calibración de los primeros.

- En la operación del sistema, solamente se tienen desplazamientos horizontales en las direcciones X e Y de la hoja. No hay desplazamientos angular ni vertical.

RECOMENDACIONES

- Se recomienda que las personas que van a estar encargadas del manejo del sistema de impresión braille, reciban la asesoría por parte del programador para que lo puedan adecuar a sus propias necesidades, y así evitar posibles daños en el *hardware*.

- Con este sistema de impresión braille, quedan abiertas posibles mejoras a futuro tanto en *hardware* como en *software*.

- Se deben investigar más opciones para el punteo grupal de las celdas braille, por cuanto, al depender de una sola aguja, la impresión es relativamente lenta.

- Se recomienda que no se imprima un trabajo extenso de manera continua, debido a que se si prolonga la operación por más de 60 minutos, se podría producir el calentamiento de los *drivers* que controlan los motores.

- Se debe evitar ambientes con excesiva luz, puesto que el sensor TCRT 5000 variaría en sus mediciones y por ende haría que el sistema no funcione correctamente con los datos por defecto que le fueron configurados.

BIBLIOGRAFÍA

ABASCAL, J. U. L. I. O. (2002). Interacción persona-computador y discapacidad. Revista Minusval, número especial dedicado a la" Discapacidad y las Nuevas Tecnologías", Ministerio de Trabajo y Asuntos Sociales, 18-21. Consulta 22-06-2014.

ARDUINO. Examples 2014. http://www.arduino.cc/en/Tutorial/AnalogInOutSerial. Consulta: 22-07-2014.

BITTER, Rick, MOHIUDDIN Taqi, NOWROCKI Matt. LabVIEW AdvancedProgramming Techniques.Second Edition 2008. Consulta 18-07-2014.

CARMJ, C. (2005). Las personas con discapacidades: los excluidos de la educación superior. Consulta 18-06-2014.

CARPIO Miranda, M. E., Cárdenas Sanchez, T. A., &Chavez Burbano, P. X. (2014). Desarrollo e implementación de un sistema de seguridad y confort para hogares monitoreado y administrado a través de una aplicación web. Consulta 20-08-2014.

EJEMPLOS DE PROGRAMACIÓN CON ARDUINO. 2014. http://www.makeblock.es/tutoriales/ejemplos_programacion_arduino/ Consulta: 20-01-2015.

ESCUELA Politécnica Nacional. (n.d.). Capítulo 3, Desarrollo del software e interfaz de usuario. Consulta 12-09-2014.

ESPAÑOL, U. (2004). Necesidades educativas especiales en el contexto universitario español. Consulta 15-05-2014.

LILLO Barberá, C. (2012). Validación del campímetro ATD para el estudio línico de la retinopatía diabética. Consulta 19-06-2014.

MANUAL DE ARDUINO. (2009). rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf. Consulta 12-12- 2014.

Márquez Montalvo, S. A., & Miranda Vega, M. A. (2012). Análisis, diseño e implementación de un Portal Web para el Consejo Nacional de Discapacidades del

Ecuador (CONADIS) aplicando estándares de usabilidad, accesibilidad web utilizando un CMS (administrador de contenidos) y herramientas web 2.0 (Doctoral dissertation, SANGOLQUÍ/ESPE/2012). Pg. 158. Consulta 01-06-2014.

Herrador, R. E. (2009). Guía de Usuario de Arduino. Universidad de Córdoba. Pg. 8

MARTÍNEZ-LIÉBANA, I., & Chacón, D. P. (2004). Guía didáctica para la lectoescritura braille. Organización Nacional de Ciegos Españoles.

MUNDIAL, B. 2001. Informe mundial sobre la discapacidad. Consulta 22-07-2014

NATIONAL INSTRUMENTS. 2001 LabVIEW Advanced Course Manual. Consulta 27-11-2014.

NATIONAL INSTRUMENTS. NI Developer Zone. LabVIEW Web Services [Material gráfico proyectable]. 2011. http://zone.ni.com/wv/app/doc/p/id/wv-810. Consulta 02-08-2014.

NATIONAL INSTRUMENTS. NI Suppor. Using Libraries in LabVIEW Projects. 2011. http://zone.ni.com/reference/en-XX/help/371361F-01/lvconcepts/project_libraries/. Consulta: 29-08-2014.

NATIONAL INSTRUMENTS. NI Developer Zone. Using the LabVIEW Shared Variable.2011. http://zone.ni.com/devzone/cda/tut/p/id/4679.Consulta: 01-09-2014.

s.a. (2012). Discapacidad Ecuador Misión solidaria Manuela Espejo. Obtenido de Discapacidad online: http://www.discapacidadonline.com/discapacidad-ecuador-mision-solidaria-manuela-espejo.html. Consulta 15-07-2014.

SCIELO, (2014) http://www.scielo.cl/scielo.php?pid=S0718-221X2004000100006&script=sci_arttext. Consulta 07-11-2014.

SAMBRANO, R. 2007, Caracterización de los estudiantes con discapacidad, Venezuela. 2007. Pg.47, 48.

TRAVIS, J., & Kring, J. (2006). LabVIEW for Everyone: Graphical Programming Made Easy and Fun (National Instruments Virtual Instrumentation Series). Prentice Hall PTR.

TUTORIALES ARDUINO. 2014. http://www.prometec.net/curso-taller-de-arduino/. Consulta: 15-01-2015.

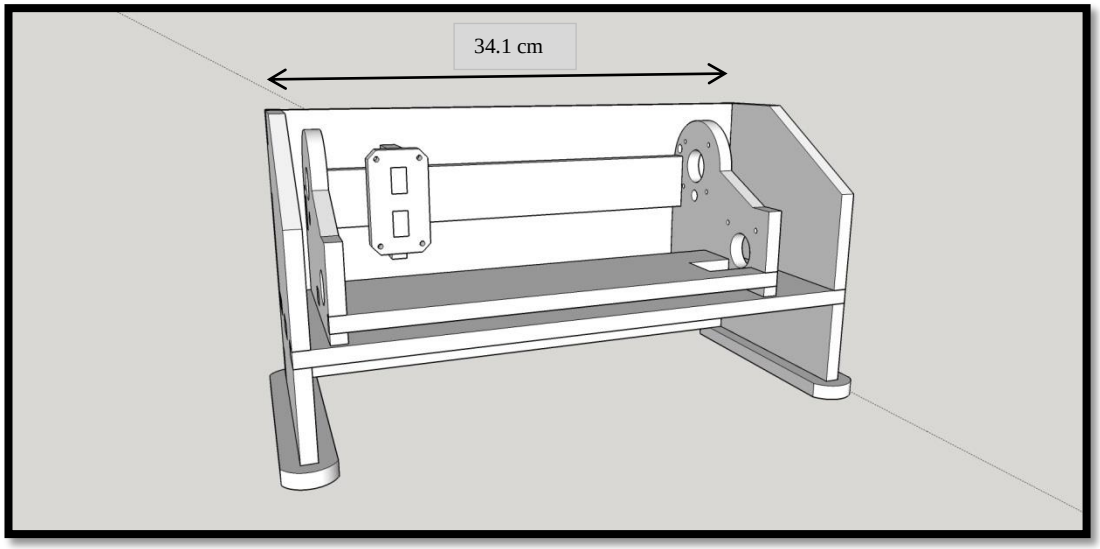
VISUALES, L., SALAS, A. C., RIVERA, C. A. C., & OLARTE, L. F. S (2005). Sistema de enseñanza del código braille para niños con discapacidades visuales. Pg. 19-20.

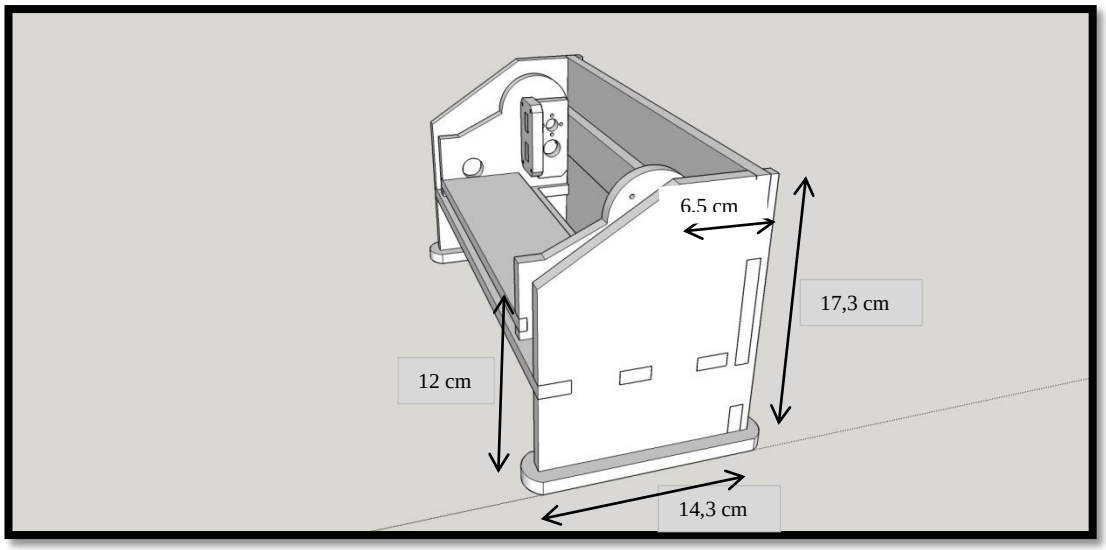
Vizcaíno, J. R. L., &Sebastiá, J. P. (2011). LabView: entorno gráfico de programación. Marcombo. Pg. 22.

Zappalá, D., Köppel, A., &Suchodolski, M. (2011). Inclusión de TIC en escuelas para alumnos con discapacidad visual. Buenos Aires, Argentina: Ministerio de Educación de la Nación. Pg. 9.

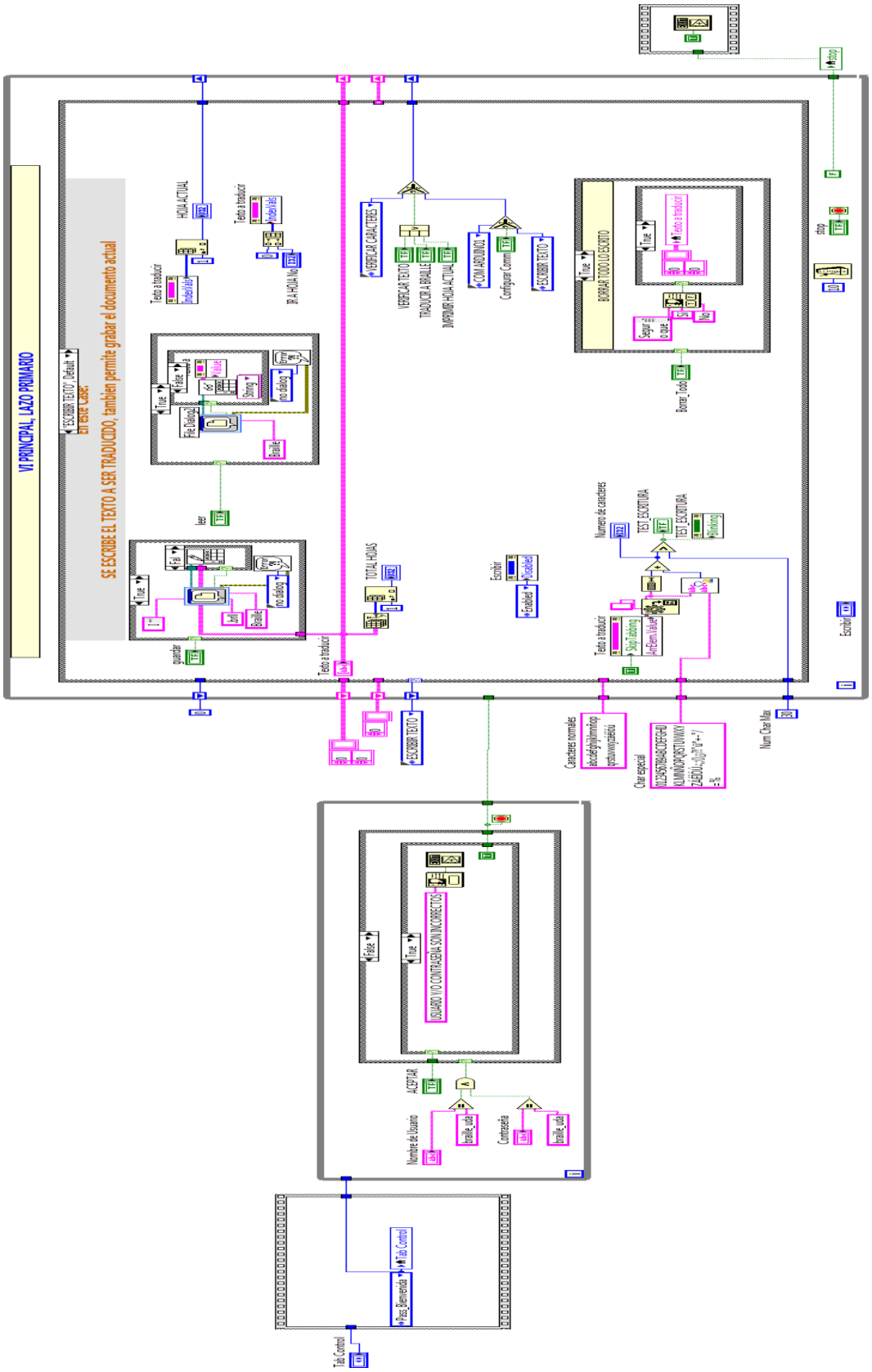
ANEXOS

ANEXO 1: Medidas de la maqueta de la impresora





ANEXO 2: Diagrama de bloques total en LabVIEW



ANEXO 3: Programación en Arduino

```
#include<DELduino.h>
PinOutdire1(2);
PinOutstep1(3);
PinOutenab1(4);
PinOutdire2(5);
PinOutstep2(6);
PinOutenab2(7);
PinOutbobin(52);
PinOut L_R(26);
PinOut L_G(24);
PinOut L_B(22);
int puls1;
int puls2;
int puls3;
intvec = 0;
int pun = 0;
chardato;
intpos = 0;
char caracter;
charletras[61] =
"0000000000000000000000000000000000000000000000000000000000000000";
intletra = 0;
boolean ban = false;
void setup()
{
Serial.begin(9600);
  L_R.Out0();
  L_G.Out0();
  L_B.Out0();
enab1.Out0();
enab2.Out0();
bobin.Out0();
```

```
dire1.Out0();
enab1.Out1();
while(1)
  {
    L_R.Out1();
    puls1 = analogRead(A1);
if(puls1 < 500){break;}
step1.Out1(); delayMicroseconds(500); step1.Out0(); delayMicroseconds(300);
  }
dire1.Out1();
delay(100);
for(vec = 0; vec< 330; vec++)
  {
step1.Out1(); delayMicroseconds(500); step1.Out0(); delayMicroseconds(500);
  }
enab1.Out0();
  L_R.Out0();
}
void loop()
{
while(1)
  {
  puls1 = analogRead(A3);
if(puls1 < 700)
  {
dire2.Out1();
enab2.Out1();
while(1)
   {
    puls1 = analogRead(A2);
if(puls1 < 600)
    {
for(vec = 0; vec< 50; vec++)
     {
```

```
step2.Out1(); delayMicroseconds(500); step2.Out0(); delayMicroseconds(500);
    }
enab2.Out0();
delay(500);
break;
   }
step2.Out1(); delayMicroseconds(500);
step2.Out0(); delayMicroseconds(500);
  }
 }
 puls1 = analogRead(A2);
if(puls1 < 600)
 {
break;}
 }
while(1)
 {
if (Serial.available())
   {
dato = Serial.read();
if(dato == 'F')
    {
ban = false;
imprime();
    }
if(ban == true)
    {
letras[letra] = dato; letra++;
}
if(dato == 'I')  {
ban = true;
}
if(dato == 'E')
    {
```

```
enab2.Out1();
while (1)
      {
        puls1 = analogRead(A3);
if(puls1 > 700){break;}
step2.Out1(); delayMicroseconds(500); step2.Out0(); delayMicroseconds(500);
      }
for(vec = 0; vec< 100; vec++)
      {
step2.Out1(); delayMicroseconds(500); step2.Out0(); delayMicroseconds(500);
      }
enab2.Out0();
      }
   }
  }
}
voidimprime()
{
letra = letra / 2;
  L_B.Out0();
dire1.Out1();  enab1.Out1();
for(intfila = 0; fila< 3; fila++)
{
for(int y = 0; y < letra; y++)
  {
   L_G.Out1();
caracter = letras[pos];
if(bitRead(caracter, fila) == 1)
   {
bobin.Out1(); delay(100); bobin.Out0(); delay(300);
   }
for(vec = 0; vec< 58; vec++)
   {
step1.Out1(); delayMicroseconds(500);  step1.Out0(); delayMicroseconds(300);
```

```
  }
pos++;
if(bitRead(caracter, fila) == 1)
  {
bobin.Out1(); delay(100);  bobin.Out0(); delay(300);}
for(vec = 0; vec< 90; vec++)//el 90 es la separación entre letra y letra
{
step1.Out1();  delayMicroseconds(500);  step1.Out0();  delayMicroseconds(300);
  }
pos++;
 }
pos=0;
L_G.Out0();  dire1.Out0();
while(1)
 {
   L_R.Out1();
   puls1 = analogRead(A1);
if(puls1 < 500)
   {
break;
   }
step1.Out1();  delayMicroseconds(400);  step1.Out0();  delayMicroseconds(300);
 }
dire1.Out1();
delay(100);
for(vec = 0; vec< 330; vec++)
 {
step1.Out1();  delayMicroseconds(500);  step1.Out0();  delayMicroseconds(500);
 }
 L_R.Out0();
dire2.Out1();
enab2.Out1();
for(vec = 0; vec< 10; vec++)
 {
```

```
step2.Out1(); delayMicroseconds(400);  step2.Out0();  delayMicroseconds(300);
  }
enab2.Out0();  delay(500);
}
enab1.Out0();
letra = 0;
L_B.Out1();
dire2.Out1();  enab2.Out1();
for(vec = 0; vec< 25; vec++)
  {
step2.Out1(); delayMicroseconds(400);  step2.Out0();  delayMicroseconds(300);
  }
enab2.Out0();
Serial.println("X");
}
```

ANEXO 4: Video demostrativo de funcionamiento.