Universidad Tecnológica Nacional

Facultad Regional La Rioja

Carrera de Ingeniería Electrónica

Trabajo Final de grado:

Adaptación de Impresora Convencional a Impresora Braille - Modelo Funcional

---

Autor:

Agüero Hemmes, Matias Gabriel

La Rioja, Mayo de 2016

Cátedra de Proyecto Final

Prof. Ing. Walter J. D. Cova

Trabajo Final de Grado: ADAPTACION DE IMPRESORA CONVENCIONAL
A IMPRESORA BRAILLE – MODELO FUNCIONAL

Autor: Agüero Hemmes, Matias Gabriel
DNI: 33.394.375
Legajo: 30-3133

Docente Tutor: Ing. Esteban Antonio Sarroca

Fecha de Examen: ____-____-_____

….. EL EXAMEN DE DEFENSA DEL PRESENTE TRABAJO FINAL DE GRADO HA MERECIDO LA CALIFICACIÓN DE ..... PUNTOS CONCEPTO: .....

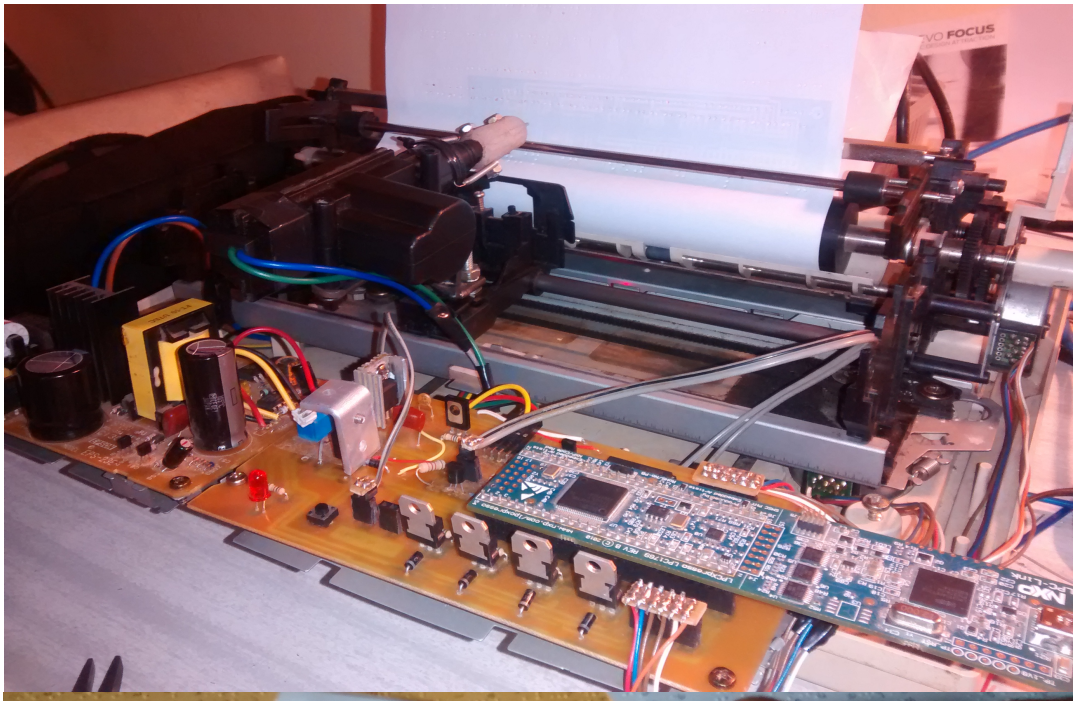
Tribunal Examinador:		
Presidente	Vocal 1°	Vocal 2°

Trabajo Final de Grado

Adaptación de Impresora Convencional a Impresora Braille - Modelo Funcional

Alumno: Agüero Hemmes, Matias Gabriel. Legajo: 30-3133

Tutor: Ing. Esteban Antonio Sarroca



Agradecimientos

A mi familia, y en especial a mi abuela Chicha, por todo su apoyo incondicional durante cada etapa de mi vida, hasta la culminación de mis estudios universitarios; y a mi novia, quien me acompañó durante toda mi carrera de formación como ingeniero.

Agradecimientos especiales

- Tutor de tesis: Ing. Esteban Sarroca, quien fue guía de este Proyecto Final, aportando la impresora de base y otros elementos imprescindibles para concretarlo.
- Lic. Patricia Castellanos, especialista en ciegos, por su aporte de material e información inicial para comenzar este proyecto.
- Dirección de Discapacidad de la Provincia, por su aporte de información y acceso a la impresora braille comercial.
- Compañeros de la fábrica Ritex: Jorge Luna, Sr. Alveró, Ing. Jorge López, por su ayuda para conseguir los Solenoides y entender su funcionamiento.
  Sr. Fabricio Silva, por su ayuda para conseguir repuestos de la impresora de base.
- Sr. José Codosea, por su ayuda con la fabricación de piezas especiales.
- Sr. Jorge Baldissone, quien se encargó de realizar y adaptar piezas especiales a la impresora base.
- Sr. Daniel Nieto, quien realizó la base de madera para la presentación de la impresora.
- **Leandro Paz**, joven ciego que evaluó la calidad de las impresiones realizadas, y su mamá, Roxana, por su colaboración y aporte de material.

Índice de contenido

Introducción

Una impresora convencional permite producir una copia permanente de textos o gráficos de documentos almacenados en formato electrónico, imprimiendolos en medios físicos, como papel, para tenerlos disponibles como respaldo de la información almacenada en medios electrónicos. Como así también, para cubrir requerimientos formales, legales y/o burocráticos. Las impresoras para el hogar suelen diseñarse para realizar trabajos de poco volumen, que no requieran virtualmente un tiempo de configuración para conseguir una copia de un determinado documento.

Las personas ciegas se ven privadas de la lectura y el acceso a información en documentos impresos de manera convencional. Es gracias al sistema **braille** que los disminuidos visuales pueden acceder vía tacto a los que sus ojos le niegan. El sistema braille utiliza una serie de puntos en relieve sobre el soporte (papel) que se interpretan como letras del alfabeto y es utilizado por las personas ciegas que aprendieron el método.

Una **impresora braille** es prácticamente igual a una impresora de tinta o láser. La principal diferencia está en el mecanismo de impresión, que utiliza **percutores** para realizar los puntos en relieve sobre el papel.

El problema se presenta al querer adquirir una de estas impresoras: el costo ronda los U$S 5.700 (cinco mil setecientos dólares estadounidenses); sólo se fabrica en países del extranjero (como Suecia); y en nuestro país solo se adquieren mediante representantes exclusivos (como tiflonexos). A estas dificultades hay que agregarle el requerimiento de un servicio de mantenimiento especializado y de alto costo económico.

El objetivo del presente proyecto es la construcción de un modelo funcional de impresora Braille, a partir de la modificación de una impresora convencional reciclada tipo “matriz de punto”, con componentes perfectamente accesibles, tanto por su disponibilidad, como por su precio.
Además se presenta con una conexión USB y un software de interfaz gráfica de usuario (I.U.) amigable, simple e intuitivo para la fácil y rápida utilización por parte del usuario final.

El proyecto comienza, en su primera parte, describiendo lo que es el lenguaje braille y una impresora braille; el capitulo 2, reseña proyectos similares al abordado en este informe.

En la segunda parte se plantean los objetivos (capitulo 3); y se analizan una impresora braille comercial, y la impresora base a modificar (capitulo 4).

La tercera parte describe el desarrollo del proyecto: comenzando por el capitulo 5, donde se describe el método de adaptación de un sistema percutor, en reemplazo del cabezal original. El capitulo 6 describe el desarrollo de la placa de control, gobernada por un microcontrolador para manejo de los motores paso a paso y percutor antes mencionado; mientras que en los capítulos 7 y 8 se detallan los métodos utilizados para programar el microcontrolador y la interfaz de usuario, respectivamente.
El desarrollo del proyecto finaliza con los capítulos 9, donde se realizan pruebas de desempeño y mediciones varias; y 10, indicando los costos monetarios y tiempos invertidos.

El proyecto puede generar una oportunidad tanto a las familias con miembros ciegos, como a las instituciones que trabajan con personas con discapacidad visual y no disponen de

ayuda financiera; brindándoles la posibilidad de generar impresiones Braille a un costo reducido.
Además el proyecto ayudará a reducir la denominada “chatarra electrónica”, reciclando impresoras en desuso.

Parte I:
Conceptos y Antecedentes

1. Conceptos

1.1 Sistema Braille

El braille es un sistema de lectura y escritura táctil pensado para personas ciegas. Se conoce también como cecografía. Fue ideado por el francés Louis Braille a mediados del siglo XIX, quien se quedó ciego debido a un accidente durante su niñez mientras jugaba en el taller de su padre. Cuando tenía 13 años, el director de la escuela de ciegos y sordos de París –donde estudiaba el joven Braille– le pidió que probara un sistema de lecto-escritura táctil inventado por un militar llamado Charles Barbier para transmitir órdenes a puestos de avanzada sin tener necesidad de delatar la posición durante las noches. Louis Braille descubrió al cabo de un tiempo que el sistema era válido y lo reinventó utilizando un sistema de ocho puntos. Al cabo de unos años lo simplificó dejandolo en el sistema universalmente conocido y adoptado de 6 puntos.

El sistema braille no es un idioma, sino un alfabeto. Con el braille pueden representarse las letras, los signos de puntuación, los números, la grafía científica, los símbolos matemáticos, la música, etc. El braille suele consistir en celdas de seis puntos en relieve, organizados como una matriz de tres filas por dos columnas, que convencionalmente se numeran de arriba a abajo y de izquierda a derecha, tal y como se muestra en la siguiente figura:

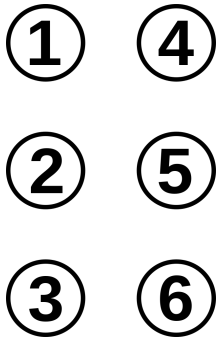


Fig.1. 1: Celda braille

La presencia o ausencia de puntos permite la codificación de los símbolos. Mediante estos seis puntos se obtienen 64 combinaciones diferentes. La presencia o ausencia de punto en cada posición determina de qué letra se trata. Puesto que estas 64 combinaciones resultan claramente insuficientes, se utilizan signos diferenciadores especiales que, antepuestos a una combinación de puntos, convierten una letra en mayúscula, bastardilla, número o nota musical. En el braille español, los códigos de las letras minúsculas, la mayoría de los signos de puntuación, algunos caracteres especiales y algunas palabras se codifican directamente con una celda, pero las mayúsculas y números son representados además con otro símbolo como prefijo.

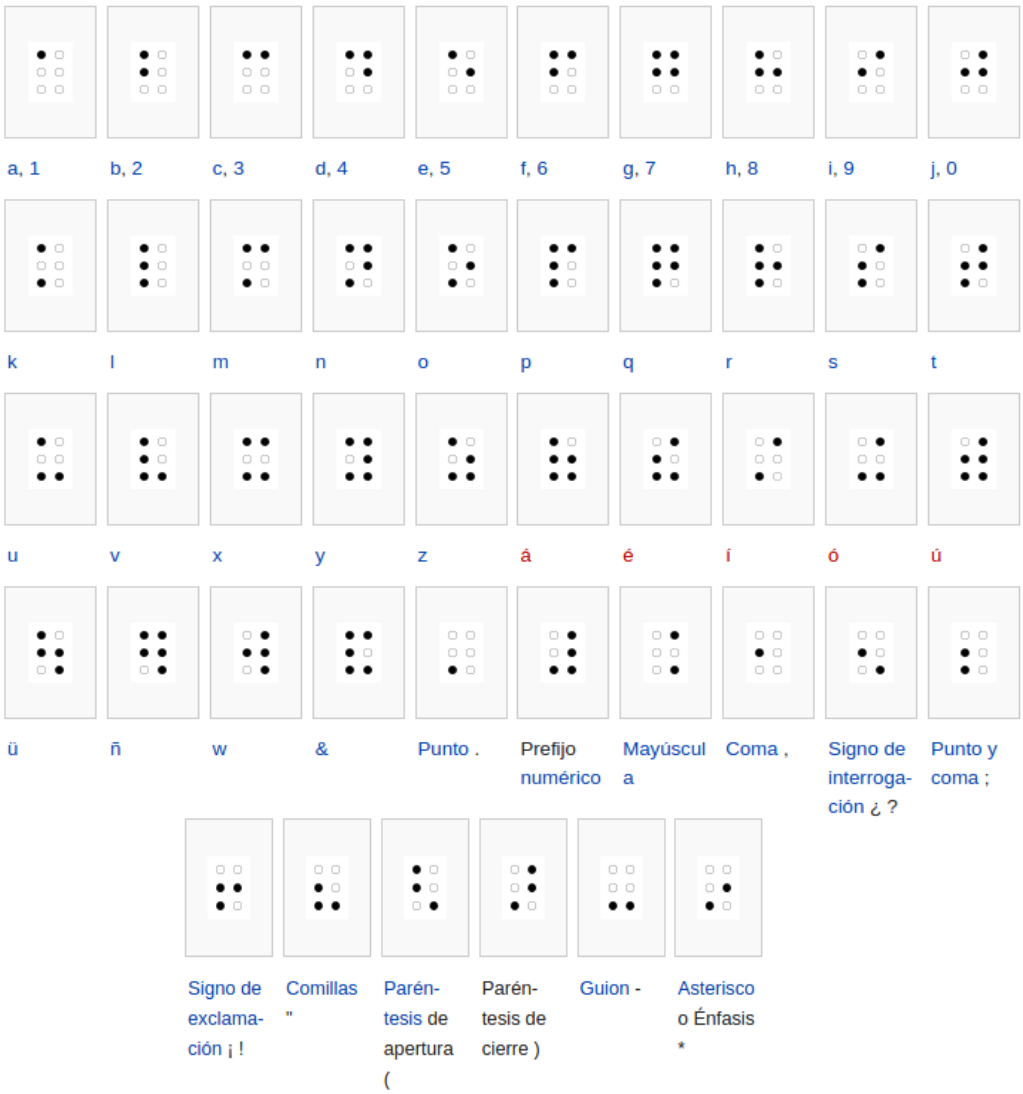


Fig.1. 2: Caracteres braille

Con la introducción de la informática, el braille se amplió a un código de ocho puntos, de tal manera que una letra individual puede ser codificada con una sola celda, pudiendo representar una celda cualquier carácter ASCII. Las 256 combinaciones posibles de los ocho puntos están codificadas según el estándar Unicode.

Alfabeto braille español en codificación Unicode

El estándar Unicode codifica patrones de braille de 8 puntos de acuerdo a su apariencia binaria, en vez de seguir un orden alfabético. Unicode define el bloque de caracteres para patrones de braille en el rango hexadecimal entre 2800 y 28FF.
Se detallan a continuación algunos de los 256 patrones codificados, junto con su significado en el braille español. Nótese, de nuevo, que en Unicode no se hace referencia alguna al significado de los patrones codificados.

Tabla 1.1: Alfabeto braille codificado en Unicode

Braille	significado	-	Braille	significado	-	Braille	significado	
⠁	a, 1		⠞	t		⠷	á	
⠃	b, 2		⠥	u		⠮	é	
⠉	c, 3		⠧	v		⠌	í	
⠙	d, 4		⠺	w		⠬	ó	
⠑	e, 5		⠭	x		⠾	ú	
⠋	f, 6		⠽	y		⠳	ü	
⠛	g, 7		⠵	z				
⠓	h, 8		⠠	Signo de mayúsculas				
⠊	i, 9		⠼	Signo de número				
⠚	j, 0		⠲	Punto (.) *(punto 3)*				
⠅	k		⠂	Coma (,) *(punto 2)*				
⠇	l							
			⠦	Signos de interrogación (¿?)				
⠍	m		⠒	Punto y coma (;)				
⠝	n		⠖	Signos de exclamación (¡ !)				
⠻	ñ		⠐	Dos puntos (:)				
⠕	o		⠶	Comillas (de cualquier tipo)				
⠏	p		⠣	Abrir paréntesis "("				
⠟	q		⠜	Cerrar paréntesis ")"				
⠗	r		⠤	Guión (-)				
⠎	s			Espacio *(ningún punto)*				

Escritura Braille

Existen diversos métodos de Transcripción Braille, conocidos como "Grado 1", "Grado 2" y "Grado 3". El braille de Grado 1 es el sistema de transcripción más empleado y el método

único y oficial para la publicación en España, según el acuerdo adoptado por la Comisión Braille Española. Este sistema de transcripción **sustituye** las notaciones en tinta del original por las correspondientes en braille. Los sistemas de transcripción correspondientes a los Grados 2 y 3 son conocidos como estenotipia. Su principio rector es el de economizar caracteres para ahorrar espacio, puesto que los caracteres en braille **no se pueden alterar de tamaño** –como sucede en el caso de la tinta–.
Existen múltiples extensiones del braille para incluir letras adicionales con diacríticos, como *Ç*, *Ô*, *Å*.

Dimensiones de los caracteres braille:

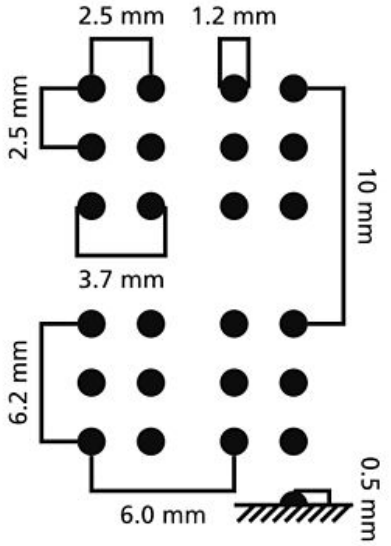


Fig.1. 3: Dimensiones de los caracteres braille

En todo el mundo existen centros de producción de libros y documentos accesibles para personas ciegas y deficientes visuales graves. Entre estos centros, se destacan el NLS de la Biblioteca del Congreso en Estados Unidos y la Red de Producción del Servicio Bibliográfico de la ONCE (Organización Nacional de Ciegos de España).

Accesibilidad

Un ejemplo de la accesibilidad del braille se encuentra en los billetes canadienses en curso, que constan de una serie de puntos que indican su denominación y pueden ser fácilmente identificados por gente con problemas de vista. El Banco Central del Paraguay, desde el año 2009 puso en circulación un billete de 2000 guaraníes con el sistema braille para los invidentes.

En España, a partir de las Elecciones Generales y Autonómicas andaluzas de marzo de 2008, es posible utilizar este sistema para emitir el voto de forma autónoma y anónima, lo cual supone un importante avance social para la integración de los ciegos y deficientes visuales severos.

A pesar de que el braille fue ideado como el principal sistema de lectura y escritura para personas ciegas, en el Reino Unido se estima que, de entre dos millones de personas con problemas de vista, sólo entre 15.000 y 20.000 utilizan el sistema braille. La gente joven prefiere el texto electrónico, ya que es portátil y les permite comunicarse con sus amigos.

Actualmente hay un debate abierto sobre cómo hacer más atractivo el braille, y cómo conseguir más profesores que sean capaces de enseñarlo.[1]

1.2. Impresora Braille

Una impresora braille es un dispositivo electrónico que permite imprimir textos e imágenes simples empleando puntos en relieve en papel.

Mecanismo de funcionamiento

Una impresora braille es prácticamente igual a una impresora de tinta, láser o térmica. La principal diferencia está en el mecanismo de impresión. Si en los formatos anteriores se realizaba a través de chorros de tinta, toner o dispositivos térmicos, en este caso se utilizan percutores.

Los percutores realizan la misma función que un punzón para escribir braille. Éstos son lanzados contra el papel con la fuerza suficiente para que lo marquen, sobresaliendo los puntos por la cara contraria a la percutida. La fuerza que se aplica al papel debe ser suficiente para que la persona pueda sentir el punto, pero no puede ser excesiva para evitar perforar el papel o el desgaste prematuro de los puntos.

Formatos

Este tipo de impresoras permiten imprimir braille convencional de 6 puntos, y algunas de ellas también permiten el de 8 puntos.

Al igual que en las impresoras de tinta se pueden realizar dibujos simples con los caracteres, en las impresoras braille podemos emplear los puntos para realizar dibujos en el papel de manera que la persona ciega pueda sentirlos al tocarlos.

El aspecto de una impresora braille no difiere demasiado de una impresora convencional. Suelen disponer de una bandeja de entrada y otra de salida. Su tamaño puede ser algo mayor. Esto es debido principalmente a las dimensiones del papel que se emplea

Hoy en día existen en el mercado impresoras portables de peso y dimensiones reducidas, de manera que es posible llevarlas a clase, reuniones de trabajo, etc.

Hay impresoras de papel continuo y de papel previamente cortado en diversos formatos.

Muchas impresoras permiten marcar puntos braille por las dos caras del papel. Para conseguirlo se ajustan los puntos de manera que no coincidan en ambas caras. Debido a la naturaleza del braille, con unas medidas establecidas y márgenes entre los puntos, no es difícil diseñar el mecanismo para conseguirlo. [2]

---

1. Los precedentes conceptos, tablas y figuras han sido extractados de la fuente: https://es.wikipedia.org/?title=Braille_(lectura)

2. Los precedentes conceptos han sido extractados de la fuente: https://es.wikipedia.org/wiki/Impresora_braille

Modelo de ejemplo: Braille Basic-D V4

Fig.1. 4: Impresora Braille comercial: Braille Basic-D V4

A modo de ejemplo puede mencionarse el modelo de impresora comercial “Braille Basic-D V4”: una impresora de tamaño compacto (similar a las chorro de tinta), “económica” y de buen rendimiento.
Además es la impresora de este tipo más vendida en el mundo.

Origen: Suecia

algunas características de la misma son:[1]

- Velocidad de impresión 300 páginas por hora (100 CPS)
- alimentación: continua
- Impresión doble faz (interpunto) y simple faz
- Capacidad de impresión de gráficos, con una distancia entre puntos de hasta 1.6 milímetros
- nivel de ruido: 76 dB
- peso neto: 9,5 kg
- Panel de comandos en tinta y Braille, con guía en audio en español.
- Impresión en hoja de formulario contínuo
- Conexiones: puerto serie, puerto de red, USB
- **Precio: U$S 5700**

1. Fuentes: http://www.tiflonexos.com.ar/basicV4.htm
http://www.tecno-ayudas.com.ar/productos_impresora_braille_basic-d.html

2. Antecedentes

La idea de poder brindar una solución a los altos costos que presenta el adquirir una impresora Braille comercial no es nueva: muchos son los casos de personas que idearon diversas formas de lograr la funcionalidad de una impresora braille a partir de modelos factibles y sobre todo, económicos. Utilizando en general impresoras recicladas y elementos disponibles y estándares para su adaptación. En algunos casos como proyectos de tesis de carreras o postgrados, como es este caso, en otros, sólo como pasatiempos.
Se citan a continuación los proyectos más completos que se encontraron publicados en internet con una breve descripción de lo realizado; luego se hace mención de algunos ejemplos más. Se brinda además la dirección web de origen por si el lector desea profundizar en alguno de los ejemplos citados.

2.1. *Desarrollo de impresora braille con componentes reciclados*

Un grupo de estudiantes de la Facultad de Informática de la Universidad Nacional de La Plata logró desarrollar una impresora Braille utilizando repuestos y componentes de impresoras comerciales convencionales en desuso. El novedoso desarrollo permite, a un costo muy bajo, mejorar las condiciones de acceso de las personas con discapacidad a las nuevas tecnologías.
La impresora de código Braille en papel desarrollada en la UNLP consta de un sistema electromecánico que se anexa a las impresoras comerciales, tanto matriciales como chorro de tinta. De esta manera, la gran ventaja es que el sistema permite la reutilización de los motores y la mayoría de las partes de las impresoras comunes, simplemente descartando la plaqueta original de la impresora por una nueva de fabricación casera. Esta nueva plaqueta es controlada por un programa basado en Software Libre, también desarrollado en la facultad de informática.[1]

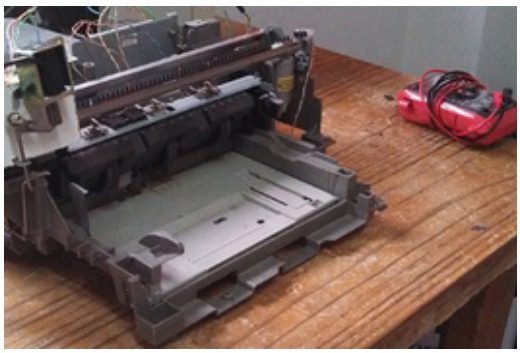


Fig.2. 1: impresora braille con componentes reciclados - UNLP

1. *fuente: http://www.unlp.edu.ar/articulo/2014/12/1/desarrollan_impresora_braille_de_bajo_costo_con_componentes_reciclados*

2.2. Diseño y construcción de una impresora Braille

Autores: Julio Camino, Liliana Ligña, Nelson Sotomayor, Escuela Politécnica Nacional (EPN), Quito – Ecuador

En el presente proyecto se tiene el diseño y ensamblaje de una Impresora Braille que tiene por objeto a través de un punzón plasmar en el papel los puntos que conforman los caracteres en Braille que son enviados desde una PC. Para la impresión de los caracteres se ejecuta el programa en una PC donde se explora y carga el documento, se traduce el texto a código Braille y se muestra en una pantalla contigua a la del texto original. Por el puerto USB se envían los caracteres al módulo en donde son comparados con tablas de correspondencia para la impresión de los puntos.[1]

2.3. Otros proyectos similares:

- **Problema de ingeniería: impresora braille**
  fuente: http://www.frsn.utn.edu.ar/tecnicas3/problemas/Impresora%20Braille.pdf

- **UN ESTUDIANTE MEXICANO DISEÑA UNA IMPRESORA EN SISTEMA BRAILLE**

fuente:http://noticias.universia.es/ciencia-nn-tt/noticia/2004/03/06/614300/estudiante-mexicano-disena-impresora-sistema-braille.html

- **Desarrollo de un Prototipo de Impresora Braille de Bajo Coste como Apoyo a la Discapacidad Visual**

fuente: http://rcs.cic.ipn.mx/2014_76/Desarrollo%20de%20un%20Prototipo%20de%20Impresora%20Braille%20de%20Bajo%20Coste%20como%20Apoyo%20a%20la%20Discapacidad%20Visual.html

1. *fuente:http://bibdigital.epn.edu.ec/bitstream/15000/4891/1/Dise%C3%B1o%20y%20Construcci%C3%B3n%20de%20una%20Impresora%20Braille.pdf*

Parte II:
Objetivos y análisis de funcionamiento

3. Objetivo del proyecto

El objetivo del proyecto es la construcción de un modelo funcional de una impresora Braille, a partir de la modificación de una impresora convencional; la cual debe ser de sencilla operación aplicando conectividad USB (interfaz de comunicación estándar en las PC actuales), el desarrollo de drivers (controladores); y un software de apoyo sencillo y amigable.

Para lo cual los siguientes objetivos específicos se plantearon:

1. Adquirir conocimiento y experiencia en la programación y conexionado de microcontroladores.
2. Adquirir conocimiento y experiencia en el uso de dispositivos de interfaces de comunicación.
3. Adquirir conocimientos y experiencia para desarrollar software de alto nivel destinado a la interfaz de usuario final.
4. Adquirir mayores conocimientos sobre los sistemas de impresión convencionales, control de motores y los requerimientos del sistema Braille.
5. Lograr el desarrollo de un modelo funcional de impresora Braille de costo reducido.
6. Formular los requerimientos de ulteriores desarrollos para convertir el modelo funcional en un prototipo para producción.

4. Análisis de Funcionamiento

4.1. Impresora comercial “Index Braille Basic-D V4”

Se realizó una visita a la Dirección de Discapacidad ubicada en Centro Administrativo Provincial (C.A.P.) Av. Ortiz de Ocampo 1700 Ala Sur, Bloque 5, Planta Baja, en la ciudad de La Rioja. La misma cuenta con una división para personas ciegas y disminuidas visuales, donde cuentan con una impresora Braille, modelo “Index Braille Basic-D V4”.
Se solicitó permiso para tener acceso a la misma y observar su funcionamiento, con la ayuda de una persona encargada de su uso, y se realizó la impresión de algunos textos de prueba.

Software

El software resultó muy sencillo e intuitivo: cuenta con dos ventanas, una donde se presenta un editor de texto normal, identificada con un fondo blanco (Fig.4. 1); y otra donde se muestra el texto “filtrado” para la impresión a Braille, de fondo fucsia (Fig.4. 2). El filtrado elimina cualquier carácter que no sea imprimible en formato Braille, además agrega los signos especiales de mayúscula, número, etc. para ser impresos.

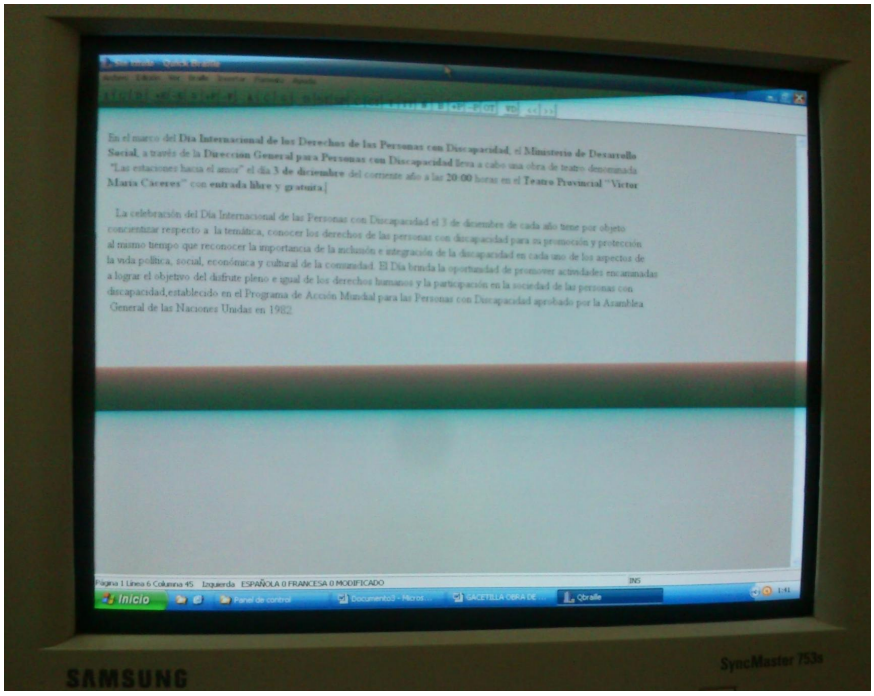


Fig.4. 1: Software - Editor de texto común

Una observación que se realizó en este punto fue la necesidad del personal de tener una “vista previa” de la impresión para conocer con anticipación la cantidad de hojas que se utilizarán. La importancia de este punto es que los caracteres Braille, como se mencionó antes, tienen un tamaño fijo que no se puede alterar (como se lo hace en un texto común) y ocupan relativamente mucho espacio en una hoja.

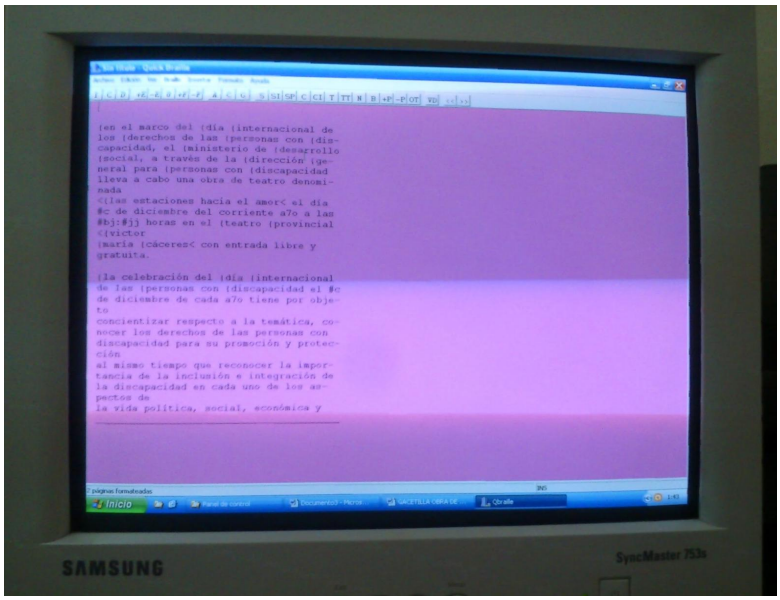


Fig.4. 2: Software - Editor de texto en braille

Adicionalmente se debe tener en cuenta que luego de una determinada cantidad de hojas impresas, el cabezal debe “enfriarse”, produciéndose una pausa durante la impresión.

Hardware

La forma y dimensiones son similares a una impresora convencional moderna. El sistema de impresión es el llamado “continuo”, similar al de las antiguas impresoras matriz de punto donde las hojas están unidas, debiéndose cortar luego de la impresión. Este tipo de sistema ayuda a mantener la hoja en su lugar por medio de dos guías perforadas en los bordes laterales, necesario por la alta vibración producida por el golpe de los percutores; lo que trae aparejado también mucho ruido.
La impresora es capaz de imprimir en sistema “interpunto”, esto es en ambas caras del papel, sin que los puntos se interfieran (en realidad a veces los hacen y rompen la hoja)

Las hojas utilizadas son especiales para la impresión de Braille: son de 142 gr., más gruesas que las comunes (de 75gr).

Presenta ayudas para la utilización por parte de personas ciegas: indicaciones en relieve sobre el frente para la manipulación de los comandos y un parlante donde se emiten mensajes cortos como “imprimiendo”, “fin de impresión” y “falta papel”.

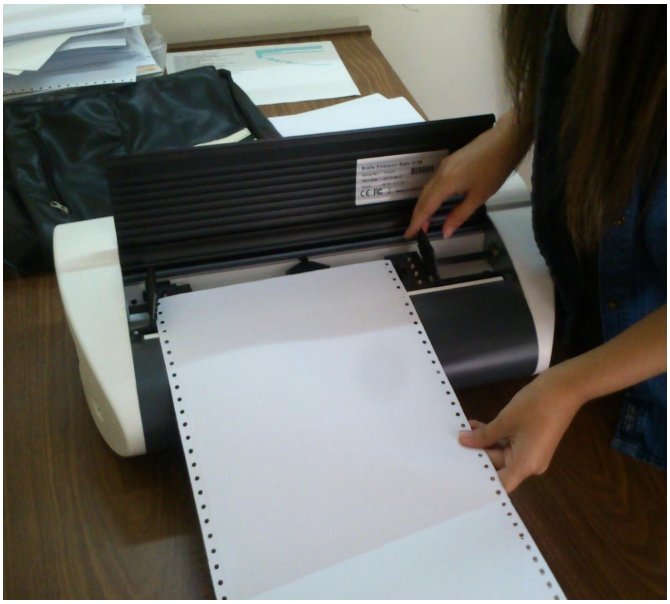


Fig.4. 3: Hardware - Colocación del papel en el sistema continuo

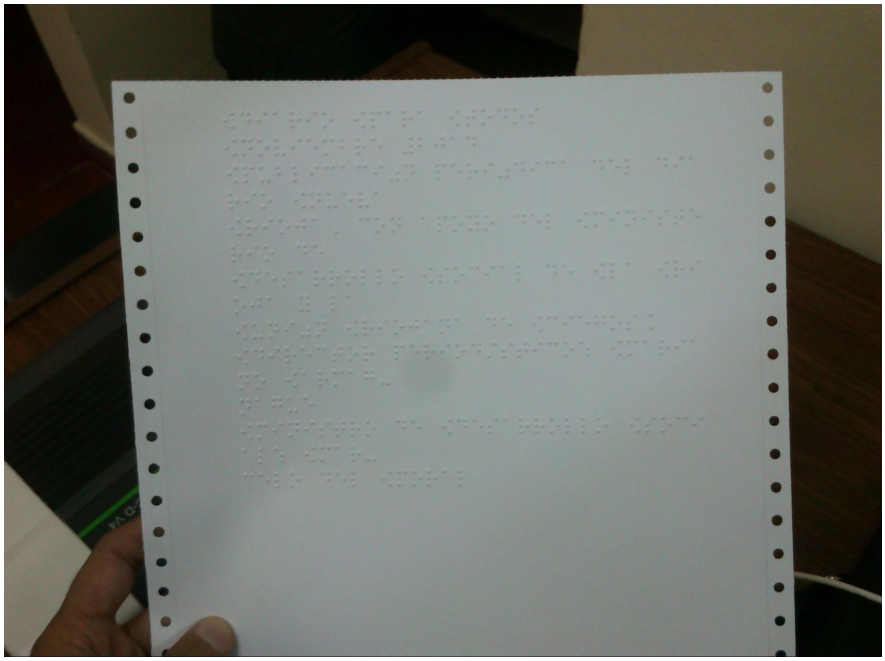


Fig.4. 4: hoja impresa en braille terminada

Conexión

La impresora se conecta por USB y figura en la lista de Hardware de la PC como otra impresora convencional más; pero si se le manda un texto en formato común (sea txt, doc, odt, etc) ésta no reacciona. Solo reconoce el formato especial de texto de extensión ".bra" generado por el soft de aplicación.

Fig.4. 5: Conexión - Puertos disponibles

4.2. Selección de la Impresora convencional de base

Para la selección de la impresora de base se tuvieron en cuenta los siguientes aspectos:

1. **Robustez:** Necesariamente la estructura debe ser resistente a vibraciones y golpes constantes por parte del percutor. Adicionalmente se consideró positiva la existencia de tractores de sistema continuo.
2. **Disponibilidad de Información**: Es fundamental disponer de información (hojas de datos) de los componentes básicos de manejo de impresión: Motores y sensores. Además es favorable la disponibilidad de manuales de usuario y técnicos. Para esto es ventajoso una impresora de una marca conocida y estándar.
3. **Disponibilidad de repuestos:** Los repuestos deben estar disponibles ya sea por internet o en tiendas de reparación de estos productos. Nuevamente es de gran ventaja contar con un producto popular.
4. **Buen estado general - funcionando:** La impresora seleccionada debe encontrarse lista para funcionar, para no perder tiempo en reparaciones y busqueda de repuestos antes de comenzar con el proyecto. Esto comprende tanto componentes electrónicos: Motores, sensores y **fuente de alimentación**; como componentes mecánicos: rodillos, carro, polea, y mecanismos en general.

El primer modelo elegido fue una impresora matriz de punto “**Citizen 200GX**” (Fig.4. 6). Se encontraba en perfecto estado en cuanto a la parte mecánica, era muy robusta y contaba con sistema continuo de impresión. El problema era la parte eléctrica: tenía dos motores paso a paso Sanyo de los cuales no se encontró información y la fuente de alimentación no funcionaba, además que se alimentaba con 110V. Se buscaron repuestos y fuentes alternativas sin éxito.

Se consiguió luego, con ayuda del tutor de tesis, otro modelo de impresora matriz de punto: “Epson LX-300” (Fig.4. 7). Ésta no se encontraba en las mejores condiciones mecánicas: estaba sucia y algunos mecanismos trabados.

En cuanto a la parte electrónica tenía dos motores paso a paso de los cuales uno tenía una bobina quemada. Debido a estos inconvenientes se llevó a reparar; donde se lubricaron y limpiaron los mecanismos, se consiguió un motor de repuesto y se compró otra fuente similar a la original para tener de repuesto. Además se comprobó la existencia de todo los repuestos en internet, por lo que se podría decir que cumple el requisito buscado de“popular”.

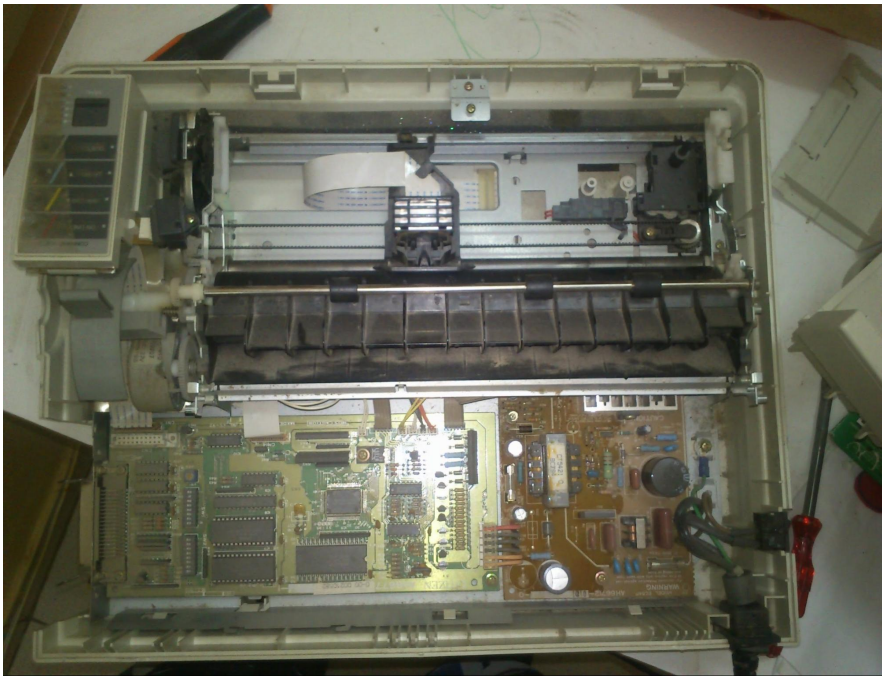


Fig.4. 6: Impresora de base - "Citizen 200 GX"

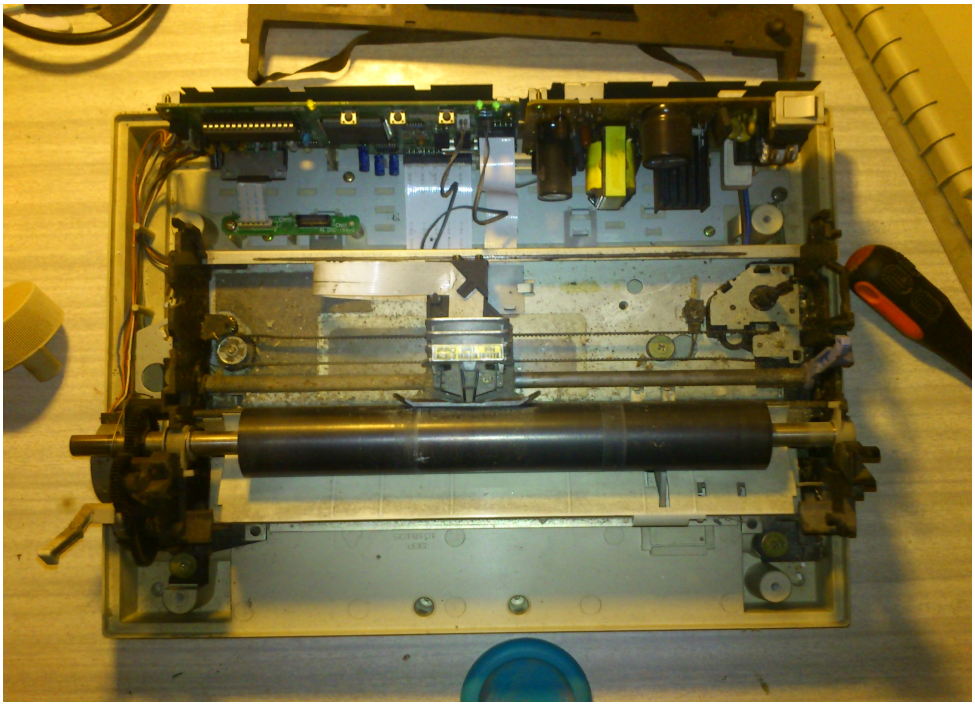


Fig.4. 7: Impresora base - "Epson LX-300"

Parte III:
Desarrollo: Materiales y Métodos

5. Percutor

El desarrollo del proyecto se comenzó con la modificación más importante: cambiar el cabezal de impresión por un percutor.

5.1. Punzón y Plantilla braille

El sistema que utilizan las personas para escribir “a mano” el braille consiste en un punzón y una plantilla especial. La hoja, que debe poseer una densidad de 120gr o mayor, se coloca en la plantilla, donde queda sujetada firmemente; y luego se realiza presión con el punzón hasta que el punto queda marcado en la hoja.

Fig.5. 1: Punzón para escribir braille

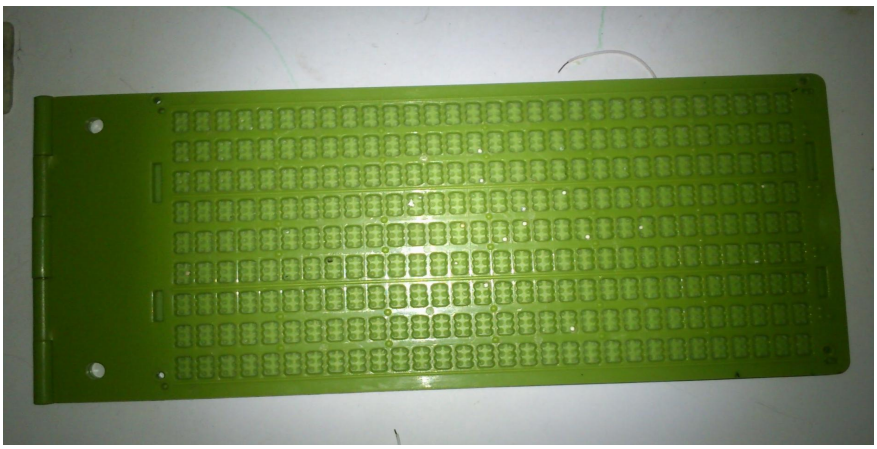


Fig.5. 2: Plantilla para escribir braille

Una cuestión a tener en cuenta para la programación posterior, es que quien escribe debe hacerlo “al revés”, ya que los puntos se escriben (marcan) de un lado, pero se leen del lado opuesto (puntos en relieve).

La idea básica fue desarrollar un dispositivo accionado eléctricamente que realice este mismo proceso (presionar el punzón contra la plantilla) de manera automática.

5.2. Actuador solenoide

La primera prueba se realizó con un actuador solenoide.

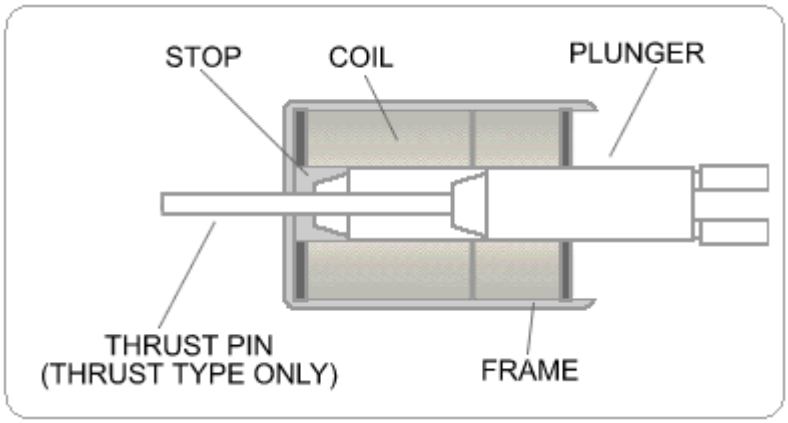


Fig.5. 3: Componentes de un solenoide

Un actuador solenoide (o simplemente solenoide) se compone de un marco (frame), el cual sostiene una bobina de alambre conductor (coil). El centro de la bobina es atravesado por un émbolo (plunger), que puede moverse hacia adelante y atrás unos centímetros, dependiendo el diseño, y un tope (stop) para detener el avance del émbolo. Unido al émbolo y sobresaliendo del marco se encuentra un pin, que es utilizado para transmitir el movimiento del émbolo y realizar el trabajo necesario de empuje o de tracción.

Principio de funcionamiento

Un solenoide es cualquier dispositivo físico capaz de crear un campo magnético sumamente uniforme e intenso en su interior, y muy débil en el exterior.

El campo magnético es un campo angular con forma circular, cuyas líneas encierran la corriente. La dirección del campo en un punto es tangencial al círculo que encierra la corriente. El campo magnético generado dentro del solenoide puede verse en la Fig.5. 4

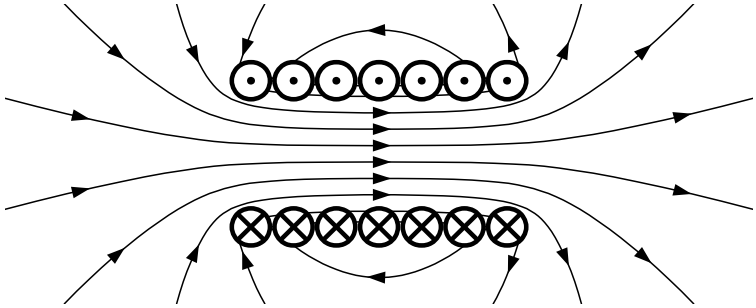


Fig.5. 4: Lineas del campo magnético generado dentro de un solenoide

Debido a que el embolo actúa como núcleo del solenoide y está fabricado de un material no ferromagnético, se produce la magnetización del mismo al aparecer el campo magnético; quedando el campo total definido como:

$$B = \mu_0 H + \mu_0 M$$

Siendo el primer termino la **Intensidad del campo magnético H**, el cual mide el campo magnético producido por la circulación de corriente real existente; y el segundo termino, la **magnetización M**, que es el aporte al campo total del material usado como núcleo.[1]

El actuador solenoide conforma un circuito magnético, como el mostrado en la Fig.5. 5:

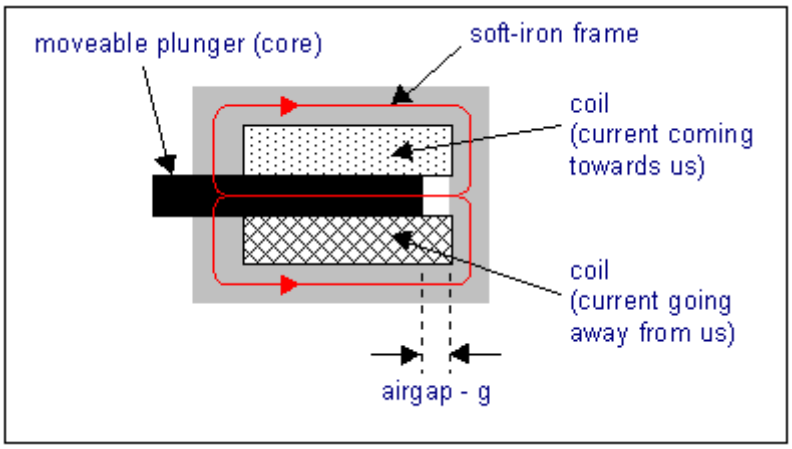


Fig.5. 5: Circuito magnético de un actuador solenoide

La bobina crea una FMM que impulsa el flujo Φ (que se muestra en rojo en el diagrama) a través del émbolo, alrededor del marco del solenoide, y luego a través del entrehierro (airgap) y de nuevo en el émbolo. La **reluctancia** de este circuito es en su mayoría compuesto por el entrehierro (ya que la reluctancia que presenta el marco y el embolo es muy pequeña).
Cuando el émbolo está hacia afuera, como se muestra en el diagrama, la reluctancia es bastante alta. Cuando se aplica corriente a la bobina, el émbolo se mueve a la derecha, y la reluctancia disminuye. Este es un ejemplo de lo que sucede con las fuerzas en los sistemas magnéticos: que siempre actúan para reducir la resistencia (reluctancia), o aumentar la inductancia. Eventualmente, el émbolo chocará con el marco en el lado derecho, y el entrehierro será cero, y la reluctancia estará en su mínimo.[2]

En el primer instante, cuando la reluctancia del circuito es alta, el flujo magnético en el circuito es relativamente bajo. Cuando se desplaza el embolo y disminuye el entrehierro, baja la reluctancia y el flujo magnético aumenta.
La ley de Lenz indica que al aumentar el flujo en la misma dirección que el producido por la bobina, aparecerá una **fuerza contra electro-motriz**, que se opondrá a dicho aumento. Esta contra fem produce una corriente de sentido opuesto (al que genera la fmm inicial) en la bobina, lo que se traduce en una disminución de la corriente total (Fig.5. 7).

---

1. *fuentes: http://www.sabelotodo.org/fisica/propiedadesmagneticas.html;*

Hugh D. Young, Roger A. Freedman – FISICA UNIVERSITARIA VOL. 2 - 12º Ed. (Addison-Wesley, año 2008)

2. *fuente: http://homepages.which.net/~paul.hills/Solenoids/SolenoidsBody.html*

Factores intervinientes

El principal factor a tenerse en cuenta para convertir el solenoide en el percutor buscado es que la fuerza de empuje sea suficiente para producir el punto en la hoja.

La fuerza de empuje depende directamente del flujo magnético producido, el cual a su vez depende de la corriente que circule por la bobina, la cual depende finalmente de la tensión aplicada.
Ahora, esta tensión no puede ser aumentada a voluntad ya que existen límites como:

a. **La aislación eléctrica del bobinado**: que limita directamente la tensión que podemos aplicar
b. **La disipación de calor:** depende del tamaño de la bobina, el material aislante y la sección del conductor. Limita la intensidad de corriente que puede atravesar la bobina.
c. **Saturación**: (ver Fig.5. 6)El flujo magnético generado aumenta linealmente con la corriente hasta un cierto punto “máximo” (b), a partir del cual, el seguir aumentando la intensidad de corriente no produce un aumento notable del flujo magnético, pero sí produce aumento de temperatura (a).

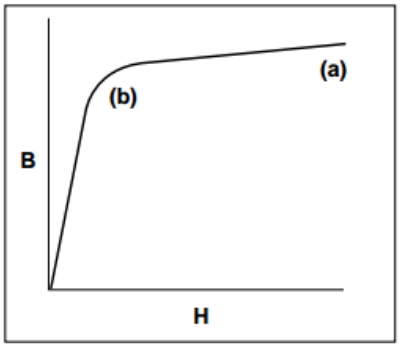


Fig.5. 6: Saturación del flujo magnético

d. **Ciclo de trabajo:** Un solenoide se diseña para trabajar con una tensión aplicada durante un tiempo determinado (ON time); y luego se debe quitar la tensión durante otro cierto tiempo (OFF time), para permitir la disipación de calor; la suma de estos tiempos conforman el ciclo de trabajo. Si se sobrepasa el ON time máximo, se produce el deterioro del solenoide por aumento de la temperatura. Se puede, en teoría, aumentar la tensión de trabajo, pero disminuyendo el ciclo proporcionalmente. Por ejemplo, si el diseño del solenoide se contempla para un ciclo del 100% (tensión aplicada constantemente) a una tensión de 12V; se podría trabajar con una tensión de 24V a un ciclo del 50%.

Se debe considerar que es necesario un tiempo ON mínimo, para que el émbolo llegue al tope. A continuación se muestra en un gráfico el consumo de corriente en función del tiempo (Fig.5. 7): el solenoide es energizado en el instante “0”, la corriente aumenta hasta el punto “a” sin producirse aun el movimiento del émbolo; con el movimiento del émbolo la curva se comporta como se muestra desde “a” hasta “b” (el émbolo llega al tope). Luego la curva asciende hasta el estado estable (dado por la ley de ohm, con la resistencia de la bobina).

La curva “c” muestra el caso donde el émbolo debe empujar una carga más pesada; mientras que la primera curva muestra el caso en el que no es posible mover la carga para el solenoide (fuerza insuficiente).[1]

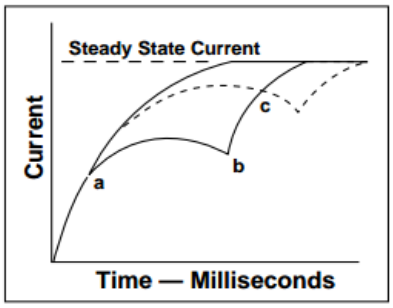


Fig.5. 7: Corriente en función del tiempo

Prueba del percutor

Se consultó, primeramente a un fabricante nacional por un modelo que satisfaga el requisito de fuerza necesaria. El modelo recomendado fue el siguiente:

Fig.5. 8: Solenoide de producción nacional

Tiene un diámetro de 48mm y una longitud de 52mm. Se alimenta con 24Vcc (20W); el émbolo recorre 7mm y está diseñado para un ciclo de trabajo del 30%. Ejerce una fuerza de 2,5 kg.
Precio: U$S 250,00.
Debido a su alto costo, se decidió realizar primero pruebas con solenoides disponibles, y en caso de obtener los resultados buscados, realizar la compra e incorporarlo en el diseño definitivo.

Las pruebas se realizaron con distintos solenoides facilitados por una fábrica, de máquinas en desuso, buscando incrementar gradualmente la fuerza sobre el punzón en cada caso:

a. Kendrion: 24V; 0,4A; 25%Ciclo de trabajo

1. *fuente: Technical information - saia-burgess Solenoids*

Fig.5. 9: Solenoide de prueba Kendrion

b. Rinder: 24V; 5% Ciclo de trabajo

Fig.5. 10: Solenoide de prueba Rinder

c. EKS: 24V; 100% Ciclo de Trabajo

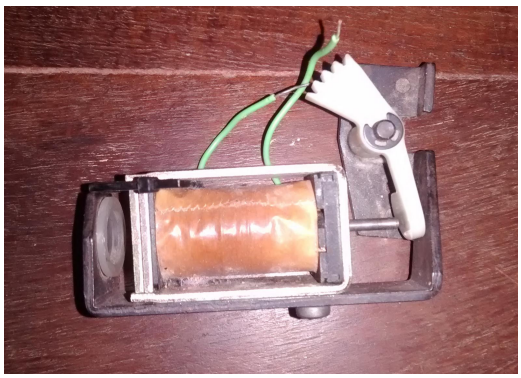


Fig.5. 11: Solenoide de prueba EKS

d. EKS Modificado: 24V; 100% Ciclo de Trabajo

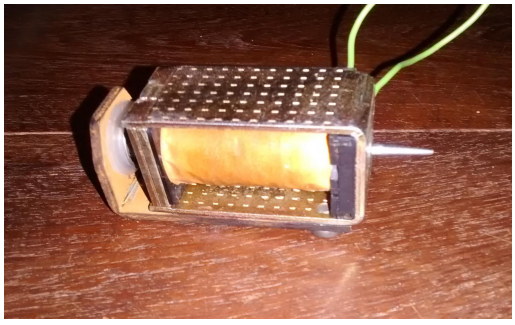


Fig.5. 12: Solenoide de prueba EKS modificado

La modificación consistió en darle al pin del embolo la misma forma que el punzón, e integrar un resorte en la estructura. En la siguiente figura se muestra, de izquierda a derecha, el émbolo original; el émbolo modificado; y el punzón de donde se tomó el modelo de punta.

Fig.5. 13: Modificación del pin del embolo EKS

Lamentablemente ninguno pudo lograr el requisito básico de marcar el papel: no tenían la fuerza suficiente.

5.3. Actuador eléctrico universal para seguro de auto

Se decidió buscar una solución alternativa, antes de seguir aumentando la potencia de los solenoides, debido a que el peso y consumo ya resultaban excesivos.

Se realizó una prueba con un actuador eléctrico para seguro se las puertas de automóvil.

Fig.5. 14: Actuador eléctrico universal

Principio de funcionamiento

Este actuador está compuesto por un motor cc, el cual acciona un sistema de piñón y cremallera, que mueve el vástago (ver Fig.5. 15 y Fig.5. 16). Dependiendo el sentido de giro del motor (la polaridad de la tensión aplicada) el vástago puede moverse hacia adelante o hacia atrás.

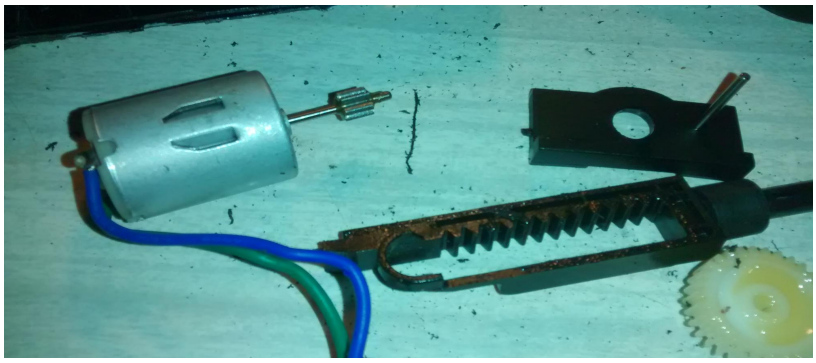


Fig.5. 15: Componentes del actuador

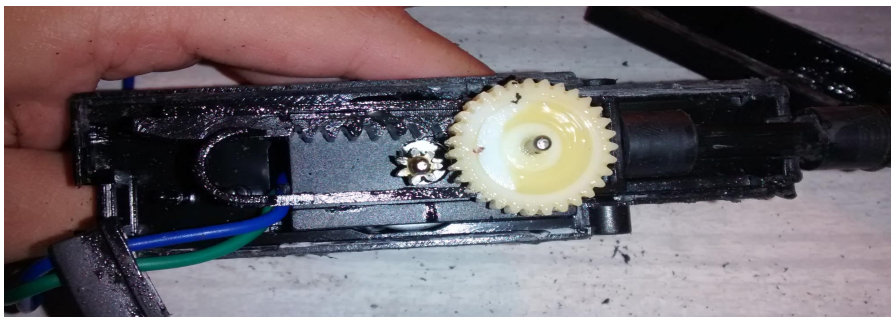


Fig.5. 16: Vista interna del actuador

Motor de cc (corriente continua)

El motor de corriente continua convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción que se genera del campo magnético.
Se compone principalmente de dos partes: el estator, que da soporte mecánico al aparato y contiene los devanados principales de la máquina, los polos, que pueden ser de imanes permanentes o electroimánes; y el rotor, que consiste en un devanado de alambre de cobre con un núcleo de hierro, alimentado con corriente continua mediante escobillas fijas.

Funcionamiento como percutor

El sistema cumplió el requisito de fuerza, pero fue superior a la necesaria.
La fuerza del motor viene dada por los siguientes factores:

$$F = I.L.B$$

- **F**: Fuerza
- **I:** Intensidad de corriente que recorre el conductor del rotor
- **L:** Longitud del conductor del rotor (espira)
- **B**: Densidad de campo magnético del estator

Dado que L y B no se pueden modificar: L depende de la construcción de la bobina del rotor y B es generado por imanes permanentes; sólo se puede regular la fuerza, variando I (la corriente). Esto se logra variando la tensión aplicada.

El problema de fuerza excesiva se solucionó entonces regulando la tensión aplicada al motor mediante la técnica de PWM[1] y el control del tiempo de accionamiento.

Finalmente se decidió adoptar este sistema debido a las siguientes ventajas sobre un solenoide:

a. Amplia disponibilidad: se encuentran en cualquier tienda de repuestos de autos o de electricidad.
b. Accesibilidad económica: $120 (ciento veinte pesos argentinos)
c. Alimentación con 12Vcc
d. Reducción de peso: son más livianos que un actuador solenoide
e. Generan mayor fuerza de empuje y tracción.

Adaptación a la impresora convencional

El actuador se sujetó al carro original mediante una pieza de hierro realizada especialmente para este propósito (Fig.5. 17).

1. *ver capitulo 6.1 “Control de motores paso a paso” - Motor de movimiento de carro: EM-210*

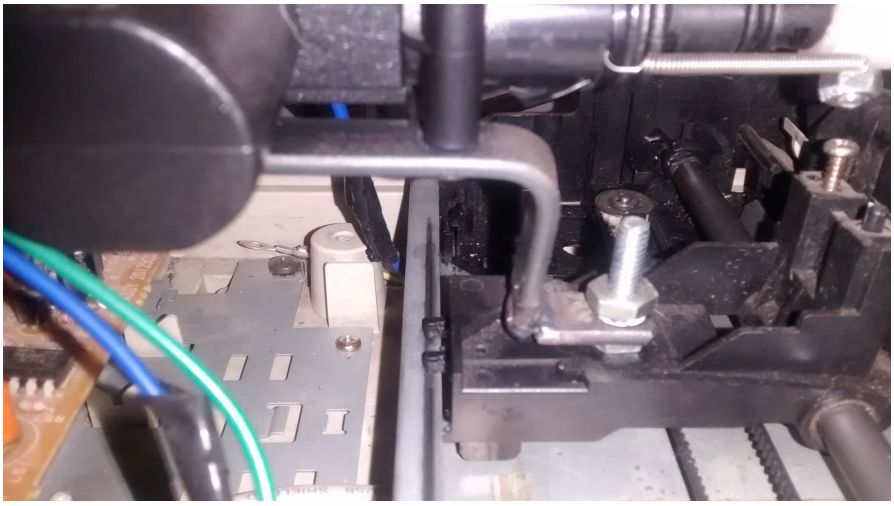


Fig.5. 17: Sujeción del actuador al carro

Se le adaptó al actuador la punta del punzón braille y un par de resortes para que el vástago regrese automáticamente luego de un golpe, para lo cual se utilizó una pieza de teflón (Fig.5. 18).

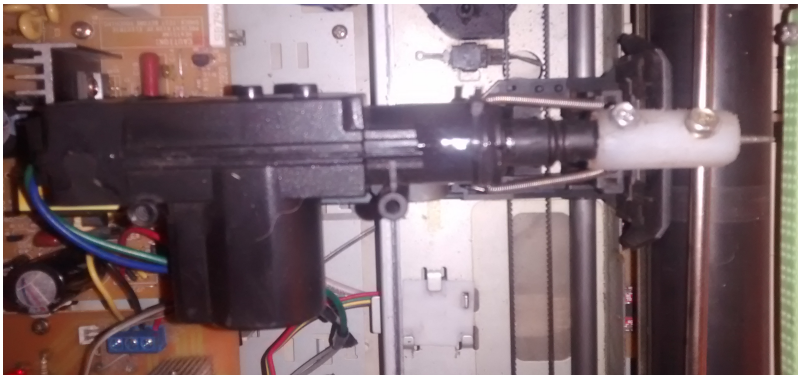


Fig.5. 18: Actuador adaptado como percutor

Por ultimo se agregó, sobre el rodillo, un yunque con una pieza plástica (que es un renglón completo de la plantilla para escribir braille) para que los puntos en la hoja se formen de manera correcta al impactar el percutor (Fig.5. 19).

Una descripción de las piezas realizadas en detalle pueden verse en el Anexo 1.

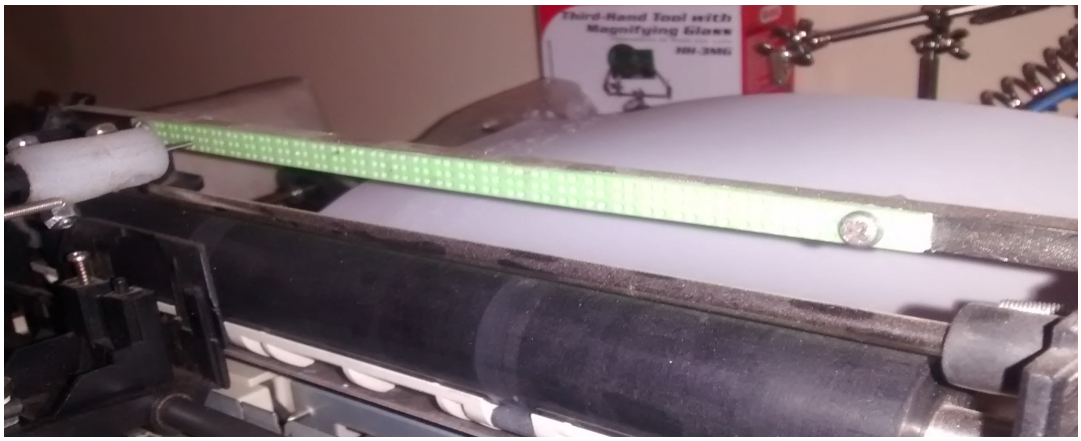


Fig.5. 19: yunque y pieza plástica de la plantilla

6. Placa electrónica

A continuación se detallan los pasos llevados a cabo para construir la placa electrónica de control.

6.1. Control de motores paso a paso

La placa electrónica debe cumplir principalmente la función de controlar los motores paso a paso (pap) que manejan la traslación del carro, que lleva el percutor, y el avance de la hoja. A continuación se describen los métodos para controlar estos motores.

Motor paso a paso

El motor a paso es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de avanzar una serie de grados (paso) dependiendo de sus entradas de control. Este motor presenta las ventajas de tener precisión y repetitividad en cuanto al posicionamiento (mientras la cupla de carga no supere la cupla motriz).

Fig.6. 1: Motor paso a paso

Los motores pap tienen un comportamiento del todo diferente al de los motores de corriente continua. En primer lugar, no giran libremente por sí mismos. Los motores pap, como lo indica su nombre, avanzan girando por pequeños pasos. También difieren de los motores de CC en la relación entre velocidad y torque (un parámetro que también es llamado "par motor" y "par de giro"). Los motores de CC no son buenos para ofrecer un buen torque a baja velocidad sin la ayuda de un mecanismo de reducción. Los motores paso a paso, en cambio, trabajan de manera opuesta: su mayor capacidad de torque se produce a baja velocidad.

Los motores pap tienen una característica adicional: el torque de detención (que se puede ver mencionado también como "par de detención", e incluso par/torque "de mantenimiento"), que no existe en los motores de CC. El torque de detención hace que un motor paso a paso se mantenga firmemente en su posición cuando no está girando. Esta característica es muy útil cuando el motor deja de moverse y, mientras está detenido, la cupla de carga permanece aplicada a su eje. Se elimina así la necesidad de un mecanismo de freno.

Secuencia de control

Estos motores tienen varios bobinados que, para producir el avance por pasos, deben ser alimentados en una adecuada secuencia. Si se invierte el orden de esta secuencia, se logra que el motor gire en sentido opuesto. Si los pulsos de alimentación no se proveen en el

orden correcto, el motor no se moverá apropiadamente. Puede ser que zumbe y no se mueva, o puede ser que gire, pero de una manera tosca e irregular.

Unipolares:
Estos motores suelen tener 5 o 6 cables de salida dependiendo de su conexión interna. Este tipo se caracteriza por ser más simple de controlar, estos utilizan un cable común a la fuente de alimentación y posteriormente se van colocando las otras líneas a tierra en un orden específico para generar cada paso, si tienen 6 cables es porque cada par de bobinas tienen un común separado, si tiene 5 cables es porque las cuatro bobinas tienen un polo común; un motor unipolar de 6 cables puede ser usado como un motor bipolar si se deja las líneas del común al aire.

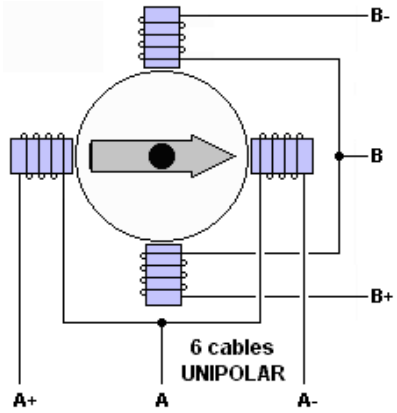


Fig.6. 2: Motor pap unipolar de 6 hilos

Bipolares:
Estos tienen generalmente 4 cables de salida. Necesitan ciertos trucos para ser controlados debido a que requieren del cambio de dirección de flujo de corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento.

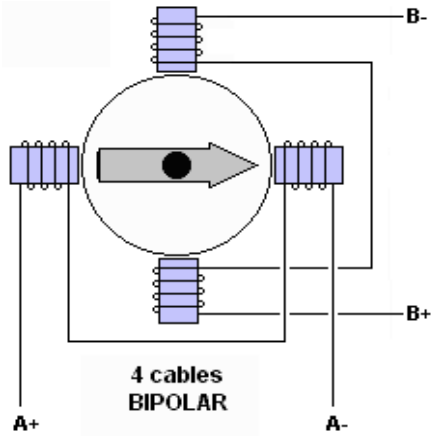


Fig.6. 3: Motor pap bipolar de 4 hilos

Los motores disponibles en la impresora de base son motores paso a paso unipolares de 6 hilos; y constructivamente del tipo denominado “de imán permanente”, lo que significa que tienen un rotor multipolar de imán permanente, que son atraídos a los dientes del estátor cuando son electromagnéticamente energizados.

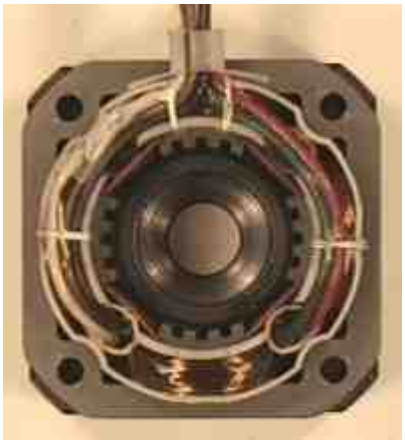


Fig.6. 4: Estator de motor pap de imán permanente

Fig.6. 5: Rotor de motor pap de imán permanente

Secuencia para motores unipolares

En el esquema más común de conexión se unen los "puntos medios" de ambos ejes y se les conecta al positivo de la alimentación del motor. El circuito de control de potencia, entonces, se limita a poner a masa los bobinados de manera secuencial.[1]

La secuencia y el comportamiento en cada paso puede verse en la tabla 6.1.

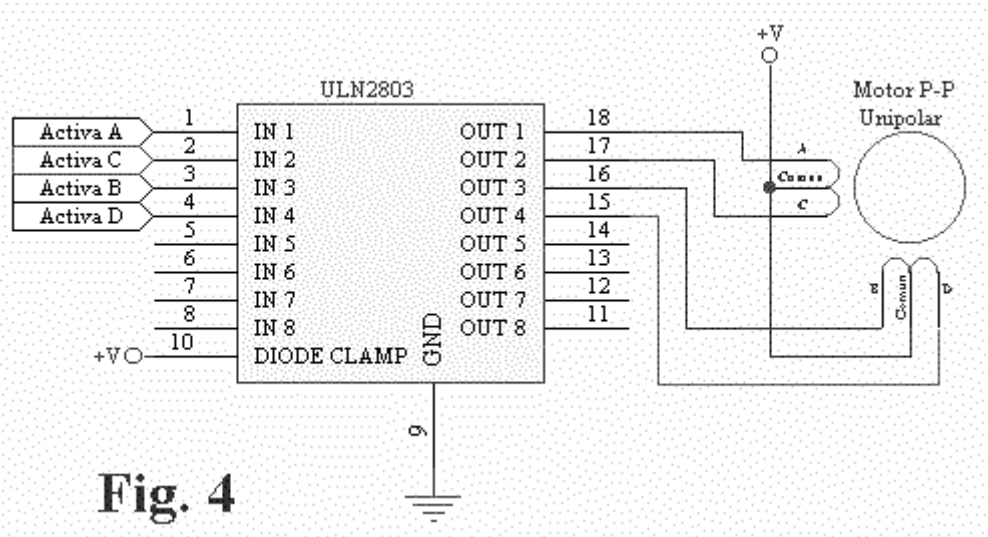
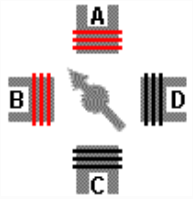
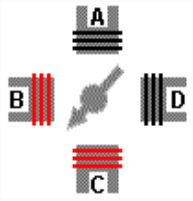
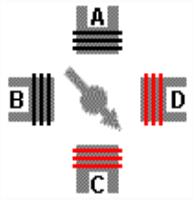
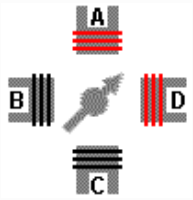


Fig.6. 6: Esquema de control de motor pap unipolar

1. Fuentes: *http://robots-argentina.com.ar/MotorPP_basico.htm*

http://www.todorobot.com.ar/tutorial-sobre-motores-paso-a-paso-stepper-motors/

Tabla 6.1. Secuencia de control de motor pap unipolar

PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	**ON**	**ON**	OFF	OFF	
2	OFF	**ON**	**ON**	OFF	
3	OFF	OFF	**ON**	**ON**	
4	**ON**	OFF	OFF	**ON**	

Motor de avance de hoja: EM-211

Características de motor pap de avance de hoja: EM-211

- Tipo: Unipolar de 6 hilos
- Número de pasos: 50
- Angulo por paso: 7,2º
- Corriente por paso: 0,3A
- Tensión por paso: 9V
- Revoluciones por minuto: trabaja bien entre 50RPM y 300RPM
- Hilos: +A = Naranja, -A = Azul, +B = Blanco, -B = Rojo, COMUNES = Marrones

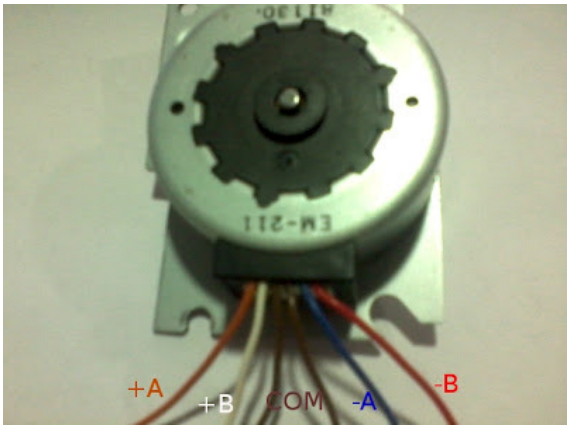


Fig.6. 7: Motor pap EM-211

En el caso del motor utilizado para el avance de la hoja, EM-211, el circuito de control se realizó como el indicado en la Fig.6. 6. El driver utilizado fue el ULN2003A, el cual proporciona un manejo de corriente de hasta 0,5A por canal y ademas implementa **diodos volante** para el manejo de cargas inductivas, los cuales proporcionan un camino libre para la circulación de la corriente almacenada en la bobina al desenergizarla, para evitar que se produzcan sobretensiones peligrosas en el circuito.

Un calculo mas detallado del dimensionamiento de los componentes puede verse en el Anexo 2.

El circuito de control realizado puede verse en la Fig.6. 8, donde el microcontrolador utilizado es el LPCXpresso, identificado en el circuito como “U7”.

En la figura puede apreciarse la utilización del pin 27 del microcontrolador, para controlar la alimentación del motor (+9Vcc); esto permite cortar la alimentación cuando se manipula el rodillo de forma manual.

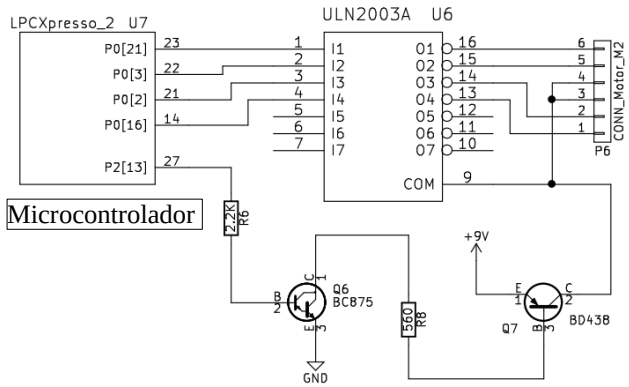


Fig.6. 8: Circuito de control realizado para motor de avance de hoja

Tanto el integrado ULN2003A, como los transistores utilizados fueron seleccionados en base a los cálculos de corrientes de control y alimentación del motor.

Motor de movimiento de carro: EM-210

Características de motor pap de movimiento de carro: EM-210

- Tipo: Unipolar de 6 hilos
- Número de pasos: 50
- Ángulo de paso: 7,2º
- Corriente por paso: 0,47A
- Tensión por paso : 9V
- Revoluciones por minuto: funciona bien entre 50 rpm y 300 rpm
- Cableado: + A = Naranja, -A = Azul, B + = Blanco, -B = Rojo, COMUNES = Marrón

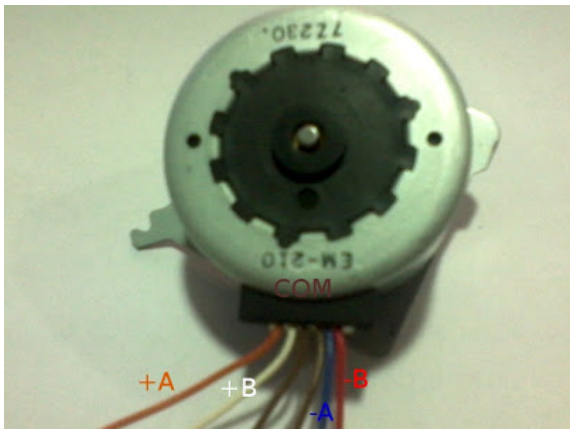


Fig.6. 9: Motor pap EM-210

Para el control de este motor debieron aplicarse técnicas adicionales. Este motor se encuentra alimentado con una tensión superior a la nominal, lo que le proporciona mayor torque y mejor desempeño a alta velocidad; pero esta sobre-tension debe compensarse con un control de la corriente, de otra forma circularía por los bobinados del estator una corriente excesiva (ley de ohm), que destruiría el motor por sobrecalentamiento. La forma de limitar esta corriente es mediante la técnica de modulación de ancho de pulso, o PWM (pulse width modulation).

La modulación PWM consiste básicamente en convertir una tensión continua en una serie de “pulsos” a una frecuencia **constante** (ver Fig.6. 10). La energía transferida, a la bobina del motor en este caso, depende del **ancho** de los pulsos en ON (pulse width); conformando así un “ciclo de trabajo” (duty cycle) determinado por el ancho del pulso ON, sobre la duración total del ciclo (period).

El circuito original de la impresora base incorpora un Driver de control especifico para tal propósito: SLA7029. El cual se basa en comparadores, y a partir de algunos elementos de circuito adicional proporciona la modulación para limitar la corriente.

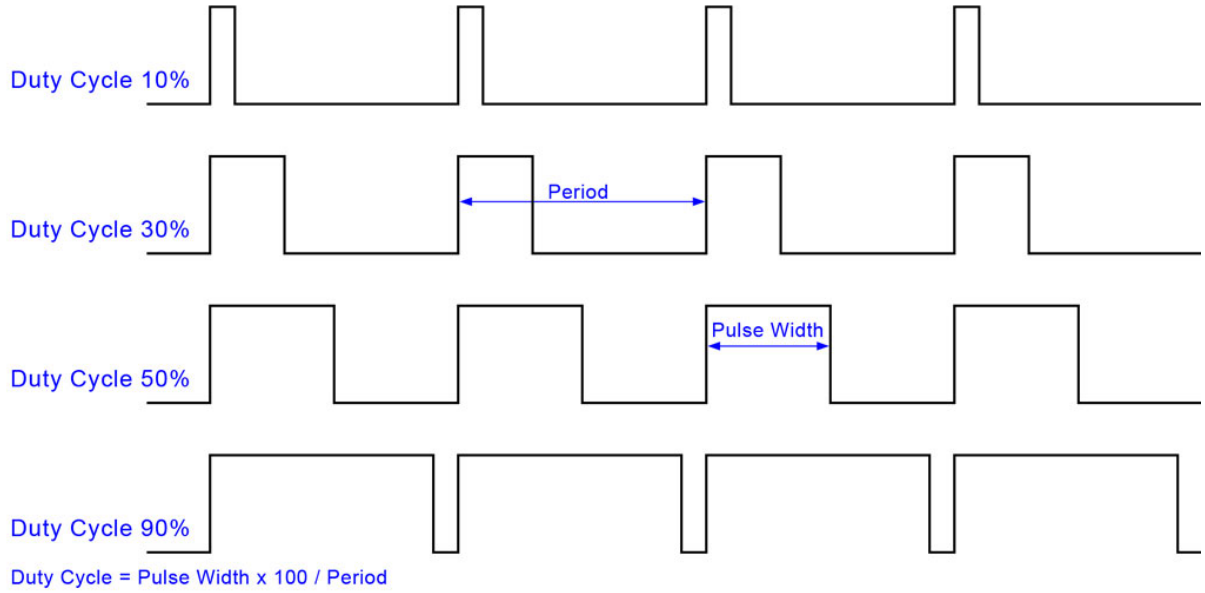


Fig.6. 10: Modulacion PWM

Se decidió para el proyecto, prescindir de un driver adicional y aprovechar las características del microcontrolador, aplicando la modulación PWM proporcionada por este ultimo; de esta manera se tendría una cierta flexibilidad para el control, simplemente variando parámetros en la programación.

A partir de la hoja de datos del driver y el análisis del circuito de la placa original se extrajeron los valores de frecuencia y ciclo de trabajo originales, para ser emulados en la programación del microcontrolador.

Debido a alta corriente que maneja este motor no era posible utilizar un amplificador integrado como en el caso anterior; por lo que se utilizaron transistores individuales, en una configuración Darlighton, con sus respectivos diodos volantes.

Un calculo mas detallado del dimensionamiento de los componentes puede verse en el Anexo 2.

La Fig.6. 11 muestra el circuito de control terminado.

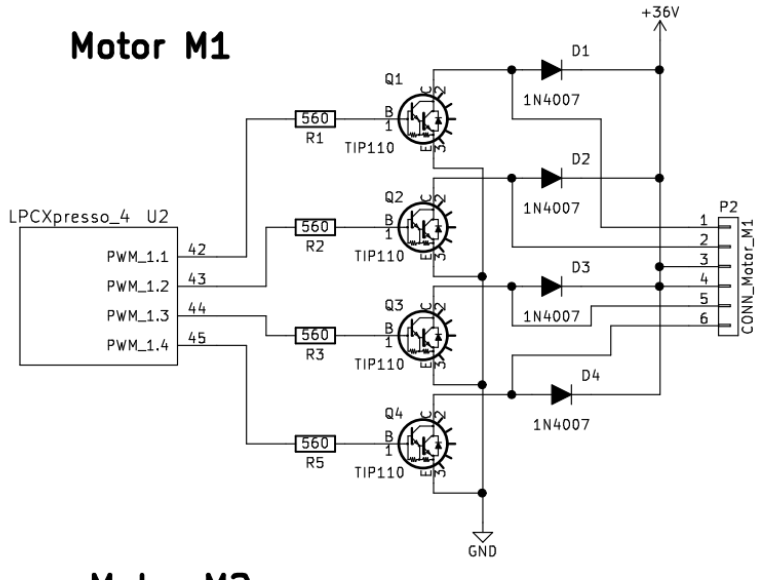


Fig.6. 11: Circuito de control realizado para motor de movimiento de carro

6.2 Control del percutor

En el caso del percutor, al tratarse en principio de un motor de cc, se realizó un circuito de control similar al de una bobina de un motor pap.
La utilización de la técnica de PWM para controlar la energía suministrada permitió reducir la fuerza de impacto (que, como se mencionó, era excesiva); ademas permitió prescindir de elementos reguladores de tensión adicionales, ya que la fuente original no tenia salida de voltaje de 12V, como requería el actuador.

Un calculo mas detallado del dimensionamiento de los componentes puede verse en el Anexo 2.

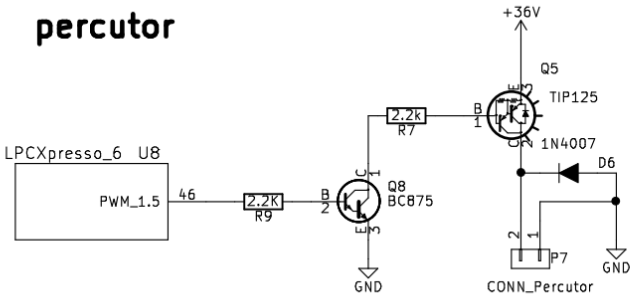


Fig.6. 12: Circuito de control realizado para percutor

6.3. Comunicación serial – USB

Uno de los requisitos establecidos para la impresora braille construida fue que sea de sencilla de operación, tanto como una impresora convencional moderna. Para lo cual era necesario una conexión USB.

La manera mas sencilla de realizarlo resultó ser la incorporación de un circuito adicional de adaptación uart – usb (usb to uart bridge). El adaptador elegido fue el que se muestra en la Fig.6. 13. El cual está basado en el CI CP2102.

Fig.6. 13: Adaptador UART - USB

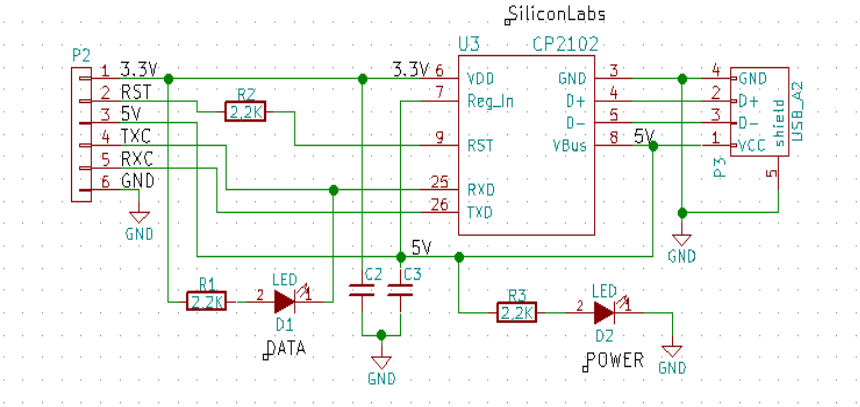


Fig.6. 14: Esquematico del adaptador UART - USB

En la Fig.6. 15 se muestra un diagrama esquemático del funcionamiento del chip CP2102 que es el que realiza la conversión de protocolos. Se puede ver un bloque que contiene las señales clásicas de la comunicación UART a la derecha como TXD, RXD, RTS, CTS, etc.; y los bloques que componen la comunicación USB a la izquierda: un oscilador de 48MHz, dos buffers de Tx y Rx, una memoria E2PROM donde se almacena la configuración y datos del tipo de dispositivo USB como fabricante, consumo, clasificación del dispositivo, etc.

El adaptador tiene una sencilla conexión con el microcontrolador: sólo se conectan las terminales Rx y Tx del adaptador, con las terminales correpondientes del canal UART del microcontrolador (Fig.6. 16). De esta forma los protocolos de comunicación se programaron simplemente como una comunicación UART en el micro.

En la PC, en caso de tener el S.O. Windows®, se debe instalar un driver para que el sistema reconozca esta conexión USB como un puerto serie (virtual); y de igual forma se utiliza la comunicación serial (por este puerto) para programar el software de usuario.

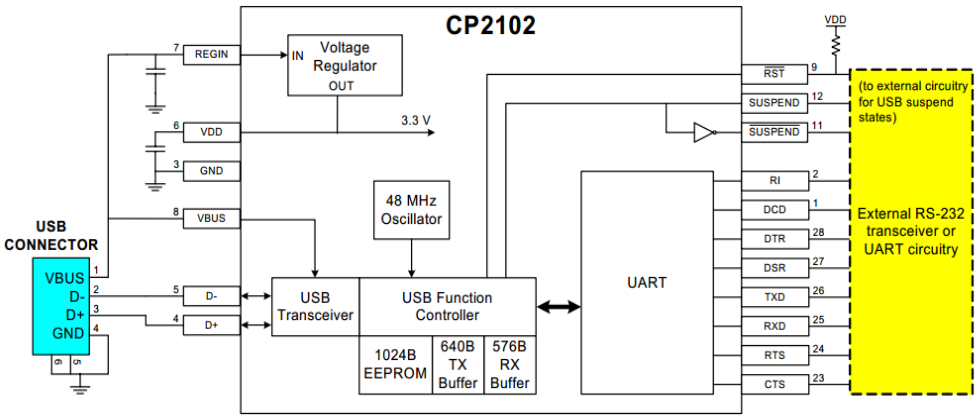


Fig.6. 15: CI CP2102

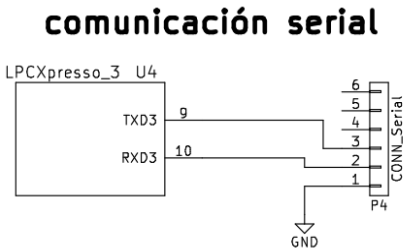


Fig.6. 16: Conexión microcontrolador - adaptador

6.4. Normas de seguridad eléctrica para protección del usuario

La **Resolución Nº 508/15** de la secretaria de comercio establece en su **ARTICULO 1º**, que: "El equipamiento eléctrico de baja tensión que se comercialice en la REPÚBLICA ARGENTINA deberá contar con una certificación que acredite el cumplimiento de los requisitos esenciales de seguridad"[1]:

1. El equipamiento eléctrico deberá contener información acerca de las características fundamentales de cuyo conocimiento y observancia dependa su utilización acorde con el destino y el empleo seguro. Esta información figurará sobre el producto o, cuando esto no sea posible, en la nota que lo acompañe, en ambos casos redactadas en idioma nacional.
2. El equipamiento eléctrico deberá contener la siguiente información: el país de origen, la razón social del fabricante o la marca comercial registrada, su domicilio legal, la razón social y domicilio legal del importador y del distribuidor en el país y el modelo

1. *fuente: http://www.cda.org.ar/index.php/normativa-movil/22312-seguridad-electrica-equipamiento-de-baja-tension-normas-iram-modificaciones*

del producto. Irán colocados de manera distinguible e indeleble en el equipamiento eléctrico o, no siendo esto posible, al menos la marca comercial registrada y el modelo irán colocados en el equipamiento eléctrico y el resto de la información en el envase primario.

3. El equipamiento eléctrico y sus partes constitutivas se fabricarán de modo que permitan una conexión segura y adecuada.
4. El equipamiento eléctrico habrá de diseñarse y fabricarse de modo que quede garantizada la protección contra los peligros a que se refieren los puntos 2 y 3 del presente listado, a condición de que su uso sea el indicado por el fabricante y sea objeto de adecuado mantenimiento.
5. La clase de aislación será la adecuada para las condiciones de utilización previstas, quedando expresamente prohibidas las clases de aislación 0 y 0I.
6. Protección contra los peligros originados en el propio equipamiento eléctrico.
7. Se preverán medidas de índole técnica conforme al punto 1, a fin de que:
   a) Las personas y los animales domésticos queden adecuadamente protegidos contra el riesgo de heridas y otros daños que puedan sufrir a causa de contactos directos o indirectos.
   b) No produzcan temperaturas, arcos o radiaciones peligrosas.
   c) Se proteja convenientemente a las personas, animales domésticos y los bienes contra los peligros de naturaleza no eléctrica causados por el equipamiento eléctrico.
8. Protección contra los peligros causados por efecto de influencias exteriores sobre el equipamiento eléctrico.
9. Se establecerán medidas de orden técnico conforme al punto 1, a fin de que:
   a) El equipamiento eléctrico responda a las exigencias mecánicas previstas con el objeto de que no corran peligro las personas, los animales domésticos y los bienes.
   b) El equipamiento eléctrico resista las influencias no mecánicas en las condiciones previstas de medio ambiente con objeto de que no corran peligro las personas, los animales domésticos y los bienes.
   c) El equipamiento eléctrico no ponga en peligro a las personas, los animales domésticos y los bienes en las condiciones previstas de sobrecarga.

Todas las medidas de seguridad mencionadas anteriormente deberán ser cumplidas en caso de pretender llevar este modelo funcional a una etapa posterior como modelo de producción. En su actual condición la impresora cumple con los puntos 3, 6, 7b, 7c y 9; pudiéndose incorporar por ejemplo, una carcasa adecuada que recubra el aparato para una adecuada aislación y protección contra contactos, y de esta forma cumplir con los puntos 4, 5 y 7.a.

Además, la misma resolución establece en su **ARTÍCULO 6°** que:
“Las certificaciones exigidas por la presente resolución deberán acreditar que los productos alcanzados por la misma cumplan los requisitos de seguridad establecidos por la Norma IRAM correspondiente. En caso de inexistencia de Norma IRAM, o de no encontrarse ésta vigente, las certificaciones citadas deberán acreditar el cumplimiento de la Norma IEC aplicable.”

Siendo una norma IRAM aplicable la **NM 60335-1:** "*Seguridad de aparatos electrodomésticos y similares*"[1]; se pueden destacar los siguientes puntos, indicando en cada uno si la impresora los cumple o no:

- El **punto 6** de la norma establece que el producto debe pertenecer a alguna de las clases mencionadas con respecto a la protección de choques eléctricos. La impresora, en tal caso, pertenece a la clase 1, ya que ademas de la aislación básica proporcionada por la carcasa de material no conductor (no quedando ninguna parte conductora accesible), posee un conductor de protección que forma parte del cableado fijo.
- El **punto 7** trata del "*marcado e instrucciones*", el cual establece que el aparato debe marcarse con la tensión nominal, símbolo de tipo de corriente, frecuencia nominal, etc. Este punto puede fácilmente ser realizado si se desea llevar a un modelo de producción.
- **Punto 8**: "*protección contra la accesibilidad a las partes activas*". Como se mencionó anteriormente, el diseño de una carcasa adecuada que encierre el aparato y evite el contacto accidental con las partes activas puede dar cumplimiento a este punto.
- El **punto 11** de la norma trata del "*calentamiento*", una cuestión que no es menor, si se tiene en cuenta el calentamiento de los motores pap. El motor de movimiento del carro es el de mayor consumo y el que se calienta durante el funcionamiento; el mismo no es accesible al usuario, ademas cuenta con una adecuada ventilación (en la carcasa original) y el armazón de aluminio que lo sostiene actúa como disipador y le permite enfriarse rápidamente.
- El **punto 17** trata de la "*protección contra la sobrecarga de transformadores*..". Considerando que la fuente utilizada es la original y que la misma posee un circuito de protección contra cortocircuitos se puede decir que cumple con este punto.
- Otro punto importante a tener en cuenta es el **20**, que habla sobre la "*estabilidad y peligros mecánicos*". En cuanto a la estabilidad, el modelo de impresora elegido es suficientemente robusto para soportar los golpes del percutor y ademas posee bujes de gomas en la fijación a la carcasa que reduce la vibración al mínimo.
  El punto 20.2 indica que las partes móviles deben estar **encerradas** para proporcionar protección al usuario. Este es uno de los puntos que el modelo funcional de impresora no cumple, ya que la carcasa solo brinda protección eléctrica. La norma podría cumplirse adecuando una cubierta de protección al sistema de carro-percutor, de modo que el usuario no tenga acceso al mismo.
- El **punto 22.2** de la norma trata de la "*desconexión omnipolar de la alimentación*" en el apartado "*construcción*". La impresora posee un cable de alimentación con ficha y ademas un interruptor que desconecta la alimentación principal, pero no es "bipolar" como exige la norma.
- El **punto 25.5** trata de los métodos para ensamblar el cable de alimentación, siendo el método de "fijación *tipo x*", el que cumple la impresora, en el cual el cable de alimentación puede ser reemplazado fácilmente. El mismo cumple con todo el **punto 25** de la norma para este tipo de fijación.
- Por ultimo mencionar el **punto 27**: "*Disposiciones para la puesta a tierra*". Se cumple, dado que todas las partes metálicas accesibles (como el soporte del carro porta-cabezal) están conectadas al borne de tierra.

1. *Normas IRAM – PC de consulta habilitada en Biblioteca de UTN - FRLR*

7. Programación de microcontrolador LPCXpresso

Se decidió utilizar para el proyecto la placa de control LPCXpresso, y una programación basada en el concepto de “maquina de estado”.

7.1. LPCXpresso

LPCXpresso ™ es una plataforma de desarrollo de bajo costo disponibles en NXP Semiconductors N.V. como soporte a microcontroladores basados en ARM de NXP. La plataforma está compuesta por un IDE simplificado basado en Eclipse y placas de bajo costo (target board), que incluyen un depurador JTAG adjunto (LPC-Link). LPCXpresso ™ es una solución de extremo a extremo, que permite a los ingenieros de sistemas embebidos desarrollar sus aplicaciones desde la evaluación inicial hasta la producción final.

Incorporando el LPC1769 para mostrar las características LPC176x, el LPCXpresso LPC1769 combina el alto nivel de integración y de bajo consumo de energía del LPC1769 y el bajo precio de LPCXpresso ™.[1]

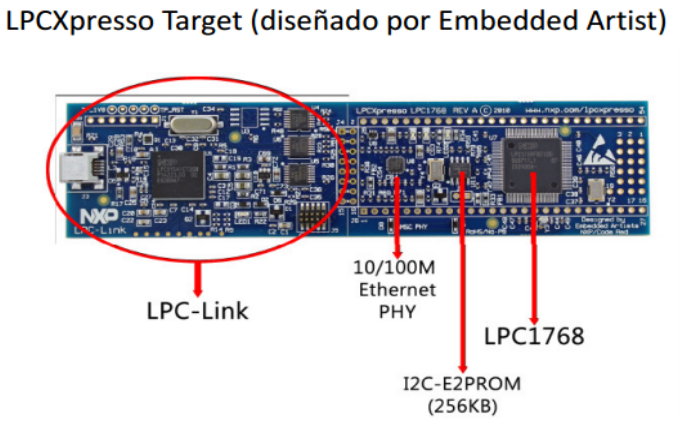


Fig.7. 1: LPCXpresso target board y LPC-Link

Como muestra la Fig.7. 2, el LPCXpresso se compone de una placa “target”, que contiene el microcontrolador LPC1769 y componentes adicionales (como la memoria E2PROM y el controlados de Ethernet); y de una placa “Link”, que incorpora el programador con conexión usb y el depurador (debugger) JTAG.

1. *fuente: http://www.nxp.com/board/OM13000.html*

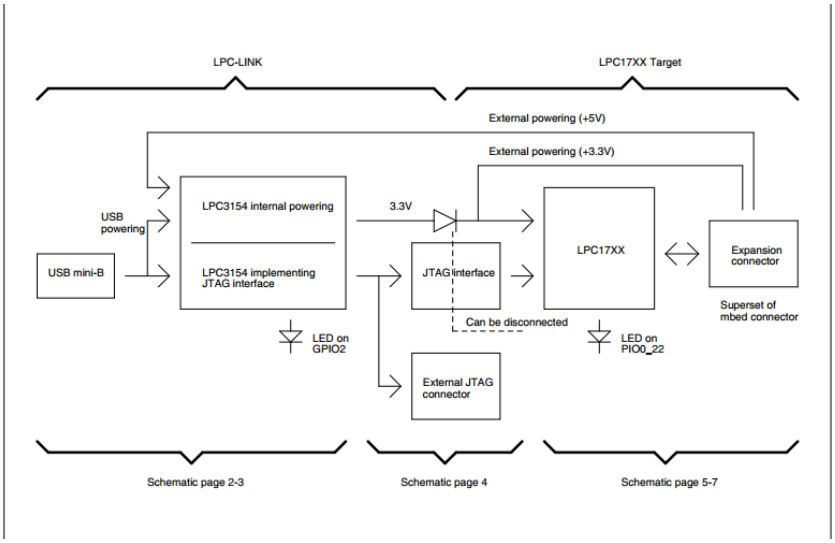


Fig.7. 2: Esquemático LPCXpresso

Microcontrolador LPC1769

El LPC1769 es un microcontrolador fabricando por NXP Semiconductors N.V. (una empresa fundada por Philips).
Este microcontrolador utiliza el procesador Cortex M3 diseñado por ARM.
NPX junta en el mismo bloque de silicio el procesador diseñado por ARM y diferentes periféricos. El microcontrolador LPC1769 puede operar a una frecuencia de 120MHz.
El procesador Cortex M3 utiliza una arquitectura Harvard (instrucciones y los datos separados).

Bloques agregados por NXP:
• 512 KByte de memoria flash
• 61 KByte de memoria SRAM
• Multilayer AHB matriz
• Dos buses APB
• DMA (8 canales) (puede ser usado con SSP, I2C, UART, ADC, DAC, Timers, GPIO, memoria-memoria)
• Ethernet
• USB 2.0
• UART (4 canales) (FIFOs internas)
• I2C – SSP – SPI (interfaz serial)
• ADC (12 bits) – DAC (10 bits)
• Timers (4 canales)
• PWM
• RTC incoporado
• JTAG (debug)

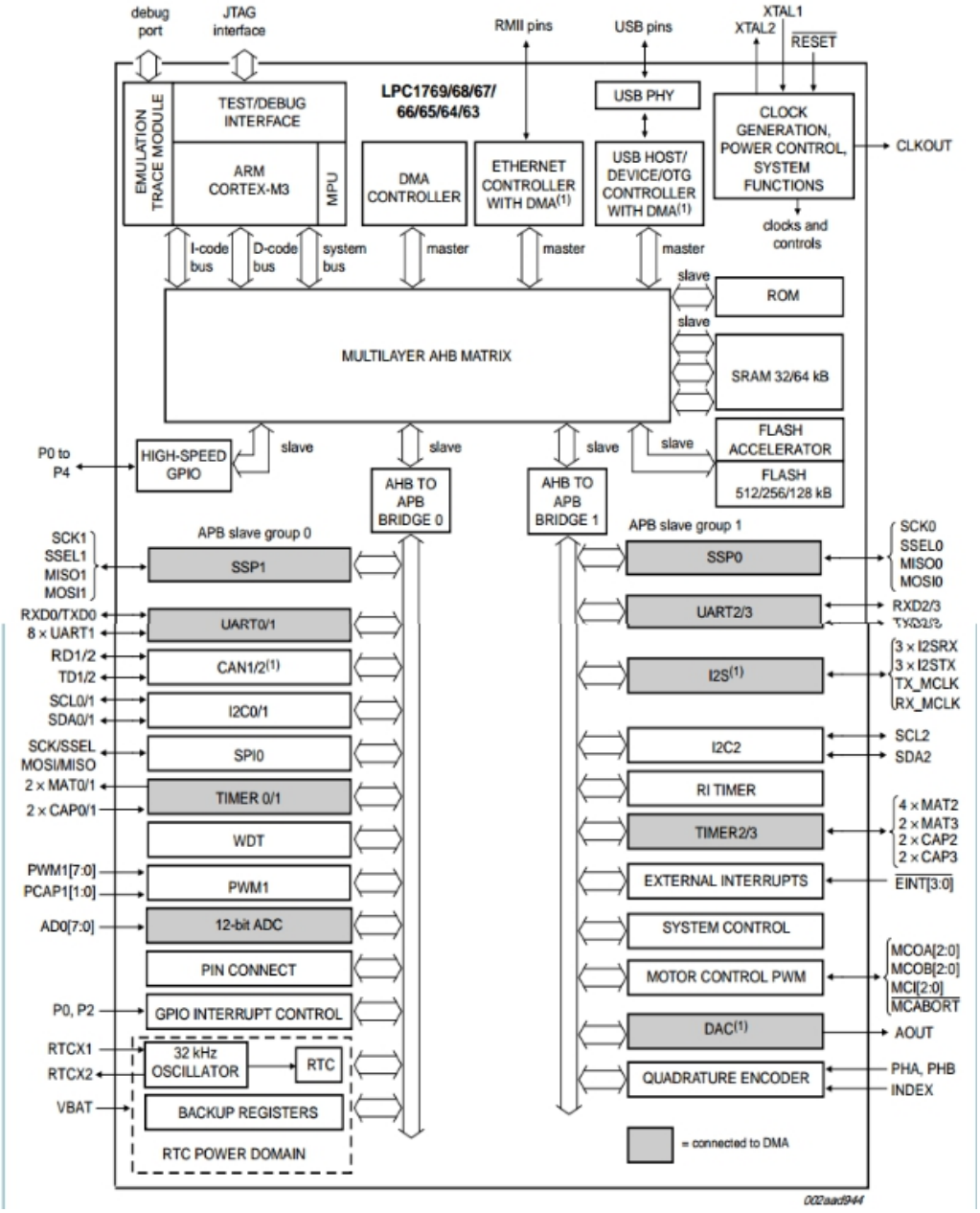


Fig.7. 3: Diagrama en bloques de LPC1769

Microprocesador ARM Cortex-M3

El ARM Cortex M3 es un procesador de 32 bits utilizado ampliamente en el mercado internacional para aplicaciones de tiempo real que requieren alta performance y bajo costo de desarrollo.

ARM vende la propiedad intelectual del Cortex M3 para que diferentes fabricantes de silicio lo utilicen en sus diseños, en particular, en microcontroladores.

El procesador desarrolla un elevado poder computacional, y al mismo tiempo, se ajusta a requerimientos exigentes de potencia (dinámica y estática).

El NVIC (Nested Vector Interrupt Controller) es uno de los principales bloques de los procesadores de la línea Cortex M3.
El Cortex M3 soporta hasta 240 fuentes de interrupciones (IRQs), una interrupción no enmascarable y varias fuentes de excepciones de sistema.

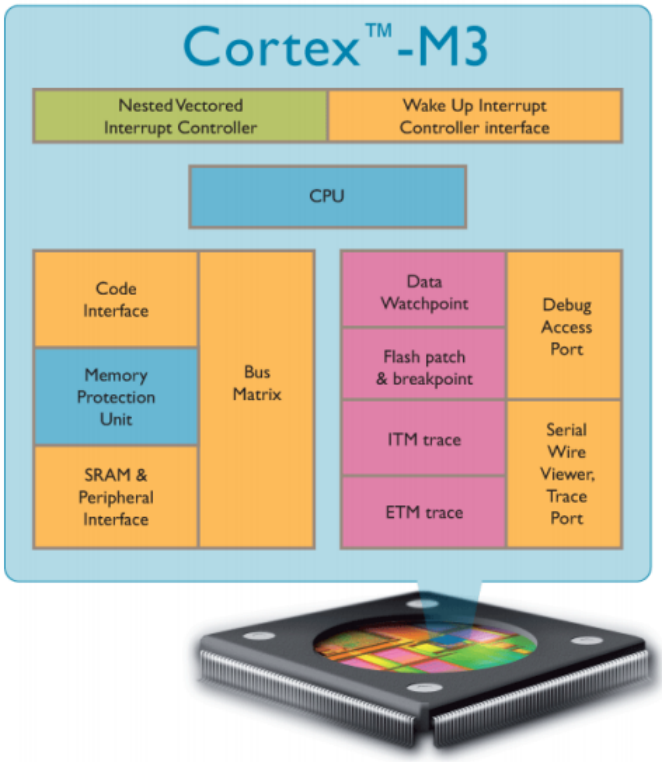


Fig.7. 4: Componentes del ARM Cortex-M3

Firmware CMSIS

CMSIS (Cortex Microcontroller Software Interface Standard) es una capa de abstracción del hardware independiente del fabricante del silicio. ARM exige que los fabricantes de silicio le brinden a los clientes esta capa de abstracción del hardware.
Se trata de código en lenguaje C que se encuentra dividido en dos partes:
• Core Support: Interfaz de acceso al núcleo
• Device Support: Interfaz de acceso a periféricos propios del fabricante

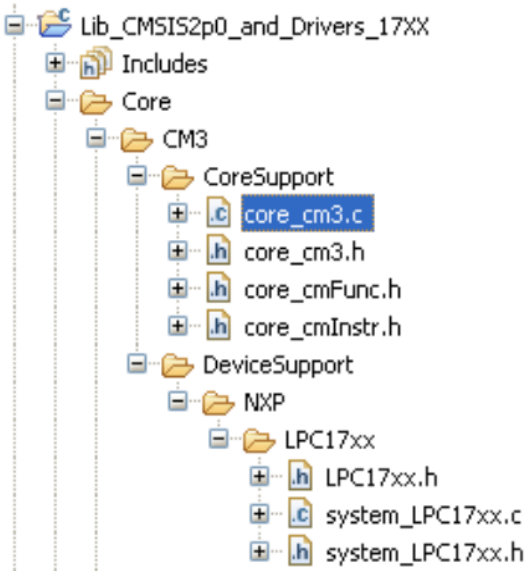


Fig.7. 5: Libreria de CMSIS

IDE LPCXpresso

El IDE LPCXpresso es un entorno de desarrollo de software con todas las funciones de MCU (microcontroladores) basado en ARM de NXP, e incluye todas las herramientas necesarias para desarrollar aplicaciones de software de alta calidad integrados en una manera optima y rentable.
El IDE LPCXpresso se basa en el IDE de Eclipse y cuenta con muchas facilidades de uso y mejoras específicas de MCU. El IDE también incluye las herramientas GNU ARM estándar de la industria que proporcionan herramientas de calidad profesional a un bajo costo. El depurador con todas las funciones admite la depuración tanto SWD y JTAG, y cuenta con descarga directa a la memoria flash del chip.[1]

1. *Todas las definiciones y figuras anteriores fueron extraídos de las siguientes fuentes:*

- L*PCXpresso User Guide 1 – Rev 7.4 - NXP Semiconductors*
- *SASE 2012 – Workshop LPCXPresso - Jorge Ezequiel Espósito y Federico Roasio – Laboratorio de Sistemas Embebidos – Facultad de Ingenieria UBA*

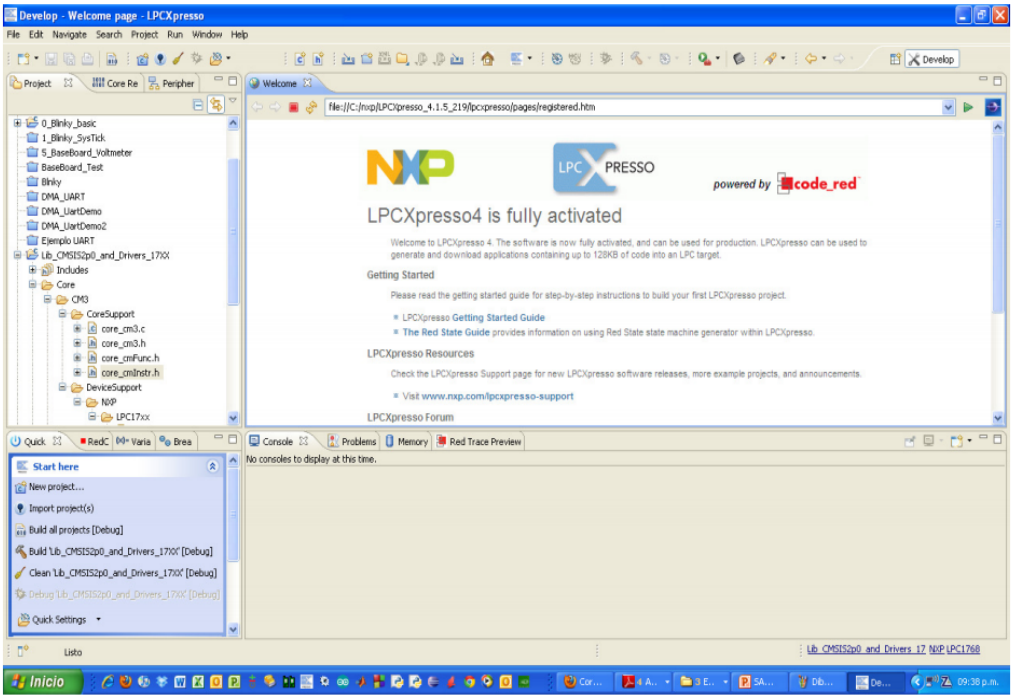


Fig.7. 6: IDE LPCXpresso

7.2. Programa de control

Para el desarrollo del software de control se aplicó el modelo de “maquina de estado”.

Máquina de Estado

Una máquina de estado es un modelo matemático de un sistema con entradas y salidas discretas, que posee sintaxis y semántica discretas formales. Útiles para representar aspectos dinámicos no representables por diagramas de otro tipo. Los diagramas de estado son particularmente útiles para modelar sistemas embebidos porque estos suelen ser reactivos, es decir responden a eventos externos que muchas veces son asincronicos.
En un proyecto de un sistema embebido la utilización de maquinas de estados sirve para:

- Ordenar las ideas, compartir ideas con el equipo de trabajo y documentar el proyecto.
- Simular las maquinas de estado para corregir errores y explorar otras alternativas.
- Traducirlos al lenguaje de programación deseado, en nuestro caso C. La traducción puede realizarse mecánicamente de forma manual o de forma automática si se cuenta con la herramienta adecuada.

Una maquina de estado esta formada, principalmente por los siguientes tres elementos:

- **ESTADOS:** Son una configuración única de información en un programa, es decir un conjunto particular de instrucciones las cuales serán ejecutadas en respuesta a una entrada. Su nombre debería describir aquellas características que hagan al estado. Ejemplos: APAGADO, PRENDIDO, TITILANDO, ESPERANDO, etc.
- **EVENTOS:** Son las señales que determinan el pasaje de un estado a otro o a sí mismo nuevamente.

- **ACCIONES:** Representan las reacciones del sistema frente a los eventos desarrollados en el mismo.[1]

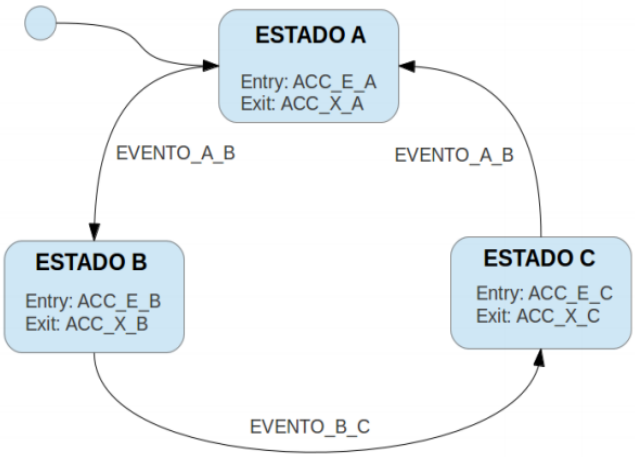


Fig.7. 7: Máquina de estado

Implementación

La maquina de estados diseñada para el proyecto se muestra en la Fig.7. 8.

A continuación se describe brevemente cada uno de los estados, las acciones y los eventos intervinientes:

Estado: 0
Al momento de encender la impresora, el programa entra automáticamente al estado "ENCENDIDO"

Estado: ENCENDIDO
La impresora está encendida. Se ejecuta la función "encender", la cual envía un mensaje de bienvenida (visible sólo mediante una consola de interface serial).
Se espera recibir el comando "enc" (**enc**ender), para producir el evento "e_iniciar" y pasar al estado "ESPERANDO"

1. *Fuente: SASE 2012 – Workshop LPCXPresso - Jorge Ezequiel Espósito y Federico Roasio – Laboratorio de Sistemas Embebidos – Facultad de Ingenieria UBA*

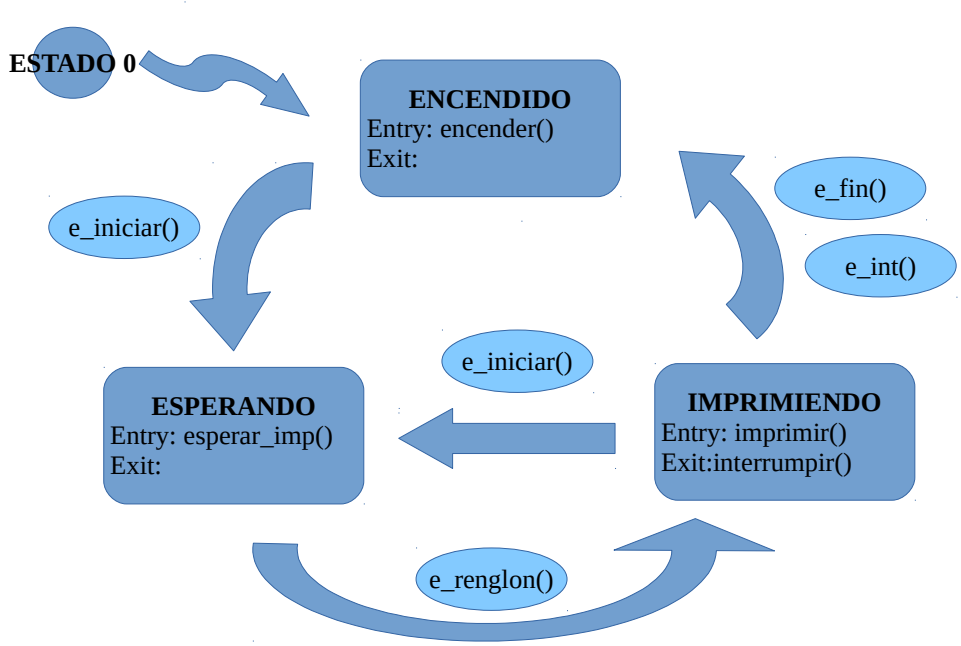


Fig.7. 8: Maquina de estados de impresora braille

Estado: ESPERANDO

La impresora espera que se envíe un renglón para ser impreso. La función activada es “esperar_imp”, que envía el mensaje “esperando renglón” por el puerto serie; y recibe 30 caracteres (el renglón).

Luego de recibir los 30 caracteres del renglón, se espera recibir el comando “imp” (**imp**rimir), que produce el evento “e_renglon” para pasar al estado “IMPRIMIENDO”

Estado: IMPRIMIENDO

Comienza la impresión del renglón enviado. La función “imprimir” comanda los motores pap y el percutor para producir los puntos en el papel, de acuerdo a los caracteres enviados.

En este punto pueden suceder 3 situaciones distintas:

- Se imprime el renglón por completo y se espera el siguiente: se produce automáticamente el evento “e_iniciar” para volver al estado “ESPERANDO”.
- Se imprime el ultimo renglón del documento: se produce el evento “e_fin”, mediante el envío del comando “fin”, por lo que al finalizar la impresión se vuelve al estado “ENCENDIDO”
- Se produce una interrupción de la impresión: si se envía el comando “int”, se ejecuta la función “interrumpir” y se interrumpe inmediatamente la impresión. Se genera el evento “e_int” y se vuelve al estado “ENCENDIDO”

La programación de la maquina de estados se tradujo al lenguaje “C” por el método de codificación con “SWITCH”, el cual tiene la siguiente forma básica:

```
switch ( estado ) {
        case ESTADO_A:
        if ( evento == e_evento_A_B) {
                estado = ESTADO_B;
                funcion_1( );
```

```
        evento = e_nulo;
        }
break ;
case ESTADO_B:
if ( evento == e_evento_B_A ) {
        estado = ESTADO_A;
        funcion_2( );
        evento = eNulo;
        }
break ;
}
```

7.3. Resumen de la programación

El proceso básico programado del firmware consiste en recibir 30 caracteres (un renglón completo) por el puerto serie (estado: ESPERANDO); luego invertirlo, debido a que la impresión de los puntos se realiza del lado opuesto al que se lee.

Este renglón invertido se convierte en 3 renglones (correspondientes a las 3 filas de los caracteres braille) mediante una tabla de conversión: la librería “abcbraille.h”, creada especialmente. Esta tabla convierte cada carácter alfanumérico en un par (correspondiente a las 2 columnas de los caracteres braille) de unos o ceros dependiendo si la columna correspondiente al carácter en braille contiene o no un punto.

Finalmente se recorre cada “fila”, haciendo avanzar el motor pap de carro, y accionando el percutor sólo si el dígito leído es “1”. al finalizar los 60 dígitos (2 dígitos por carácter) se hace avanzar la hoja mediante el motor pap de avance del rodillo.

La programación detalla del firmware se encuentra en el Anexo 5.

Se hizo uso ademas de las bibliotecas de “drivers” proporcionadas por CMSIS:

- “PINSEL”, para manejo de los pines y sus funciones;
- “UART”, para la implementación de la comunicación serial
- “PWM”, para el manejo y configuración de las salidas PWM para el control de los motores y el percutor.

8. Software de usuario

El software de usuario fue desarrollado en Visual Basic, debido a que existe gran cantidad de información sobre su uso y desarrollo de aplicaciones especificas en internet, facilitando el aprendizaje del mismo. Ademas se centra en el aspecto y la presentación del software, que es lo que se busca en este punto; presentando al usuario un aspecto conocido e intuitivo de usar.

8.1. Utilización General

La pantalla principal del software trató de resumir las características mas importantes del software de la impresora comercial: dos ventanas principales, una para insertar el texto de forma tradicional (ya sea escribiendo o cortando y pegando), presentada con fondo blanco a la izquierda; y otra ventana donde se puede visualizar el texto traducido a Braille, esto es, eliminando los símbolos que no se puedan imprimir, agregando los símbolos de mayúscula ($) y número (#) y también acomodando los caracteres para que ocupen un máximo de 30 en un renglón. De esta forma el usuario puede tener una idea de como quedará la hoja impresa (el cual era un requisito propuesto). Dicha ventana se presenta con fondo naranja a la derecha. Una captura de pantalla del soft ejecutándose puede verse en la Fig.8. 1.

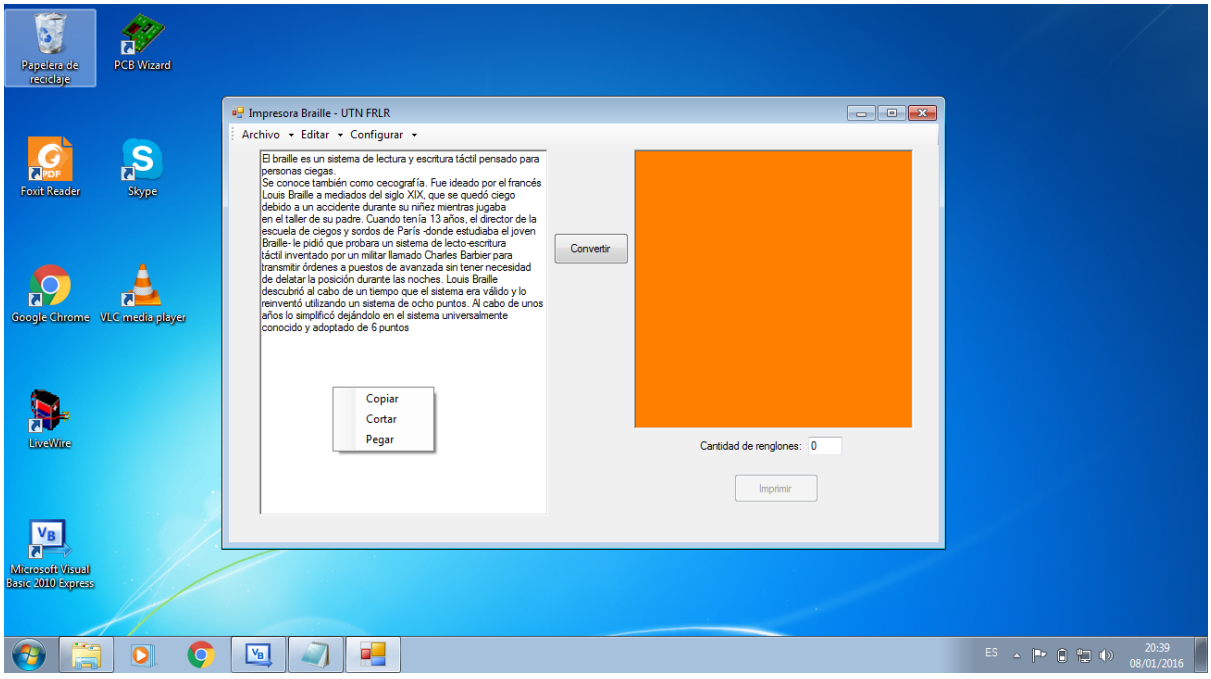


Fig.8. 1: Captura de pantalla del software de usuario

Una vez introducido el texto que se desea imprimir, el usuario debe realizar la conversión a Braille haciendo click en el botón “Convertir”, ubicado entre las dos ventanas. De esta forma aparecerá en la ventana de la derecha la previsualización del texto a imprimirse. Ver Fig.8. 2.
Debajo de la ventana de visualización se indica la cantidad de renglones a imprimirse. En cada hoja caben hasta 25 renglones El usuario deberá acomodar cada hoja en la impresora de manera manual, de a una por vez.

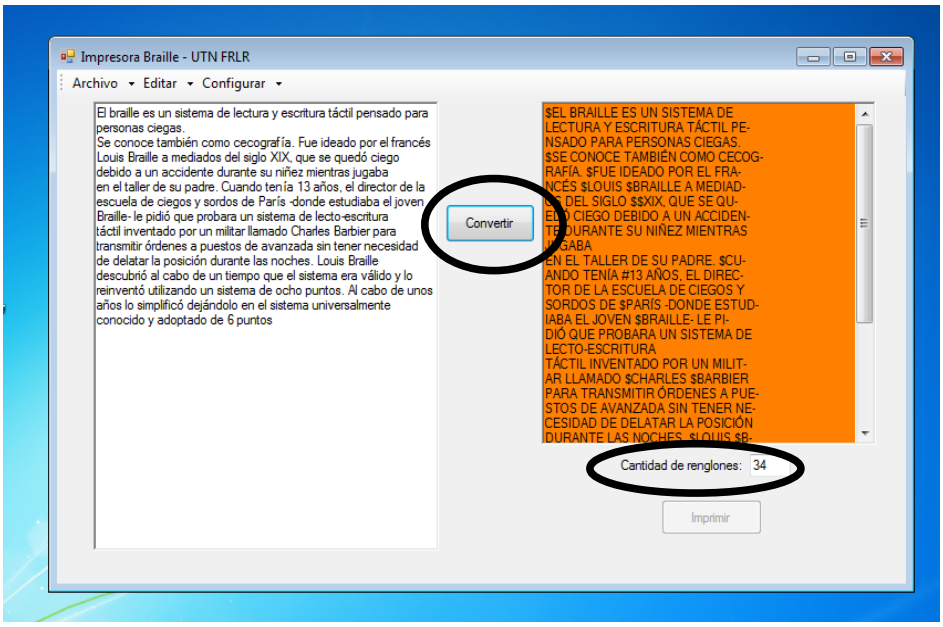


Fig.8. 2: Texto original (izq.) y texto "convertido" (der.)

Selección del puerto de impresión

En la pestaña de “configurar” se puede acceder a la ventana de configuración del puerto de impresión (Fig.8. 3).

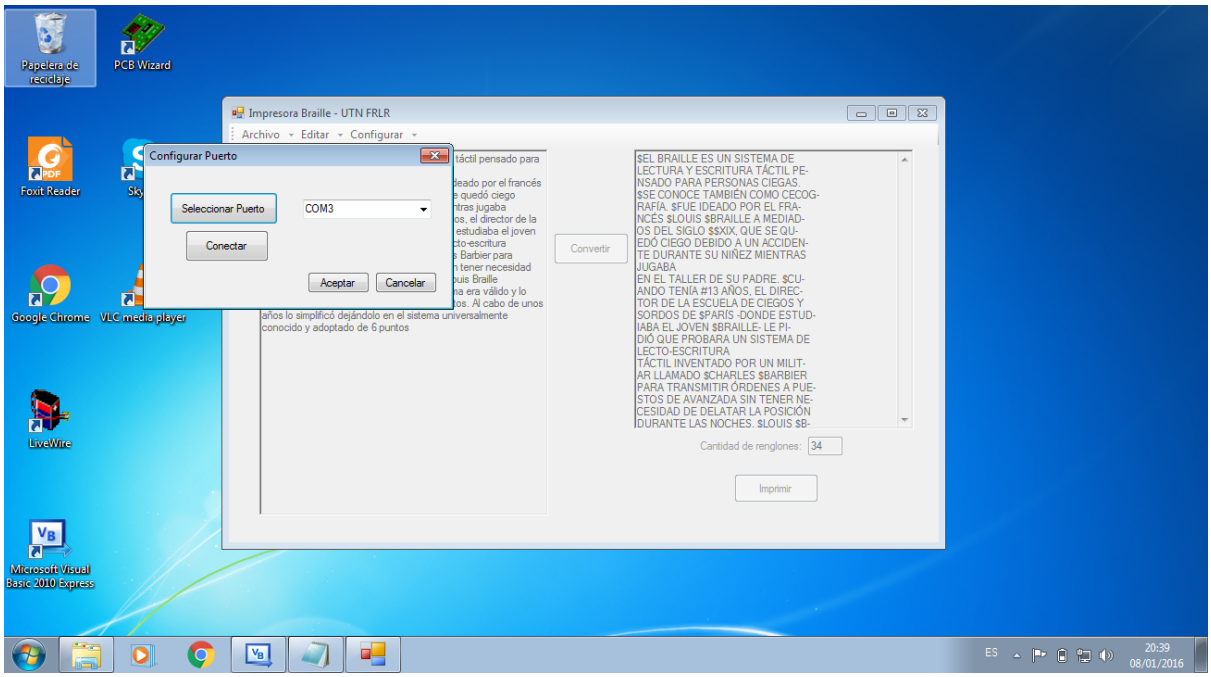


Fig.8. 3: Ventana de configuración de puerto

Haciendo click en “Seleccionar puerto” el usuario debe seleccionar aquel puerto serie virtual que se creó al conectar la impresora (recordemos que la conexión USB es con el USB – UART bridge, el cual crea un puerto virtual en la PC); y luego hacer click en “Conectar”.

De esta forma la impresora queda lista para imprimir, habilitándose el botón “Imprimir” (ubicado debajo del contador de renglones).

Ventana de Impresión

Una vez terminada la conversión y la selección del puerto de impresión, se deberá hacer click en el botón “imprimir”; de esta forma aparecerá una ventana de aviso indicando la cantidad de hojas a imprimirse y la opción de “aceptar” para comenzar la impresión; o “cancelar” para volver a la edición del texto. (Fig.8. 4)

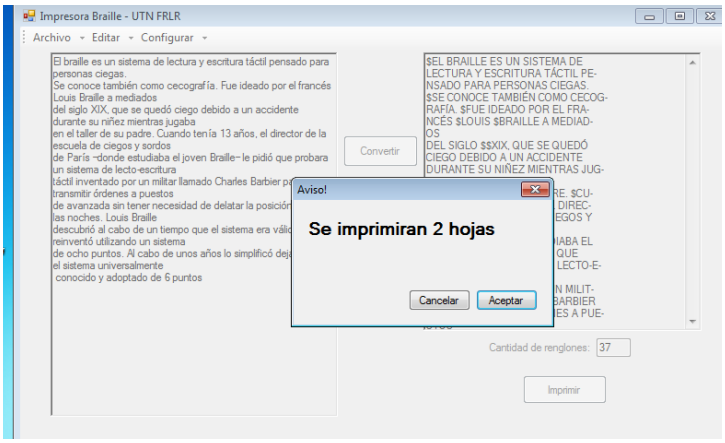


Fig.8. 4: Aviso de cantidad de hojas a imprimirse

Durante la impresión se muestra una ventana con una pequeña consola para observar la comunicación entre el software y la impresora (microcontrolador), y se presenta un botón “Cancelar”, el cual produce la interrupción inmediata de la impresión. Ver Fig.8. 5

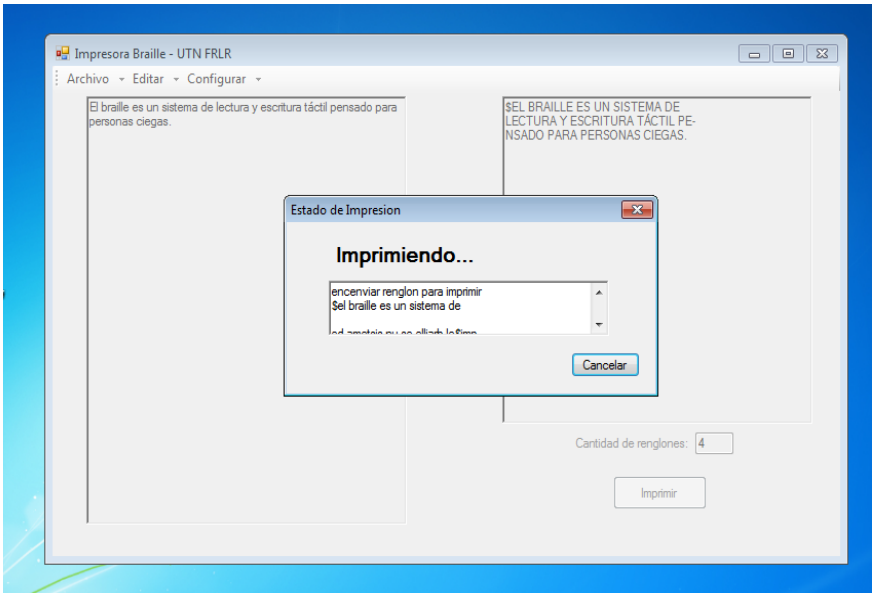


Fig.8. 5: Ventana de Impresión

Si la impresión ocupa mas de una hoja, al completar cada una se mostrará un mensaje indicando al usuario que inserte una nueva hoja en la impresora (de manera manual), para continuar con la impresión.

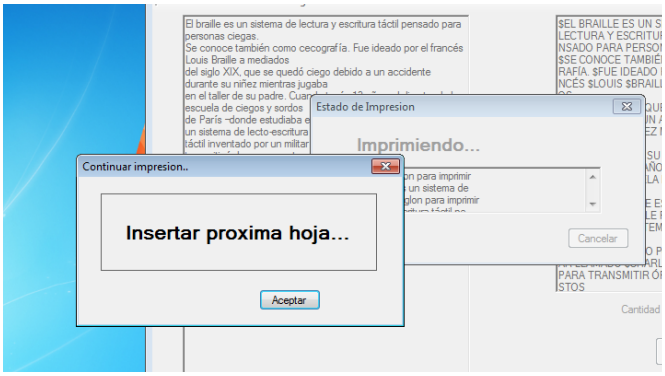


Fig.8. 6: Aviso de insertar próxima hoja

Finalmente si la impresión se llevó a cabo sin problemas se muestra un mensaje de impresión finalizada. Fig.8. 7. Presionando aceptar se vuelve a la ventana inicial.

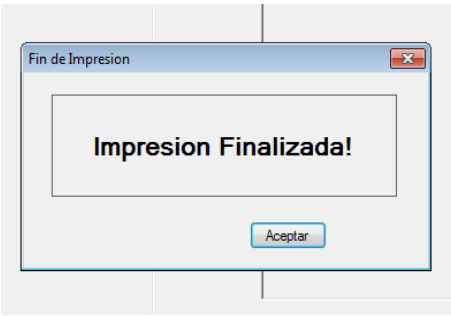


Fig.8. 7: Mensaje de Fin de Impresión

8.2. Resumen de la programación

El software comienza con la adaptación del texto convencional al texto “braille”, al pulsar el botón “convertir”; para lo cual se toma el texto contenido en la ventana izquierda, y se lo hace pasar por 4 filtros:

- Filtro de caracteres: elimina todos los caracteres no contemplados en la tabla de braille
- Filtro de mayúsculas: convierte todos los caracteres a mayúscula (en braille los caracteres minúsculos y mayúsculos tienen la misma representación) y se agrega un símbolo “$” antes de una letra mayúscula, o “$$” antes de una palabra mayúscula.
- Filtro de números: agrega un símbolo “#” antes de cada numero (los números se representan con los mismos símbolos que las letras en braille)

- Filtro de alineación de renglón: se divide el texto en renglones de 30 caracteres máximo; en caso de palabras largas, se cortan y se agrega un guion “-” de continuidad.

Finalmente este texto adaptado se muestra en la ventana principal derecha (de fondo naranja) a modo de “visualización” de la impresión.

Protocolo de Comunicación

Al presionar el botón “imprimir” se produce la comunicación con la impresora:

- Se envía el comando “enc”; y se espera recibir el mensaje “enviar renglon para imprimir” desde la impresora.
- Se envía el renglón de 30 caracteres. En caso de tener menos de 30 caracteres el renglón se completa con espacios en blanco. Se envía el comando “imp” para comenzar la impresión.
- La impresora imprime el renglón y al finalizar envía nuevamente el mensaje “enviar renglon para imprimir”, y espera el próximo renglón.
- El software busca el 2do renglón y se repite el proceso con todos los renglones.
- Luego de enviar el último renglón el software envía el comando “fin”, para indicarle a la impresora que es el ultimo renglón y que vuelva al estado de reposo.

La programación completa del software de interfaz se muestra en el Anexo 6.

8.3. Referencia de Software utilizado

Un listado del software utilizado tanto para realizar la programación del microcontrolador, como para desarrollar el software de interfaz de usuario (IU) y realizar el informe final se encuentra en el Anexo 3.

9. Ensayos y mediciones de desempeño

Una vez finalizada la etapa de desarrollo se procedió a las mediciones de desempeño (performance) y limitaciones. Las mismas permitieron descubrir aspectos erróneos o no contemplados en la planificación y realizar las correcciones y ajustes necesarias.

9.1. Calibración

El primer punto que se debió llevar a cabo antes de un ensayo de performance fue la calibración del equipo. Este proceso contempló:

- Ajuste de la altura del punzón y la plantilla para que coincidan.
- Ajuste de la distancia de “margen” hasta el primer punto. Para lo cual se corrigieron valores en la programación del microcontrolador que controla la cantidad de pasos del motor pap M1.
- Distancia entre puntos de un mismo carácter y de caracteres sucesivos. Mediante un procedimiento similar al del punto anterior.
- Ajuste de la distancia de puntos vertical. De manera similar a los puntos anteriores, pero ajustando el motor de movimiento de hoja, M2.
- Fuerza de impacto de percutor. Variando el ancho de la modulación PWM en el actuador. También se ajustó el tiempo de aplicación de la tensión.
- Frecuencia del PWM del motor M1. Se buscó una frecuencia que no produzca cabeceos, vibraciones o sobrecalentamiento excesivo.
- Velocidad del motor M1.

Los ajustes de la modulación PWM se llevaron a cabo con la ayuda de un osciloscopio marca: **Tektronix**, modelo: **TDS2022B**; del laboratorio de electrónica de la Facultad (Fig.9. 1)

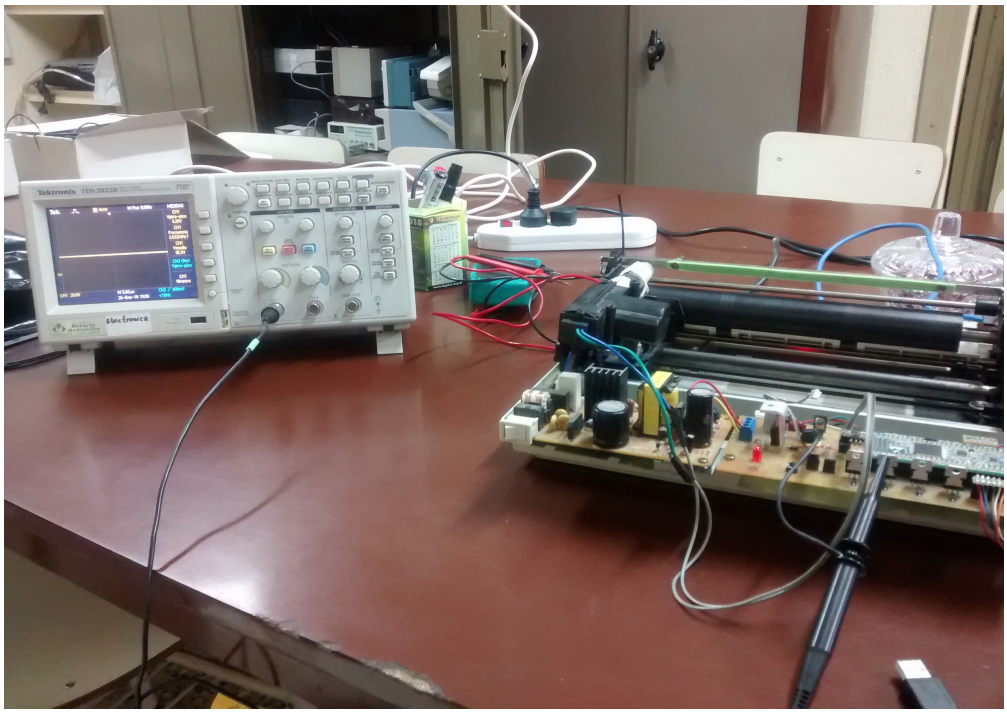


Fig.9. 1: Calibración de PWM con osciloscopio

Se evaluó posteriormente el funcionamiento del dispositivo mediante una filmación en cámara lenta del proceso; y se corroboró que todos los puntos realizados coincidieran con la plantilla.

9.2. Performance y consumo

Se llevó a cabo la impresión de varias hojas de prueba para medir el desempeño y el consumo. Los resultados se muestran a continuación:

Tabla 9.1: Mediciones de performance y consumo

Medición	Resultado
Capacidad de producción	0,5 CPS (caracteres por segundo)
Consumo de la fuente de 9V	• 150mA en “espera” (alimenta el micro y el led) • 450mA (pico) cuando se acciona el motor M2
Consumo de la fuente de 36V	• 0mA en “espera” • Entre 350 y 550mA imprimiendo (M1 + percutor)
Temperatura de motor M1	Se registró un aumento progresivo hasta 60ºC al finalizar la impresión de 10 renglones.

Por otro lado ni el motor M2 ni el percutor registran aumento de temperatura considerable. Tampoco se genera ruido excesivo (como ocurre con la impresora comercial).

Las mediciones de consumo de corriente y temperatura se llevaron a cabo con un multímetro marca **UNI-T**, modelo **UT30C**. En la Fig.9. 2 se muestra la medición de temperatura llevada a cabo.

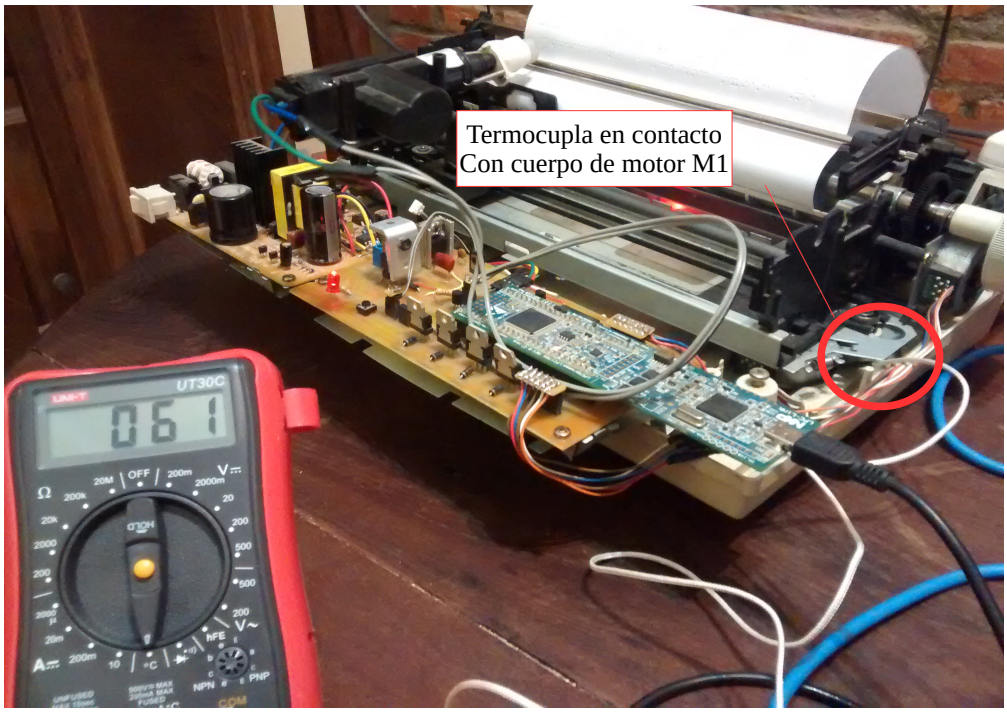


Fig.9. 2: Medición de temperatura con multímetro

9.3. Limitaciones

La Principal limitación encontrada fue el calentamiento del motor M1, a pesar de los ajustes de PWM llevados a cabo para reducirla lo mayor posible (sin perder fuerza de torque).
Se encontró información en Internet que indica que motores pap utilizados en estas condiciones pueden trabajar hasta en 80ºC sin producirse daños.
Aun así se recomienda mantener el tiempo de operación continuada en no mas de una hoja por vez; tiempo luego del cual se deberá dejar enfriar el motor por 2 o 3 minutos.
Teniendo en cuenta que en una hoja A4 se imprimen 20 renglones, y cada renglón tiene 30 caracteres, se limitaría la velocidad a 2,5 hojas A4 por hora.

La regleta y el punzón son los que definirán la calidad de los puntos impresos, un aspecto destacado del proyecto; por lo cual es importante determinar su desgaste.
Durante las pruebas se imprimieron alrededor de 200 renglones, sin percibirse una reducción de la calidad de los puntos.
Podría, por precaución reemplazarse la regleta luego de unos 1000 renglones, esto es 50 hojas A4.
El punzón al ser de acero registraría aún menos desgaste, por lo que podría recomendarse cambiarlo luego de 10 cambios de regleta; lo que representaría una duración de 300 mil caracteres.

9.4. Calidad de impresión producida

La calidad del texto producido fue evaluada con la ayuda de Leandro Paz, un joven ciego de 16 años quien aceptó formar parte de este proyecto como evaluador de la calidad.

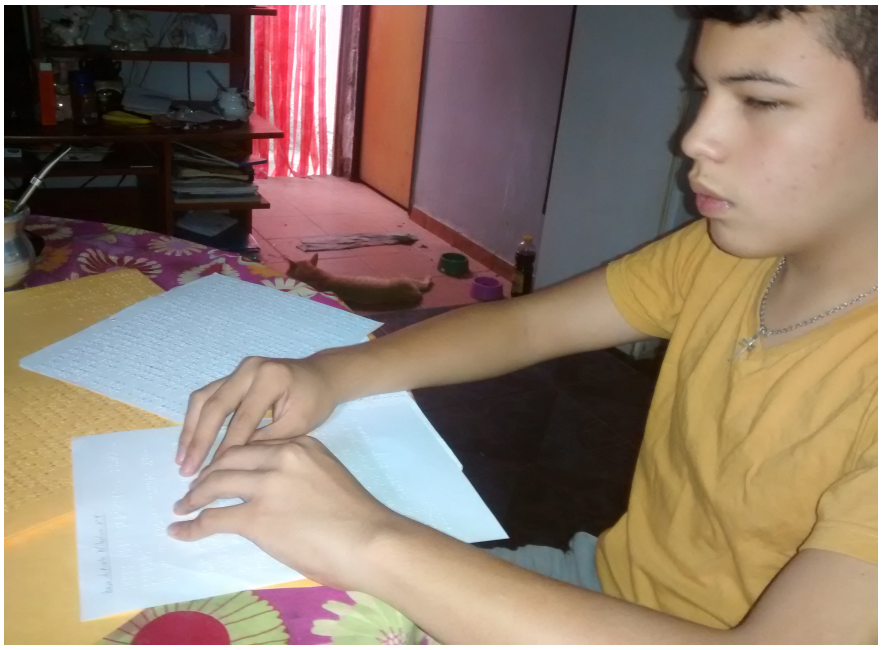


Fig.9. 3: Leandro testeando algunas hojas de prueba

Con la ayuda de Leandro se comprendieron los aspectos que hacen a la calidad de la impresión:

Papel

El factor principal que determina la calidad es el papel utilizado. Se realizaron impresiones de prueba en papel común de impresión (de 72gr.), y a pesar de mostrar una muy buena calidad en cuanto al punto realizado, y ser poder ser entendido; presentaba la dificultad de que se perdía rápidamente el relieve al ser “leído” (tocado con las yemas de los dedos); y luego de hacerlo unas 3 veces se advertía un desgaste notable del papel.
Otro inconveniente presentado fueron los puntos “rotos” producidos por impactos con fuerza excesiva sobre la hoja. Estos puntos se deshacían al leerse y desaparecían del texto.

Se realizó una prueba con papel mas grueso de 140gr., y aunque se mejoraron los aspectos mencionados anteriormente, Leandro recomendó la utilización de un papel especial para impresiones en braille: el papel MANILA.
Este papel no se encuentra en nuestra provincia y sólo se consigue en casas especializadas en Córdoba y Bs As. Para evitar la compra de una resma completa de hojas, teniendo en cuenta que solo se utilizarían algunas para impresiones de prueba, Leandro decidió donar 10 hojas para el proyecto.
Este papel prácticamente no presenta desgaste al ser leído y los puntos que se rompen por excesiva fuerza de impacto no se deshacen al pasar las yemas de los dedos; pudiendo entenderse el texto perfectamente y ser releído varias veces.

La Fig.9. 4 muestra una hoja impresa en papel manila (adelante), y una hoja escrita a mano en el mismo papel (atrás).

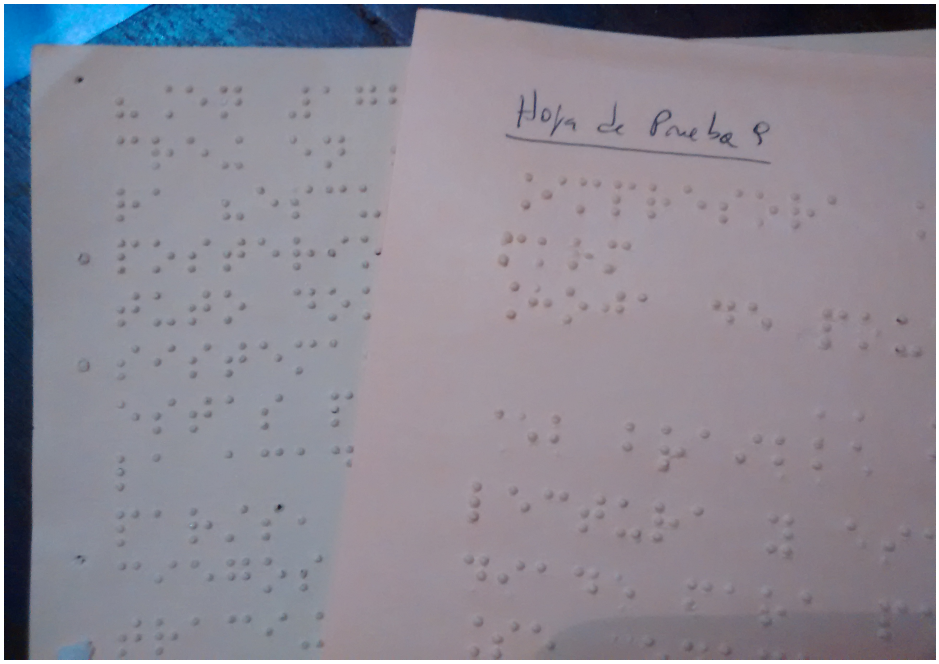


Fig.9. 4: Impresion en papel MANILA

Alineación de los puntos verticales

Un aspecto acertado del proyecto, que lo distingue de los otros mencionados, fue el de utilizar una plantilla de escritura braille para realizar los puntos; esto no solo permite una perfecta formación del punto y sus dimensiones, sino también permite que la separación horizontal sea la adecuada y lo mas importante, una alineación de los puntos verticales: Leandro comentaba que si la hoja era muy liviana se corría durante la impresión, y los puntos verticales no quedaban alineados, provocando desaciertos en la interpretación de los caracteres.

9.5. Síntesis de Performances

En la tabla 9.2 Se incluye una síntesis de los valores de performance obtenidos.

Tabla 9.2: Síntesis de performances

Característica	Performance
Caracteres por linea	30
Lineas por página A4	20
Velocidad de impresión	0,5 CPS
Rendimiento	2,5 paginas A4 por hora
Potencia consumida en operación	25W MAX

10. Costos y tiempo de trabajo

Para cerrar el trabajo presentado, se expone un resumen de los costos asociados al proyecto en cada una de sus partes. Ademas se presenta un diagrama de Gantt, mostrando el tiempo consumido por el presente proyecto.

10.1 Costos

Se detalla a continuación una lista de los elementos componentes y software utilizado, con el respectivo costo real, y en caso de ser necesario, un costo estimado para los elementos que fueron obsequiados o que ya tenia de antemano.

Los costos monetarios citados corresponden a diciembre del año 2015.

Hardware

Tabla 10.1: Costo de componentes de hardware

Componente	**Costo real ($AR.)**	**Costo estimado ($AR)**
Impresora base: Epson LX-300 (usada)	$0	$800
Percutor: Actuador eléctrico de seguro	$120	
Elementos de adaptación de percutor	$0	$300
Fuente de alimentación de repuesto	$0	$500
Motor pap em-210 de repuesto	$0	$250
Punzón braille	$50	
Pizarra braille	$120	
Cable USB impresora	$0	$60
Tornillos, tuercas, resortes y bulones varios	$0	$100
Cable de Alimentación	$60	
Papel de impresión 142gr	$0	$140 (resma 250 u.)
Total	**$350**	**$2150**

Placa electrónica

Tabla 10.2: Costo de componentes de placa electrónica de control

Componente	**Costo ($AR)**
Microcontrolador: NXP LPCXpresso	$1500
Cable USB – Micro usb (para programar)	$60
Adaptador USB – UART bridge	$250
Pertinax 10x20 cm	$30
Bornera, zócalo, micro-switch y tiras de pines	$75
CI: ULN2003A, 7805, disipador	$50
Semiconductores individuales: diodos 1N4007, LED; transistores BC875, BD438, TIP110, TIP125	$100

Elementos pasivos ind.: Capacitores y resistencias varias, cables y estaño	$30
Total	**$2095**

Software

Para la programación del microcontrolador, edición del informe final, confección del esquemático del circuito electrónico y confección de los planos de piezas realizadas se utilizaron recursos de software libre; mientras que para la programación de la interfaz de usuario, y confección de la placa de circuito impresa se utilizó software privativo.

Tabla 10.3: Costo de software de desarrollo

Software	Costo
Linux (Ubuntu)	$0
LPCXpresso IDE	$0
Libre Office	$0
kiCad	$0
LibreCad	$0
Windows 7	$?
Visual Basic 2010 Express	$0 (licencia gratuita)
PCB Wizard	$0 (versión gratuita)
Total	**$0**

10.2. Tiempo de Trabajo

Se presenta a continuación un diagrama de Gantt indicando la distribución y el tiempo consumido en cada etapa del proyecto.

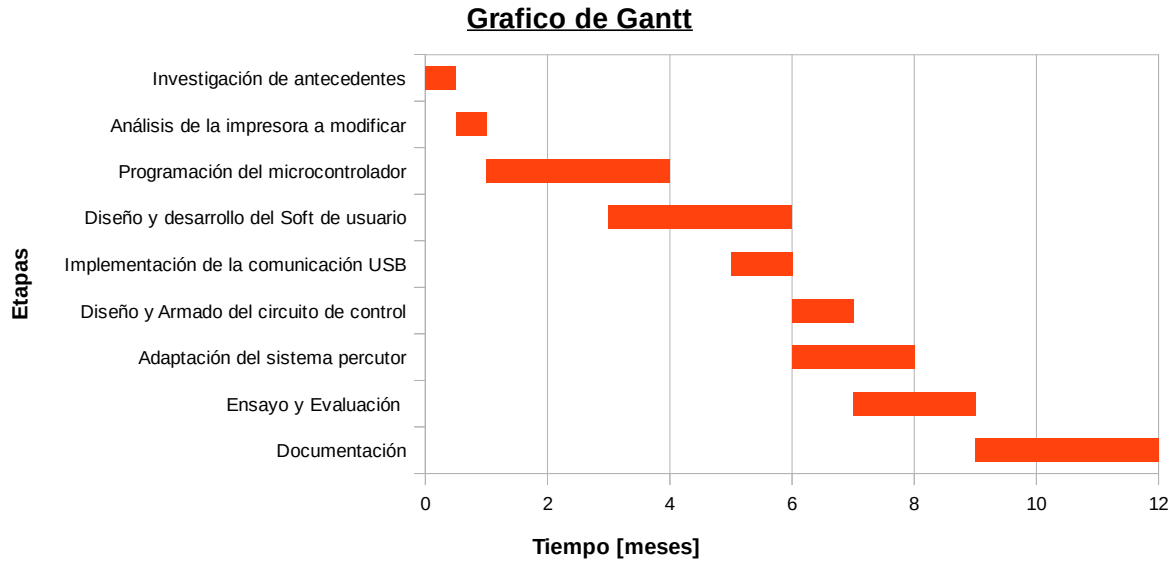


Fig.10. 5: Grafico de Gantt

11. Conclusión

El presente proyecto nace con la necesidad de tener una mayor inclusión de las personas ciegas, facilitándoles material de lectura impreso, principalmente para educación; mediante la impresión de textos en formato braille.

Se cumplió con el objetivo propuesto de la construcción de un modelo funcional de impresora braille, utilizando elementos de alta disponibilidad y accesibilidad, ayudando a reciclar dispositivos en desuso e implementando un software de interfaz de usuario amigable y sencillo de utilizar. Todo esto posibilita que su uso sea similar a la de cualquier otra impresora para el usuario final.

Los conocimientos adquiridos a lo largo de la carrera de ingeniería electrónica permitieron el desarrollo de un circuito de control gobernado por un microcontrolador actual y una interfaz de comunicación USB con la PC; ademas el desarrollo de una interfaz moderna para el manejo del mismo.

El punto menos favorable de realizar fue la adaptación de un sistema percutor, el cual pudo lograrse gracias a la ayuda de un familiar.

Como puntos a mejorar en el futuro y acercarse a un modelo de producción se plantean:

- Construcción de una carcasa adecuada que proteja y envuelva todo el dispositivo.
- Desarrollo de una Interfaz de usuario multiplataforma (Linux, Windows y Mac).
- Implementación de un protocolo de comunicación estandarizado por puerto USB. Esto incluye un reconocimiento como dispositivo “impresora” por parte de la PC y utilización de archivos de “lenguaje de descripción de paginas” (PDL) braille estándares.
- Mejora del sistema percutor, mediante desarrollo de un cabezal con múltiples punzones, e incrementar la velocidad de impresión.
- Reducción de la temperatura de trabajo del motor de avance de carro, mediante incorporación de un ventilador, para aumentar el tiempo de uso continuado.
- Capacidad de imprimir gráficos en relieve.
- Desarrollo de un sistema de alimentación de papel de hoja suelta (como el de las impresoras modernas).
- Incorporación de una interfaz de comunicación auditiva para la manipulación por parte de una persona ciega.

Se espera que el proyecto y las mejoras propuestas puedan ser aprovechadas para desarrollar modelos de producción de bajo costo y poder implementar impresoras braille en centros educativos y casas de familias donde se necesiten; y de esta forma acercar más a las personas ciegas a la lectura.

Parte IV:
Anexos

Anexo 1: Planos de piezas realizadas

En este anexo se muestran planos con la forma y dimensiones de las piezas realizadas para la adaptación del sistema de percutor a la impresora original[1].

1.1 Soporte para percutor

En la Fig.A1. 1 se muestra un plano de la pieza realizada para sostener el percutor en su lugar.

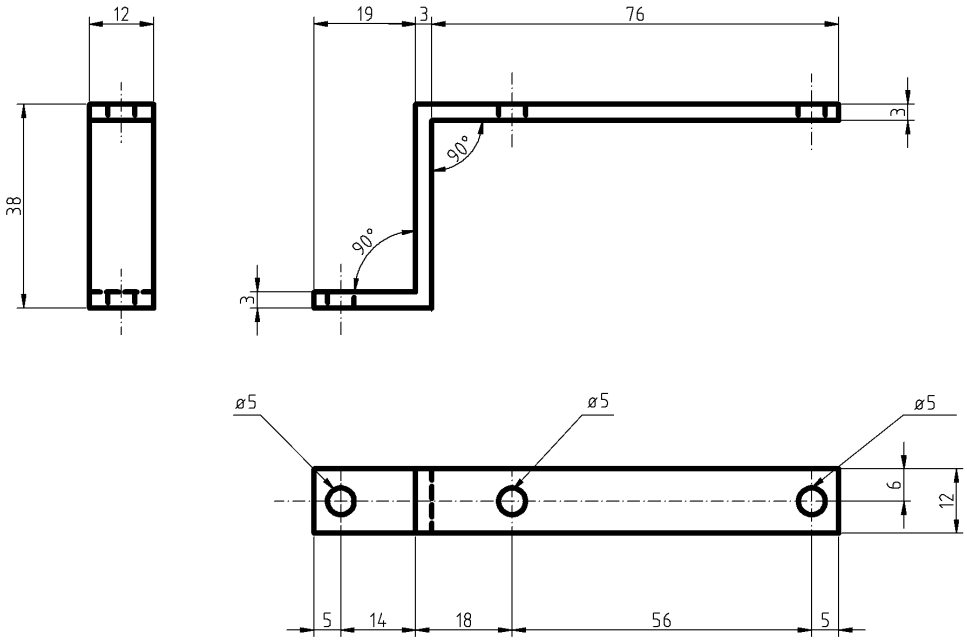


Fig.A1. 1: Soporte para percutor

La pieza fue realizada en hierro.
Los dos agujeros en la parte superior coinciden con los agujeros de sujeción del actuador. Este se fijó con dos tornillos M4x1,5x15 al soporte.
El agujero en la parte inferior se utiliza para sujetar el soporte al carro portacabezal de la impresora con un bulón M5x1,5x25 y dos tuercas (una para fijar el bulón al carro y otra para fijar el soporte al bulón).

Lo ideal hubiera sido utilizar dos bulones para sujetar el soporte al carro portacabezal, evitando así cualquier posibilidad de rotación no deseada del soporte, pero se dejó de esta forma por motivos de simplicidad, espacio reducido en el carro y porque en funcionamiento normal, este único punto de anclaje fue suficiente para mantener el percutor firmemente en su posición. Ademas se prevé la posibilidad de re-ajuste de la posición del actuador en caso de ser necesario en el futuro.

---

1. *NOTA: Todas las cotas de las figuras están indicadas en milímetros (mm).*

1.2. Punta del percutor

En la Fig.A1. 2 se muestra el plano con la forma y dimensiones de la pieza utilizada como punta del percutor. La misma consiste en un cilindro “adaptador” para sostener la punta del punzón braille. El cilindro adaptador está hecho de teflón, con dos orificios: uno para sostener el cilindro a la punta del actuador con un bulón M4x1,5x25 y tuerca; y otro para sostener la punta del punzón al cilindro, mediante un tornillo M4x1,2x15 y una tuerca.
La punta fue extraída del punzón original para escribir a mano, de esta forma las medidas son idénticas y la marca que deja en el papel también. El material de la punta es acero.

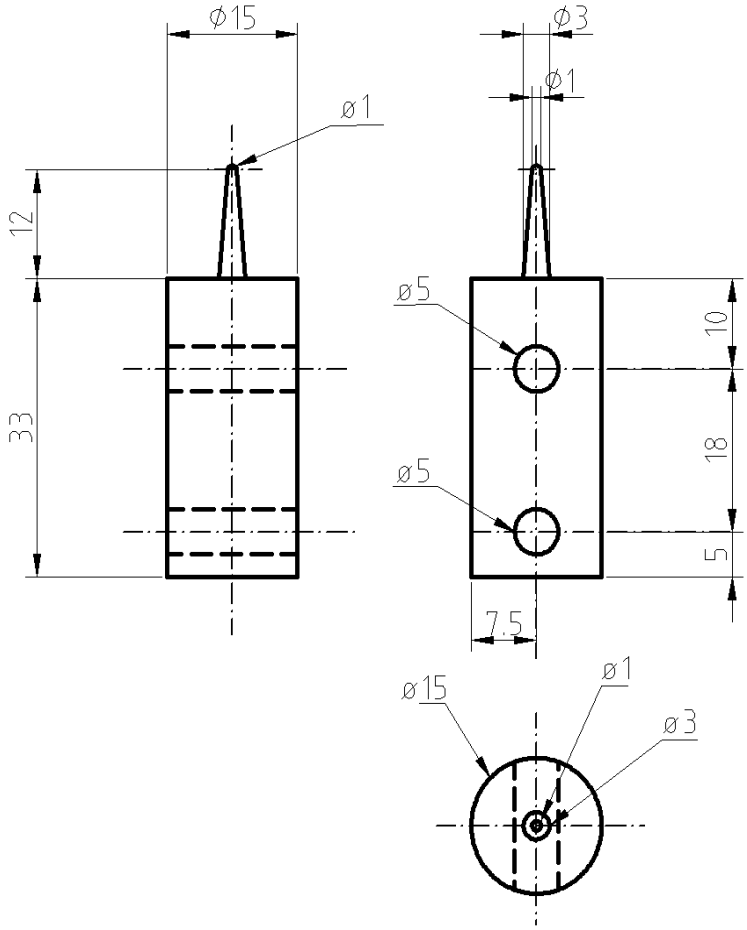


Fig.A1. 2: Punta del percutor

Ademas se agregaron dos resortes de 30mm de longitud y 3mm de diámetro, sujetados a los tornillos del cilindro en un extremo, y al actuador en el otro, para provocar el retroceso automático del vástago luego de un golpe del percutor.

1.3. Yunque para percutor

Para realizar el yunque que recibirá los golpes del percutor se utilizó una pieza extraída de otra impresora. La forma y dimensiones se muestran en la Fig.A1. 3. La misma está hecha de acero.
La pieza se sujetó a la impresora por los extremos con resortes de 25mm de longitud y 5mm de diámetro.
Los agujeros se utilizaron para sujetar la pieza de la pizarra, con tornillos sin cabeza M4x15mm.

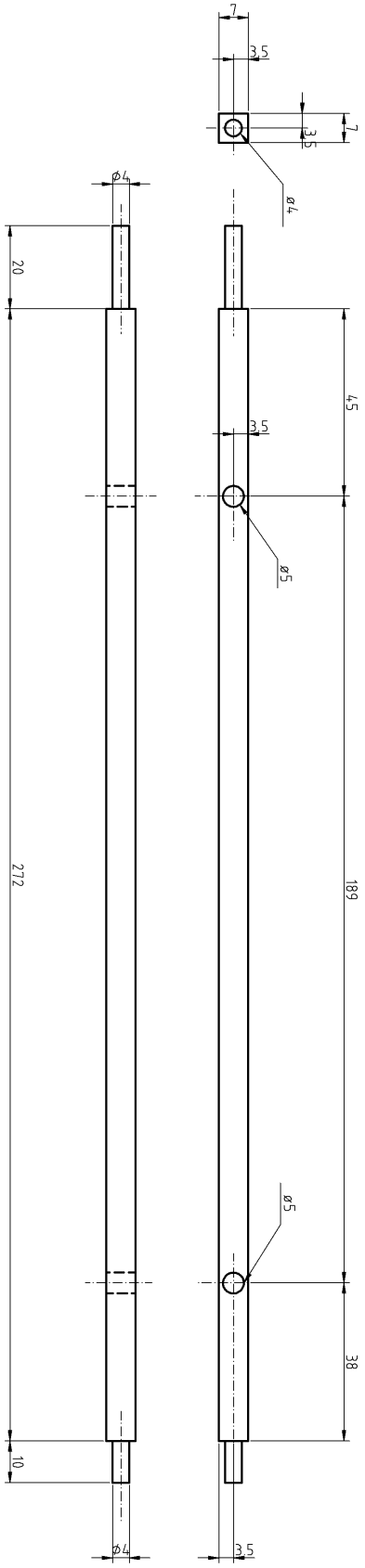


Fig.A1. 3: Yunque para percutor

Anexo 2: Cálculos de dimensionamiento del circuito electrónico

En el presente anexo se muestran los cálculos realizados para dimensionar los componentes del circuito electrónico de control.

2.1. Motor de avance de hoja EM-211 (M2)

Primero se realizó una medición de la resistencia que presenta cada bobina individual del motor, con un tester; a partir de la cual, y conociendo la tensión nominal de alimentación, se pudo calcular la corriente necesaria por bobina:

$$R_b = 30\,\Omega \quad ; \quad I_b = \frac{V_{nominal}}{R_b} = \frac{9\,V}{30\,\Omega} = 0{,}3\,A$$

La corriente necesaria, al ser menor a 0,5A, permite utilizar un driver integrado (un array de transistores y diodos, que por lo general manejan corrientes de hasta 0,5A), permitiendo ahorrar espacio en la placa.
El driver utilizado es el ULN2003A, que tiene una capacidad de manejo de corriente de 0,5A por canal. Ver hoja de datos anexada al final del documento para mas información.

En el caso del transistor elegido para controlar la desconexión de la alimentación, se debe tener en cuenta que durante el funcionamiento se accionan dos bobinas por paso; por lo tanto la corriente en cada momento será:

$$I_s = 2 \times I_b = 2 \times 0{,}3\,A = 0{,}6\,A$$

Se buscó un transistor PNP que soportara por lo menos el doble de ésta corriente (por ocurrencia de picos en cargas inductivas).
El transistor encontrado fue el **BD438**, que maneja una corriente de 4A max.

Debido a que en saturación este transistor consume una corriente de base de 200mA, se decidió añadir un transistor de paso, para el manejo de dicha corriente: el **BC875**.
El BC875 maneja una corriente de hasta 1A y se excita con una corriente de base muy pequeña: 0,5mA en saturación.
El microcontrolador maneja hasta 40mA por pin.

2.2. Motor de movimiento de carro EM-210 (M1)

En el caso de este motor, la medición de bobinas y corriente nominal resultó:

$$R_b = 19\,\Omega \quad ; \quad I_b = \frac{V_{nominal}}{R_b} = \frac{9\,V}{19\,\Omega} = 0{,}47\,A$$

En este caso, la corriente nominal es muy alta como para utilizar un CI como en el caso anterior.
El driver debe realizarse con transistores individuales.

Se debe tener en cuenta que el manejo de este motor se realiza con la técnica de PWM, por lo tanto la corriente nominal será la corriente promedio, pero el transistor debe soportar una tensión pico de 36V (proveniente de la fuente original), y una corriente pico de:

$$I_P = \frac{V_s}{R_b} = \frac{36\,V}{19\,\Omega} = 1{,}89\,A$$

Ademas de trabajar en la frecuencia de conmutación de 25KHz.

El transistor elegido fue el **TIP110**, el cual soporta una corrientecontinua de hasta 2A, y de 4A pulsante. Al tener una configuración Darlighton la corriente de base es muy pequeña: 50mA máximo y 8mA en saturación (hfe mínimo de 1000). Por lo tanto se puede manejar directamente con la salida del microcontrolador (hasta 40mA por pin).
Ademas, soporta una tensión Vce de 60V, y trabaja bien en frecuencias de alrededor de 100KHz.

Los diodos volante se dimensionaron para soportar la corriente nominal de 0,47A. Se eligió el modelo **1N4007** que maneja una corriente directa de 1A, y soporta una tensión inversa de hasta 1000V.

2.3. Percutor

En el caso del percutor, recordemos que es básicamente un motor de corriente continua; para el dimensionamiento del transistor a utilizar se procedió con la medición de la corriente nominal con un tester:

$$I_N = 3\,A$$

Se eligió entonces para controlar el percutor el transistor **TIP125**, que maneja una corriente continua de hasta 5A, y picos de hasta 8A. Ademas soporta una Vce de 60V, recordando que debe soportar los picos de la tensión de fuente de 36V en la modulación PWM.

La corriente de base en saturación es de sólo 12mA (para Ic=3A), al tratarse de un transistor en configuración Darlighton. Aun así se decidió por precaución agregar un transistor de paso, que al igual que en el caso anterior es el **BC875**.

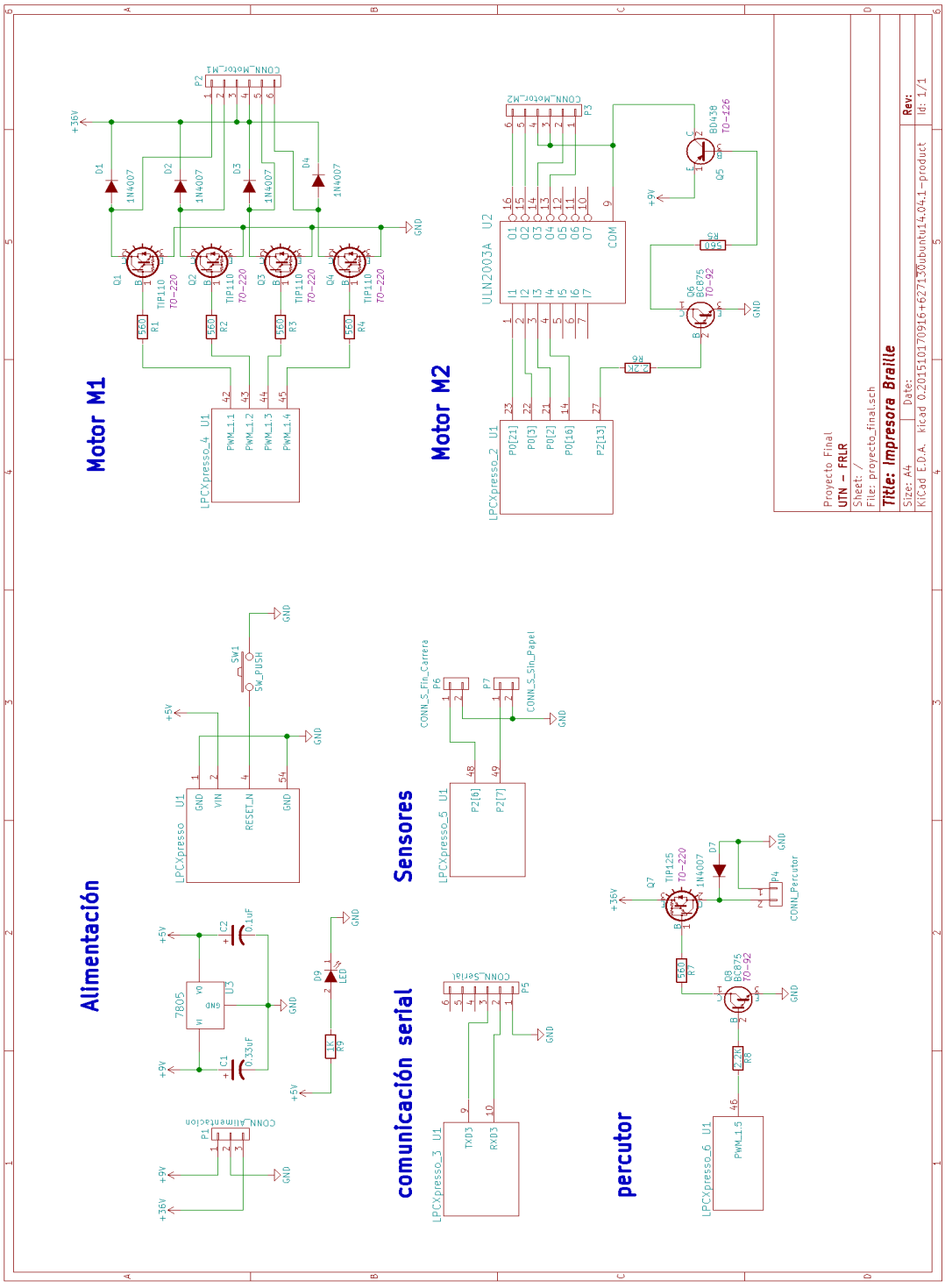
La modulación de PWM se fue variando progresivamente hasta alcanzar una fuerza de impacto satisfactoria.

Anexo 3: Lista de Software utilizado

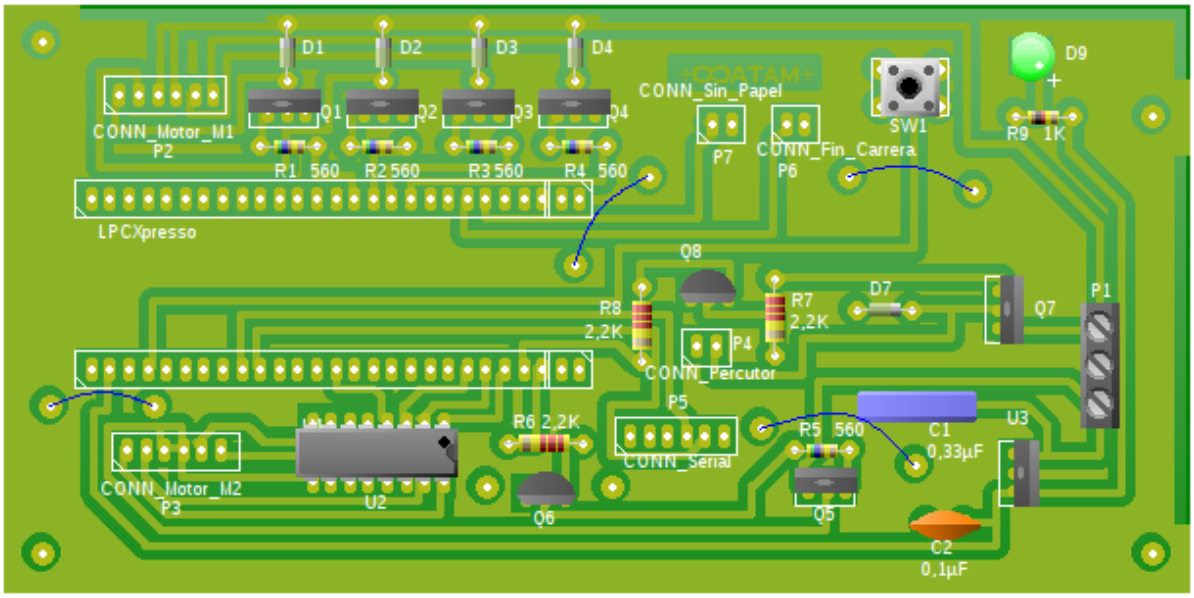
Software	**Utilización**
Ubuntu 14.04 LTS (Linux)	Sistema Operativo principal
LPCXpresso IDE v7.4.0	IDE para programación del microcontrolador LPCXpresso
Libre Office v4.2	Edición del Informe definitivo
KiCad v4.0.1	Creación del Circuito esquemático
LibreCad v2.0.2	Creación de Planos de piezas de adaptación
Windows 7	Sistema Operativo secundario
Visual Basic 2010 Express	Programación del software de interfaz de usuario
PCB Wizard 4.0	Diseño de PCB de plaqueta electrónica

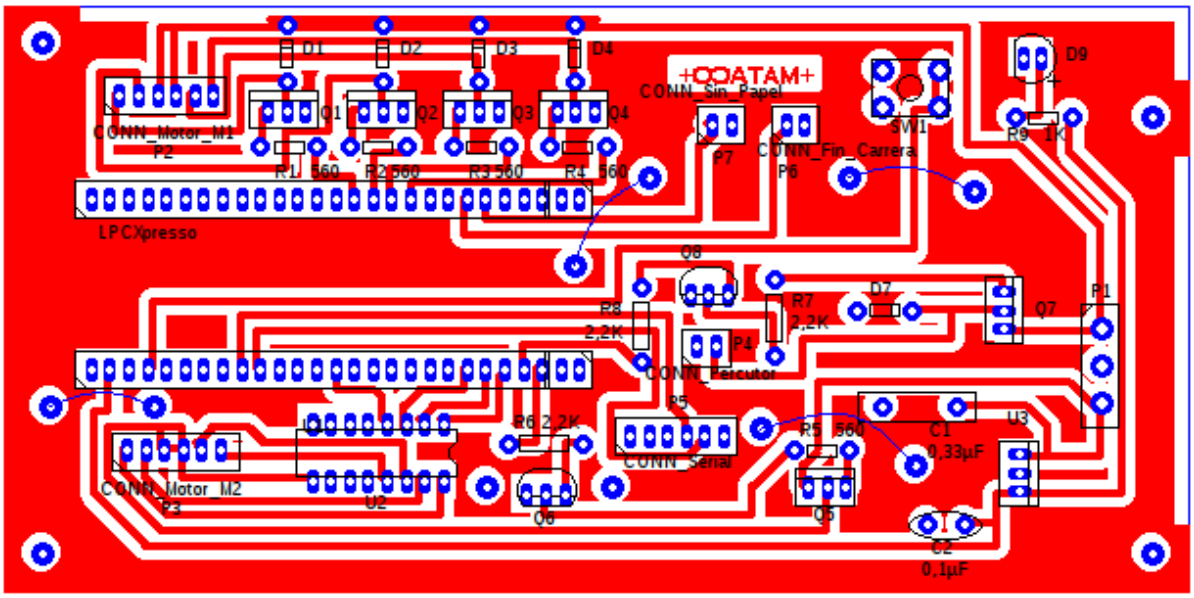
Anexo 4: Circuito Electrónico

Esquemático



Placa PCB





Componentes de la Placa Electrónica

Nombre	Componente
Placa PCB	Pertinax de 10x20cm
U1	Microcontrolador LPCXpresso
U2	ULN2003A (driver de motor M2)
U3	LM7805
U4	USB – UART Bridge (a traves de conector P5)
P1	Bornera de 3 cables (conector de alimentación)
P2, P3, P4, P5, P6, P7	Tira de pines hembra. (conectores de: Motor M1, Motor M2,

	Percutor, USB – UART Bridge, Sensor fin de carrera, sensor sin hojas).
Q1, Q2, Q3, Q4	Transistores TIP110 (TO-220)
Q5	Transistor BD438 (TO-126)
Q6, Q8	Transistores BC875 (TO-92)
Q7	Transistor TIP125 (TO-220)
D1, D2, D3, D4, D7	Diodos 1N4007
D9	Diodo LED 5mm Rojo
SW1	Pulsador micro-switch 6mm
R1, R2, R3, R4, R5, R7	Resistencias 560 Ω - 0,25W
R6, R8	Resistencias 2,2 KΩ – 0,5W
R9	Resistencia 1KΩ – 0,25W
C1	Capacitor de polyester 0,33µF - 50V
C2	Capacitor cerámico 0,1µF - 500V

Anexo 5: Programación del microcontrolador (firmware)

La programación del microcontrolador se realizó en lenguaje “C”, utilizando el software “LPCXpresso IDE”, sobre la plataforma Linux (Ubuntu).

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include "lpc17xx_uart.h"
#include "lpc17xx_pinsel.h"
#include "lpc17xx_pwm.h"
#include "lpc_types.h"

#include <string.h>
#include <stdio.h>
#include "abcbraille.h"
#include "lib_imprimir3.h"

#include <cr_section_macros.h>
#include <NXP/crp.h>

void UART3_IRQHandler (void);


__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;

enum estados_imp {ENCENDIDA, ESPERANDO, IMPRIMIENDO};
typedef enum estados_imp estados_imp_t;


#define estado_ini ENCENDIDA

/*enum eventos_imp {e_nulo, e_iniciar, e_int, e_renglon};
typedef enum eventos_imp eventos_imp_t;*/
eventos_imp_t evento;

void inicializar_imp (void);
void maq_est_imp(void);
void encender(void);
void esperar_imp(void);
void interrumpir(void);

char c_UART;
char rengbra1[60], rengbra2[60], rengbra3[60];
int b_int=0; //bandera de interrupcion
int b_enc = 0; //bandera de encendido
int b_renglon = 0; //bandera de renglon
int b_fin = 0; // bandera de fin de impresion

estados_imp_t estado;
```

```
int main(void) {

	uint8_t temp;

	PWM_TIMERCFG_Type PWMCfgDat;
	PWM_MATCHCFG_Type PWMMatchCfgDat;

	PINSEL_CFG_Type PinCfg;
	UART_CFG_Type UARTConfigStruct;

	LPC_GPIO0->FIODIR |= (1 << 16); //salida motor2_A
	LPC_GPIO0->FIODIR |= (1 << 2); //salida motor2_B
	LPC_GPIO0->FIODIR |= (1 << 3); //salida motor2_C
	LPC_GPIO0->FIODIR |= (1 << 21); //salida motor2_D

	//LPC_GPIO0->FIODIR |= (1<<22); //salida percutor

	LPC_GPIO2->FIODIR |= (1<<13); //salida habilitar motor_2

	//configurar pines de entrada (sensor)
	PinCfg.Funcnum = PINSEL_FUNC_0;
	PinCfg.OpenDrain = PINSEL_PINMODE_NORMAL;
	PinCfg.Pinmode = PINSEL_PINMODE_PULLUP;
	PinCfg.Pinnum = PINSEL_PIN_6;
	PinCfg.Portnum = PINSEL_PORT_2;

	PINSEL_ConfigPin(&PinCfg);

	/*//configurar pines salida habilitar motor_2
	PinCfg.Funcnum = PINSEL_FUNC_0;
	PinCfg.OpenDrain = PINSEL_PINMODE_NORMAL;
	PinCfg.Pinmode = PINSEL_PINMODE_PULLDOWN;
	PinCfg.Pinnum = PINSEL_PIN_27;
	PinCfg.Portnum = PINSEL_PORT_0;

	PINSEL_ConfigPin(&PinCfg);*/

	//configurar pines de UART 3
	PinCfg.Funcnum = PINSEL_FUNC_2;
	PinCfg.OpenDrain = PINSEL_PINMODE_NORMAL;
	PinCfg.Pinmode = PINSEL_PINMODE_TRISTATE;
	PinCfg.Pinnum = PINSEL_PIN_0;
	PinCfg.Portnum = PINSEL_PORT_0;

	PINSEL_ConfigPin(&PinCfg);

	PinCfg.Pinnum = PINSEL_PIN_1;

	PINSEL_ConfigPin(&PinCfg);

	// Configuramos la UART
	UARTConfigStruct.Baud_rate = 115200;
	UARTConfigStruct.Databits = UART_DATABIT_8;
```

```
UARTConfigStruct.Parity = UART_PARITY_NONE;
UARTConfigStruct.Stopbits = UART_STOPBIT_1;

// Inicializamos la UART
UART_Init(LPC_UART3, &UARTConfigStruct);

/* Initialize PWM peripheral, timer mode
 * PWM prescale value = 1 (absolute value - tick value) */
PWMCfgDat.PrescaleOption = PWM_TIMER_PRESCALE_USVAL;
PWMCfgDat.PrescaleValue = 40;
PWM_Init(LPC_PWM1, PWM_MODE_TIMER, (void *) &PWMCfgDat);

/* Initialize PWM pin connect */
PinCfg.Funcnum = 1;
PinCfg.OpenDrain = 0;
PinCfg.Pinmode = PINSEL_PINMODE_PULLDOWN;
PinCfg.Portnum = 2;
for (temp = 0; temp <= 4; temp++)          //activo desde 1.1 hasta 1.5 (percutor)
{
        PinCfg.Pinnum = temp;
        PINSEL_ConfigPin(&PinCfg);
}
/* Set match value for PWM match channel 0 = 256, update immediately */
PWM_MatchUpdate(LPC_PWM1, 0, 25, PWM_MATCH_UPDATE_NOW);
/* PWM Timer/Counter will be reset when channel 0 matching Enable interrupt when match
 * no stop when match */
PWMMatchCfgDat.IntOnMatch = ENABLE;
PWMMatchCfgDat.MatchChannel = 0;
PWMMatchCfgDat.ResetOnMatch = ENABLE;
PWMMatchCfgDat.StopOnMatch = DISABLE;
PWM_ConfigMatch(LPC_PWM1, &PWMMatchCfgDat);

/* Configure each PWM channel: -------------------------------------------- */
/* - Single edge */
for (temp = 2; temp < 6; temp++)
{
        PWM_ChannelConfig(LPC_PWM1, temp, PWM_CHANNEL_SINGLE_EDGE);
}

/* Disable PWM interrupt */
NVIC_DisableIRQ(PWM1_IRQn);
/* preemption = 1, sub-priority = 1 */
NVIC_SetPriority(PWM1_IRQn, ((0x01<<3)|0x01));

/* Configure match value for each match channel */
for (temp = 1; temp < 5; temp++)
{
        /* Set up match value */
        PWM_MatchUpdate(LPC_PWM1, temp, 5, PWM_MATCH_UPDATE_NOW);
        /* Configure match option */
        PWMMatchCfgDat.IntOnMatch = DISABLE;
        PWMMatchCfgDat.MatchChannel = temp;
        PWMMatchCfgDat.ResetOnMatch = DISABLE;
        PWMMatchCfgDat.StopOnMatch = DISABLE;
        PWM_ConfigMatch(LPC_PWM1, &PWMMatchCfgDat);
```

```
	}

	//configuro el pwm para percutor

	PWM_MatchUpdate(LPC_PWM1, 5, 6, PWM_MATCH_UPDATE_NOW);
	/* Configure match option */
	PWMMatchCfgDat.IntOnMatch = DISABLE;
	PWMMatchCfgDat.MatchChannel = 5;
	PWMMatchCfgDat.ResetOnMatch = DISABLE;
	PWMMatchCfgDat.StopOnMatch = DISABLE;
	PWM_ConfigMatch(LPC_PWM1, &PWMMatchCfgDat);



	//inicializar la maquina de estados
	inicializar_imp();

	while(1) {

		//correr la maquina de estados
			maq_est_imp();

	}
	return 0 ;
}

//INTERRUPCIONES
void UART3_IRQHandler (void)
{
	//recibo caracter
	c_UART = UART_ReceiveByte(LPC_UART3);

	//envio caracter
	UART_SendByte(LPC_UART3, c_UART);
	if (estado == ENCENDIDA)
	{
		switch(b_enc)
		{
		case 0:
			if(c_UART=='e')
				b_enc=1;
			break;

		case 1:
			if(c_UART=='n')
				b_enc=2;
			else
				b_enc=0;
			break;

		case 2:
			if(c_UART=='c')
				{b_enc=0;
				evento=e_iniciar;
				}
```

```
                else
                        b_enc=0;
                break;
        }
}

if (estado == ESPERANDO)
{
        switch(b_renglon)
        {
        case 0:
                if(c_UART=='i')
                        b_renglon=1;
                break;

        case 1:
                if(c_UART=='m')
                        b_renglon=2;
                else
                        b_renglon=0;
                break;

        case 2:
                if(c_UART=='p')
                        {b_renglon=0;
                        evento=e_renglon;
                        }
                else
                        b_renglon=0;
                break;
        }
}

if (estado == IMPRIMIENDO)
{
        switch(b_int)
        {
        case 0:
                if(c_UART=='i')
                        b_int=1;
                break;

        case 1:
                if(c_UART=='n')
                        b_int=2;
                else
                        b_int=0;
                break;

        case 2:
                if(c_UART=='t')
                        {b_int=0;
                        evento=e_int;
                        }
                else
```

```
                        b_int=0;
                    break;
                }

                switch(b_fin)
                {
                case 0:
                    if(c_UART=='f')
                        b_fin=1;
                    break;

                case 1:
                    if(c_UART=='i')
                        b_fin=2;
                    else
                        b_fin=0;
                    break;

                case 2:
                    if(c_UART=='n')
                        {b_fin=0;
                        evento=e_fin;
                        }
                    else
                        b_fin=0;
                    break;
                }
        }
}


void inicializar_imp (void)
{
        estado = estado_ini;
        evento = e_nulo;
        encender();
}

void maq_est_imp(void)
{
        /* Primero valido la variable de estado */
        if (estado >= 3)
        {
                inicializar_imp();
                return ;
        }
        switch (estado)
        {
                case ENCENDIDA:
                if ( evento == e_iniciar )
                {
                        evento = e_nulo ;
                        estado = ESPERANDO;
                        NVIC_DisableIRQ(UART3_IRQn); //Deshabilita interrupcion  de uart 3 en el
NVIC
```

```
                        esperar_imp();
                        NVIC_EnableIRQ(UART3_IRQn); //Habilita interrupcion  de uart 3 en el NVIC
                }
                break ;

                case ESPERANDO:
                if (evento == e_renglon)
                {
                        estado = IMPRIMIENDO;
                        evento = e_nulo ;
                        //UART_IntConfig(LPC_UART3, UART_INTCFG_RBR, ENABLE); //Habilita
interrup UART
                        imprimir(rengbra1, rengbra2, rengbra3, &evento);
                }
                break ;

                case IMPRIMIENDO:
                if (evento == e_iniciar)
                {
                        NVIC_DisableIRQ(UART3_IRQn); //Deshabilita interrupcion  de uart 3 en el
NVIC
                        evento = e_nulo ;
                        estado = ESPERANDO;
                        esperar_imp();
                        NVIC_EnableIRQ(UART3_IRQn); //Habilita interrupcion  de uart 3 en el NVIC
                }

                if (evento == e_fin)
                {
                        NVIC_DisableIRQ(UART3_IRQn); //Deshabilita interrupcion  de uart 3 en el
NVIC
                        evento = e_nulo ;
                        estado = ENCENDIDA;
                        encender();
                }

                if (evento == e_int)
                {
                        estado = ENCENDIDA;
                        evento = e_nulo;
                        NVIC_DisableIRQ(UART3_IRQn); //Deshabilita interrupcion  de uart 3 en el
NVIC
                        interrumpir();
                        NVIC_EnableIRQ(UART3_IRQn); //Habilita interrupcion  de uart 3 en el NVIC
                }
                break ;
        }

}

void encender(void)
{
        const char welcome[]="Impresora Braille UTN FRLR \r\n";

        UART_TxCmd(LPC_UART3, ENABLE);
```

```
	// Mensaje de bienvenida
	UART_Send(LPC_UART3, welcome, sizeof(welcome), BLOCKING);

	UART_IntConfig(LPC_UART3, UART_INTCFG_RBR, ENABLE); //Habilita interrup UART
	NVIC_EnableIRQ(UART3_IRQn); //Habilita interrupcion  de uart 3 en el NVIC

	return;
}

void esperar_imp(void)
{
	char listo[]="enviar renglon para imprimir \r\n";
	char renglon[31];
	char linea[]="\r\n";

	renglon[0] = '\0';


	UART_Send(LPC_UART3, listo, sizeof(listo), BLOCKING); //envio "enviar renglon"

	UART_Receive(LPC_UART3, renglon, 30, BLOCKING); //recibo 30 caracteres de la uart
	renglon[30]='\0';

	invierte(renglon); //invierto el renglon

	UART_Send(LPC_UART3, renglon, sizeof(renglon), BLOCKING); //muestro el renglon
invertido en pantalla
	UART_Send(LPC_UART3, linea, sizeof(linea), BLOCKING); //espacio

	renglonbraille (renglon,rengbra1,rengbra2,rengbra3); //convierto el renglon a 0 y 1 en 3 lineas
distintas
	invierte2(rengbra2); //invierto el renglon del medio

	UART_Send(LPC_UART3, rengbra1, sizeof(rengbra1), BLOCKING); //muestro los 1 y 0 en
pantalla
	UART_Send(LPC_UART3, linea, sizeof(linea), BLOCKING);
	UART_Send(LPC_UART3, rengbra2, sizeof(rengbra2), BLOCKING);
	UART_Send(LPC_UART3, linea, sizeof(linea), BLOCKING);
	UART_Send(LPC_UART3, rengbra3, sizeof(rengbra3), BLOCKING);
	UART_Send(LPC_UART3, linea, sizeof(linea), BLOCKING);

	//evento = e_renglon;

}

void interrumpir(void)
{

	/*desactivo todas las salidas*/
	PWM_Cmd(LPC_PWM1, ENABLE);

	PWM_ChannelCmd(LPC_PWM1, 1, DISABLE);
	PWM_ChannelCmd(LPC_PWM1, 2, DISABLE);
	PWM_ChannelCmd(LPC_PWM1, 3, DISABLE);
```

```
	PWM_ChannelCmd(LPC_PWM1, 4, DISABLE);

	LPC_GPIO0->FIOCLR = (1 << 16);
	LPC_GPIO0->FIOCLR = (1 << 21);
	LPC_GPIO0->FIOCLR = (1 << 2);
	LPC_GPIO0->FIOCLR = (1 << 3);

	LPC_GPIO2->FIOCLR = (1<<13);

	LPC_GPIO0->FIOCLR = (1<<22);

	apagar_motor();

	return;
}
```

Libreria: imprimir3.h

```
void PWM1_IRQHandler(void);

void invierte (char reng[30]);
void invierte2 (char reng[60]);
void paso_motorM2 (int pasos);
void apagar_motor(void);

void percutor (void);
void nueva_linea_1(void);	//separacion entre renglones de un mismo caracter
void nueva_linea_2(void);	//separacion entre renglones de distintos caracteres
void delay (void);
void delay2 (void);
void delay3 (void);

#define margen 60		//cantidad de pasos para hacer el margen
#define dist_punto_1 5		//cantidad de pasos para hacer la disancia entre dos puntos de un
caracter
//#define dist_punto_2 9		//cantidad de pasos para hacer la disancia entre puntos de dos
caracteres
#define dist_linea_1 22		//cantidad de pasos para hacer la disancia entre dos lineas de
caracter
#define dist_linea_2 38		//cantidad de pasos para hacer la disancia entre dos lineas de dos
caracteres
#define t_percutor	60		//tiempo de accionamiento de percutor
#define t_m2_avance 35		//tiempo entre pasos de motor 2 al avanzar
#define LENTO 35		//avance o retroceso lento de motor M1 -> para imprimir
#define RAPIDO 12		//avance o retroceso rapido de motor M1 -> para retroceso
del carro

int match_cnt = 0;
int num_pasoM1 = 1;
int num_pasoM2 = 1;

enum eventos_imp {e_nulo, e_iniciar, e_int, e_renglon, e_fin};
typedef enum eventos_imp eventos_imp_t;

typedef enum {
```

```
	AVANZAR,		/*motor hacia a delante*/
	RETROCEDER		/*motor hacia atras */
} MOTOR_DIRECCION;


void imprimir (char reng1[60],char reng2[60],char reng3[60],eventos_imp_t *evento);
void paso_motorM1 (int pasos, MOTOR_DIRECCION dir, int rapidez);


void invierte (char reng[30])
{
	char aux;
	int i=0;

	for(i=0;i<15;i++)
	{
		aux = reng[i];
		reng[i] = reng[29-i];
		reng[29-i]=aux;
	}

	return;
}

void invierte2 (char reng[60])
{
	char aux;
	int i=0;

	for(i=0;i<30;i++)
	{
		aux = reng[i];
		reng[i] = reng[59-i];
		reng[59-i]=aux;
	}

	return;
}

void imprimir (char reng1[60],char reng2[60],char reng3[60], eventos_imp_t *evento)
{
	const char fin_ok[]="finok";
	int i=0;

	int dist_punto_2 = 8;

	/* Enable PWM interrupt */
	NVIC_EnableIRQ(PWM1_IRQn);

	/* Reset and Start counter */
	PWM_ResetCounter(LPC_PWM1);
	PWM_CounterCmd(LPC_PWM1, ENABLE);

	/* Start PWM now */
	PWM_Cmd(LPC_PWM1, ENABLE);
```

```
	paso_motorM1(10, AVANZAR, RAPIDO);
	nueva_linea_1();

	for(i=0; i<59; i=i+2) //imprimir renglon 1
	{
		if (*evento==e_int)
		{
			return;
		}

		if ((i==6)||(i==12)||(i==18)||/*(i==24)||*/(i==30)||(i==36)||(i==42)||(i==48)||(i==54))
			dist_punto_2 = 6;
		else
			dist_punto_2 = 8;

		if (reng1[i]=='1')		//hacer un punto si es 1; sino no hacer nada
		{
			percutor();
		}
		paso_motorM1(dist_punto_1, AVANZAR, LENTO);		//espacio en el mismo
caracter

		if (reng1[i+1]=='1')		//hacer un punto si es 1; sino no hacer nada
		{
			percutor();
		}
		paso_motorM1(dist_punto_2, AVANZAR, LENTO);		//espacio entre caracteres

	}

	paso_motorM2(dist_linea_1); //avanzar hoja (sin retroceder carro
	paso_motorM1(dist_punto_2, RETROCEDER, LENTO);

	for(i=0; i<59; i=i+2) //imprimir renglon 2
	{
		if (*evento==e_int)
		{
			return;
		}

		if ((i==6)||(i==12)||(i==18)||(i==24)||(i==30)||/*(i==36)||*/(i==42)||(i==48)||(i==54))
			dist_punto_2 = 6;
		else
			dist_punto_2 = 8;

		if (reng2[i]=='1')		//hacer un punto si es 1; sino no hacer nada
		{
			percutor();
		}
		paso_motorM1(dist_punto_1, RETROCEDER, LENTO);  //espacio en el mismo
caracter

		if (reng2[i+1]=='1')		//hacer un punto si es 1; sino no hacer nada
		{
```

```
                percutor();
            }
            paso_motorM1(dist_punto_2, RETROCEDER, LENTO);  //espacio entre caracteres

        }

        nueva_linea_1();

        for(i=0; i<59; i=i+2) //imprimir renglon 3
        {
            if (*evento==e_int)
            {
                return;
            }

            if ((i==6)||(i==12)||(i==18)||/*(i==24)||*/(i==30)||(i==36)||(i==42)||(i==48)||(i==54))
                dist_punto_2 = 6;
            else
                dist_punto_2 = 8;

            if (reng3[i]=='1')          //hacer un punto si es 1; sino no hacer nada
            {
                percutor();
            }
            paso_motorM1(dist_punto_1, AVANZAR, LENTO);         //espacio en el mismo
caracter

            if (reng3[i+1]=='1')        //hacer un punto si es 1; sino no hacer nada
            {
                percutor();
            }
            paso_motorM1(dist_punto_2, AVANZAR, LENTO);         //espacio entre caracteres

        }

        nueva_linea_2();

        apagar_motor();

        if (*evento == e_fin){
            UART_Send(LPC_UART3, fin_ok, sizeof(fin_ok), BLOCKING);
            return;
        }

        else
            *evento = e_iniciar;

        return;
}


void PWM1_IRQHandler(void)
{
        /* Check whether if match flag for channel 0 is set or not */
        if (PWM_GetIntStatus(LPC_PWM1, PWM_INTSTAT_MR0))
```

```
	{
		match_cnt++;
		/* Clear the interrupt flag */
		PWM_ClearIntPending(LPC_PWM1, PWM_INTSTAT_MR0);
	}
}

void delay (void)
{

	while (match_cnt < t_m2_avance);
	return;
}

void delay2 (void)
{

	while (match_cnt < 25);
	return;
}

void delay_percutor (void)
{

	while (match_cnt < t_percutor);
	return;
}
void paso_motorM1 (int pasos, MOTOR_DIRECCION dir, int rapidez)
{
	int cont_pasos = 0;

	while(cont_pasos <= pasos)
	{
		match_cnt = 0;
		switch (num_pasoM1)
		{
		case 1:
			if (dir== AVANZAR)//hago el paso 2
			{
				PWM_ChannelCmd(LPC_PWM1, 1, DISABLE);
				PWM_ChannelCmd(LPC_PWM1, 2, ENABLE);
				PWM_ChannelCmd(LPC_PWM1, 3, ENABLE);
				PWM_ChannelCmd(LPC_PWM1, 4, DISABLE);
				num_pasoM1 = 2;
				while (match_cnt < rapidez);
			}

			if (dir== RETROCEDER)//hago el paso 4
			{
				PWM_ChannelCmd(LPC_PWM1, 2, DISABLE);
				PWM_ChannelCmd(LPC_PWM1, 3, DISABLE);
				PWM_ChannelCmd(LPC_PWM1, 4, ENABLE);
				PWM_ChannelCmd(LPC_PWM1, 1, ENABLE);
				num_pasoM1 = 4;
				while (match_cnt < rapidez);
```

```
		}

		cont_pasos ++;
		break;

	case 2:
		if (dir== AVANZAR)//hago paso 3
		{
			PWM_ChannelCmd(LPC_PWM1, 1, DISABLE);
			PWM_ChannelCmd(LPC_PWM1, 2, DISABLE);
			PWM_ChannelCmd(LPC_PWM1, 3, ENABLE);
			PWM_ChannelCmd(LPC_PWM1, 4, ENABLE);
			num_pasoM1 = 3;
			while (match_cnt < rapidez);
		}

		if (dir== RETROCEDER)//hago el paso 1
		{
			PWM_ChannelCmd(LPC_PWM1, 1, ENABLE);
			PWM_ChannelCmd(LPC_PWM1, 2, ENABLE);
			PWM_ChannelCmd(LPC_PWM1, 3, DISABLE);
			PWM_ChannelCmd(LPC_PWM1, 4, DISABLE);
			num_pasoM1 = 1;
			while (match_cnt < rapidez);
		}

		cont_pasos ++;
		break;

	case 3:
		if (dir== AVANZAR)//hago paso 4
		{
			PWM_ChannelCmd(LPC_PWM1, 2, DISABLE);
			PWM_ChannelCmd(LPC_PWM1, 3, DISABLE);
			PWM_ChannelCmd(LPC_PWM1, 4, ENABLE);
			PWM_ChannelCmd(LPC_PWM1, 1, ENABLE);
			num_pasoM1 = 4;
			while (match_cnt < rapidez);
		}

		if (dir== RETROCEDER)//hago el paso 2
		{
			PWM_ChannelCmd(LPC_PWM1, 1, DISABLE);
			PWM_ChannelCmd(LPC_PWM1, 2, ENABLE);
			PWM_ChannelCmd(LPC_PWM1, 3, ENABLE);
			PWM_ChannelCmd(LPC_PWM1, 4, DISABLE);
			num_pasoM1 = 2;
			while (match_cnt < rapidez);
		}

		cont_pasos ++;
		break;

	case 4:
		if (dir== AVANZAR)//hago paso 1
```

```
                        {
                                PWM_ChannelCmd(LPC_PWM1, 1, ENABLE);
                                PWM_ChannelCmd(LPC_PWM1, 2, ENABLE);
                                PWM_ChannelCmd(LPC_PWM1, 3, DISABLE);
                                PWM_ChannelCmd(LPC_PWM1, 4, DISABLE);
                                num_pasoM1 = 1;
                                while (match_cnt < rapidez);
                        }

                        if (dir== RETROCEDER)//hago el paso 3
                        {
                                PWM_ChannelCmd(LPC_PWM1, 1, DISABLE);
                                PWM_ChannelCmd(LPC_PWM1, 2, DISABLE);
                                PWM_ChannelCmd(LPC_PWM1, 3, ENABLE);
                                PWM_ChannelCmd(LPC_PWM1, 4, ENABLE);
                                num_pasoM1 = 3;
                                while (match_cnt < rapidez);
                        }

                        cont_pasos ++;
                        break;
                }
        }

        PWM_ChannelCmd(LPC_PWM1, 1, DISABLE);
        PWM_ChannelCmd(LPC_PWM1, 2, DISABLE);
        PWM_ChannelCmd(LPC_PWM1, 3, DISABLE);
        PWM_ChannelCmd(LPC_PWM1, 4, DISABLE);

        cont_pasos = 0;

        return;
}

void paso_motorM2 (int pasos)
{
        int cont_pasos = 0;

        LPC_GPIO2->FIOSET = (1<<13);

        while(cont_pasos <= pasos)
        {
                match_cnt = 0;
                switch (num_pasoM2)
                {
                case 1: //hago el paso 2
                        LPC_GPIO0->FIOCLR = (1 << 16);
                        LPC_GPIO0->FIOCLR = (1 << 21);
                        LPC_GPIO0->FIOSET = (1 << 2);
                        LPC_GPIO0->FIOSET = (1 << 3);
                        delay();
                        cont_pasos ++;
                        num_pasoM2 = 2;
                        break;
```

```
		case 2: //hago paso 3
			LPC_GPIO0->FIOCLR = (1 << 16);
			LPC_GPIO0->FIOCLR = (1 << 2);
			LPC_GPIO0->FIOSET = (1 << 3);
			LPC_GPIO0->FIOSET = (1 << 21);
			delay();
			num_pasoM2 = 3;
			cont_pasos ++;
			break;

		case 3: //hago paso 4
			LPC_GPIO0->FIOCLR = (1 << 2);
			LPC_GPIO0->FIOCLR = (1 << 3);
			LPC_GPIO0->FIOSET = (1 << 16);
			LPC_GPIO0->FIOSET = (1 << 21);
			delay();
			num_pasoM2 = 4;
			cont_pasos ++;
			break;

		case 4: //hago paso 1
			LPC_GPIO0->FIOCLR = (1 << 3);
			LPC_GPIO0->FIOCLR = (1 << 21);
			LPC_GPIO0->FIOSET = (1 << 16);
			LPC_GPIO0->FIOSET = (1 << 2);
			delay();
			num_pasoM2 = 1;
			cont_pasos ++;
			break;
		}
	}
	LPC_GPIO2->FIOCLR = (1<<13);
	cont_pasos =0;

	return;
}

void percutor (void)
{
	match_cnt = 0;
	//LPC_GPIO0->FIOSET = (1<<22);
	PWM_ChannelCmd(LPC_PWM1, 5, ENABLE);
	delay_percutor();
	//LPC_GPIO0->FIOCLR = (1<<22);
	PWM_ChannelCmd(LPC_PWM1, 5, DISABLE);

	return;
}

void apagar_motor(void)
{
	PWM_Cmd(LPC_PWM1, DISABLE);
	PWM_CounterCmd(LPC_PWM1, DISABLE);
	NVIC_DisableIRQ(PWM1_IRQn);
```

```
	return;
}

void nueva_linea_1(void)
{
	do
	{
		paso_motorM1(1, RETROCEDER, RAPIDO);
	}while((LPC_GPIO2->FIOPIN0 & 0x40) == 0); //verifico el 1 en P2(6) -> sensor de retorno de
carro
	paso_motorM2(dist_linea_1);
	paso_motorM1(margen, AVANZAR, RAPIDO);

	return;
}

void nueva_linea_2(void)
{
	do
	{
		paso_motorM1(1, RETROCEDER, RAPIDO);
	}while((LPC_GPIO2->FIOPIN0 & 0x40) == 0); //verifico el 1 en P2(6) -> sensor de retorno de
carro
	paso_motorM2(dist_linea_2);

	return;
}
```

Libreria: abcbraille.h

```
void renglonbraille (char reng[30],char rengbra1[60],char rengbra2[60],char rengbra3[60]);


void renglonbraille (char reng[30],char rengbra1[60],char rengbra2[60],char rengbra3[60])
{
	int i=0;


	for(i=0;i<30;i++)
	{
		switch(reng[i])
		{
		//prefijo mayusculas
		case '$': //prefijo mayuscula
			rengbra1[2*i]  ='1';
			rengbra1[2*i+1]='0';
```

```
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='0';
            break;

//minusculas
case 'a':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case 'b':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case 'c':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case 'd':
```

```
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case 'e':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case 'f':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case 'g':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;
```

```
case 'h':
        rengbra1[2*i]  ='0';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='1';
        rengbra2[2*i+1]='1';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='0';
        break;

case 'i':
        rengbra1[2*i]  ='1';
        rengbra1[2*i+1]='0';
        rengbra2[2*i]  ='0';
        rengbra2[2*i+1]='1';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='0';
        break;

case 'j':
        rengbra1[2*i]  ='1';
        rengbra1[2*i+1]='0';
        rengbra2[2*i]  ='1';
        rengbra2[2*i+1]='1';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='0';
        break;

case 'k':
        rengbra1[2*i]  ='0';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='0';
        rengbra2[2*i+1]='0';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='1';
        break;
```

```
case 'l':
        rengbra1[2*i]  ='0';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='0';
        rengbra2[2*i+1]='1';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='1';
        break;

case 'm':
        rengbra1[2*i]  ='1';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='0';
        rengbra2[2*i+1]='0';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='1';
        break;

case 'n':
        rengbra1[2*i]  ='1';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='1';
        rengbra2[2*i+1]='0';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='1';
        break;

case 'ñ':
        rengbra1[2*i]  ='1';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='1';
        rengbra2[2*i+1]='1';
        rengbra3[2*i]  ='1';
        rengbra3[2*i+1]='0';
```

```
            break;

case 'o':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case 'p':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case 'q':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case 'r':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
```

```
            rengbra3[2*i+1]='1';
            break;

case 's':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case 't':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case 'u':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'v':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
```

```
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'w':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='0';
            break;

case 'x':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'y':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'z':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
```

```
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

//espacio
case ' ':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

//acentos
case 'á':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'é':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'í':
```

```
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case 'ó':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'ú':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='1';
            break;

case 'ü':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='0';
            break;
```

```
//numeros
case '#': //simbolo numerico
        rengbra1[2*i]  ='1';
        rengbra1[2*i+1]='0';
        rengbra2[2*i]  ='1';
        rengbra2[2*i+1]='0';
        rengbra3[2*i]  ='1';
        rengbra3[2*i+1]='1';
        break;

case '1':
        rengbra1[2*i]  ='0';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='0';
        rengbra2[2*i+1]='0';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='0';
        break;

case '2':
        rengbra1[2*i]  ='0';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='0';
        rengbra2[2*i+1]='1';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='0';
        break;

case '3':
        rengbra1[2*i]  ='1';
        rengbra1[2*i+1]='1';
        rengbra2[2*i]  ='0';
        rengbra2[2*i+1]='0';
        rengbra3[2*i]  ='0';
        rengbra3[2*i+1]='0';
```

```
            break;

case '4':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case '5':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case '6':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case '7':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
```

```
            rengbra3[2*i+1]='0';
            break;

case '8':
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case '9':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case '0':
            rengbra1[2*i]  ='1';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

//signos de puntuacion
case '.': //punto
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
```

```
            rengbra2[2*i+1]='0';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case ',': //coma
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='0';
            break;

case ';': //punto y coma
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case '¿': //interrogacion abre
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='0';
            break;

case '?': //interrogacion cierra
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
```

```
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='0';
            break;

case '¡': //exclamacion abre
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case '!': //exclamacion cierra
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='0';
            rengbra2[2*i]  ='1';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='0';
            rengbra3[2*i+1]='1';
            break;

case '(': //parentesis abre
            rengbra1[2*i]  ='0';
            rengbra1[2*i+1]='1';
            rengbra2[2*i]  ='0';
            rengbra2[2*i+1]='1';
            rengbra3[2*i]  ='1';
            rengbra3[2*i+1]='0';
            break;

case ')': //parentesis cierra
            rengbra1[2*i]  ='1';
```

```
                    rengbra1[2*i+1]='0';
                    rengbra2[2*i]  ='1';
                    rengbra2[2*i+1]='0';
                    rengbra3[2*i]  ='0';
                    rengbra3[2*i+1]='1';
                    break;

                case "": //comilla abre y cierre
                    rengbra1[2*i]  ='0';
                    rengbra1[2*i+1]='0';
                    rengbra2[2*i]  ='0';
                    rengbra2[2*i+1]='1';
                    rengbra3[2*i]  ='1';
                    rengbra3[2*i+1]='1';
                    break;

                case '-': //guion
                    rengbra1[2*i]  ='0';
                    rengbra1[2*i+1]='0';
                    rengbra2[2*i]  ='0';
                    rengbra2[2*i+1]='0';
                    rengbra3[2*i]  ='1';
                    rengbra3[2*i+1]='1';
                    break;
                case '*': //asterisco
                    rengbra1[2*i]  ='0';
                    rengbra1[2*i+1]='0';
                    rengbra2[2*i]  ='1';
                    rengbra2[2*i+1]='0';
                    rengbra3[2*i]  ='0';
                    rengbra3[2*i+1]='1';
                    break;
                }
            }
}
```

Anexo 6: Programación de software de interfaz de usuario

La programación del soft de interfaz se realizó en lenguaje “basic”, utilizando el software “Visual Basic Express”, bajo la plataforma “Windows 7”.

```
Public Class Form1


    Public textobraille As String
    Dim cant_hojas As Decimal


    Private Sub mnuArchNuevo_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mnuArchNuevo.Click
        txtNorm.Clear()
    End Sub

    Private Sub mnuArchAbrir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mnuArchAbrir.Click
        Dim Open As New OpenFileDialog()
        Dim myStreamReader As System.IO.StreamReader
        Open.Filter = "Text [*.txt*]|*.txt|All Files [*,*]|*,*"
        Open.CheckFileExists = True
        Open.Title = "Abrir Archivo"
        Open.ShowDialog(Me)
        Try
            Open.OpenFile()
            myStreamReader = System.IO.File.OpenText(Open.FileName)
            txtNorm.Text = myStreamReader.ReadToEnd()
        Catch ex As Exception

        End Try
    End Sub

    Private Sub mnuArchGuardar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuArchGuardar.Click
        Dim Save As New SaveFileDialog()
        Dim myStreamWriter As System.IO.StreamWriter
        Save.Filter = "Text (*.txt)|*.txt|All files(*,*)|*,*"
        Save.CheckPathExists = True
        Save.Title = "Guardar como"
        Save.ShowDialog(Me)
        Try
            myStreamWriter = System.IO.File.AppendText(Save.FileName)
            myStreamWriter.Write(txtNorm.Text)
            myStreamWriter.Flush()
        Catch ex As Exception

        End Try
    End Sub
```

```
    Private Sub mnuArchCerrar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mnuArchCerrar.Click
        End
    End Sub

    Private Sub mnuEditCopiar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mnuEditCopiar.Click
        txtNorm.Copy()
    End Sub

    Private Sub mnuEditPegar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mnuEditPegar.Click
        txtNorm.Paste()
    End Sub

    Private Sub mnuEditCortar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mnuEditCortar.Click
        txtNorm.Cut()
    End Sub

    Private Sub mnuEditAtras_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mnuEditAtras.Click
        txtNorm.Undo()
    End Sub

    Private Sub mnuEditAdelante_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuEditAdelante.Click
        txtNorm.Redo()
    End Sub

    Private Sub opcConv_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles opcConv.Click

        Dim texto1 As String

        texto1 = txtNorm.Text
        filtro_car(texto1)
        filtro_mayus(texto1)
        filtro_num(texto1)
        filtro_alin30(texto1)
        txtBra.Text = texto1
        Control_renglones(texto1)
        textobraille = txtBra.Text


    End Sub

    Private Sub filtro_car(ByRef texto As String)
        'elimina simbolos no imprimibles
        Dim i As Integer, car As String

        For i = 1 To Len(texto)
            car = Mid(texto, i, 1)
            If (car >= "a" And car <= "z") Or (car >= "A" And car <= "Z") Or (car >= "0" And car <= "9") Or _
                (car = " ") Or (car = "ñ") Or (car = "á") Or (car = "é") Or (car = "í") Or (car = "ó") Or _
```

```
            (car = "ú") Or (car = "Ñ") Or (car = "Á") Or (car = "É") Or (car = "Í") Or (car = "Ó") Or _
            (car = "Ú") Or (car = "ü") Or (car = ".") Or (car = ",") Or (car = ";") Or (car = ":") Or _
            (car = "-") Or (car = "(") Or (car = ")") Or (car = "¿") Or (car = "?") Or (car = "¡") Or _
            (car = "!") Or (car = """) Or (car = Chr(10)) Or (car = Chr(13)) Then

        Else
            texto = Replace(texto, car, " ", 1, 1) 'reemplaza el caracter no imprimible por un espacio en
blanco
        End If
    Next i

End Sub

Private Sub filtro_mayus(ByRef texto As String)
    'agrega un signo $ antes de una letra mayuscula y $$ antes de una palabra mayuscula
    Dim i As Integer, car As String, txtaux As String, contcar As Integer, txtaux1 As String,
txtaux2 As String, carmayus1 As Boolean, carmayus2 As Boolean

    txtaux = texto
    contcar = 0
    carmayus1 = False
    carmayus2 = False

    For i = 1 To Len(texto)
        car = Mid(texto, i, 1)
        If (car >= "A" And car <= "Z") Then
            If carmayus2 = True Then 'si ya habian 2 letras mayusculas antes, no hacer nada

            ElseIf carmayus1 = True Then 'si habia 1 letra mayuscula antes, poner $$ a la palabra
                txtaux1 = Replace(txtaux, "$", "$" & "$", i + contcar - 2, 1)
                txtaux2 = Mid(txtaux, 1, i + contcar - 3)
                contcar = contcar + 1
                txtaux = txtaux2 & txtaux1
                carmayus2 = True 'indica q ahora hay 2 letras mayusculas seguidas
            Else 'si no habia ninguna letra mayuscula antes, poner $ a la letra
                txtaux1 = Replace(txtaux, car, "$" & car, i + contcar, 1)
                txtaux2 = Mid(txtaux, 1, i + contcar - 1)
                contcar = contcar + 1
                txtaux = txtaux2 & txtaux1
                carmayus1 = True 'indica q ahora hay 1 letra mayuscula

            End If

        Else 'convertir la letra minuscula a mayuscula
            carmayus1 = False
            carmayus2 = False
            txtaux = Replace(txtaux, car, UCase(car), 1, 1) 'convierte car a mayuscula

        End If

    Next i

    texto = txtaux

End Sub
```

```
    Private Sub filtro_num(ByRef texto As String)
        'agrega un signo # antes de un numero
        Dim i As Integer, car As String, txtaux1 As String, txtaux As String, txtaux2 As String,
            contcar As Integer, carnum As Boolean

        txtaux = texto
        carnum = False
        contcar = 0

        For i = 1 To Len(texto)
            car = Mid(texto, i, 1)
            If (car >= "0" And car <= "9") Or (car = ".") Or (car = ",") Then
                If carnum = True Then

                ElseIf (car >= "0" And car <= "9") And carnum = False Then
                    txtaux1 = Replace(txtaux, car, "#" & car, i + contcar, 1)
                    txtaux2 = Mid(txtaux, 1, i + contcar - 1)
                    contcar = contcar + 1
                    txtaux = txtaux2 & txtaux1
                    carnum = True 'indica q ahora hay 1 letra mayuscula
                End If
            Else
                carnum = False
            End If
        Next i

        texto = txtaux
    End Sub

    Private Sub filtro_alin30(ByRef texto As String)
        Dim i As Integer, renglon As String, car As String, txtaux As String, rengaux As String


        For i = 1 To Len(texto)
            car = Mid(texto, i, 1)
            renglon = renglon & car
            If (car = Chr(10) Or i = Len(texto)) Then 'se extrae un renglon
                If (Len(renglon) <= 30) Then 'si el renglon tiene menos de 30 caract. se pasa como está
                    txtaux = txtaux & renglon
                    renglon = vbNullString
                Else                    'si tiene mas de 30 caracteres se lo divide sucesivamente en
                    Do                  'renglones de 29 caract + un guion "-", a menos que:
                        If (renglon(30) = " ") Then 'despues del car 30 siga un "espacio"
                            rengaux = Mid(renglon, 1, 30) & Chr(10)
                            renglon = Mid(renglon, 32, Len(renglon))
                            txtaux = txtaux & rengaux
                        ElseIf (renglon(29) = " ") Then 'el car 30 sea justo un "espacio"
                            rengaux = Mid(renglon, 1, 29) & Chr(10)
                            renglon = Mid(renglon, 31, Len(renglon))
                            txtaux = txtaux & rengaux
                        ElseIf (renglon(28) = " ") Then 'el car 29 sea un "espacio" (no se debe agragar un
guion en car 30)
                            rengaux = Mid(renglon, 1, 28) & Chr(10)
                            renglon = Mid(renglon, 30, Len(renglon))
```

```
                txtaux = txtaux & rengaux
            ElseIf (renglon(27) = " ") Then 'el car 28 sea justo un "espacio" (para mas prolijidad)
                rengaux = Mid(renglon, 1, 27) & Chr(10)
                renglon = Mid(renglon, 29, Len(renglon))
                txtaux = txtaux & rengaux
            ElseIf (renglon(29) = "$" Or renglon(29) = "#") Then 'el car 30 sea un signo $ o #
                If (renglon(28) = "$") Then 'si es un $$
                    rengaux = Mid(renglon, 1, 28) & "-" & Chr(10)
                    renglon = Mid(renglon, 29, Len(renglon))
                    txtaux = txtaux & rengaux
                Else
                    rengaux = Mid(renglon, 1, 29) & "-" & Chr(10)
                    renglon = Mid(renglon, 30, Len(renglon))
                    txtaux = txtaux & rengaux
                End If
            ElseIf (renglon(28) = "$" Or renglon(28) = "#") Then 'si el car 29 es un signo $ o #
                If (renglon(27) = "$") Then 'si el signo es $$
                    If (renglon(26) = " ") Then 'si justo hay un espacio antes, quitar el guion
                        rengaux = Mid(renglon, 1, 27) & Chr(10)
                        renglon = Mid(renglon, 28, Len(renglon))
                        txtaux = txtaux & rengaux
                    Else
                        rengaux = Mid(renglon, 1, 27) & "-" & Chr(10)
                        renglon = Mid(renglon, 28, Len(renglon))
                        txtaux = txtaux & rengaux
                    End If

                Else
                    rengaux = Mid(renglon, 1, 28) & "-" & Chr(10)
                    renglon = Mid(renglon, 29, Len(renglon))
                    txtaux = txtaux & rengaux
                End If

            Else 'sino, se divide y agrega el guion
                rengaux = Mid(renglon, 1, 29) & "-" & Chr(10)
                renglon = Mid(renglon, 30, Len(renglon))
                txtaux = txtaux & rengaux
            End If
        Loop While (Len(renglon) >= 30)
        txtaux = txtaux & renglon
        renglon = vbNullString
      End If
    End If
  Next i
  texto = txtaux

End Sub

Private Sub Control_renglones(ByRef texto As String)
  Dim i As Integer, c As Char, cant_renglon As Integer

  cant_renglon = 0
  cant_hojas = 0

  For i = 1 To Len(texto)
```

```
            c = Mid(texto, i, 1)
            If (c = Chr(10)) Then
                cant_renglon = cant_renglon + 1
            End If
        Next i
        cant_renglon = cant_renglon + 1
        cant_hojas = Fix(cant_renglon / 21) + 1
        TextBox1.Text = cant_renglon
    End Sub



    Private Sub ToolStripMenuItem4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ToolStripMenuItem4.Click
        dialConfigPuerto.Show()
        dialConfigPuerto.Enabled = True


    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        opcImpr.Enabled = False
        dialConfigPuerto.Enabled = False
        Dial_impresion.Enabled = False

    End Sub

    Private Sub opcImpr_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles opcImpr.Click

        Dialog4.Label1.Text = "Se imprimiran " & cant_hojas & " hojas"
        Dialog4.Show()
        Dialog4.Enabled = True
        Me.Enabled = False

    End Sub

    Private Sub ToolStripMenuItem5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ToolStripMenuItem5.Click
        txtNorm.Copy()
    End Sub

    Private Sub CortarToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CortarToolStripMenuItem.Click
        txtNorm.Cut()
    End Sub

    Private Sub PegarToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles PegarToolStripMenuItem.Click
        txtNorm.Paste()
    End Sub

End Class
```

```
Public Class dialConfigPuerto

    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles OK_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.OK
        Form1.Enabled = True
        Me.Close()
    End Sub

    Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Cancel_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
        Form1.Enabled = True
        Me.Close()
    End Sub

    Private Sub Dialog1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        btnConectar.Enabled = False
        Form1.Enabled = False

    End Sub

    Private Sub btnDetConex_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnDetConex.Click
        cboPuertos.Items.Clear()

        For Each PuertoDisponible As String In My.Computer.Ports.SerialPortNames
            cboPuertos.Items.Add(PuertoDisponible)
        Next

        If cboPuertos.Items.Count > 0 Then
            cboPuertos.Text = cboPuertos.Items(0)
            MessageBox.Show("Seleccione el puerto")
            btnConectar.Enabled = True
        Else
            MessageBox.Show("Ningun puerto encontrado")
            btnConectar.Enabled = False
            cboPuertos.Items.Clear()
            cboPuertos.Text = ("")
        End If
    End Sub

    Private Sub btnConectar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnConectar.Click
        If btnConectar.Text = ("Conectar") Then
            Form1.SerialPort1.PortName = cboPuertos.Text
            btnConectar.Text = ("Desconectar")
            Form1.opcImpr.Enabled = True
            'Dial_impresion.Timer1.Enabled = True
            Form1.SerialPort1.Open()
        ElseIf btnConectar.Text = ("Desconectar") Then
            btnConectar.Text = ("Conectar")
            Form1.opcImpr.Enabled = False
```

```
            Dial_impresion.Timer1.Enabled = False
            Form1.SerialPort1.Close()

        End If
    End Sub
End Class



Public Class Dial_impresion
    Public cont As Integer = 1, renglon As String, b_ult_reng As Boolean = False, cont_reng As Integer
= 0, b_renglon As Boolean = False


    Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Cancel_Button.Click
        Form1.SerialPort1.Write("int int")
        Form1.Enabled = True
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
        Me.Close()
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick
        Dim txtleido As String, cadena As Integer, fin_ok As Integer

        cadena = 0

        txtleido = Form1.SerialPort1.ReadExisting 'guardar lo que se encuentre en puerto serial

        cadena = InStr(txtleido, "enviar renglon para imprimir") 'espero que la impresora este lista
        If (cadena >= 1) Then 'si ya esta lista para recibir renglon
            If (cont_reng = 20) Then 'se se llego al ultimo renglon de la hoja
                cont_reng = 0 'reseteo contador de renglones
                Dialog3.Label1.Text = "Insertar proxima hoja..." 'espero que se inserte hoja nueva
                Dialog3.Text = "Continuar impresion.."
                Dialog3.fin_impresion = False
                Dialog3.Show()
                Me.Enabled = False
                Timer1.Enabled = False
            Else 'si no se llego al ultimo renglon de la hoja
                b_renglon = True 'pongo la bandera en 1 para enviar renglon
            End If
        End If

        fin_ok = InStr(txtleido, "finok") 'confirmacion de la impresora de que es el ultimo renglon

        If txtleido <> "" Then 'mostrar lo leido del puerto en pantalla y borrar buffer
            txtRecibido.Text = txtRecibido.Text & txtleido & Chr(10)
            txtleido = ""
            Form1.SerialPort1.DiscardInBuffer()
        End If
```

```
    If (cont >= Len(Form1.textobraille)) Then 'si se llego al final del texto
        If (b_ult_reng = False) Then    'si la bandera no es 1
            Form1.SerialPort1.Write("fin") 'informar a la impresora que es el ultimo renglon
        End If
        b_ult_reng = True   'poner bandera de ultimo renglon en 1
    End If


    If (b_ult_reng = True) Then 'si se alcanzo el ultimo renglon
        If (fin_ok >= 1) Then   'si la impresora confirmo el ultimo renglon
            Dialog3.Text = "Impresion Finalizada"
            Dialog3.Label1.Text = "Impresion Finalizada!"
            Dialog3.fin_impresion = True
            Dialog3.Show()  'mostrar cartel de impresion finalizada
            Timer1.Enabled = False  'cortar el timer de impresion
            Me.Enabled = False 'inhabilitar la ventana
            Me.Close()
        End If

    Else
        If (b_renglon = True) Then  'si la impresora esta lista para recibir renglon
            armarRenglon() 'preparar un renglon de 30 caracteres para enviar
            cont_reng = cont_reng + 1 'contar un renglon
            Form1.SerialPort1.Encoding = System.Text.Encoding.Default 'codificar el texto a ascii
            Form1.SerialPort1.Write(renglon) 'enviar renglon a la impresora
            txtRecibido.Text = txtRecibido.Text & renglon
            delay() 'esperar un tiempo
            Form1.SerialPort1.Write("imp") 'enviar comando para imprimir
            b_renglon = False 'bajar bandera de impresora lista
        End If
    End If

End Sub



Private Sub armarRenglon()
    Dim c As Char, contaux As Integer, aux As Integer, i As Integer

    renglon = ""
    c = ""
    i = 1

    For i = 1 To 31
        c = Mid(Form1.textobraille, cont, 1)
        c = LCase(c)
        renglon = renglon & c
        cont = cont + 1
        If c = Chr(10) Then 'si encuentro un cambio de linea
            aux = cont - 1
            renglon = Mid(renglon, 1, aux) 'le saco el caracter "\n" del renglon
            contaux = i - 1
            While (contaux < 31)
                renglon = renglon & " " 'lleno el renglon con espacios
                contaux = contaux + 1
```

```
            End While
            Exit For
        End If
    Next i

End Sub

Private Sub delay()
    Dim retardo As Integer = 1

    For retardo = 1 To 100000000
    Next retardo

End Sub

Private Sub Dial_impresion_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Form1.Enabled = False
End Sub
End Class


Public Class Dialog3
    Public fin_impresion As Boolean

    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles OK_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.OK
        If (fin_impresion = True) Then 'si es el ultimo renglon del texto terminar la impresion
            Dial_impresion.Timer1.Enabled = False
            Form1.Enabled = True
        Else
            Dial_impresion.Timer1.Enabled = True 'si es es el ultimo renglon de la hoja volver a la
impresion
            Dial_impresion.Enabled = True
            Dial_impresion.b_renglon = True
        End If

        Me.Close()
    End Sub

    Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel
        Me.Close()
    End Sub

    Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Label1.Click

    End Sub
End Class


Public Class Dialog4
```

```
    Private Sub OK_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles OK_Button.Click
        Me.DialogResult = System.Windows.Forms.DialogResult.OK

        Dial_impresion.Show()
        Dial_impresion.Enabled = True
        Dial_impresion.cont = 1

        Dial_impresion.Timer1.Enabled = True
        Form1.SerialPort1.DiscardOutBuffer()
        Form1.SerialPort1.Write("enc")
        Me.Close()
    End Sub

    Private Sub Cancel_Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Me.DialogResult = System.Windows.Forms.DialogResult.Cancel

        Form1.Enabled = True

        Me.Close()
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

        Form1.Enabled = True

        Me.Close()
    End Sub
End Class
```

Bibliografía

- “Introducción a la programación en C”. Marco A. Peña, Jose M. Cela. Departament d’Arquitectura de Computadors Universitat Politecnica de Catalunya. Año 2000
- “Universal Serial Bus Specification”. Revision 2.0 . April 27, 2000
- “Universal Serial Bus Device Class Definition for Printing Devices”. Version 1.1 . January 2000
- “LPC17xx User manual”. Rev. 2 — 19 August 2010.
- “LPCXpresso v7 User Guide”. Rev. 7.4 — 16 September, 2014.
- “Workshop LPCXPresso”. SASE 2012. Jorge Ezequiel Espósito y Federico Roasio – Laboratorio de Sistemas Embebidos – Facultad de Ingenieria UBA
- “Aprenda Visual Basic 6.0 como si estuviera en primero”. Javier García de Jalón, José Ignacio Rodríguez, Alfonso Brazález. Escuela Superior de Ingenieros Industriales de San Sebastián - UNIVERSIDAD DE NAVARRA.
- “El sistema Braille”. https://es.wikipedia.org/?title=Braille_(lectura)
- “Impresora Braille”. https://es.wikipedia.org/wiki/Impresora_braille
- “Información de Impresora Braille Basic DV4”. http://www.tiflonexos.com.ar/basicV4.htm , http://www.tecno-ayudas.com.ar/productos_impresora_braille_basic-d.html
- “Impresora Braille y Equipo Multifunción - Orientaciones de Instalación y Uso”. DIRECCIÓN DE EDUCACIÓN ESPECIAL, DIRECCIÓN DE TECNOLOGÍA EDUCATIVA , PROGRAMA CONECTAR IGUALDAD.
- “LENGUAJE DE DESCRIPCION DE PAGINAS (PDL)”. https://es.wikipedia.org/wiki/Lenguaje_de_descripci%C3%B3n_de_p%C3%A1ginas
- “FISICA UNIVERSITARIA VOL. 2 - 12º Ed.”. Hugh D. Young, Roger A. Freedman. Addison-Wesley, año 2008
- “Solenoids - Technical Data”. Technical Information. saia-burgess Solenoids
- “Propiedades magnéticas de los materiales”. http://www.sabelotodo.org/fisica/propiedadesmagneticas.html
- “Solenoids and Actuators”. http://homepages.which.net/~paul.hills/Solenoids/SolenoidsBody.html
- “Motores paso a paso - Características básicas”. Eduardo J. Carletti. http://robots-argentina.com.ar/MotorPP_basico.htm
- “Tutorial sobre Motores Paso a Paso”. http://www.todorobot.com.ar/tutorial-sobre-motores-paso-a-paso-stepper-motors/
- “SINGLE-CHIP USB TO UART BRIDGE – CP2102/9”. Silicon Labs. Rev. 1.6 12/13
- “Normas de seguridad eléctrica para protección del usuario”. Secretaria de Comercio - Resolución Nº 508/15 - articulo 1º y 6º.
- “Seguridad de aparatos electrodomésticos y similares”- Normas IRAM - NM 60335-1.
- “Kicad – Tutorial”. kicad-pcb.org
- Hoja de datos (Datasheets) varios. Componentes electrónicos presentes en la impresora original de base, y la placa electrónica desarrollada (ver Anexo 4).