UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COLEGIO DE CIENCIA Y TECNOLOGÍA
LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

DISEÑO, DESARROLLO Y CONSTRUCCIÓN DE UNA SUPERFICIE MULTITÁCTIL, PARA EL CONTROL DE VELOCIDAD Y POSICIÓN DE MOTORES DE CORRIENTE DIRECTA

TRABAJO RECEPCIONAL
PARA OBTENER EL TÍTULO DE LICENCIADO EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

P R E S E N T A:

ISMAEL JIMÉNEZ OSORIO

DIRECTOR DE TESIS:
DR. JUAN CARLOS MARTÍNEZ ROSAS

MÉXICO, CIUDAD DE MÉXICO, MAYO 2023

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

JURADO ASIGNADO:

Lectora : Mtra.Catalina Trevilla Román
Lector : Dr. Efren Bernardo Ramírez Solíz
Lector : Dr. José Alberto Aparicio Santos

UACM, plantel San Lorenzo Tezonco, CDMX.

DIRECTOR DE TESIS:
Dr. Juan Carlos Martínez Rosas

_____

FIRMA

Dedicado a mi madre:
quien con su ejemplo y amor
me enseñó el camino
para ser una mejor persona cada día
y forjar en mí un espíritu inquebrantable.

Agradecimientos

A la UACM por brindarme la oportunidad de conocer un nuevo y maravilloso mundo de conocimientos.

Al extinto Instituto de Ciencia y Tecnología Del Distrito Federal por el soporte financiero para esta investigación.

A la Universidad Autónoma Veracruzana por invitarnos a participar con nuestro proyecto en su semana tecnológica.

Al Instituto Tecnológico de Acayucan.

A mis lectores: Mtra. Catalina Trevilla Román, Dr. Efren Bernardo Ramírez Solíz, Dr. José Alberto Aparicio Santos. Por fomentar en mí la perfección y hacer de este trabajo un proyecto de excelencia.

A mi director de tesis, Dr. Juan Carlos Martínez Rosas, por brindarme la oportunidad de trabajar en un proyecto innovador que me hizo crecer como estudiante y como ser humano. Trazar un camino donde no había.

Agradezco a mis padres y hermanos por su eterno apoyo.

Ismael Jiménez Osorio.

Reconocimientos

Quiero agradecer a la Universidad Autónoma de la Ciudad de México (UACM) y al Instituto de Ciencia y Tecnología del Distrito Federal (ICyTDF), el apoyo económico proporcionado para la compra de equipo de laboratorio y material necesario para la realización de este proyecto. Así como la beca con la que fui beneficiado durante el desarrollo del mismo, por medio del convenio número UACM-ICyTDF/211/2010.

A la Universidad Autónoma de la Ciudad de México (UACM) por prestar las instalaciones de los laboratorios B-403, C-005, y C-401. Además, agradezco los viáticos que nos proporcionó para acudir a la Universidad Autónoma del Estado de Hidalgo para participar como ponentes en el ”1er Congreso Nacional en Tecnologías de la Información”, así como los viáticos para acudir al instituto Tecnológico Superior de Acayucan para exponer el trabajo realizado. Y los viáticos para acudir al foro MexEEdev 2012 (México Electronics and Embedded Developers Forum) que se desarrolló en la ciudad de Guadalajara.

A la Universidad Autónoma de Baja California (UABC), por la asesoría técnica aportada por el Dr. Miguel E. Rosas Martínez y el Dr. Eduardo A. Murillo Bracamontes, los cuales fueron claves para llevar a buen fin el proyecto.

Resumen

En el presente trabajo, se desarrolla un dispositivo capaz de convertir un monitor de computadora en una pantalla táctil, a un costo inferior al de un dispositivo con tales características. El sistema es empleado para realizar de manera demostrativa el control de posición de un motor de corriente directa.

El presente trabajo está organizado como sigue: en el Capítulo 1, se introduce al lector en el tema, describiendo qué es, cómo está compuesta y para qué se utiliza una pantalla táctil, así como su historia. En el Capítulo 2, se describe el diseño de las etapas que componen al sistema: transmisión, recepción, acondicionamiento de señal, procesamiento, interfaz gráfica, y potencia. En el Capítulo 3, se describen las pruebas realizadas. En el Capítulo 4, se describen las conclusiones, así como el trabajo a futuro. Finalmente, se incluye un apéndice que contiene el código fuente de las rutinas utilizadas en el microcontrolador, así como el código fuente de la interfaz gráfica de usuario creada en MATLAB.

Índice

Lista de figuras

Capítulo 1

Introducción

En la actualidad la mayoría de los procesos industriales requieren de algún tipo de actuador para poner en movimiento sus componentes y de esta manera realizar un tipo de tarea específica (cortar, trasladar, pulir, perforar, acelerar, etc). Para llevar a cabo este tipo de aplicaciones, se hace uso de servosistemas, los cuales pueden ser de tipo electrónico (motores de corriente continua - c.d., corriente alterna- a.c.) o tipo mecánico (motores neumáticos o hidráulicos). Su principal ventaja radica en la posibilidad de controlar dinámicamente tanto su posición como velocidad con un alto grado de precisión, permitiendo que el acabado de un proceso industrial sea de mayor calidad conforme la precisión de los servosistemas lo permita.

Sin caer en la exageración se puede aseverar que la mayoría de las actividades que acontecen en la vida moderna giran en torno a los servosistemas, logrando la movilidad de sistemas industriales, automotrices, transporte, aviación, navegación, hogar, telecomunicaciones, cómputo, construcción, medicina, investigación, etc.

El motor de corriente continua es una máquina utilizada para convertir la energía eléctrica en mecánica, mediante movimiento rotatorio o lineal. Esta máquina de c.d. es una de las más versátiles por su simplicidad y eficiencia dinámica. Su fácil control de posición y de velocidad la han convertido en una de las mejores opciones en aplicaciones de control y automatización de procesos.

Los motores de c.d. se usan en aplicaciones de potencia (trenes, tranvías, autos eléctricos, grúas, etc.) o de precisión (máquinas *CNC*, telescopios, robots, etc.) así como en aplicaciones de asistencia de movimiento en pacientes con movilidad

restringida, etc.

La principal característica del motor de c.d., es la posibilidad de regular la velocidad de cero a máxima velocidad, o su posición angular en radianes o microradianes en forma regulada. El modelo matemático de un motor de c.d. de imán permanente es un sistema que permite ser controlado con una variedad de sistemas de control, los cuales van desde un controlador Proporcional Integral Derivativo (PID), pasando por otros tipos de controladores: adaptable, robusto, difuso, redes neuronales, etc. Es por ello que es considerada una máquina universal de aplicación extendida.

Sin embargo, se encuentra ausente el uso de plataformas táctiles en aplicaciones que permiten su encendido, apagado, control y comunicación en tiempo real.

1.1 Motivación

Motivados por el desarrollo de una arquitectura de bajo costo, sofisticada funcionalidad y aplicación en la educación, se plantea el desarrollo de una plataforma completa para el control retroalimentado de servomotores de c.d. mediante un conjunto de herramientas que involucran el diseño de *software, hardware,* protocolos de comunicación y el diseño y construcción de una superficie táctil para combinar la recepción, detección, envío y manipulación de datos en tiempo real.

La introducción de la tecnología táctil en el área de control es con el fin de facilitar la interacción hombre-máquina haciéndola más fluida, ya que como se ha observado en otros ámbitos, resulta más intuitivo interactuar con una interfaz táctil que con botones o perillas. Un claro ejemplo son los teléfonos celulares, actualmente nos resulta fácil desplazarnos entre los diferentes menús que contienen, con unos cuantos movimientos de nuestros dedos podemos acceder rápidamente a todas sus funciones, mientras que en los teléfonos que carecen de esta tecnología nos resulta complicada la navegación en sus menús.

Viendo las bondades que tiene esta tecnología, se decidió construir un dispositivo que convierte una pantalla normal en pantalla táctil, este dispositivo es de bajo costo, por lo que se puede utilizar para proyectos académicos como una interfaz

hombre máquina amigable.

1.2 Objetivo

En el presente trabajo se pretende realizar una pantalla táctil, la cual se desea utilizar para relacionar conceptos teóricos del área de control automático con su correspondiente aplicación experimental, y que, por medio de la misma, el usuario pueda realizar pruebas de sintonización de un controlador PID, realizando sus propios cálculos, para vaciarlos en la interfaz de usuario del sistema.

El control de posición de un motor de corriente directa, así como la modificación de las ganancias del controlador PID, deberán ser por medio de ordenes introducidas de forma táctil a través de una interfaz gráfica de usuario *(Graphical User Interface o GUI por sus siglas en inglés)*.

1.3 Formulación del problema

El diseño del prototipo se dividirá en etapas bien definidas y se realizará cada una de ellas por separado para garantizar su correcto funcionamiento. Posteriormente, se conectarán entre ellas para formar un solo sistema.

Se requiere diseñar y construir el prototipo de un dispositivo electrónico que permita introducir órdenes o comandos de forma táctil a un *software* o programa gráfico ejecutado sobre una computadora personal con sistema operativo *Windows*.

Se requiere diseñar e implementar el programa gráfico, que deberá fungir como una interfaz gráfica de usuario para que el operador utilice el sistema; ya sea cambiando el comando de posición o velocidad de la flecha del motor, o modificando los parámetros del controlador.

Se requiere implementar un controlador PID para realizar el control del motor de corriente directa.

Se requiere diseñar y construir la etapa de potencia que proporciona alimentación al motor de c.d.

En los siguientes párrafos se da una breve introducción acerca de lo que es una pantalla táctil, así como una línea de tiempo acerca del desarrollo de este tipo de pantallas, con la finalidad de introducir al lector en el tema.

1.4 ¿Qué es una pantalla táctil?

Una pantalla táctil es una pantalla que mediante sensores es capaz de identificar la posición de uno o varios objetos sobre su superficie, lo que permite la entrada de datos e instrucciones a dispositivos como computadoras, teléfonos inteligentes, cajeros automáticos, entre otros. Dependiendo del tipo de tecnología utilizada para realizar el sensado, las instrucciones pueden ser introducidas por medio de herramientas especiales como lápices también llamados *stylus*, o directamente con los dedos.

1.5 Clasificación por tipo de tecnología

Existen diferentes clasificaciones de las pantallas táctiles, en función de la forma en la cual se realiza la detección del contacto con la pantalla o por el tipo de sensor empleado. Para el caso de este trabajo consideraremos una clasificación por tipo de sensor.

1.5.1 Pantalla táctil resistiva

Su nombre se debe a que el principio de funcionamiento utilizado se basa en la resistencia eléctrica. Una pantalla táctil resistiva está compuesta por varias capas, de las cuales, las más importantes son dos placas de vidrio o acrílico colocadas muy juntas una de la otra sin que tengan contacto. Una de las capas está recubierta de un material semiconductor transparente, mientras que a la otra se le recubre con un material resistivo. Durante la operación de la pantalla la electricidad pasa a través de la superficie conductiva. Cuando se realiza una presión sobre la pantalla, se establece un ligero contacto entre las dos capas, y ocurre un cambio en el flujo de la electricidad debido a la resistencia creada por la superficie resistiva.

Un circuito electrónico detecta el cambio, y lo utiliza para localizar la posición de la presión realizada sobre la pantalla.

Aunque las pantallas resistivas tienden a ser menos sensibles, siguen manteniendo un lugar importante en el mercado, fundamentalmente porque tienden a ser más resistentes que las capacitivas, además su precio es inferior al de otras tecnologías. Su diseño en capas, también permite determinar niveles de presión sin incrementar demasiado su costo.

En la tabla 1.1: se muestran algunas de las ventajas y desventajas que tiene este tipo de tecnología (Ramírez et al., 2014).

Ventajas	Desventajas
Bajo costo de producción.	Menor tiempo de vida, a causa del desgaste por el uso.
Capacidad de medir la presión ejercida sobre su superficie.	Pérdida de brillo de hasta un 25% a causa de las múltiples capas que incorporan.
Resistentes al polvo y al agua.	Su comportamiento es no lineal.

Tabla 1.1: Pantalla táctil resistiva, ventajas y desventajas.

1.5.2 Pantalla táctil capacitiva

Una pantalla táctil capacitiva es aquella que utiliza sensores capacitivos para detectar las pulsaciones sobre la pantalla.

En la capa exterior se genera un campo eléctrico de muy baja intensidad. Si se toca esa capa con un objeto conductor, se extrae una pequeña parte de la carga. Una pantalla táctil capacitiva se construye uniendo un panel de vidrio recubierto con un material conductor transparente. Su funcionamiento básico se basa en aprovecharse de la capacidad del cuerpo humano de conducir electricidad, es decir que cuando el usuario toca la superficie de la pantalla, activa un campo eléctrico, el cual es registrado e informado al hardware del dispositivo, traduciéndolo en comandos.

En la tabla 1.2: se muestran algunas de las ventajas y desventajas que tiene este tipo de tecnología (Ramírez et al., 2014).

Ventajas	Desventajas
Alta claridad de pantalla.	Sólo reconoce cuerpos conductores.
No se ven alteradas por elementos externos.	No responden ante stylus o lápices.
Capacidad de interacción multipunto.	Su precio es mayor al de las pantallas resistivas.

Tabla 1.2: Pantalla táctil capacitiva, ventajas y desventajas.

1.5.3 Pantalla táctil infrarroja

Las pantallas táctiles infrarrojas siguen un circuito de dos o más sensores situados alrededor de la pantalla. Cuando el usuario toca sobre la pantalla, la sombra provocada es captada por los sensores, los cuales triangulan la posición para reconocer las coordenadas.

La tecnología infrarroja se está empezando a introducir en la construcción de monitores táctiles para ordenador, ya que además de su eficacia, proporcionan versatilidad y asequibilidad, especialmente en pantallas de gran tamaño.

En la tabla 1.3: se muestran algunas de las ventajas y desventajas que tiene este tipo de tecnología.

Ventajas	Desventajas
Iluminación uniforme, siempre y cuando los leds sean distribuidos de forma correcta.	Dificultad de construcción ya que requiere una superficie con características específicas que permitan la colocación de los sensores.
El costo de producción es el más bajo de las tecnologías analizadas.	
Capacidad de implantación en pantallas de gran tamaño a bajo costo.	

Tabla 1.3: Pantalla táctil infrarroja, ventajas y desventajas.

1.6 Ventajas y desventajas de la pantalla táctil

La principal razón de utilizar aplicaciones con tecnología táctil es el incremento sustancial en la facilidad de interacción hombre-máquina, lo que se traduce en una notable disminución de tiempo en la curva de aprendizaje, por lo cual, el usuario final puede familiarizarse rápidamente con un producto nuevo, aunque no tenga experiencia en su uso.

La principal desventaja que tienen las pantallas táctiles capacitivas y resistivas respecto a las infrarrojas es el costo, ya que es superior, y se incrementa considerablemente respecto a sus dimensiones, por lo que las pantallas no suelen ser muy grandes, lo que nos lleva a otro punto en contra, ya que si la pantalla es demasiado pequeña se dificulta su manejo para personas cuyos dedos sean anchos y esto hace que pierdan su característica más valiosa que es la facilidad de uso.

En la tabla 1.4: se muestran algunas de sus ventajas y desventajas.

Ventajas	Desventajas
La facilidad de su uso.	Son muy delicadas al sol y a la suciedad.
Es el sistema más intuitivo para manejar cualquier elemento electrónico.	Pérdida de brillo.
La amplia gama de sectores en la cual puede ser aplicada dicha tecnología.	Dependiendo del tamaño de la pantalla, la fisonomía de los dedos puede ser un problema.
La disminución del uso de periféricos de entrada en el computador como teclados, ratones, etc.	

Tabla 1.4: Pantalla táctil, ventajas y desventajas.

1.7 Cronología del desarrollo de la pantalla táctil

El primer trabajo en esta área se remonta al año 1965, en el cual, el científico *Edward A. Johnson* publicó un artículo acerca de su trabajo en *The Royal Radar*

Establishment en Malvern, Inglaterra, sobre una pantalla táctil de tecnología capacitiva, la cual proporcionaba un acoplamiento muy eficiente entre hombre y máquina que hasta el momento no se había logrado (Johnson and Establishment, 1966). Posteriormente en 1967 *Johnson* publicó otro artículo donde daba una explicación completa del dispositivo (Johnson, 1967).

En el año de 1971 el profesor *Sam Hurst* de la Universidad de *Kentucky*, desarrolló un sensor táctil, que llamó *"Elograph"*, el cual fue la base para que en el año 1974 desarrollara la primera pantalla táctil resistiva. El profesor continuó con sus investigaciones, y en el año 1977, su empresa *"Elographics"* desarrolla y patenta la primera pantalla táctil de tecnología resistiva. Ese mismo año la empresa *"Siemens Corporation"* financió a *"Elographics"* para producir la primera interface táctil para pantallas, siendo el primer dispositivo conocido con el nombre de pantalla táctil (From, 2015).

En el año de 1983, la compañía *"Hewlett-Packard"* lanza al mercado la primera computadora personal con una pantalla táctil, su modelo *"HP-150"*, la cual fue diseñada con tecnología infrarroja (Hawkins, 1983).

En el año de 1993, la empresa *"Apple"* saca al mercado su *"Newton PDA"* la cual venia equipada con reconocimiento de escritura (Service, 1993), y ese mismo año, la compañía *"IBM"* comercializa el primer teléfono inteligente, llamado *"Simon"* el cual tenía una interface táctil, entre otras características muy destacables para su época (Roebuck, 2012).

En el año de 1996, la empresa *"Palm Computing"* saca al mercado su asistente digital personal *(PDA) Palm Pilot*, que incorporaba una pantalla táctil para interactuar con el usuario, la cual tuvo gran éxito (Korhonen and Ainamo, 2013).

En el año de 2006, la empresa *"Apple"* presentó la primera pantalla Multitáctil en la conferencia de Tecnología, Entretenimiento y Diseño de 2006 *(TED)* celebrada en *EEUU*, la cual fue usada en su producto *"iPhone"* (Han, 2006).

1.8 El futuro de la tecnología táctil

Recientemente la compañía *"Leap Motion, Inc"* ha desarrollado un nuevo producto que podrá sustituir las actuales pantallas táctiles, ya que el usuario no requiere tocar físicamente una superficie para poder controlar el dispositivo que está manipulando, el cual parece ser el futuro de los dispositivos táctiles.

El controlador *"Leap Motion"* es un dispositivo periférico de conexión por puerto USB, diseñado para ser conectado en una computadora. Para su funcionamiento utiliza dos cámaras y tres diodos emisores de luz (*Light-Emitting Diode ó LED* por sus siglas en inglés), de tecnología infrarroja. Su funcionamiento es como sigue: El dispositivo monitorea un área semiesférica de aproximadamente 1 metro. Para ello ilumina el área con los leds, y cuando un objeto la atraviesa, éste refleja la luz que incide sobre él, lo que genera un patrón en tercera dimensión de puntos de luz infrarroja, que es detectado por las cámaras, generando cerca de 300 cuadros por segundo de datos reflejados, los cuales se envían por medio de un puerto USB a la computadora donde son analizados por el software y traducidos a comandos.

Capítulo 2

Diseñó de la superficie táctil

La pantalla detecta las pulsaciones por medio de sensores colocados en su periferia. El circuito detector se compone de emisores y sensores infrarrojos.

Frente a cada emisor se coloca un sensor. Los emisores están colocados en los laterales izquierdo e inferior de la pantalla y los sensores están colocados en los laterales derecho y superior de la misma. La disposición de los emisores y sensores se muestra en la figura 2.1, en la cual, los emisores se representan con círculos, y los sensores con cuadrados.

Al colocar emisores y sensores en toda la periferia de la pantalla, nos permite formar una cuadrícula infrarroja que abarca toda la pantalla. Para detectar una pulsación, el sistema busca la ausencia de señal en alguno de los sensores, cuando lo encuentra, significa que un objeto se interpuso entre el sensor y su correspondiente emisor, de esta manera el sistema conoce el punto donde se realizó la pulsación.

El sistema se dividió en cinco etapas individuales: emisión, sensado, acondicionamiento de señal, procesamiento e interfaz gráfica.

Para lograr el sensado de un objeto dentro del área de la pantalla, se emite una señal lumínica por medio del primer led infrarrojo, colocado en la parte inferior izquierda de la pantalla.

La señal emitida es detectada por el sensor infrarrojo colocado justo frente al emisor, en la esquina superior izquierda de la pantalla. Una vez detectada la señal es filtrada, amplificada, y finalmente guardada para su posterior análisis.

Lo anterior se repite para cada uno de los sensores colocados en el contorno de la pantalla. En la figura 2.1 se muestra el orden en el cual se leen los sensores. Una vez terminado de leer los 53 sensores, se analizan sus lecturas y en caso de existir una pulsación sobre la pantalla, se determina su posición dentro de ella.

En la figura 2.2 se muestra el diagrama de bloques de la superficie táctil, cada uno de los bloques representa un sub-sistema que fue desarrollado independientemente del resto, cuando cada bloque funcionó debidamente se procedió a conectarlos entre sí.

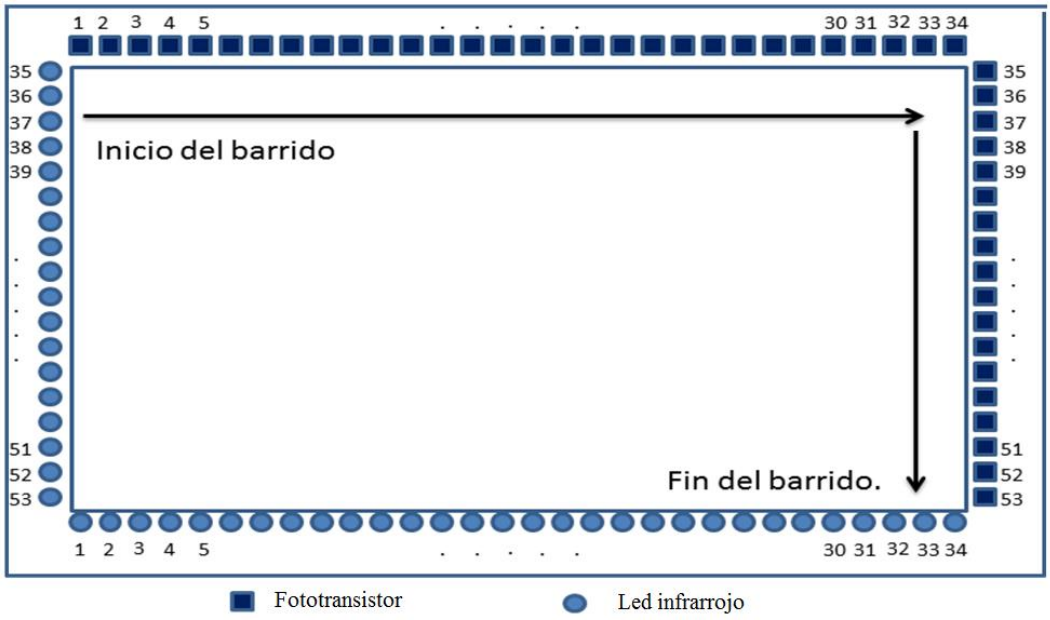


Figura 2.1: Distribución de emisores y sensores

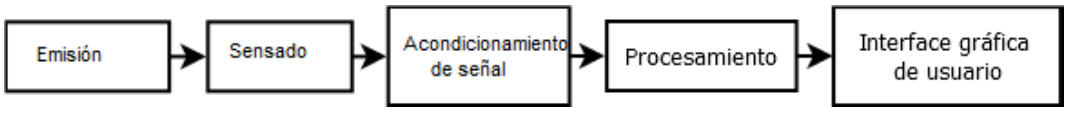


Figura 2.2: Diagrama a bloques del sistema

2.1 Etapa de emisión

Para realizar la emisión se requiere de tres sub etapas, las cuales como podemos observar en la Figura 2.3 son: el generador de señal, la alimentación de los leds y el emisor.

Como generador de señal se utiliza el microcontrolador MCF52223, el cual se describe en la sección 2.4.

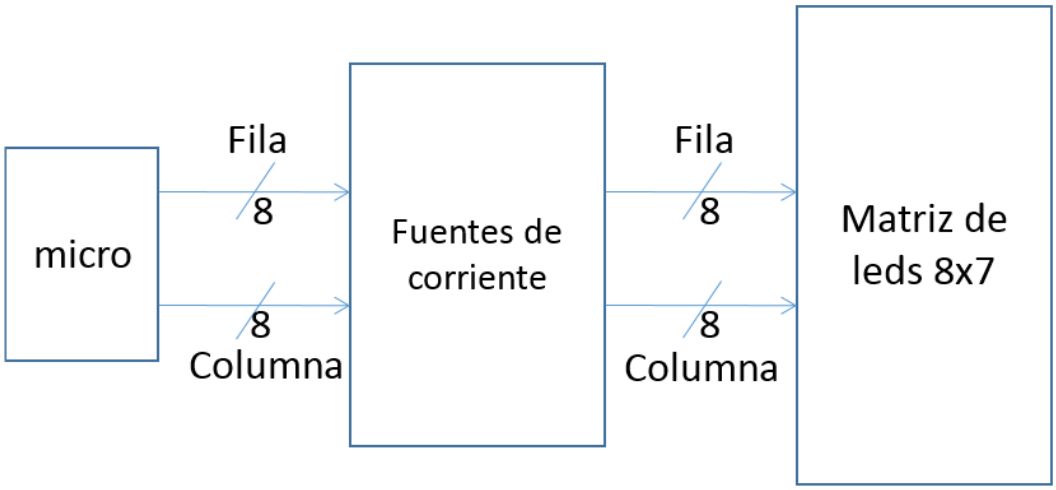


Figura 2.3: Diagrama a bloques de la etapa de transmisión

Como emisor se utiliza el diodo emisor de luz ”infrarroja” *(Infrared Emitting diode)* modelo APA3010F3C, de la marca *King-Bright*.

La emisión es realizada por leds infrarrojos, los cuales son excitados por una señal modulada a una frecuencia de $10KHz$ procedente de un microcontrolador. Cada led emite una señal lumínica modulada, cuya fuente de excitación procede de una fuente de corriente, lo cual garantiza que la cantidad de luz emitida por cada led sea la misma.

Cuando un led es excitado, se hace parpadear 5 veces a una frecuencia de 10 KHZ, esto se decidió así, debido a que en las pruebas experimentales, cuando se realizaba la lectura del sensor con diferente número de parpadeos, se determinó que el promedio de las lecturas del sensor no variaba significativamente cuando

se realizaban más de 5 parpadeos.

Para cubrir el área de la pantalla se requiere utilizar 53 leds, esto se debe a que se decidió dejar una separación de media pulgada entre cada uno de ellos. Debido a la gran cantidad de leds requeridos, se decide conectarlos en forma de un arreglo tipo matricial, tal y como se muestra en la figura 2.4, con lo cual, se requieren siete fuentes de corriente para alimentar cada una de las columnas de la matriz de leds, de tal forma que se disminuyeron las necesidades de recursos del microcontrolador.

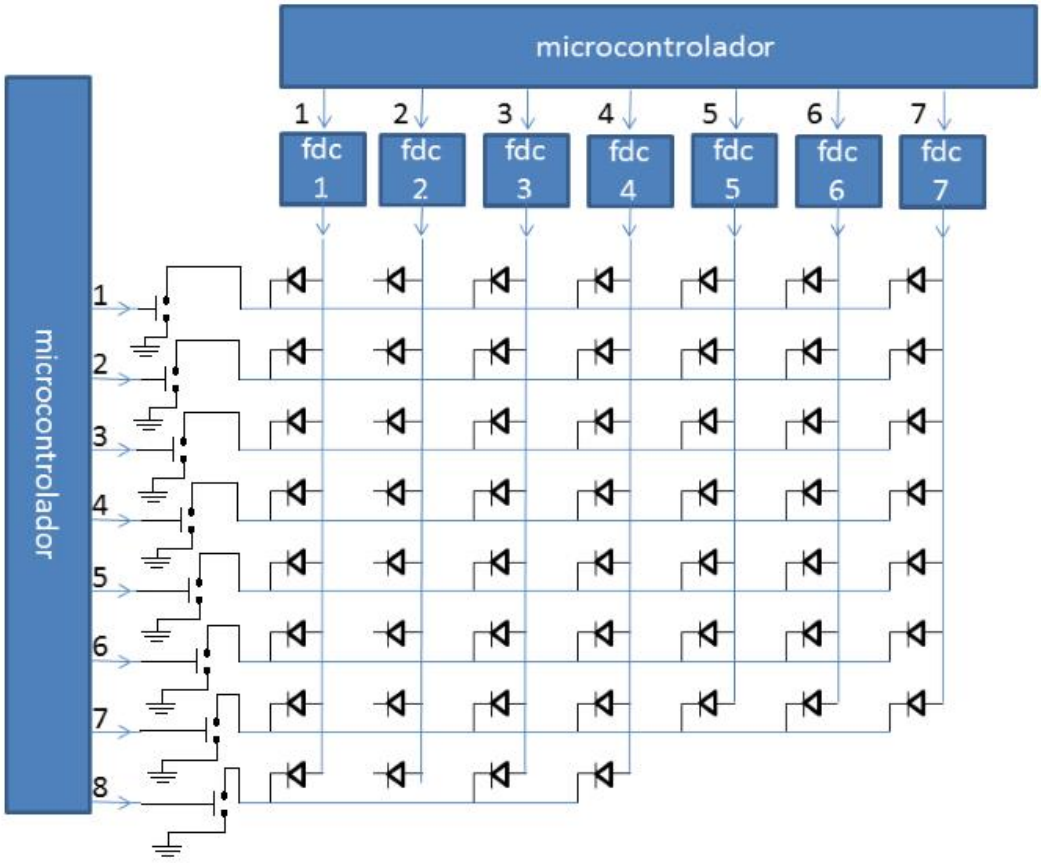


Figura 2.4: Matriz de leds

2.1.1 Generación de señal

Para que el sensor utilizado distinga entre la luz ambiental y la señal en ambientes ruidosos, esta debe de ser modulada previo a ser enviada, lo que ayuda a reducir el efecto que tiene la luz ambiental en el sensor.

Una de las tareas que realiza el microcontrolador utilizado en este proyecto, es producir una señal cuadrada de 10 KHZ de frecuencia y 3.3 volts de amplitud, y con una duración de 500 μs, lo que genera cinco pulsos a la frecuencia mencionada.

Para seleccionar qué led parpadea, el microcontrolador utiliza dos puertos de salida. Como se muestra en la figura 2.4, uno de los puertos envía la señal de 10 KHZ a la fuente de corriente correspondiente a la columna de la matriz de leds que se desea excitar.

Por otro lado, el microcontrolador utiliza el otro puerto para conectar la referencia de voltaje a la fila de la matriz de leds que corresponda.

2.1.2 Fuentes de corriente

Esta etapa es la encargada de proporcionar de manera constante, la corriente eléctrica requerida por los leds.

La intensidad de luz emitida por el led depende directamente de la corriente que lo alimenta, ya que la salida de luz es casi lineal respecto a la corriente dentro de su región activa como se muestra en la figura 2.5 de la hoja de especificaciones del led APA3010F3C. Por lo que, si queremos que la luz emitida por cada led al ser excitado por la señal de 10KHZ mantenga características similares, debemos garantizar que la corriente mantenga las mismas características en cada uno de ellos, o lo que es lo mismo, se requiere garantizar que el flujo de corriente sea constante en cada led.

La fuente de corriente se muestra en la figura 2.6, la cual es calculada haciendo la consideración que los transistores $Q2$ y $Q3$ están saturados, ya que estos funcionan como interruptores para que el microcontrolador active o desactive la fuente de corriente.

El análisis es como sigue:

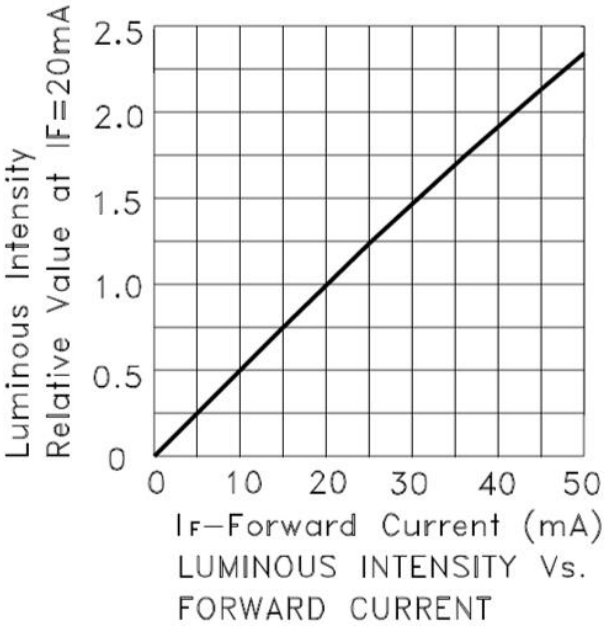


Figura 2.5: LED APA3010F3C - Corriente vs intensidad luminosa

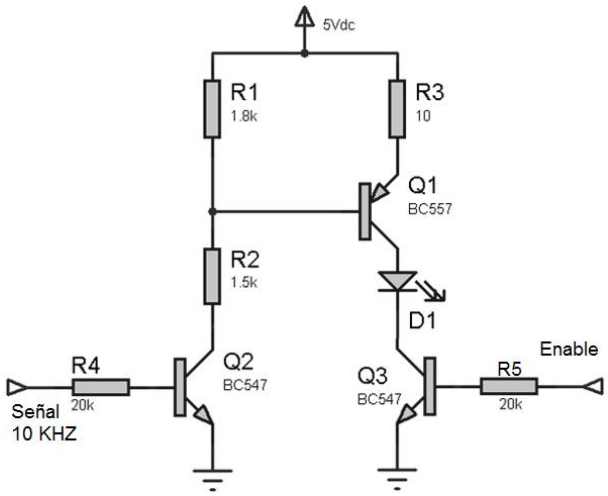


Figura 2.6: Fuente de corriente de 202 mA

El voltaje respecto a tierra, de la base del transistor $Q1$ es un divisor de voltaje formado por los resistores denominados $R1$ y $R2$

$$V_{B(Q1)} = \frac{(V_{CC} - V_{CE(Q2)})R_2}{R_1 + R_2} \tag{2.1}$$

El voltaje en el emisor del transistor $Q1$ es

$$V_{E(Q1)} = V_{B(Q1)} + V_{BE(Q1)} \tag{2.2}$$

Sustituyendo (2.1) en (2.2) se tiene que el voltaje del emisor del transistor $Q1$ es:

$$V_{E(Q1)} = \frac{(V_{CC} - V_{CE(Q2)})R_2}{R_1 + R_2} + V_{BE(Q1)} \tag{2.3}$$

El voltaje a través del resistor $R3$ es:

$$V_{R3} = (R_3)(I_{R3}) = V_{CC} - V_{E(Q1)} \tag{2.4}$$

Sustituyendo la ecuación (2.3) en (2.4) se tiene que:

$$V_{R3} = V_{CC} - \frac{(V_{CC} - V_{CE(Q2)})R_2}{R_1 + R_2} + V_{BE} \tag{2.5}$$

Por otro lado, sabemos que la corriente que circula por el resistor $R3$ es:

$$I_{R3} = \frac{V_{R3}}{R_3} \tag{2.6}$$

y que la corriente del led es aproximadamente igual a la corriente que circula por el resistor R3 que se puede expresar como:

$$I_{LED} \cong IR_3 \tag{2.7}$$

De las ecuaciones (2.5), (2.6) y (2.7) obtenemos que:

$$I_{LED} = \frac{V_{CC} - \frac{(V_{CC} - V_{CE(Q2)})R_2}{R_1 + R_2} + V_{BE}}{R_3} \tag{2.8}$$

Fijando los siguientes valores:

$$\begin{aligned} V_{CC} &= 5Vdc \\ V_{BE} &\cong 0.7Vdc \\ R_1 &= 1.8K\Omega \\ R_2 &= 1.5K\Omega \\ I_{LED} &= 202mA\Omega \end{aligned}$$

Despejando $R3$ de (2.8) y sustituyendo los valores seleccionados, obtenemos que el valor del resistor $R3$ es de:

$$R_3 \cong 10\Omega$$

Utilizando los valores de los componentes calculados, así como los transistores $BC547$, se realizaron las 7 fuentes de corriente para excitar las 7 columnas de la matriz de leds.

2.1.3 Emisores

Considerando que actualmente la mayoría de los sensores fotoeléctricos utilizan leds como fuentes de luz, debido entre otros factores a su eficiencia energética, linealidad y velocidad de respuesta, se decidió que la transmisión fuera realizada por medio de leds infrarrojos (led IR). El modelo seleccionado es $APA3010F3C$, de la marca *King Bright*.

Un led es un semiconductor, eléctricamente similar a un diodo, pero con la característica que emite luz cuando una corriente circula por él en forma directa. Los leds pueden ser construidos para que emitan en diferentes colores como verde, azul, amarillo, rojo, infrarrojo, etc. Los colores más comúnmente usados en aplicaciones de detección son rojo e infrarrojo, aunque en aplicaciones donde se necesite detectar contraste, la elección del color de emisión suele ser el verde.

Los leds son utilizados ampliamente como indicadores visuales en muchos dispositivos, aunque en los últimos años se ha popularizado su uso en iluminación debido a su alta eficiencia.

Al inicio de su historia, los leds emitían únicamente luz roja de baja intensidad, pero los dispositivos actuales son capaces de emitir luz de alto brillo tanto en el espectro visible como en el infrarrojo, y ultravioleta.

Debido a su capacidad de operación a altas frecuencias, son también útiles en tecnologías avanzadas de comunicaciones y control. Los leds infrarrojos son a menudo utilizados en unidades de control remoto de muchos productos comerciales incluyendo equipos de audio y video.

Para el diseño de esta aplicación se utilizaron 53 leds distribuidos en el contorno de la pantalla, colocados con una separación de media pulgada entre cada uno de ellos. La pantalla mide 12 pulgadas de alto, por 19.5 pulgadas de ancho. La distribución de los leds, así como de los sensores, se muestra en la figura 2.1.

2.2 Etapa de sensado

Consta de sensores infrarrojos alineados frente a cada uno de los leds, estos sensores detectan si existe un objeto opaco en la línea directa entre emisor y sensor. El sensor recibe la señal lumínica emitida por el led posicionado frente a él, la transforma en una señal eléctrica y la envía hacia la siguiente etapa del sistema, para ello se conectan los 53 sensores fotoeléctricos a un multiplexor de 64 a 1.

2.2.1 Sensores

Para detectar el haz de luz emitido por los leds se utiliza el fototransistor $APA-3010P3C$, de la marca *Kingbright*, ya que sus características tanto mecánicas

como espectrales están diseñadas para trabajar con el led $APA3010F3C$ utilizado en la etapa de transmisión.

El sensor debe ser capaz de leer la señal emitida por los leds, lo que significa que su velocidad de respuesta debe ser menor a 50 μs. La velocidad de respuesta del fototransistor $APA3010P3C$ depende del resistor de carga que se conecte en su emisor, el cual se muestra en la figura 2.7. En la figura 2.7 obtenida de su hoja de especificaciones, se muestra la relación que existe entre ambos, en la cual se observa que mientras mayor sea el valor del resistor, mayor será el tiempo de respuesta, o lo que es lo mismo, el sensor reaccionará de forma más lenta.

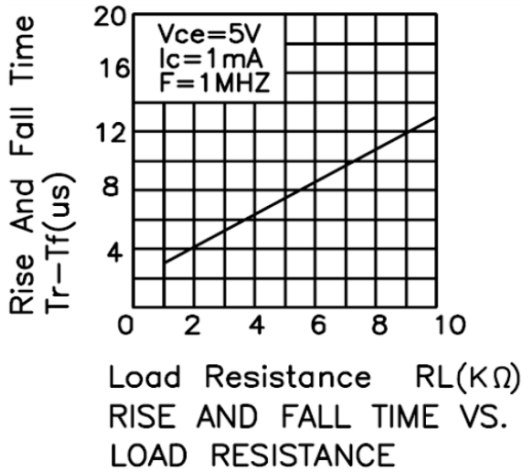


Figura 2.7: Tiempo de respuesta vs resistor de carga - Fototransistor *APA3010P3C*

Considerando las necesidades del sistema se decide utilizar un resistor de 10 $K\Omega$, lo que nos da una velocidad de respuesta de 13 μs, que es suficiente para manejar la señal recibida de 10 Khz.

En la figura 2.8, se muestra el circuito de polarización de cada fototransistor. Para calcular la potencia que disipará el resistor denominado $R1$, se hace una malla de voltaje de la fuente de alimentación a tierra.

$$V_{CC} - V_{CE(Q1)} - V_{R1} = 0 \tag{2.9}$$

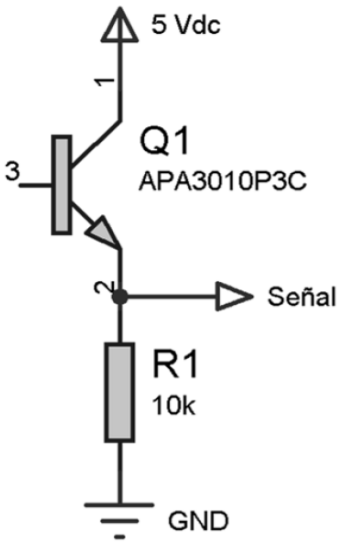


Figura 2.8: Circuito de polarización - Fototransistor *APA3010P3C*

Donde:

$$V_{CE(Q1)} = 0.8dc$$
$$V_{CC} = 3.3Vdc$$
$$V_{R1} = R_1 I_{R1}$$

Sustituyendo los valores correspondientes en (2.9) y despejando I_{R1}:

$$I_{R1} = \frac{3.3 - 0.8}{10x10^3} = 250\mu A \tag{2.10}$$

La potencia disipada por $R1$ es:

$$P_{R1} = (R_1)(I_{R1})^2 \cong (2.5V)(250X10^{-}6) = 625\mu A \tag{2.11}$$

Por lo que R_1 debe ser un resistor de 10 $K\Omega$ a una potencia de $1/4$ Watt.

2.2.2 Multiplexado

Para conducir las señales de los 53 sensores al convertidor analógico-digital del microcontrolador se necesita un multiplexor de 53 a 1, esto porque únicamente se utiliza un convertidor analógico-digital del microcontrolador.

Debido a que en el mercado actual no existen multiplexores de 53 a 1, se tuvo la necesidad de crear uno con componentes discretos. Tomando en cuenta que la señal a transportar es de naturaleza analógica, se utilizan multiplexores analógicos para su construcción, en este caso concreto el multiplexor 8 a 1 modelo 74HC4051.

Para crear un multiplexor 53 a 1, se requieren 7 multiplexores 8 a 1 conectados en paralelo. Para realizar el control de los 7 multiplexores 8 a 1 se utiliza un decodificador de binario a decimal, de esta manera, con 3 bits se tiene la posibilidad de habilitar o deshabilitar cualquiera de los 8 multiplexores, en consecuencia, en cada momento sólo hay una señal en el convertidor analógico-digital, de las 53 posibles señales. Y con otros 3 bits se controla la salida específica del multiplexor seleccionado. El diagrama electrónico del diseño se muestra en el anexo A.2.

El multiplexor utilizado es el modelo 74HC4051. Es un multiplexor analógico, triestado, lo que nos garantiza que cuando está deshabilitado su salida es puesta en alta impedancia, esta característica es indispensable debido a que se conectan las salidas de los 7 multiplexores a un mismo punto, en consecuencia, Sólo una de ellas debe estar activa a la vez.

El multiplexor cuenta con 3 pines; $S0$, $S1$, $S2$, para seleccionar qué entrada "Y" se conectará con la salida "Z". Su símbolo lógico se muestra en la figura 2.9 (a). Adicionalmente, el multiplexor cuenta con un pin de habilitación, en el símbolo lógico se muestra como el pin "E", por medio de este pin se puede poner la salida del multiplexor en alta impedancia.

El decodificador utilizado es el modelo 74F138. En la figura 2.9 (b) se muestra su símbolo lógico. Por medio de los pines llamados $A0$, $A1$, y $A2$, se selecciona la salida que se desea poner a un estado lógico 1. Las salidas son mutuamente excluyentes, por lo que sólo una de ellas estará activa a la vez. Los pines $E0$, $E1$, $E2$, son utilizados para habilitar el decodificador. Los pines $Q0$ hasta $Q7$ son las salidas.

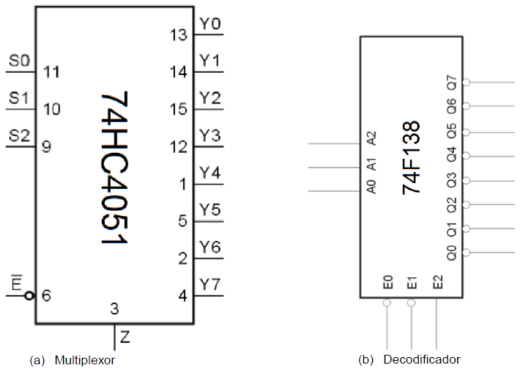


Figura 2.9: Elementos del multiplexor 53 a 1

Las salidas $Q0,\ Q1, ..., Q6$, del decodificador se conectan respectivamente a la entrada "E" de cada multiplexor; la salida $Q0$ se conecta a la entrada "E" del primer multiplexor, la salida $Q1$ se conecta a la entrada "E" del segundo multiplexor, y así sucesivamente hasta la salida $Q6$ que se conecta a la entrada "E" del séptimo multiplexor. La salida $Q7$ se deja sin conectar.

De tal manera que, si deseamos activar el primer multiplexor, debemos introducir el valor "000" en las entradas $A0,\ A1,\ A2$ del decodificador.

Y si queremos leer el sensor conectado en la entrada $Y0$ de uno de los multiplexores, debemos introducir el valor "000" en las entradas $S0,\ S1,\ S2$ de dicho multiplexor.

Posteriormente, las entradas $S0,\ S1,\ S2$ de los siete multiplexores se conectaron entre sí, de tal forma que el mismo dato les llega a los siete multiplexores al mismo tiempo. Por lo que, cuando se introduce el valor "000", los siete multiplexores conectan su entrada $Y0$ a su salida "Z".

Las salidas "Z" de los siete multiplexores también se conectaron entre sí, ya que esta es la salida general de esta etapa.

En el anexo A.2 se muestra el diagrama esquemático del multiplexor. A continuación, se explica su funcionamiento:

Cuando se quiere leer el primer sensor del sistema, el microcontrolador debe poner en los pines $A0$, $A1$, $A2$ del decodificador el valor "000", y en los pines $S0$, $S1$, $S2$, de los multiplexores (recordar que los 7 están conectados en paralelo) el valor "000".

Al poner el valor "000" en la entrada del decodificador, su salida $Q0$ toma el valor lógico 1; ésta salida está conectada al pin de habilitación "E" del primer multiplexor, en consecuencia, el primer multiplexor queda habilitado. Debido a que las salidas del decodificador son mutuamente excluyentes, todas las demás salidas toman un valor lógico 0, lo que a su vez, deshabilita los demás multiplexores.

Y al poner el valor "000" en la entrada de los multiplexores, todos los multiplexores conectan su entrada $Y0$ a su salida "Z", pero como el único multiplexor que está habilitado es el número 1, este es el único que realmente tiene su salida conectada y los demás tienen su salida en alta impedancia. Por consiguiente, el único dato que llega a la salida del multiplexor de 53 a 1, es el valor de voltaje del sensor número 1.

Finalmente, podemos comentar que para conectar cualquier sensor con la etapa siguiente (acondicionamiento de señal) el microcontrolador sólo debe escribir el número del sensor que se desea leer, en formato binario en un puerto de 6 bits. Tomando en consideración que las señales $A0$, $A1$, $A2$ son los bits menos significativos, y las señales $S0$, $S1$, $S2$ son los bits más significativos. Otra consideración a tomar en cuenta es que la numeración de los sensores inicia en cero.

2.3 Etapa de acondicionamiento de señal

El diseño de esta etapa se divide en dos secciones: El filtrado, y la amplificación. Ya que para acondicionar la señal proveniente del multiplexor son necesarias estas dos acciones, para posteriormente llevar a cabo el procesamiento de la información.

En la etapa de filtrado se toma en cuenta que la frecuencia de las señales provenientes de los sensores es de 10 KHZ. Por lo cual, se plantea el diseño de un filtro pasivo paso-altas con una frecuencia de corte de 1.5 KHz, ya que se considera que

es suficiente para atenuar las señales de ruido provenientes de la luz ambiental, la consideración de la frecuencia de corte se hizo de forma empírica.

Se decidió utilizar un filtro pasivo en lugar de uno activo porque se consideró que este tipo de filtro es suficiente para las necesidades de filtrado del sistema, además de que su implementación es más sencilla.

Para la fase de amplificación se utilizan dos etapas amplificadoras conectadas en cascada. Cada una de ellas implementada con un amplificador operacional.

Cada uno de los cuales tiene una ganancia variable para permitir seleccionar la ganancia dependiendo del lugar de procedencia de la señal ya sea que proceda de los sensores dispuestos en forma horizontal o que proceda de los sensores dispuestos en forma vertical. Esto se debe a que los monitores no son cuadrados sino rectangulares, por lo tanto, en el acondicionamiento de la señal horizontal se requiere una mayor amplificación ya que el receptor está a una distancia mayor del receptor que en el caso vertical.

Para salvar este inconveniente, el segundo amplificador tiene una ganancia que varía por medio de un transistor que sirve como interruptor para habilitar o deshabilitar el resistor denominado $R7$ del circuito de la figura 2.10, lo que permite que la ganancia del segundo amplificador varíe entre una u otra ganancia dependiendo de las necesidades de la situación.

2.3.1 Filtro pasivo y *offset*

La primera parte del circuito de la figura 2.10, es un filtro pasivo RC, paso-altas de primer orden, con una frecuencia de corte de 1.5 KHz, formado por el capacitor denominado C_1 y el resistor denominado R_3.

Primero obtendremos la función de transferencia del filtro, posteriormente se obtendrá su respuesta en frecuencia.

En la figura 2.11, se muestra el esquemático del filtro.

Las relaciones de voltaje y corriente de los componentes son las siguientes:

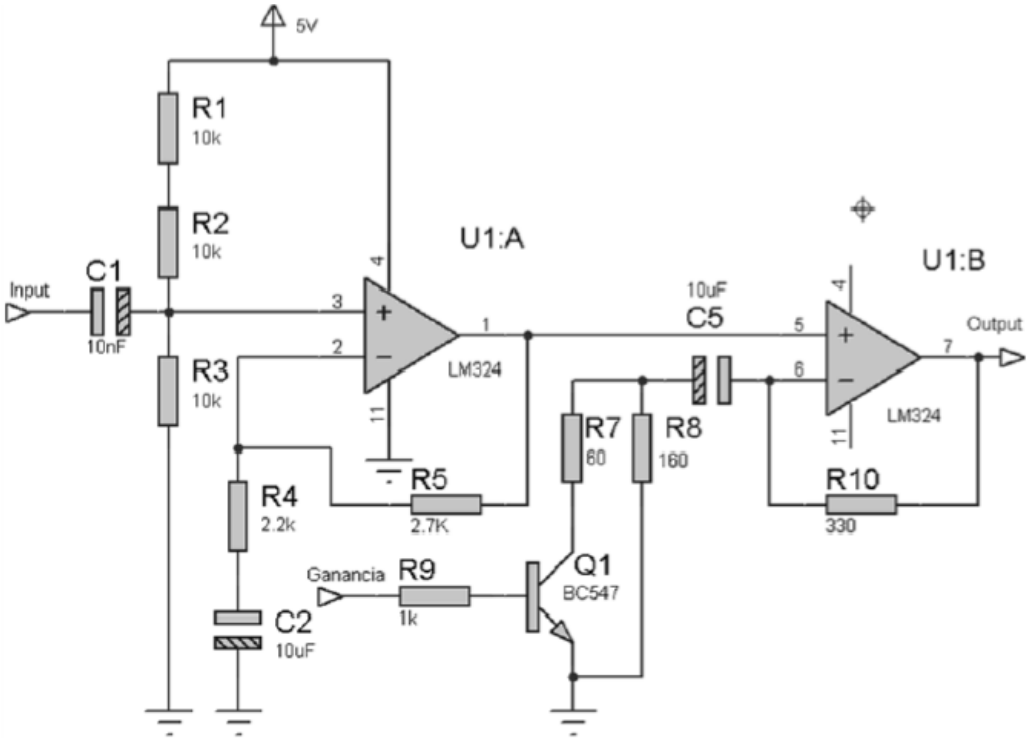


Figura 2.10: Circuito de filtrado y amplificación

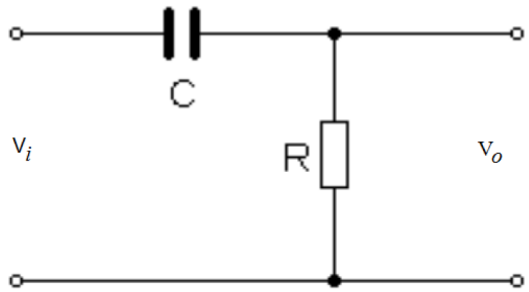


Figura 2.11: Esquemático del filtro pasivo

Resistencia:

$$V_R = R_3 I_R \tag{2.12}$$

$$I_{R3} = \frac{V_R}{R} \tag{2.13}$$

Capacitor:

$$V_C = \frac{1}{C}\int_0^t I_C \mathrm{d}t \tag{2.14}$$

$$I_C = C\frac{dV_C}{\mathrm{d}t} \tag{2.15}$$

De la figura 2.11 se hace una malla de voltaje:

$$V_i = V_C + V_R \tag{2.16}$$

Sustituyendo (2.14) en (2.16).

$$V_i = \frac{1}{C}\int_0^t I_C \mathrm{d}t + V_R \tag{2.17}$$

Se sabe que en un circuito de una sola malla la corriente es la misma en todos sus elementos, por lo que, la corriente en el capacitor es la misma que la corriente en la resistencia, tomando esto en consideración, se sustituye (2.13) en (2.17).

$$V_i = \frac{1}{C}\int_0^t \frac{V_R}{R} dt + V_R \tag{2.18}$$

Se deriva (2.18)

$$\frac{d}{dt}V_i = \frac{1}{C}\frac{V_R}{R} + \frac{d}{dt}V_R \tag{2.19}$$

Se aplica la transformada de Laplace a (2.19).

$$V_i(s)s = \frac{V_R(s)}{RC} + V_R(s)s = V_R(s)(\frac{1}{RC} + s) \tag{2.20}$$

Se multiplica (2.20) por RC y se acomoda la ecuación para obtener la función de transferencia del filtro.

$$F.T = \frac{V_R(s)}{V_i(s)} = \frac{RC_s}{1 + RC_s} \tag{2.21}$$

Para obtener la frecuencia de corte, se utiliza (2.22).

$$f_c = \frac{1}{2\pi R_3 C_1} \tag{2.22}$$

Al fijar los valores de la frecuencia de corte y el capacitor, y despejar el resistor denominado $R3$, se obtiene que su valor es:

$$R_3 = \frac{1}{2\pi(10x10^{-}9)(1.5x10^3)} = 10610.33\Omega \tag{2.23}$$

$$R_3 \cong 10\ k\Omega \tag{2.24}$$

Debido a que la frecuencia de corte no es un valor crítico en el diseño de este sistema, podemos utilizar un valor comercial próximo para el resistor, como lo es 10 $K\Omega$, sin que se vea comprometido el desempeño global del sistema.

En la figura 2.12 se muestra el diagrama de Bode utilizando una resistencia de 10 $K\Omega$ con una tolerancia del 10% . Se marca el punto donde la salida del filtro toma el valor de $-3\ dB$, que corresponde con la frecuencia de corte de 1.59 KHz.

Además de filtrar la señal la entrada del circuito se encarga de quitar cualquier componente de DC que pudiera traer y le agrega un offset de 1.66 V, ya que se

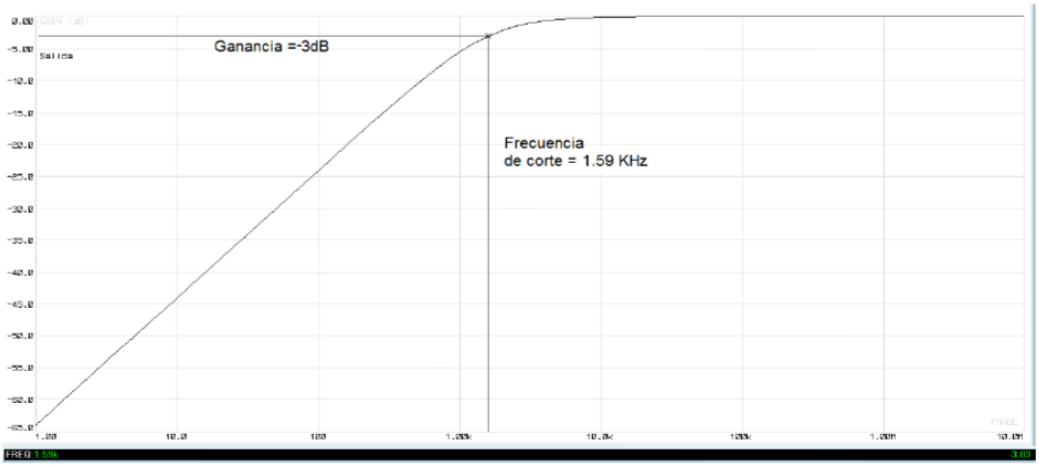


Figura 2.12: Diagrama de bode del filtro

forma un divisor de voltaje con los resistores denominados $R1$, $R2$ *y* $R3$, lo cual centra la señal justo a la mitad del rango dinámico del convertidor analógico-digital del microcontrolador, que es, el siguiente punto al que se dirige la señal.

2.3.2 Amplificador

Inmediatamente después del filtro se encuentra un primer amplificador operacional, como se puede observar de la figura 2.10, el cual tiene una configuración no-inversora. Se realizaron pruebas experimentales para determinar la ganancia que debían tener los amplificadores operacionales de cada etapa.

La ganancia del primer amplificador operacional es:

$$A_V 1 = \frac{R_5}{R_4} + 1$$
$$A_{V1} = \frac{2.7x10^3\Omega}{2.2x10^3\Omega} + 1 \cong 2.23Vdc \tag{2.25}$$

El segundo amplificador operacional que tiene el circuito diseñado, también cuenta con una configuración no-inversora, aunque presenta una gran diferencia

respecto al primero, ya que, en este caso, el resistor denominado $R7$ puede desconectarse por medio del transistor denominado $Q1$, de tal forma, que el amplificador operacional puede conmutar entre dos ganancias diferentes.

Para asegurarnos que la resistencia de la unión colector-emisor del transistor denominado $Q1$ es despreciable en su estado de saturación, recurrimos a la hoja de especificaciones del transistor $BC547$, que en el apartado "características eléctricas" se encuentra la información del voltaje colector-emisor de saturación, así como la corriente de colector, las cuales se reproducen en la tabla 2.1:

Símbolo	Parámetro	Condiciones	Típ.	Máx.	Unidad
$V_{CE(sat)}$	Voltaje colector-emisor de saturación	$I_C = 10mV, I_B = 0.5mA$	90	250	mV
		$I_C = 100mV, I_B = 5mA$	250	600	mV
$V_{BE(sat)}$	Voltaje base-emisor de saturación	$I_C = 10mV, I_B = 0.5mA$	700	-	mV

Tabla 2.1: Características eléctricas del transistor $BC547$

Con estos valores se calcula la resistencia de la unión colector-emisor cuando el transistor está en saturación.

La hoja de especificaciones muestra el voltaje colector-emisor de saturación del transistor cuando se le aplica una corriente de base de 0.5 y 5 mA respectivamente. Analizamos ambas situaciones, ya que como veremos más adelante, nuestro caso se encuentra dentro de este rango de corriente.

El primer caso es cuando la corriente aplicada a la base es de 0.5 mA, que como podemos ver en la tabla 2.1 produce un voltaje colector-emisor típico de 90 mV, y máximo de 250 mV, analizaremos el peor de los casos, ya que es cuando se genera una mayor resistencia en la unión.

$$A_{CE(max)} = R_{CE(max)} I_C \tag{2.26}$$

Despejando $R_{CE(max)}$ de la ecuación (2.26) y sustituyendo los valores obtenidos de la tabla 2.1 obtenemos:

$$R_{CE(max)} = \frac{250x10^{-3}V}{10x10^{-3}A} = 2.5m\Omega \tag{2.27}$$

Para el caso en que la corriente de base es de 5 mA, tomamos los datos de la tabla 2.1 y utilizamos el peor de los casos; cuando se genera un voltaje de 600 mV en la unión colector-emisor. Introduciendo los datos en (2.26), tenemos que:

$$R_{CE(max)} = \frac{600x10^{-3}V}{100x10^{-3}A} = 60m\Omega \tag{2.28}$$

De los datos obtenidos en (2.27) y (2.28) se puede observar que la resistencia de la unión colector-emisor cuando el transistor está en saturación está en el rango de 2.5 $m\Omega$ a 60 $m\Omega$ cuando la corriente de su base está en el rango de 0.5 mA a 2.5 mA.

A continuación, calculamos la corriente que circula por la base del transistor denominado Q_1, para asegurarnos que está dentro del rango de corrientes que analizamos.

El microcontrolador es el encargado de activar o desactivar el transistor denominado Q_1 de la figura 2.10. Por la hoja de especificaciones del microcontrolador $MCF52223$, se sabe que la amplitud de voltaje producida por un 1 lógico es de 3.3 V. Para calcular la corriente que circula por la base del transistor se hace una malla de voltaje como se muestra en la figura 2.13 (extraída de la figura 2.10).

$$V_{micro} - I_B R_9 - V_{BE} = 0 \tag{2.29}$$

Despejando I_B de (2.29) y sustituyendo el valor de la unión base-emisor, obtenido de la hoja de especificaciones del transistor $BC547$ y que se reproduce en la tabla 2.1, así como un valor especificado de 1 k del resistor denominado $R9$, tenemos que:

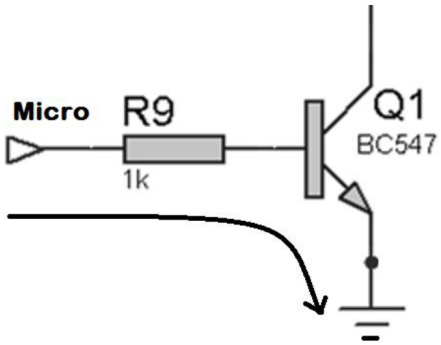


Figura 2.13: Malla de voltaje

$$I_B = \frac{3.3V - 0.7V}{1X10^3\Omega} = 2.6mA \tag{2.30}$$

La corriente de la base del transistor Q_1 está dentro del rango de 0.5 mA y 5 mA, así que la resistencia de su unión emisor-colector está en el rango de 2.5 $m\Omega$ a 60 $m\Omega$.

En conclusión, después de calcular la resistencia que se genera en la unión colector-emisor del transistor denominado Q_1 cuando está en saturación, estamos en condiciones de poder despreciar su resistencia; ya que su valor es mucho menor que el del resistor denominado R_7, y por lo tanto no afecta en el cálculo de la ganancia del amplificador operacional denominado $U1:B$.

A continuación, calculamos la ganancia de voltaje del segundo amplificador operacional, primero hacemos la suposición que el transistor Q_1 está encendido, con lo cual, el resistor denominado R_7 queda en paralelo con el resistor denominado R_8, y despreciando la resistencia que hay entre colector-emisor del transistor Q_1, calculamos la ganancia de voltaje:

$$A_{V1} = \frac{R_{10}}{R_7 \parallel R_8} + 1 \tag{2.31}$$

Donde $\|$ significa que estos dos resistores están en paralelo.

Por lo tanto, la ganancia resulta:

$$A_{V1} = \frac{330\Omega}{43.64\Omega} + 1 \cong 7.6 \tag{2.32}$$

Cuando el transistor denominado Q_1 está abierto, el resistor denominado R_7 queda desconectado y sólo se toma en cuenta el resistor denominado R_8 en la ganancia del amplificador, quedando como sigue:

$$A_{V2} = \frac{R_{10}}{R_8} + 1 \tag{2.33}$$

Por lo que se tiene que la ganancia es:

$$A_{V1} = \frac{330\Omega}{160\Omega} + 1 \cong 3 \tag{2.34}$$

2.4 Etapa de procesamiento

La etapa de procesamiento, es la encargada de administrar el circuito electrónico de la pantalla táctil. Controla, desde el encendido de un led, hasta el envió de las coordenadas hacia la computadora personal.

Para su creación, se utiliza un microcontrolador de 32 *bits* de la compañía *FreeScale*, modelo $MCF52223$. Su elección se basó en la frecuencia de trabajo de su reloj; que como puede verse en sus características listadas más adelante, es de 80 MHz, lo que, según su hoja de especificaciones, ejecuta hasta 76 millones de instrucciones por segundo, al funcionar a máxima velocidad. De esta manera, aseguramos que tiene el poder de procesamiento necesario para controlar todo el sistema.

Entre las herramientas de desarrollo que ofrece la empresa *FreeScale* para la programación de este microcontrolador, se tiene un Entorno de Desarrollo Integrado (*Integrated Development Environment o IDE* por sus siglas en inglés) amigable con el usuario, el cual incluye una herramienta gráfica para configurar los periféricos del microcontrolador, de tal forma que con unos cuantos *clicks* del ratón se tienen congelados el convertidor analógico-digital, así como los contadores, y el módulo de comunicación serial (*Universal Asynchronous/synchronous Receiver/Transmitter o UART* por sus siglas en inglés).

Este microcontrolador pertenece a la familia *ColdFire*, la cual tiene una arquitectura de tipo Conjunto de Instrucciones Reducida (*Reduced Instruction Set Computing o RISC* por sus siglas en inglés).

Sus características son:

- Núcleo $V2$ *ColdFire*.
- Reloj del sistema de $80MHZ$.
- 32 Kbytes de SRAM interna.
- 256 Kbytes de memoria flash.
- USB *On-The-Go (USB OTG) full speed/low speed.*
- 3 *UARTs.*
- 1 controlador de I2C.
- ADC de $12-bit$.
- Reloj en tiempo real.
- Módulo QSPI.
- 4 canales de 32-bit DMA.
- 4 canales de 32-bit timers de propósito general con soporte opcional de DMA.
- 2 timers de interrupción periódica de 16 bits (PITs).

- *Watchdog timer* programable por software.
- Hasta 63 fuentes de interrupción.
- Módulo de reloj de 8 *MHZ* y PLL integrado.

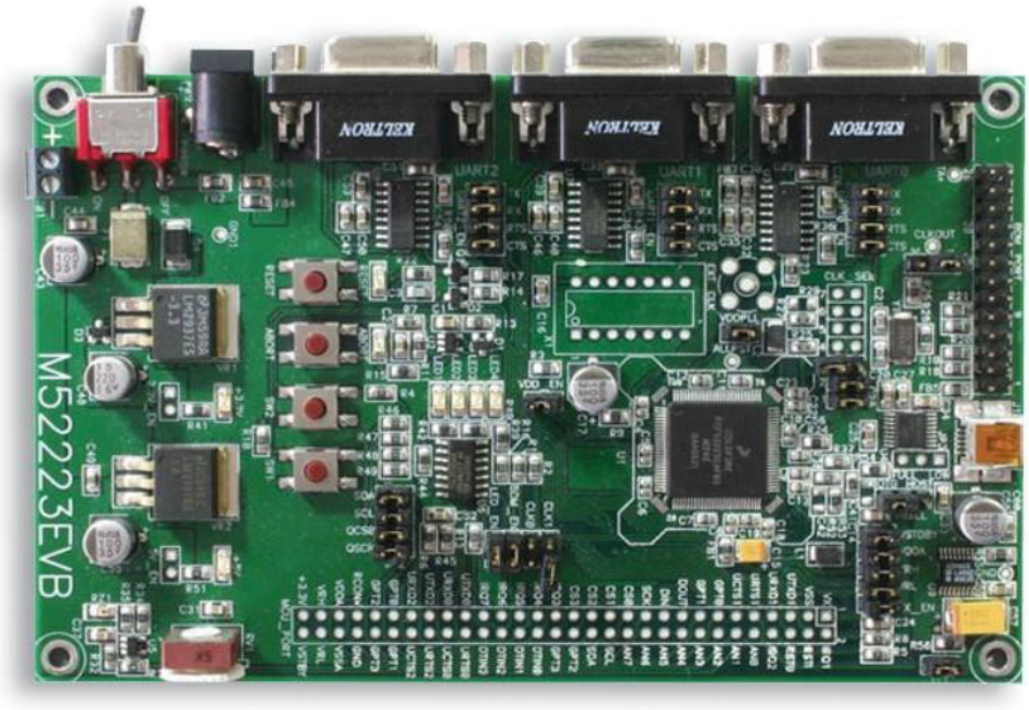


Figura 2.14: Tarjeta de desarrollo M52223EVB

De las cuales podemos resaltar el convertidor analógico-digital de 12 bits, el reloj del sistema de 80 *MHZ*, así como los contadores de 16 bits que son los que utilizaremos.

El costo del microcontrolador en el año 2016, por el proveedor internacional de componentes electrónicos *"Mouser"* es de USD $ 13.90 por pieza, mientras que por el proveedor *"Newark"* es de USD $ 5.94 por cada uno.

Para acelerar la creación del prototipo, se decide utilizar una tarjeta de desarrollo de la misma empresa *FreeScale*, modelo *M52223EVB* en la que ya viene

montado el microcontrolador, así como todo lo necesario para su utilización y programación. En la figura 2.14 se muestra su apariencia física.

Las características de la tarjeta de evaluación son:

- Dimensiones: 812 $mmx139\ mm$.
- Alimentación: de 7 a 15 Vdc.
- CPU MCF52221, de 100 pines con empaquetado LQFP.
- Puerto BDM para programación.
- Cristal de 48 MHZ.
- 3 Puertos seriales $RS-232$ con conectores DB9-S.
- 40 pines de E/S.
- Puerto miniUSB para USB *On-the-GO*.
- Switch de encendido/apagado con led indicador.
- Switch de reset con indicador.
- Regulador de voltaje de 3.3 y 5.0 volts.
- 4 leds indicadores.
- 2 botones de presión.
- 1 potenciómetro de 5 K.
- 1 sensor de luz.

El costo de esta tarjeta de desarrollo en el año 2016, consultado con el proveedor internacional de componentes electrónicos *"Mouser"* es de USD $ 388.83 por pieza, mientras que, con el proveedor *"Newark"* es de USD $ 299.10 por pieza.

Como se puede apreciar, el costo de la tarjeta es elevado, y su justificación se basa en la considerable disminución del tiempo de desarrollo y creación del prototipo. Una vez se tenga un prototipo funcional, se debe reemplazar la tarjeta de desarrollo por una tarjeta *PCB* donde se integrarán todas las etapas del sistema,

para disminuir tanto el tamaño del prototipo como su costo de construcción.

Para controlar el sistema, se diseña un programa en lenguaje ”C” encargado de gestionar cada una de las etapas del circuito electrónico.

Para configurar los módulos periféricos, como el convertidor analógico-digital, el puerto de comunicación UART, y los contadores, se utiliza *“Processor expert”* el cual, es una herramienta del *IDE Codewarrior* que sirve para configurar los módulos del microcontrolador desde un ambiente gráfico.

El programa principal, además de configurar los puertos y módulos, lleva el control del encendido y apagado de los leds de la matriz de leds, además, realiza el promedio de las lecturas de cada led, guardándolo en su correspondiente variable. Procesa la información de todos los leds para obtener las coordenadas “(x,y)” del punto donde se haya detectado el contacto con la pantalla. Por último, envía las coordenadas a la computadora por medio del puerto serial.

2.4.1 Función *Main*

Cada programa estructurado en lenguaje C, tiene una función principal que se debe llamar *main*, la cual sirve como punto de partida para la ejecución del programa. En la figura 2.15 se muestra el diagrama de flujo de la función principal diseñado para administrar el sistema, el código fuente generado a partir del diagrama de flujo se encuentra en el Anexo A.3, su funcionamiento es como sigue:

El programa inicializa las variables antes de comenzar un ciclo.

Compara la cantidad de leds que han parpadeado hasta el momento contra el número de leds que se desea que parpadeen.

Si aún no han parpadeado todos, se inicia el contador 1; el cual genera una interrupción cada 50 μs, de esta forma se genera una frecuencia de 10 KHz. Se espera que se ejecute el número de parpadeos indicado en la constante “Lecturas x led” para apagar el contador 1.

Se calcula el promedio del voltaje pico-pico de las lecturas realizadas y se decide si corresponde a un led colocado horizontal o verticalmente en la pantalla. Se

incrementa la variable “Foto” para que en el siguiente ciclo se encienda y apague el siguiente led.

Al terminar de parpadear los leds indicados en la variable “numero leds” el programa procesa la información guardada en los arreglos “Horizontal” y “Vertical” para que, en caso de que exista un objeto sobre la pantalla, obtenga sus coordenadas correspondientes.

Posteriormente envía la información a la interfaz gráfica de usuario por medio del puerto serial.

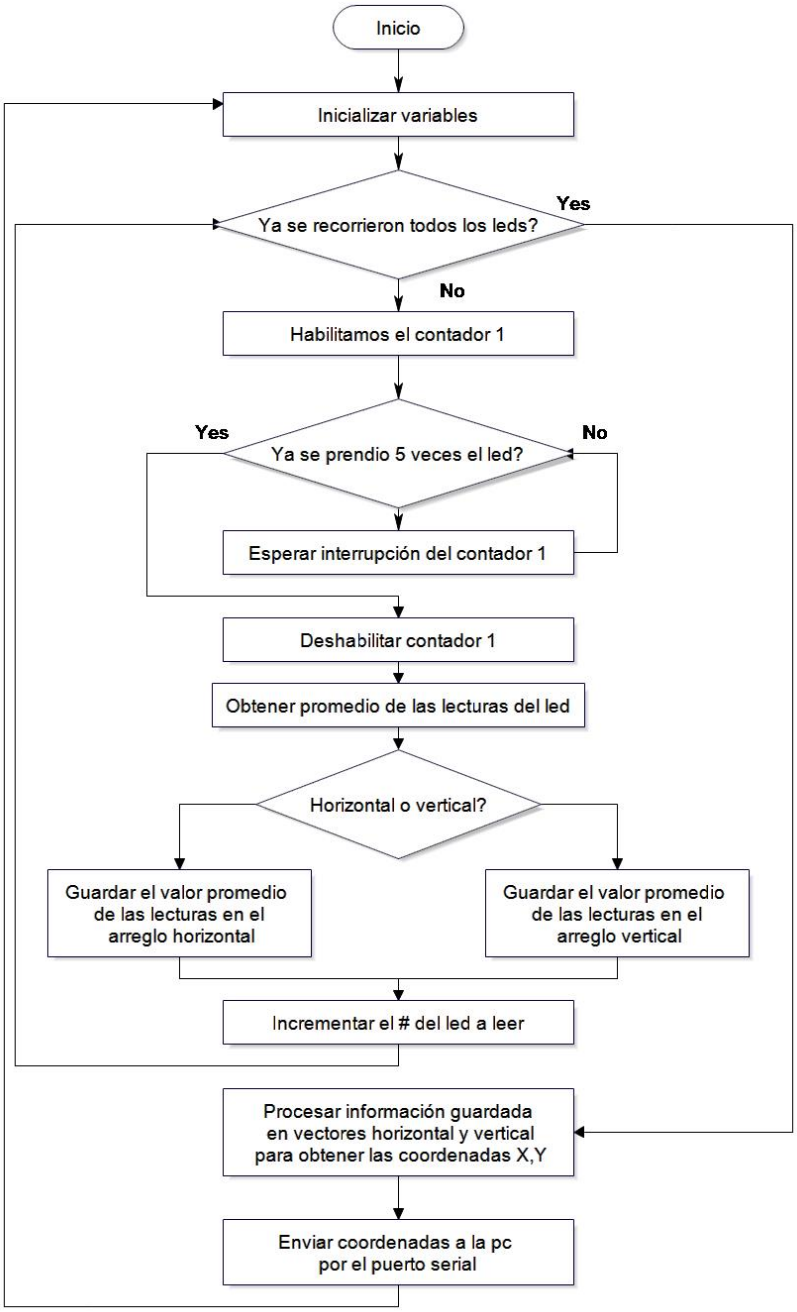


Figura 2.15: Diagrama de flujo del programa principal

El criterio que utiliza la función main para decidir si existe un objeto dentro del área de la pantalla, es la lectura de cada sensor. Para discriminar entre objeto existente y no existente, se compara el valor leído del sensor contra un valor umbral de 0.5 volts, por lo cual, si la lectura del sensor es inferior al umbral, quiere decir que existe algún objeto frente al sensor, y por el contrario, si la lectura del sensor es superior a 0.5 volts, quiere decir que no existe un objeto frente al sensor.

Se decidió utilizar un valor umbral de 0.5 volts porque experimentalmente se vio que al obstruir el sensor con un objeto opaco, la lectura del voltaje no superaba los 0.5 volts, mientras que, cuando no existía ningún objeto frente al sensor, su voltaje era superior a 1 volt.

La variable "Lecturas x led" determina el número de veces que se enciende y apaga cada led, de esta manera se varia fácilmente el número de veces que parpadea cada emisor, lo cual, en la etapa de pruebas fue de gran utilidad para determinar la cantidad de parpadeos mínima requerida para genera el mismo promedio en la lectura del sensor.

En este caso se hacen ráfagas de 5 parpadeos por led, aunque se probó con ráfagas de hasta 10 parpadeos, pero se concluyó que el promedio no variaba significativamente a partir de 5 lecturas, por lo que se decidió usar 5 parpadeos.

El promedio de voltaje leído en cada sensor es guardado en su correspondiente arreglo, dependiendo de la posición del mismo, ya que existen dos arreglos; uno para los sensores dispuestos de manera horizontal sobre la pantalla y otro para los dispuestos de manera vertical.

Al terminar de leer los 53 sensores, el programa busca en los arreglos "Horizontal" y "Vertical" valores menores a 0.5 volts para considerar que existe un objeto dentro del área de la pantalla, de esta manera se conocen sus coordenadas.

Constantes

Se crean dos constantes, "numero_leds" usada para decidir la cantidad de leds que se desean probar. Así como "Lecturas_x_led" utilizada para definir la cantidad de parpadeos que se desea que realice cada led; ya que en la etapa de pruebas, fue necesario probar con diferentes configuraciones, y la definición de estas constan-

tes facilitaba las pruebas.

Variables

Se utiliza un arreglo tipo entero de 35 elementos llamado “Horizontal” para almacenar el valor del voltaje promedio leído de los sensores dispuestos en forma horizontal sobre la pantalla, y un arreglo también de tipo entero pero de 19 elementos para almacenar el valor del voltaje promedio leído de los sensores dispuestos en forma vertical.

Se utiliza un arreglo tipo entero de 3 elementos llamada “Coordenadas” para almacenar las coordenadas del objeto detectado.

2.4.2 Contador 1

El código desarrollado para atender la interrupción del contador 1 realiza las siguientes tareas:

- Genera y envía la frecuencia de 10 KHz a la columna correspondiente de la matriz de leds; para encender y apagar el led en turno.
- Envía la señal a la fila correspondiente de la matriz de leds; para conectar la tierra al led en turno.
- Envía la señal al multiplexor para seleccionar el sensor que se debe leer.
- Envía la señal a la etapa de acondicionamiento de señal para seleccionar la ganancia correcta del amplificador operacional.
- Activa el contador 2.

La figura 2.16 muestra el diagrama de flujo de la rutina de atención a la interrupción del contador 1, cuyo funcionamiento es como sigue:

- Lo primero que realiza la rutina es una verificación de la variable “Flag” para saber si enciende o apaga el led en turno.

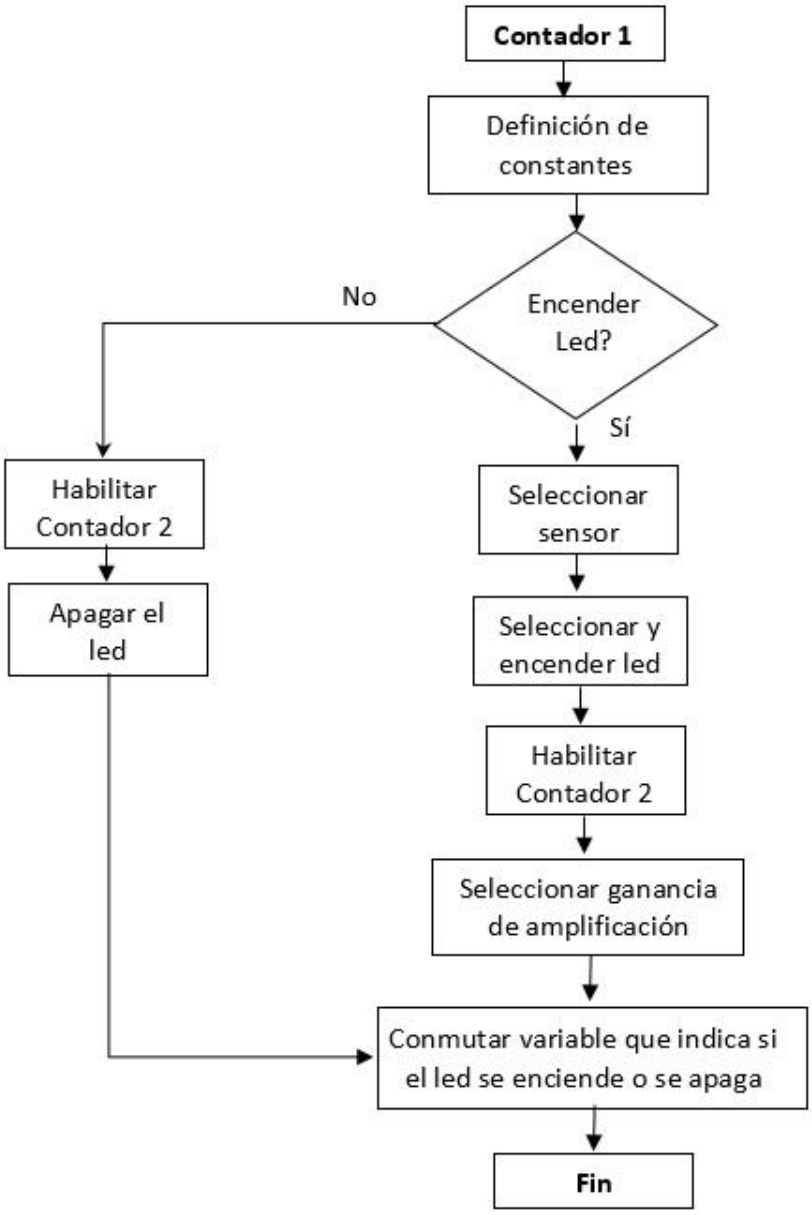


Figura 2.16: Diagrama de flujo de la rutina de interrupción del contador 1

- En caso de que se encienda; se utiliza la variable “Foto” para saber qué fila y qué columna de la matriz de leds debe encender y qué dirección debe enviar al multiplexor para conectar el correspondiente sensor a la etapa de

acondicionamiento, así como la ganancia que debe tener el amplificador. Esto lo decide dependiendo del led que deba encender; si corresponde a un led colocado en el lateral izquierdo de la pantalla, o en el lateral inferior de la pantalla. Una vez que se han tomado las decisiones mencionadas, se activa el contador 2.

- En caso de que se apague el led; se apaga la fuente de corriente correspondiente a dicho led, incrementa el valor de la variable "Cuenta" para indicar que ya se ha llevado a cabo un ciclo completo con el led actual, conmuta el valor de la variable "Flag" para que la siguiente vez que entre en esta interrupción encienda el led y finalmente inicia de nuevo el contador 2.

En la figura 2.17 se muestra un diagrama temporal de las señales involucradas en la lectura de los primeros 8 sensores.

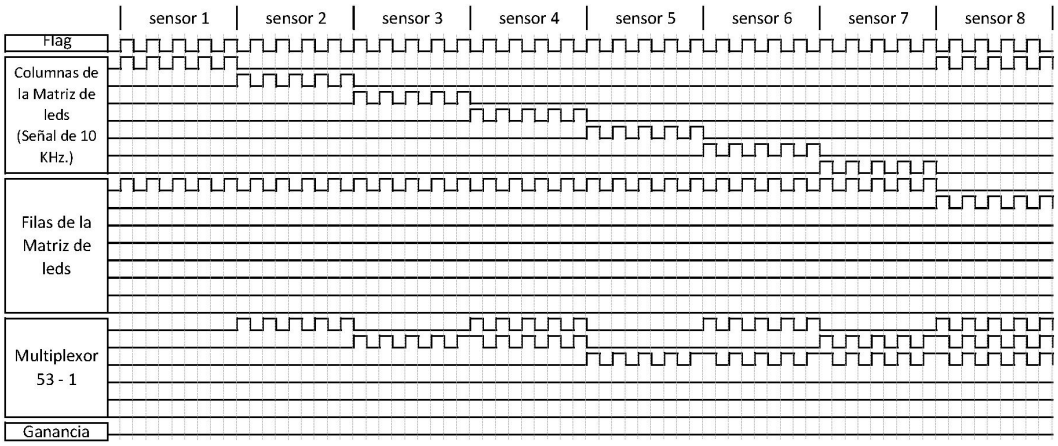


Figura 2.17: Diagrama temporal de la lectura de los primeros 8 sensores

2.4.3 Contador 2

Esta interrupción está programada para ejecutarse 15 μs después de habilitar el contador 2.

El código desarrollado para atender la interrupción del contador 2 realiza las siguientes tareas:

- Realiza las lecturas del sensor 15 μs después de encender o apagar el led en turno.

- Cuando el led está en su fase de encendido. Realiza la lectura del sensor por medio del convertidor analógico-digital y la guarda en una variable global. Cuando el led está en su fase de apagado. Realiza la lectura del sensor por medio del convertidor analógico-digital y obtiene el voltaje pico-pico de la señal restando el valor guardado del valor actual del sensor.

La razón de implementar las lecturas del sensor con un retraso de 15 μs a partir del encendido o apagado del led, es que, como se puede observar en la figura 3.3 (b) del capítulo 3, la respuesta del sensor tiene una forma senoidal, y las crestas y valles de la señal no coinciden con los flancos de subida y bajada de la señal que activa al led. En consecuencia, si leemos el sensor al mismo tiempo que activamos el led, nuestra lectura no sería de los valores máximo y mínimo de la señal.

La razón de hacer la lectura del voltaje pico-pico del sensor en lugar del voltaje pico, es debido a la forma de la respuesta del sensor, que como se mencionó en el párrafo anterior es senoidal.

En la figura 2.18 se muestra la imagen de dos señales de voltaje, denominadas como V_1 y V_2 respectivamente. La señal V_2 es igual a la señal V_1 mas un *offset* de 1.65 V.

Como se puede ver claramente, no es lo mismo leer el voltaje pico de la señal V_1 que el voltaje pico de la señal V_2, ya que, aunque es la misma señal el valor de esta medida es totalmente diferente.

A continuación, se obtiene el voltaje pico-pico de ambas señales.

$$V_{pp(V1)} = V_{max} - V_{min} = 0.89V - (-0.98V) = 1.78V \tag{2.35}$$
$$V_{pp(V2)} = V_{max} - V_{min} = 0.89V - (-0.98V) = 1.78V \tag{2.36}$$

De los datos obtenidos en (2.35) y (2.36) se puede observar que el voltaje pico-pico de una señal no varía al agregarle un offset. Por lo tanto, si en la lectura del

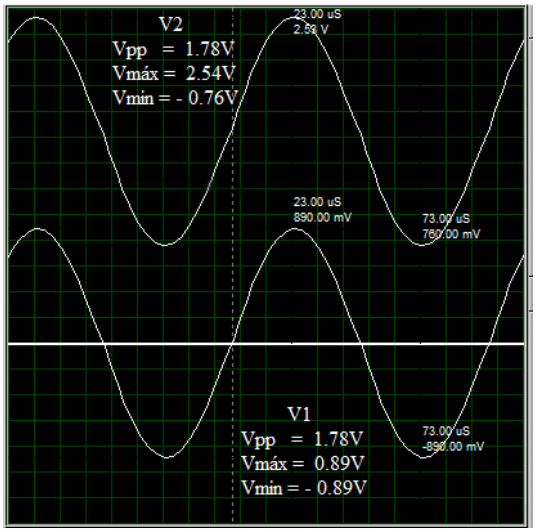


Figura 2.18: Señal de voltaje con offset

sensor existiese, este no interferirá con la lectura.

La figura 2.19 muestra el diagrama de flujo de la rutina de atención a la interrupción del contador 2, cuyo funcionamiento es como sigue:

- Lo primero que realiza la rutina es una verificación de la variable “Bandera” para saber si el led en turno está encendido o apagado.
- En caso de estar encendido, se lleva a cabo la lectura del sensor por medio del convertidor analógico-digital, dicho valor es convertido a voltaje y guardado en la variable global “valor1”, se debe utilizar una variable global para que se mantenga dicho valor cuando el flujo del programa salga de la rutina de interrupción, de lo contrario se perdería. Y finalmente, antes de que finalice la rutina de interrupción se deshabilita el contador 2.
- En caso de que el led esté apagado, se lleva a cabo la lectura del sensor por medio del convertidor analógico-digital, dicho valor es convertido a voltaje y guardado en la variable global “valor2”, a continuación, al valor1 se le resta valor2 para obtener el voltaje pico-pico, el cual es guardado en una de las posiciones de un arreglo global de 5 elementos llamado “Promedio”. Finalmente se deshabilita el contador 2 antes de salir de la rutina de atención

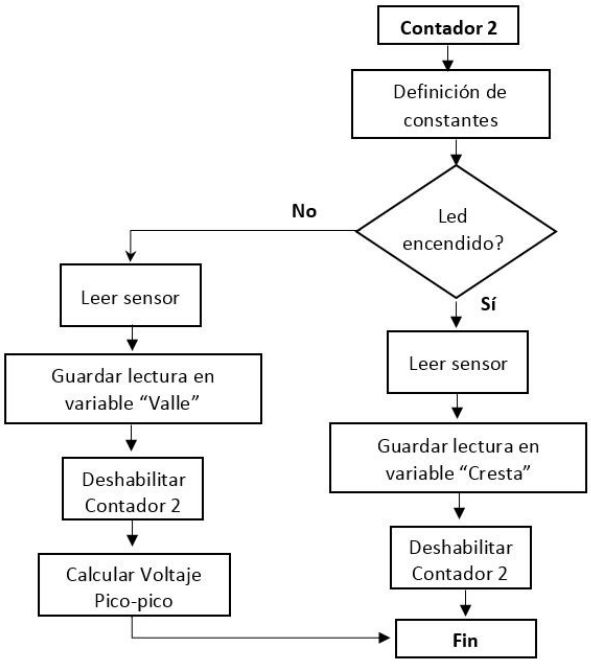


Figura 2.19: Diagrama de flujo de la rutina de interrupción del contador 2

a la interrupción del contador. Se debe de tomar en cuenta, es que se deben utilizar variables globales dentro de la rutina de interrupción, para que su valor no se pierda cuando el flujo del programa regrese a la rutina principal.

2.5 Etapa de interfaz gráfica

La Interfaz Gráfica de Usuario (*Graphic User Interface ó GUI* por sus siglas en ingles), tiene la función de proporcionar un entorno visual, sencillo de utilizar, para permitir una interacción fluida entre el usuario y el sistema, por medio de barras deslizables y botones.

La GUI está desarrollada en el software *Matlab*. Sirve para que el usuario le indique al sistema la velocidad o posición deseada del eje del motor. Así como para variar los parámetros del controlador Proporcional-Integral-Derivativo ó PID que gobierna al motor.

El circuito electrónico de la pantalla táctil envía, por medio del puerto serial, las coordenadas de las pulsaciones realizadas sobre la pantalla y la interfaz gráfica, las compara con las coordenadas de las barras y botones de la GUI, si coincide con algún control, realiza la acción correspondiente.

La Figura 2.20 muestra el aspecto que tiene la interface gráfica de usuario, que se diseñó para controlar el sistema. En la parte superior consta de tres graficadores que muestran de izquierda a derecha las siguientes señales en tiempo real:

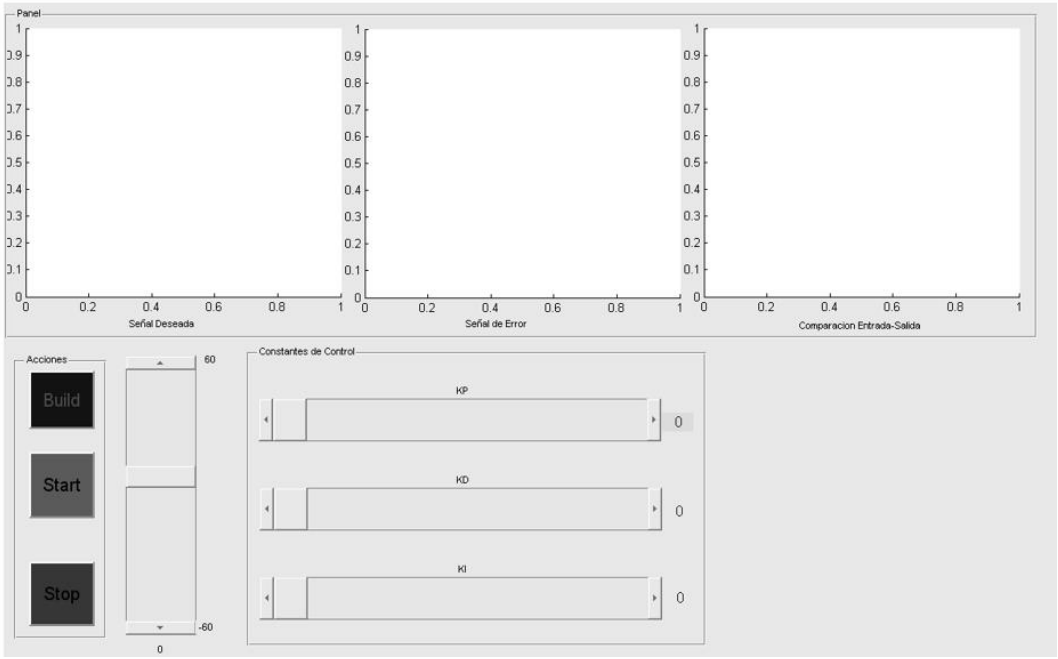


Figura 2.20: Interfaz gráfica de usuario

- La primera gráfica es la señal deseada, ya sea de posición o de velocidad, dependiendo del motor que se desea controlar.

- La segunda gráfica muestra la señal de error, la cual es la diferencia que existe entre la referencia de posición o velocidad deseada y la señal real del sensor.

- La tercera gráfica muestra la señal de posición o velocidad deseada y al mismo tiempo la señal real, esto con propósitos informativos para ver el desempeño del controlador.

Del lado izquierdo de la *GUI* se encuentra una barra vertical deslizable que sirve para asignar la posición o velocidad deseada del eje del motor. Cuando se trata del control de posición, ésta barra tiene un rango de desplazamiento de $-360°$ *a* $360°$, y cuando se trata del control de velocidad, un rango de -60 *RPM* a 60 *RPM*.

Al centro de la GUI se encuentran tres barras; las cuales sirven para modificar las constantes de un controlador proporcional-integral-derivativo PID, su esquema se puede ver en la Figura 2.21, las barras descritas de abajo hacia arriba, son las siguientes:

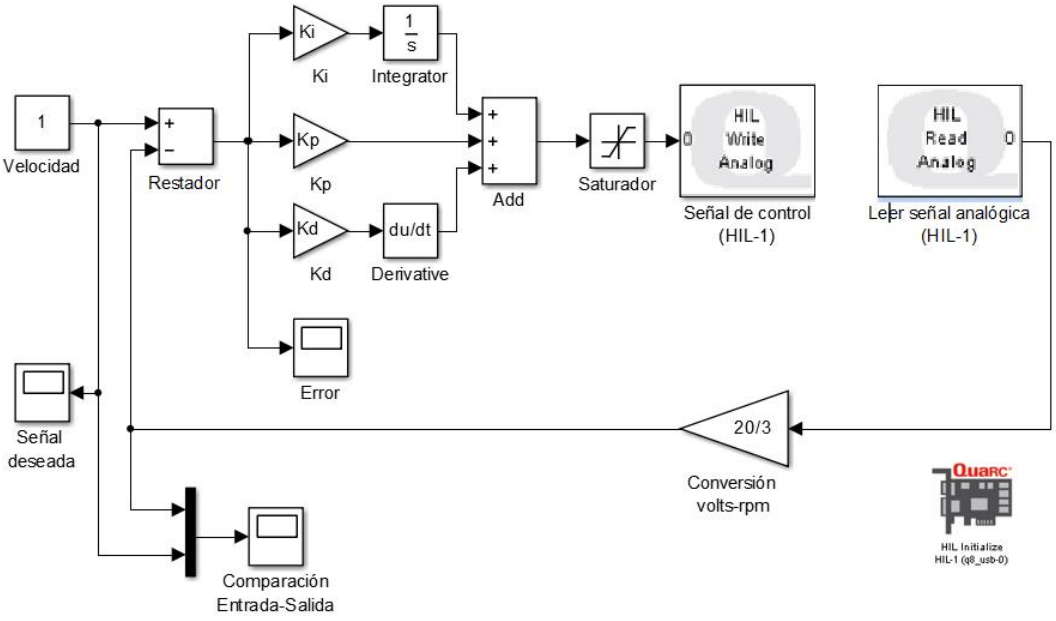


Figura 2.21: Controlador PID asociado a la GUI

- Barra KP, modifica el valor de la constante proporcional del controlador PID. Su rango va de 0 a 100, siendo cero su extremo izquierdo y 100 su extremo derecho.

- Barra KD, modifica el valor de la constante derivativa del controlador PID. Su rango va de 0 a 1, siendo cero su extremo izquierdo y 1 su extremo derecho.
- Barra KI, modifica el valor de la constante integral del controlador PID. Su rango va de 0 a 5, siendo cero su extremo izquierdo y 5 su extremo derecho.

Finalmente, la *GUI* cuenta con tres botones ubicados del lado izquierdo, descritos de abajo hacia arriba son:

- Botón *Stop*, su función es detener la operación de la *GUI*, por lo tanto, al presionarlo la *GUI* deja de recibir los datos de la pantalla táctil, y deja de enviar las señales de control a los motores.
- Botón *Start*, inicia las operaciones del *GUI*, inicia la comunicación con la pantalla táctil, y envía las señales de control a los motores.
- Botón *Built*, su función es volver a compilar el modelo cuando se han realizado cambios en el esquemático de *SIMULINK*.

La *GUI* esta enlazada a un modelo realizado en el software *SIMULINK* que se muestra en la figura 2.21, el cual, realiza el control dependiendo de los parámetros introducidos por el usuario a través de las barras deslizables.

Adicionalmente se necesita un código fuente en el cual se definen las tareas realizadas por la *GUI*, este código se muestra en el Anexo A.6.

La interfaz gráfica de usuario, además de recibir y procesar los datos provenientes de la pantalla táctil, se encarga de enviar la señal de control hacia los motores que se están controlando.

Entre los motores y la GUI se encuentran dos etapas que permiten dicha comunicación, una es una tarjeta de adquisición de datos que comunican la *GUI* con el mundo exterior, y la otra es un servo-amplificador que más adelante se describirá.

La tarjeta de adquisición de datos es de la marca *Quanser*, modelo $Q8-USB$, la Figura 2.22 muestra la tarjeta con sus cables de conexión y fuente de alimentación.

Sus características principales son:

Figura 2.22: Tarjeta de adquisición de datos *Quanser Q8-USB*

- Dimensiones: 812 x 139 mm.
- 8 entradas analógicas.
- 8 salidas analógicas.
- 8 lectores de encoder.

La señal de control proveniente de la *GUI*, se envía por medio de una de las salidas analógicas, debido a que sólo entrega una señal de voltaje en un rango de $\pm 10\ V$, es necesaria una etapa de amplificación para poder controlar los motores.

La lectura del sensor de velocidad se hace a través de una entrada analógica de la tarjeta, por lo que el rango de la señal no puede exceder los $\pm 10\ V$, considerando que el tacómetro del motor genera un voltaje máximo de $9\ V$, no es necesario acondicionar su señal. El sensado de corriente se realiza por medio de otra entrada analógica, debido a que la caída de voltaje es muy baja en el resistor de sensado no hace falta delimitar el voltaje.

2.6 Etapa de Servo-amplificador

La pantalla táctil se utiliza como una interfaz gráfica para enviar órdenes de posición o velocidad a dos motores de la marca *harmonic drive*, modelo $RH-8D6006E100$. Por lo que se requiere construir un servo-amplificador para alimentarlos. Los motores tienen las siguientes características:

- Voltaje nominal: 24 Vcd.
- Velocidad máxima: 60 rpm.
- Caja reductora con relación de engranes 50:1.
- Uno de ellos, tiene acoplado a su flecha un sensor de posición (encoder).
- El otro, un sensor de velocidad (tacómetro).

El tacómetro es un sensor de velocidad que entrega un voltaje proporcional a la velocidad de giro del eje del motor. El tacómetro utilizado tiene una resolución de $3\,V/1000\,rpm$, esto traducido a la flecha de salida de la caja reductora equivale a $3\,V/20\,rpm$.

El encoder absoluto es un sensor de posición que entrega una serie de pulsos proporcional a la posición. El encoder utilizado tiene una resolución de 1000 pulsos/revolución, esto traducido a la flecha de salida de la caja reductora equivale a $50000 pulsos/revolucion$.

El servo-amplificador acopla la interfaz gráfica de usuario con el actuador por medio de una tarjeta de adquisición de datos. Es utilizado para convertir la señal de voltaje proveniente de la tarjeta de adquisición de datos *(Q8-USB)* en una señal de voltaje que proporciona la corriente necesaria para obtener la respuesta que se requiere del motor. En la Figura 2.23 se muestra el diagrama esquemático de la etapa amplificadora.

El servo-amplificador está compuesta por las siguientes partes:

- *Buffer* de entrada.
- Amplificador de voltaje.

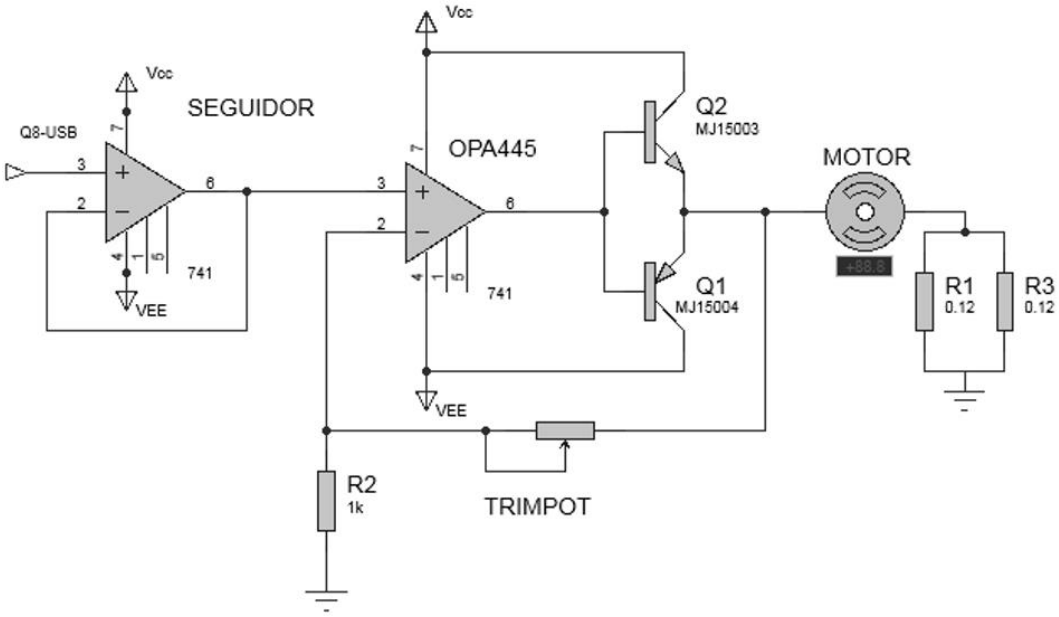


Figura 2.23: Amplificador de potencia

- Sensado de corriente.

Para el diseño se eligieron amplificadores operacionales de alto voltaje modelo *OPA*445, por su alta impedancia de entrada y su voltaje de alimentación de $45Vdc$.

2.6.1 Búfer de entrada

Un amplificador operacional en configuración seguidor de voltaje, tiene una impedancia de entrada Z_{in} muy elevada, y una impedancia de salida Z_{out} muy pequeña. Por este motivo se utiliza principalmente para aislar dos circuitos, de manera que el segundo no resulte una carga para el primero, pues la impedancia vista será la altísima Z_{in} del operacional. En este caso se dice que esta etapa sirve para adaptar impedancias, además de proteger la salida de la tarjeta de adquisición de datos *Q8-USB*.

2.6.2 Amplificador push-pull

Para amplificar el voltaje se utiliza un amplificador operacional en configuración amplificador no inversor, con ésta configuración no se desfasa la señal la ganancia de voltaje es ajustable debido a que la realimentación se hace a través de un potenciómetro $(10K\,\Omega)$, con un rango de ganancia de 1 a 10. La salida del amplificador operacional se conecta a la base de los transistores de potencia. El esquema electrónico se muestra en la Figura 2.24

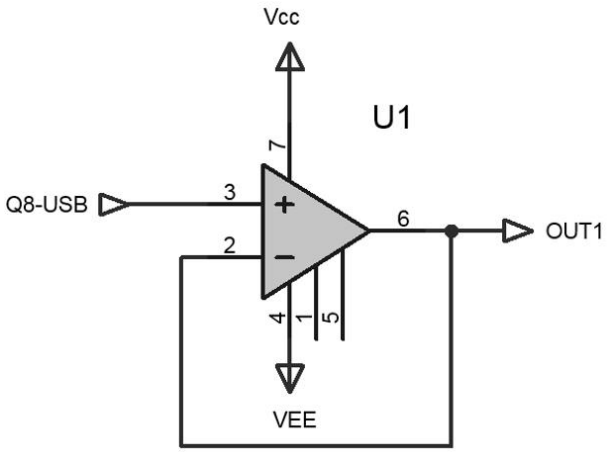


Figura 2.24: Seguidor de voltaje

Para esta etapa se usa un par de transistores un *PNP* y *NPN*, los transistores elegidos son los modelos *MJ*15003 y *MJ*15004, ambos transistores de potencia; entregan hasta 20 *A*. Fueron montados con disipador de calor, el potenciómetro de la etapa anterior se conecta a los emisores cerrando el lazo de retroalimentación, la salida *OUT2* es la salida del amplificador operacional. El esquema electrónico se muestra en la figura 2.25.

2.6.3 Sensado de corriente

Para sensar la corriente se utiliza un resistor de valor muy bajo conectado en el cable de retorno a tierra del motor, la caída de voltaje en dicho resistor es proporcional al voltaje, en este caso se utiliza un par de resistores con una resistencia de

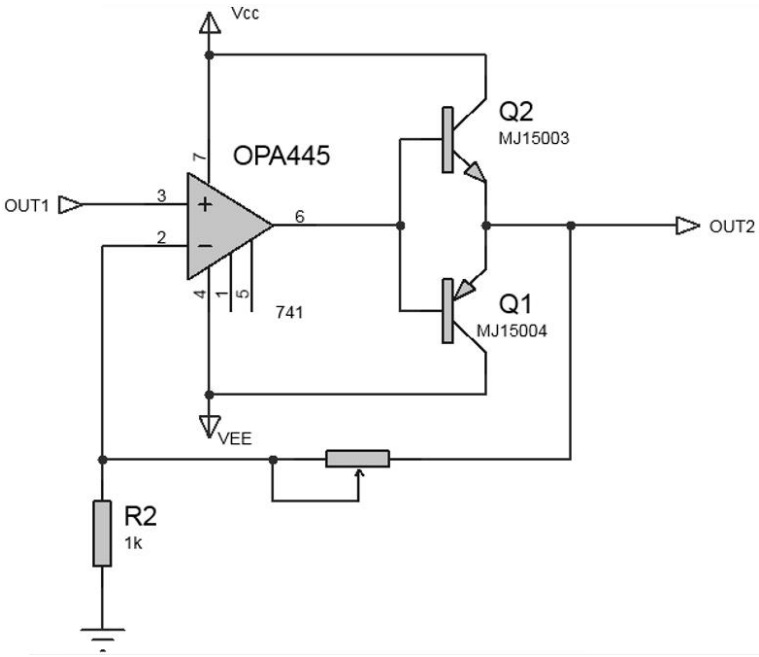


Figura 2.25: Amplificador *push-pull*

0.12 cada uno, conectados en paralelo para bajar aun más la resistencia equivalente. El resultado es un resistor de 0.06 de resistencia y dado que cada uno es de $2W$ de potencia, se alcanza una disipación de $4W$. Sabiendo esto se puede calcular la corriente máxima que se puede sensar sin dañar las resistencias.

Se sabe que la potencia eléctrica es el producto del voltaje por la corriente, por lo que la potencia en el resistor que se utiliza en este proceso es:

$$P_R = V_R I_R \tag{2.37}$$

Asimismo, se sabe por la ley de Ohm que el voltaje es directamente proporcional a la corriente de la siguiente forma:

$$V_R = R I_R \tag{2.38}$$

Sustituyendo (2.38) en (2.37) tenemos que la potencia en el resistor es:

$$P_R = RI_R^2 = 4\ W \tag{2.39}$$

Considerando que la potencia en el resistor es de 4 W como la máxima para disipar; se despeja IR de (2.39) y se sustituyen valores, con lo que se obtiene una corriente máxima que puede circular por el resistor de:

$$I_R = \sqrt{\frac{4\ W}{0.06\ \Omega}} \cong 8.16\ A \tag{2.40}$$

Como se muestra en la ecuación (2.40), la corriente máxima que se puede sensar de forma segura con el circuito, es de 8.16 A. Tomando en cuenta que el motor a alimentar consume una corriente nominal de 1 A, se considera que los valores de los componentes son suficientes para el sensado de su corriente. En el caso de que se tenga una corriente máxima de 8.16 amperes circulando por la resistencia de sensado, esto generaría una caída de voltaje en el resistor aproximada de 0.49 V. Valor que es admisible para la tarjeta de adquisición de datos, quien sería la encargada de leer este valor.

Capítulo 3

Pruebas

3.1 Detección de objetos dentro de la pantalla

3.1.1 Circuito básico de pruebas

Las primeras pruebas de detección fueron realizadas con un circuito montado sobre una tarjeta protoboard, el cual se muestra en la figura 3.1 (a).

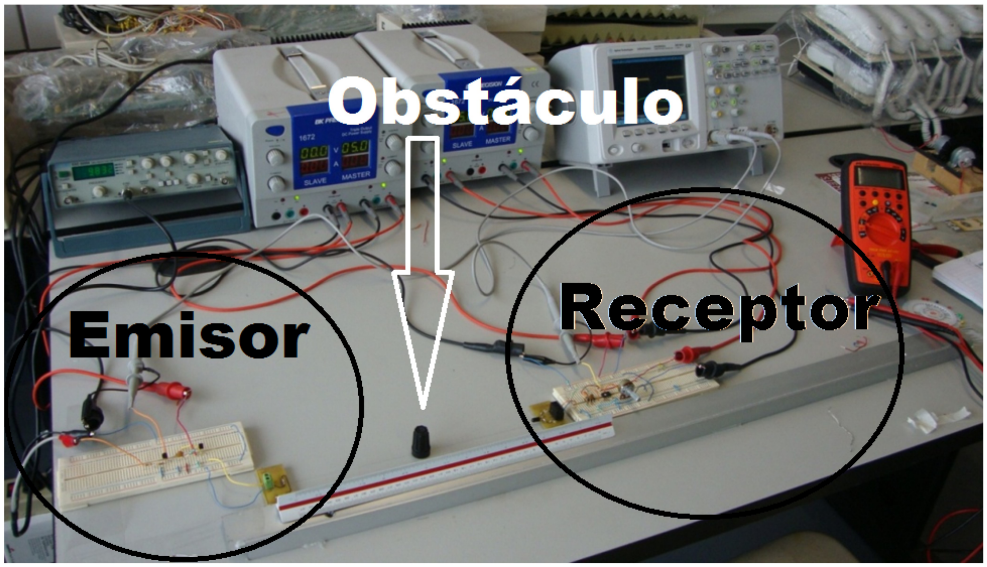
Del lado izquierdo de la imagen, se puede ver el led emisor; el cual se encuentra montado en una pequeña tarjeta de circuito impreso, que está conectada a su vez a una fuente de corriente armada sobre un protoboard.

Del lado derecho de la imagen se muestra el receptor; formado por un fototransistor montado en una pequeña placa de circuito impreso y conectado a su vez a un filtro y a un amplificador, ambos armados sobre otro protoboard, del cual se toma la salida para observarla en el osciloscopio.

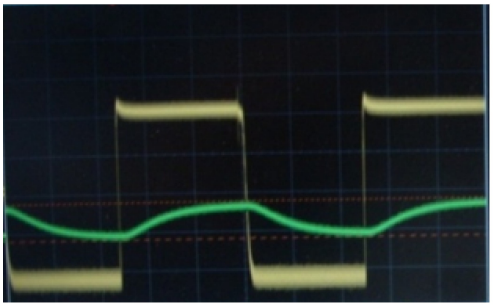
Para realizar las pruebas de detección, se le hizo pasar al led una señal de onda cuadrada con amplitud de 3 volts y frecuencia de 10 KHz proveniente de un generador de funciones.

El sensor fue colocado a una distancia de separación de 25 cm, respecto del emisor. Posteriormente se colocó un objeto entre emisor y receptor para simular la presencia de un dedo sobre la pantalla. La figura 3.1 (b), muestra la señal proveniente del generador de funciones (onda cuadrada) y la señal proveniente del sensor. Como se puede apreciar en la gráfica, la señal del sensor en estas condi-

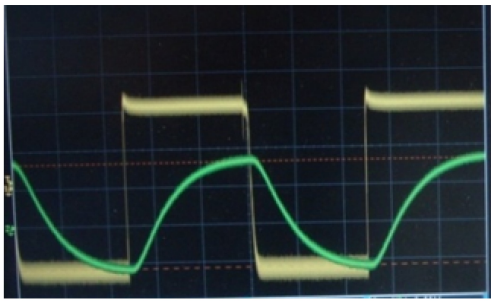
ciones está atenuada, ya que la amplitud de voltaje pico-pico es de 480 mV.



(a) *Primer prueba de detección*



(b) *Señal con obstáculo*



(c) *Señal sin obstáculo*

Figura 3.1: Prueba del sensor - Circuito de prueba.

Posteriormente, se retiró el obstáculo entre sensor y emisor y se realizó la misma medición. La figura 3.1 (c), muestra la respuesta del sensor en estas condiciones; su amplitud de voltaje es de 1.41 V, y su frecuencia es de 10 KHZ.

3.1.2 Pruebas de detección con el sistema terminado

Cuando finalmente se tuvieron terminados todos los subsistemas que componen al prototipo tema de esta tesis, se procedió a interconectarlos, como se muestra en la figura 3.2.

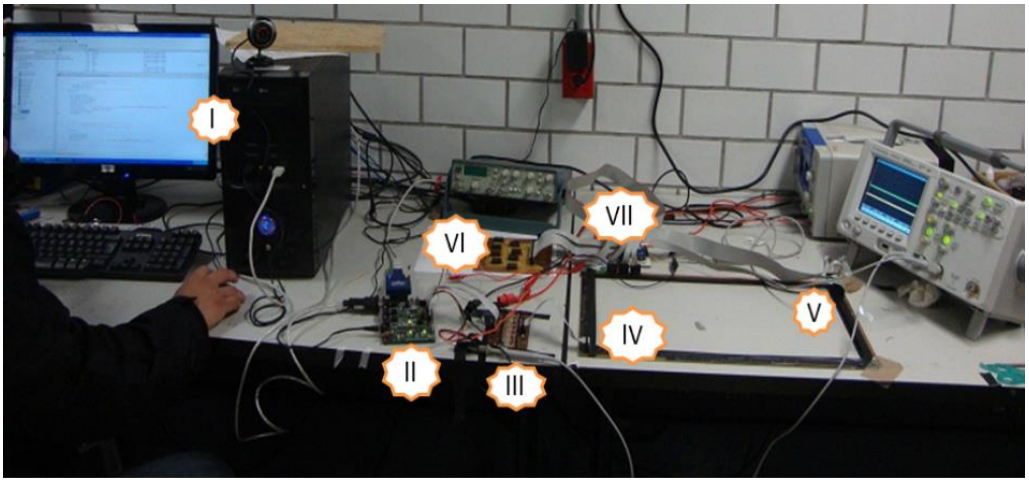


Figura 3.2: Sistema completo

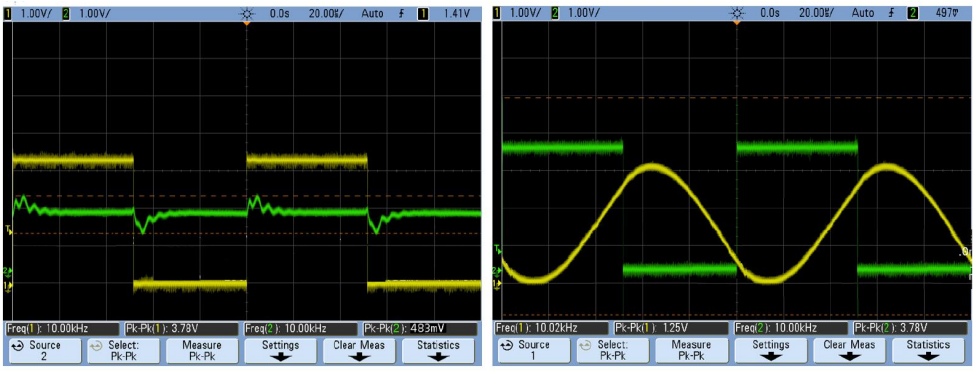
En el número I se muestra la computadora ejecutando la GUI, en el número II la tarjeta de desarrollo, en el número III las fuentes de corriente, en el número IV la matriz de led (etapa emisora), en el número V el arreglo de sensores (etapa receptora), en el número VI el multiplexor, y en el número VII el filtro y amplificadores (etapa de acondicionamiento).

En este caso se puede ver el marco formado por las etapas emisora y receptora marcadas con los números IV y V respectivamente, colocadas sobre el escritorio, ya que aún no se montaba en el monitor. De esta forma se realizó la prueba y cuando se colocaba un objeto dentro de dicha área este era detectado por los sensores y su ubicación enviada a la *GUI*.

Con la finalidad de comprobar que la señal producida por el sensor correspondía con las señales obtenidas en las pruebas del circuito básico mencionadas en la sección anterior de este mismo capítulo, se procedió a realizar la misma prueba de detección.

Se colocó un objeto opaco dentro del área de sensado, y se realizó la lectura del sensor correspondiente por medio del osciloscopio.

En la figura 3.3 (a), se muestra la señal producida por el sensor que se está probando, y como se puede apreciar, su voltaje es de 483 mV y su frecuencia de 10 KHZ. Asimismo, se puede ver la señal generada por el microcontrolador que excita al emisor correspondiente.



(a) *Señal con obstáculo* (b) *Señal sin obstáculo*

Figura 3.3: Prueba del sensor - Circuito final

Posteriormente, se retiró el objeto y se realizó la lectura del mismo sensor. En la figura 3.3 (b), se muestra la señal producida por el microcontrolador para excitar al emisor, y la señal producida por el sensor, la cual, como se puede observar tiene una amplitud de voltaje de 1.25 V, y una frecuencia de 10 KHZ.

La misma prueba se realizó en cada uno de los sensores para asegurar su correcto funcionamiento. Como se puede observar, las señales de los sensores montados en el prototipo tienen las mismas características de las señales obtenidas en el circuito básico.

3.2 Pruebas con el controlador

En esta sección, se obtiene el modelo matemático lineal de un motor de corriente directa, basándose en la bibliografía (Golnaraghi, 2010).

El modelo electromecánico del motor considera el voltaje de armadura como variable de entrada y a la posición como variable de salida. Los componentes principales de los motores de CD se muestran en la figura 3.4.

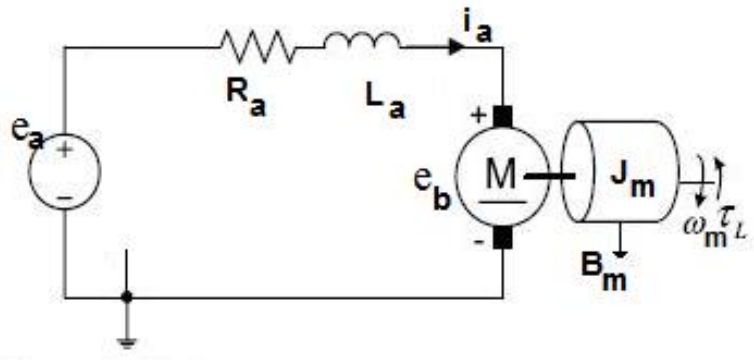


Figura 3.4: Modelo de un motor de corriente directa.

Para calcular el modelo matemático del motor de corriente continua, planteamos las ecuaciones físicas del sistema. Para ello recurrimos al diagrama del circuito eléctrico de la armadura, y al diagrama de cuerpo libre del rotor.

Las variables y parámetros del motor se definen como sigue:

- i_a=Corriente de armadura.
- R_a=Resistencia de armadura.
- $e_b(t)$=Voltaje contraelectromotriz.
- $T_L(t)$=Torque de la carga.
- $T_m(t)$=Torque del motor.
- $\Theta_m(t)$Desplazamiento del rotor.
- K_i=Constante del torque.
- L_a=Inductancia de armadura.
- $e_a(t)$=Voltaje aplicado.

- K_b=Constante de fuerza contraelectromotriz.
- ϕ=Flujo magnético en el entre hierro.
- $\omega_m(t)$=Velocidad angular del rotor.
- J_m=Inercia del motor.
- B_m=Coeficiente de fricción viscosa.

Las ecuaciones eléctricas de los componentes son:

$$V_{Ra} = Ri_a(t) \tag{3.1}$$
$$V_{La} = L\frac{d}{dt}i_a(t) \tag{3.2}$$

La ecuación mecánica utilizada es:

$$T_m(t) = J_m\frac{d^2}{dt^2}\Theta_m(t) + B_m\frac{d}{dt}\Theta_m(t) \tag{3.3}$$

Las ecuaciones que acoplan la parte eléctrica del motor con la parte mecánica, son:

$$e_b(t) = K_b\frac{d}{dt}\Theta_m(t) \tag{3.4}$$
$$T_m(t) = K_i i_a(t) \tag{3.5}$$

A continuación se calcula la función de transferencia $\frac{\Theta_m(s)}{E_a(s)}$.

Se aplica la ley de voltajes de Kirchhoff al circuito de la figura 3.4 para calcular $e_b(t)$:

$$e_a(t) = V_{La} + V_{Ra} + e_b(t) \tag{3.6}$$

Se sustituye (3.1), (3.2) y (3.4) en (3.6):

$$e_a(t) = Ri_a(t) + L\frac{d}{dt}i_a(t) + K_b\frac{d}{dt}\Theta_m(t) \tag{3.7}$$

De la parte mecánica se sustituye (3.5) en (3.4):

$$K_i i_a(t) = J_m\frac{d^2}{dt^2}\Theta_m(t) + B_m\frac{d}{dt}\Theta_m(t) \tag{3.8}$$

Se aplica la transformada de Laplace a (3.7) y a (3.8) para pasarlos al dominio de la frecuencia:

$$eE_a(s) = RI_a(s) + LI_a(s)s + K_b\Theta_m(s)s \tag{3.9}$$
$$K_i i_a(s) = J_m\Theta_m(s)s^2 + B_m\Theta_m(s)s \tag{3.10}$$

Se despeja la (s) en (3.9) y se sustituye en (3.10):

$$K_i\frac{E_a(s) - K_b\Theta_m(s)s}{R + L_s} = J_m\Theta_m(s)s^2 + B_m\Theta_m(s)s \tag{3.11}$$
$$K_iE_a(s) - K_iK_b\Theta_m(s)s = (R + L_s)(J_m\Theta_m(s)s^2 + B_m\Theta_m(s)s) \tag{3.12}$$

Realizando las simplificaciones algebráicas correspondientes a (3.12), se obtiene la función de transferencia del motor mostrada en (3.13):

$$F.T. = \frac{\Theta_m(s)}{E_a(s)} = \frac{K_i}{L_aJ_ms^3 + (R_aJ_m + B_mL_a)s^2 + (K_bK_i + R_aB_m)} \tag{3.13}$$

Parámetro	Valor	Unidades
L_a	$2.2x10^-3$	mH
R_a	$10x10^-3$	Ω
J_m	$45.37x10^-6$	Kgm^2
B_m	$66x10^-3$	Nms/rad
K_b	$22x10^-3$	Vs/rad
K_i	$2.72x10^-6$	Nm/A

Tabla 3.1: Parámetros del motor de cd

El motor utilizado tiene los siguientes parámetros:
Sustituyendo los valores del motor en (3.13), se tiene que el modelo del motor utilizado es:

$$G_p(s) = \frac{2.72x10^{-6}}{s^3 + 6s^2 + 5s} \tag{3.14}$$

A continuación, se aplica el criterio de estabilidad de Routh-Hurwitz para saber si la planta es estable.

La ecuación característica de la función de transferencia del motor, es:

$$s^3 + 6s^2 + 5s^1 = 0 \tag{3.15}$$

s^3	1	5
s^2	6	1
s^1	29/6	0
s^0	1	0

Tabla 3.2: Prueba de estabilidad de Routh-Hurwitz

Como se puede apreciar en la tabla 3.2, no hay ningún cambio de signo en la primera columna, por lo que podemos decir, que el sistema es estable.

A continuación, factorizamos el denominador de (3.14), lo que nos permite observar, que el sistema es tipo 1; ya que cuenta con un polo en el origen.

$$G_p(s) = \frac{2.7210x10^{-6}}{(s)(s+1)(s+5)} \tag{3.16}$$

Un sistema tipo 1, tiene la característica que cuando es excitado por una señal escalón, su error en estado estacionario es cero.

Se sabe que el error de posición está dado por:

$$K_p = \lim_{x\to 0} G_p(s) \tag{3.17}$$

Calculamos el error de posición para una entrada tipo escalón.

$$K_p = \lim_{x\to 0} \frac{2.7210x10^{-6}}{(s)(s+1)(s+5)} = \frac{2.7210x10^{-6}}{0} = \infty \tag{3.18}$$

Se sabe que el error en estado estacionario para una entrada tipo escalón es:

$$E_s s = \frac{1}{1+K_p} = \frac{1}{1+\infty} = 0 \tag{3.19}$$

Como se muestra en (3.19) el error en estado estacionario es cero.

3.2.1 Controlador proporcional integral derivativo

La señal de salida de un controlador PID es la suma de los efectos de los controladores por separado, es decir:

$$u(t) = K_p e(t) + k_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{3.20}$$

La función de transferencia del controlador PID se muestra a continuación:

$$G_c(s) = K_p + \frac{K_i}{s} + K_d s = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{3.21}$$

Un controlador PID incrementa el número de ceros en dos y el de polos en uno en la trayectoria directa, en consecuencia, se incrementa el tipo de sistema en 1 mejorando la estabilidad del sistema.

Para realizar la sintonización del controlador PID, primero debemos hacer que $T_i = \infty$ y $T_d = 0$ y usando solamente la acción de control proporcional se incrementa K_p desde cero hasta un valor crítico K_{cr} en donde la salida muestre oscilaciones sostenidas. Esto lo podemos hacer utilizando un controlador tipo proporcional en la trayectoria directa de la planta y obteniendo la función de transferencia en lazo cerrado.

$$G_c(s) = \frac{K_p}{s^3 + 6s^2 + 5s + K_p} \tag{3.22}$$

De la ecuación característica de la función de transferencia en lazo cerrado, sustituimos la variable compleja s por $j\omega_{cr}$ y K, por K_{cr} como sigue:

$$(j\omega_{cr})^3 + 6(j\omega_{cr})^2 + 5(j\omega_{cr}) + K_{cr} = 0 \tag{3.23}$$

$$-j\omega_{cr}^3 - 6j\omega_{cr}^2 + 5j\omega_{cr} + K_{cr} = 0 \tag{3.24}$$

Agrupamos parte real con parte real, y parte imaginaria con parte imaginaria. Al despejar ω_{cr} obtenemos la frecuencia angular crítica.

$$-\omega_{cr}^3 + 5\omega_{cr} = 0 \tag{3.25}$$

Al resolver para ω_{cr}, encontramos que:

$$\omega_{cr1} = 0 \tag{3.26}$$
$$\omega_{cr2} = +\sqrt{5} \tag{3.27}$$
$$\omega_{cr3} = -\sqrt{5} \tag{3.28}$$

Por otro lado, al tomar la parte real, se tiene que la ganancia proporcional crítica es:

$$K_{cr} = 6\omega_{cr}^2 \tag{3.29}$$

Al sustituir los valores de la frecuencia angular crítica en (3.29), obtenemos los valores de la ganancia proporcional critica.

$$K_{cr1} = 0$$
$$K_{cr2} = 30$$
$$K_{cr3} = 30$$

Tipo de controlador	K_p	T_i	T_d
P	$0.5\ K_{cr}$	∞	0
PI	$0.45\ K_{cr}$	$\frac{5}{6}\ P_{cr}$	0
PID	$0.6\ K_{cr}$	$0.5\ P_{cr}$	$0.125\ P_{cr}$

Tabla 3.3: Relación para asignación de ganancias

Usando los valores de la tabla 3.3, así como los valores de K_{cr} y Ω_{cr}, se calculan los valores del controlador:

$$K_p = 0.6K_{cr} = 18 \tag{3.30}$$
$$T_i = 0.5P_{cr} = 1.4 \tag{3.31}$$
$$T_d = 0.125P_{cr} = 0.35 \tag{3.32}$$

Para obtener la función de transferencia del controlador, se sustituye (3.30), (3.31) y (3.32), en (3.21).

$$G_c(s) = \frac{6.32s^2 + 18s + 12.81}{s} \tag{3.33}$$

La función de transferencia en lazo cerrado es:

$$G_p(s) = \frac{6.32s^2 + 18s + 12.81}{s^4 + 6s^3 + 11.32s^2 + 18s + 12.81} \tag{3.34}$$

A continuación, se muestra la gráfica de la respuesta a una entrada tipo escalón.

Como se puede apreciar en la figura 3.5 el sistema tiene un sobrepaso del 60 por ciento y un tiempo de asentamiento de 10 segundos.

3.2.2 Controlador proporcional

El primer controlador utilizado en la GUI fue un controlador proporcional, su implementación en SIMULINK se puede ver en la figura 3.6.

El valor numérico del bloque llamado “referencia” de la figura 3.6 está controlado por la barra vertical ubicada en la parte izquierda de la GUI, por medio de la cual, el usuario puede cambiar la posición del eje del motor.

En el bloque llamado “restador” se realiza la diferencia entre la señal de referencia y la señal de velocidad proveniente del tacómetro.

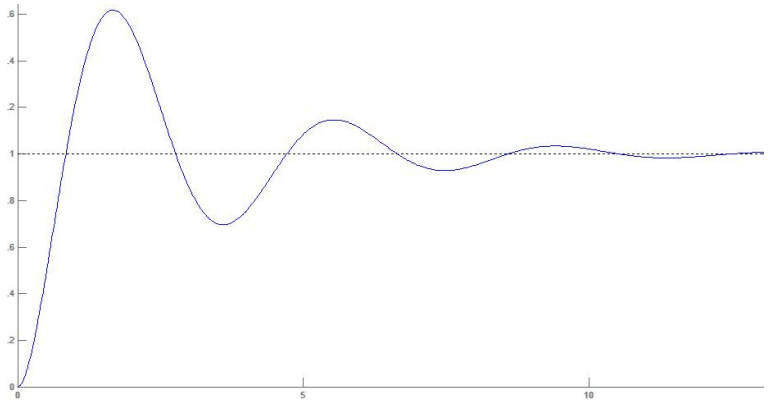


Figura 3.5: Respuesta del motor a una señal escalón unitario

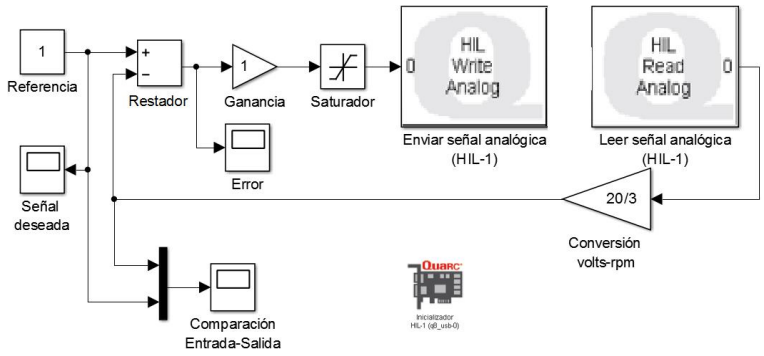


Figura 3.6: Controlador Proporcional en SIMULINK

La lectura del sensor se realiza por medio de una de las entradas analógicas de la tarjeta de adquisición de datos Q8, y que en el diagrama se representa con el bloque llamado “Leer señal analógica”. La lectura del sensor es entregada en forma de voltaje, por lo cual, para hacer la conversión a rpm (revoluciones por minuto) se debe multiplicar por la ganancia llamada “Conversión de volts a rpm” que tiene un valor de $20/3$, ya que el sensor genera 3 volts cuando gira a una velocidad de 60 rpm.

La salida del restador es multiplicada por el bloque llamado “Ganancia”, el cual está controlado por la barra “K_p” de la GUI, lo que permite variar su valor en tiempo real.

La señal de control es limitada a un valor entre 10 y -10, por medio de un bloque saturador para finalmente ser enviada a la etapa de potencia por medio de una de las salidas analógicas de la tarjeta *Q*8, representada en el diagrama por medio del bloque llamado “Enviar señal analógica”.

Los bloques llamados “Señal deseada”, “Error”, y “Comparación entrada-salida” se encargan de enviar los datos para que sean graficados en la correspondiente ventana de la GUI, las cuales fueron comentadas en el capítulo anterior.

Finalmente, al tener funcionando el sistema completo, se realizaron pruebas con la GUI. En la figura 3.7 se muestra al usuario modificando las ganancias del controlador PID a través de la interfaz gráfica.

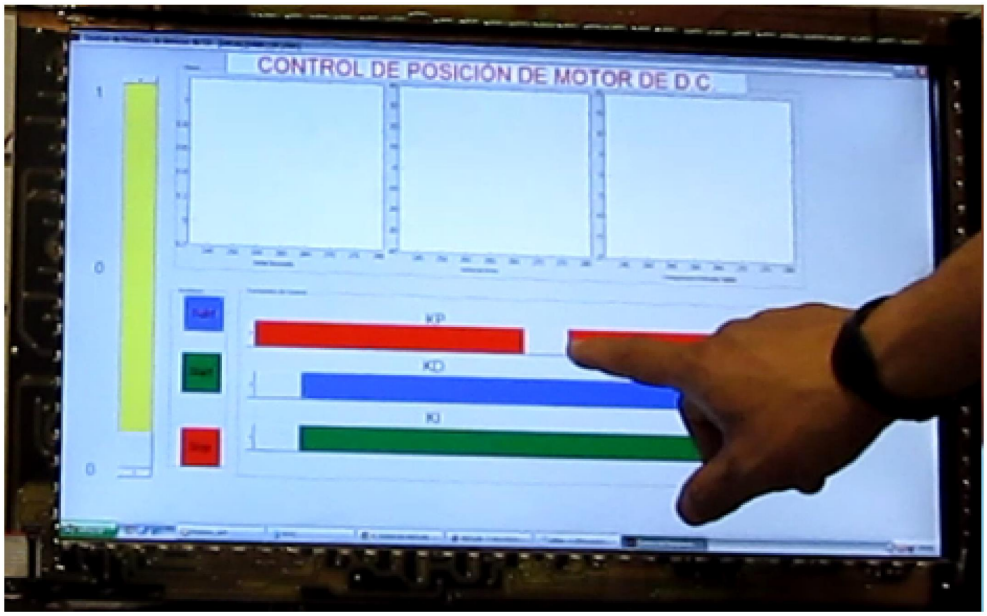


Figura 3.7: Usuario modificando valores del controlador

Capítulo 4

Conclusiones

Se construyó un prototipo funcional de una pantalla o superficie táctil. Es un dispositivo capaz de detectar objetos dentro de su área de sensado, la cual, es capaz de cubrir un monitor de 20 pulgadas. En su diseño y construcción, la única etapa que se compró, fue la tarjeta de desarrollo, las demás debieron ser diseñadas e implementadas.

Una vez que se tuvo el prototipo funcional, se integró a una interfaz gráfica de usuario creada específicamente para este proyecto, para lo cual se creó un programa en MATLAB ® que genera la interfaz gráfica y al mismo tiempo se comunica con el dispositivo táctil por medio del puerto serial.

La interfaz gráfica a su vez, se conectó a una etapa de potencia por medio de una tarjeta de adquisición de datos para controlar dos motores.

Finalmente, lo que se obtuvo fue un sistema totalmente integrado, con el cual se puede controlar dos motores de CD, por medio de una interfaz táctil, así como cambiar los parámetros de su controlador PID en tiempo real, lo que nos permite ver tanto en gráficas como en el comportamiento del motor el efecto que tiene cada constante en el desempeño del controlador.

Durante la carrera adquirí conocimientos sobre el diseño e implementación de circuitos electrónicos que me permitieron, desarrollar el prototipo expuesto en esta tesis. Gracias a las enseñanzas de todos mis profesores, actualmente cuento con los conocimientos suficientes para encarar el desarrollo de proyectos que involucran la integración de sistemas de instrumentación electrónica, adquisición

de datos y desarrollo de software, tal y como se pudo demostrar con el diseño y construcción del prototipo tema de la presente tesis.

Desde mi punto de vista, el trabajo realizado puede utilizarse como una herramienta de apoyo para los compañeros de la carrera de Ingeniería en Sistemas Electrónicos Industriales, ya que, desde mi experiencia personal, puedo decir que es complicado entender temas como el control automático, si no se relaciona la teoría con la práctica. El prototipo desarrollado, da la posibilidad de ver en tiempo real los efectos causados en el control de la posición de la flecha del motor al modificar los parámetros de su controlador PID. Efectos tales como inestabilidad en la respuesta de la planta debido a una mala sintonización de las ganancias del controlador.

Originalmente, el desarrollo del presente trabajo abarcaba la integración de la pantalla táctil con un sistema robótico de 6 grados de libertad, pero conforme se avanzó en su construcción, nos dimos cuenta que era demasiado ambicioso, por lo cual, se acotó a controlar dos motores de cd.

A la hora de implementar los circuitos electrónicos diseñados, se presentaron algunos problemas; uno de ellos, fue el ruido en las señales de los sensores, esto fue producido por residuos de pasta en las soldaduras de algunas de las tarjetas de circuito impreso que se realizaron. El problema fue difícil de detectar, porque el ruido se presentaba de forma aleatoria. Para diagnosticar el problema, se desconectó etapa por etapa, hasta lograr aislar la fuente del ruido. Esto nos enseñó que la problemática al construir un nuevo prototipo no se limita únicamente a su diseño, si no que involucra la resolución sistemática de diversos problemas que van surgiendo durante la construcción.

Finalmente, se puede decir que se cumplió con los objetivos planteados al inicio del presente trabajo, ya que, se logró construir el prototipo de un circuito electrónico capaz de detectar pulsaciones sobre una pantalla. Se logró crear una interfaz de usuario para modificar los parámetros del controlador PID, así como para indicar la posición deseada de la flecha del motor de cd. Así como la integración de todas las partes en un sólo sistema funcional.

4.1 Trabajo futuro

Falta integrar todas las etapas en una sola tarjeta de circuito impreso, para facilitar su transporte e instalación, así como diseñar la sección del microcontrolador para dejar de depender de la tarjeta de desarrollo, ya que es la parte más cara del sistema.

Aún queda por mejorar la resolución de detección de la pantalla, ya que actualmente está limitado a la distancia que hay entre sensor y sensor, la forma de aumentar la resolución sin agregar más sensores es utilizando la ventaja de leer una señal analógica de los sensores, y no, una señal digital, lo que nos permite leer variaciones de voltaje dependiendo de la posición del objeto respecto a dos sensores contiguos, ya que mientras más se aleja de un sensor más se acerca al otro, lo cual crea diferencias de voltaje que pueden ser analizadas para encontrar un patrón que puede ser introducido a un algoritmo que determine su posición .

Y finalmente, falta integrar la pantalla al sistema operativo Windows, ya que actualmente el sistema Sólo funciona dentro de la GUI que se creó, aunque ya se empezó a desarrollar un programa para integrarla el sistema operativo, y de hecho se tiene una primera versión de prueba, aún falta mucho para que esté completamente operativo. Se han hecho pruebas controlando el programa de dibujo Paint para realizar dibujos y las pruebas son alentadoras. En la figura 4.1 se muestra al usuario mientras desliza su dedo sobre la pantalla para formar la palabra UACM en una ventana del software Paint de Windows.

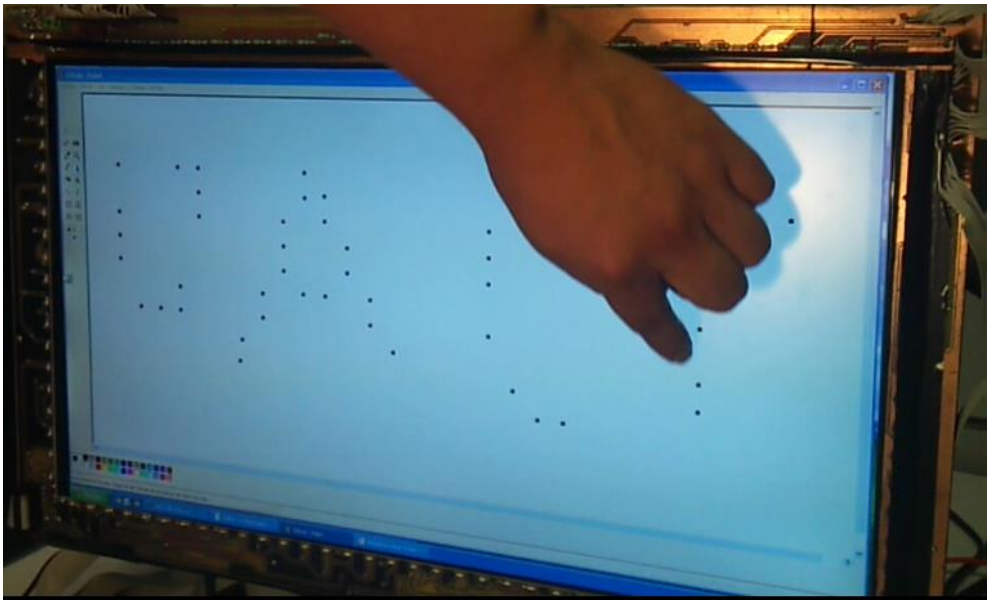


Figura 4.1: Dibujando en Paint

Anexo A

Publicaciones/Código fuente/Manuales

A.1 Ponencia en 1er Congreso de Tecnologías de la Información 2012

Partiendo del mismo trabajo descrito en esta tesis, se escribió un artículo para presentar la ponencia llamada "CONSTRUCCIÓN DE SUPERFICIE MULTITÁCTIL PARA CONTROL SERVOMOTORES DE D.C." la cual fue presentada en el "1er Congreso de Tecnologías de la Información 2012", llevado a cabo del 21 al 23 de marzo de 2012 en la Universidad Autónoma del Estado de Hidalgo (UAEH), campus Tlahuelilpan.

Las memorias del congreso fueron publicadas por la Universidad Autónoma del Estado de Hidalgo con el siguiente ISBN: 978-607-482-270-0.

A.2 Diagrama esquemático de multiplexor

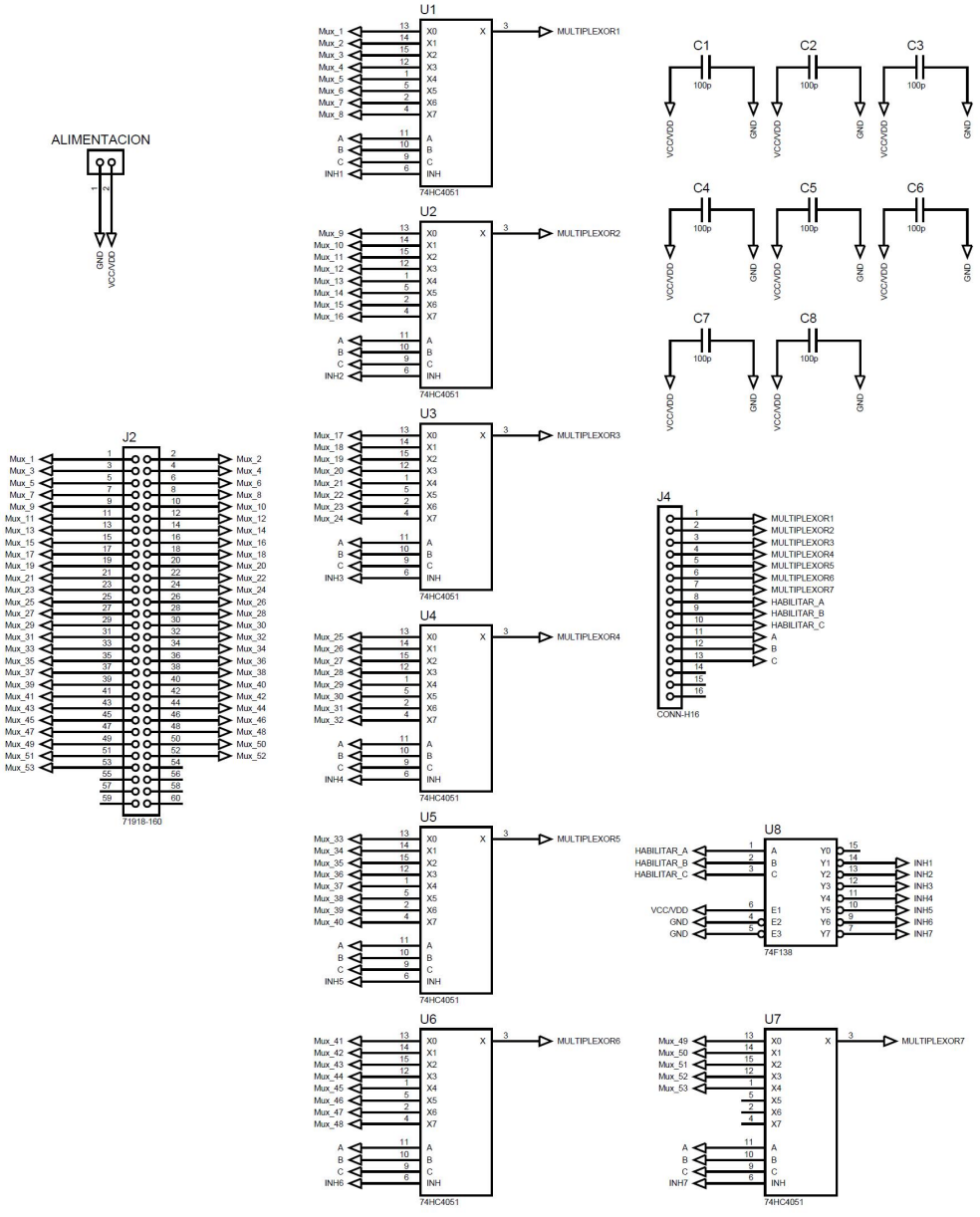


Figura A.1: Multiprexor

A.3 Código fuente del programa principal

```
1. #include"Cpu.h"
2. #include"Events.h"
3. #include"Contador1.h"
4. #include"Contador2.h"
5. #include"Multiplexor.h"
6. #include"SalidaMux.h"
7. #include"Columnas.h"
8. #include"Filas03.h"
9. #include"Filas47.h"
10. #include"Ganancia.h"
11. #include"AD1.h"
12. #include"Bit1.h"
13. #include"Initconsoleio.h"
14. #include"AS1.h"
15. #include"PETypes.h"
16. #include"PEError.h"
17. #include"PEConst.h"
18. #include"IOMap.h"
19. #include <stdio.h>
20. #include <string.h>
21.
22. constnumeroleds = 53;
23. constLecturasxled = 10;
24.
25. externintFlag = 0;
26. externintFoto = 0;
27. externintCuenta = 0;
28. externintBandera = 1;
29. externbyteerr = 0;
30. externintPromedio[Lecturasxled] =
    f0g;
31. externunsignedcharTemporal =
    0;
32. externintvalor1 = 0;
33. externintvalor2 = 0;
34.
35. inti, PromedioTotal = 0,
    Horizontal[34] = 0, Vertical[19] =
    0;
36. intCoordenadas[2] = 0,
    CoordenadasAnteriores[2] =
    {0}, cero = 0;
37.
38. PElowlevelinit();
39. AD1Enable();
40. AD1Start();
41. //AS1Enable();
```

```
42.
43. for(;;){
44. Foto = 1;
45. Cuenta = 0;
46. Flag = 1;
47.
48. while(Foto <= numeroleds)
49. {
50. Contador1Enable();
51. while(Cuenta <= Lecturasxled)
52. {
53. }
54. Contador1Disable();
55. Cuenta = 0;
56. Filas03PutVal(0);
57. Foto++;
58.
59.
60. PromedioTotal = cero;
61. for(i = 0;i < Lecturasxled;i++)
62. PromedioTotal+ = Promedio[i];
63. PromedioTotal = PromedioTotal/Lecturasxled;
64. if(PromedioTotal < 0)
65. PromedioTotal = PromedioTotal * -1;
66.
67. if(Foto <= 35)
68. Horizontal[Foto-2] = PromedioTotal;
69. if(Foto > 35&&Foto <= 54)
70. Vertical[Foto-36] = PromedioTotal;
71. }
72.
73. Coordenadas[0] = 0;
74. Coordenadas[1] = 0;
75. for(i = 0;i <= 33;i++){
76. if(Horizontal[i] <= 50)
77. Coordenadas[0] = i+1;
78. }
79.
80. for(i = 0;i <= 18;i++){
81. if(Vertical[i] <= 40)
82. Coordenadas[1] = i+1;
83. }
84.
85.
```

```
86. printf("x");
87. printf("%d",Coordenadas[0]);
88. printf("y");
89. printf("%d",Coordenadas[1]);
90. }
```

A.4 Código fuente de la interrupción del contador 1

```
1. voidContador1OnInterrupt(void)
2. {
3. //----------------
4. //VariablesGlobales
5. //----------------
6. externFlag;
7. externFoto;
8. externCuenta;
9. externunsignedcharTemporal;
10.
11. //----------------
12. //ConstantesParalasColumnas
13. //----------------
14.
15. unsignedcharColumna1 = 1;
16. unsignedcharColumna2 = 2;
17. unsignedcharColumna3 = 4;
18. unsignedcharColumna4 = 8;
19. unsignedcharColumna5 = 16;
20. unsignedcharColumna6 = 32;
21. unsignedcharColumna7 = 64;
22.
23. //----------------
24. //ConstantesParalasFilas
25. //----------------
26. unsignedcharFila1 = 1;
27. unsignedcharFila2 = 2;
28. unsignedcharFila3 = 4;
29. unsignedcharFila4 = 8;
30. unsignedcharFila5 = 1;
31. unsignedcharFila6 = 2;
32. unsignedcharFila7 = 4;
33. unsignedcharFila8 = 8;
34.
35. //----------------
36. //IniciodelaISRdelContador1
37. //----------------
```

```
38.
39. if(Flag == 1){
40. switch(Foto){
41.   case1 :
42.     Contador2Enable();
43.     GananciaClrVal();
44.     Filas47PutVal(0);
45.     ColumnasPutVal(Columna6);
46.     Filas03PutVal(Fila3);
47.     MultiplexorPutVal(1);
48.     SalidaMuxPutVal(0);
49.     break;
50.
51.   case2 :
52.     Contador2Enable();
53.     GananciaClrVal();
54.     Filas47PutVal(0);
55.     MultiplexorPutVal(1);
56.     SalidaMuxPutVal(1);
57.     ColumnasPutVal(Columna7);
58.     Filas03PutVal(Fila3);
59.     break;
60.
61.   case3 :
62.     Contador2Enable();
63.     GananciaClrVal();
64.     Filas47PutVal(0);
65.     MultiplexorPutVal(1);
66.     SalidaMuxPutVal(2);
67.     ColumnasPutVal(Columna1);
68.     Filas03PutVal(Fila4);
69.     break;
70.
71.   case4 :
72.     Contador2Enable();
73.     MultiplexorPutVal(1);
74.     SalidaMuxPutVal(3);
75.     ColumnasPutVal(Columna2);
76.     Filas03PutVal(Fila4);
77.     Filas47PutVal(0);
78.     GananciaClrVal();
79.     break;
80.
81.   case5 :
82.     Contador2Enable();
83.     MultiplexorPutVal(1);
84.     SalidaMuxPutVal(4);
```

```
85.     ColumnasPutVal(Columna3);
86.     Filas03PutVal(Fila4);
87.     Filas47PutVal(0);
88.     GananciaClrVal();
89.     break;
90.
91.  case6 :
92.     Contador2Enable();
93.     MultiplexorPutVal(1);
94.     SalidaMuxPutVal(5);
95.     ColumnasPutVal(Columna4);
96.     Filas03PutVal(Fila4);
97.     Filas47PutVal(0);
98.     GananciaClrVal();
99.     break;
100.
101.  case7 :
102.     Contador2Enable();
103.     MultiplexorPutVal(1);
104.     SalidaMuxPutVal(6);
105.     ColumnasPutVal(Columna5);
106.     Filas03PutVal(Fila4);
107.     Filas47PutVal(0);
108.     GananciaClrVal();
109.     break;
110.
111.  case8 :
112.     Contador2Enable();
113.     MultiplexorPutVal(1);
114.     SalidaMuxPutVal(7);
115.     ColumnasPutVal(Columna6);
116.     Filas03PutVal(Fila4);
117.     Filas47PutVal(0);
118.     GananciaClrVal();
119.     break;
120.
121.  case9 :
122.     Contador2Enable();
123.     MultiplexorPutVal(2);
124.     SalidaMuxPutVal(0);
125.     ColumnasPutVal(Columna7);
126.     Filas03PutVal(Fila4);
127.     Filas47PutVal(0);
128.     GananciaClrVal();
129.     break;
130.
131.  case10 :
```

```
132.     Contador2Enable();
133.     MultiplexorPutVal(2);
134.     SalidaMuxPutVal(1);
135.     ColumnasPutVal(Columna1);
136.     Filas47PutVal(Fila5);
137.     Filas03PutVal(0);
138.     GananciaClrVal();
139.     break;
140.
141.  case11 :
142.     Contador2Enable();
143.     MultiplexorPutVal(2);
144.     SalidaMuxPutVal(2);
145.     ColumnasPutVal(Columna2);
146.     Filas47PutVal(Fila5);
147.     Filas03PutVal(0);
148.     GananciaClrVal();
149.     break;
150.
151.  case12 :
152.     Contador2Enable();
153.     MultiplexorPutVal(2);
154.     SalidaMuxPutVal(3);
155.     ColumnasPutVal(Columna3);
156.     Filas47PutVal(Fila5);
157.     Filas03PutVal(0);
158.     GananciaClrVal();
159.     break;
160.
161.  case13 :
162.     Contador2Enable();
163.     MultiplexorPutVal(2);
164.     SalidaMuxPutVal(4);
165.     ColumnasPutVal(Columna4);
166.     Filas47PutVal(Fila5);
167.     Filas03PutVal(0);
168.     GananciaClrVal();
169.     break;
170.
171.  case14 :
172.     Contador2Enable();
173.     MultiplexorPutVal(2);
174.     SalidaMuxPutVal(5);
175.     ColumnasPutVal(Columna5);
176.     Filas47PutVal(Fila5);
177.     Filas03PutVal(0);
178.     GananciaClrVal();
```

```
179.      break;
180.
181.    case 15 :
182.      Contador2Enable();
183.      MultiplexorPutVal(2);
184.      SalidaMuxPutVal(6);
185.      ColumnasPutVal(Columna6);
186.      Filas47PutVal(Fila5);
187.      Filas03PutVal(0);
188.      GananciaClrVal();
189.      break;
190.
191.    case 16 :
192.      Contador2Enable();
193.      MultiplexorPutVal(2);
194.      SalidaMuxPutVal(7);
195.      ColumnasPutVal(Columna7);
196.      Filas47PutVal(Fila5);
197.      Filas03PutVal(0);
198.      GananciaClrVal();
199.      break;
200.
201.    case 17 :
202.      Contador2Enable();
203.      MultiplexorPutVal(3);
204.      SalidaMuxPutVal(0);
205.      ColumnasPutVal(Columna1);
206.      Filas47PutVal(Fila6);
207.      Filas03PutVal(0);
208.      GananciaClrVal();
209.      break;
210.
211.    case 18 :
212.      Contador2Enable();
213.      MultiplexorPutVal(3);
214.      SalidaMuxPutVal(1);
215.      ColumnasPutVal(Columna2);
216.      Filas47PutVal(Fila6);
217.      Filas03PutVal(0);
218.      GananciaClrVal();
219.      break;
220.
221.    case 19 :
222.      Contador2Enable();
223.      MultiplexorPutVal(3);
224.      SalidaMuxPutVal(2);
225.      ColumnasPutVal(Columna3);
```

```
226.      Filas47PutVal(Fila6);
227.      Filas03PutVal(0);
228.      GananciaClrVal();
229.      break;
230.
231.    case20 :
232.      Contador2Enable();
233.      MultiplexorPutVal(3);
234.      SalidaMuxPutVal(3);
235.      ColumnasPutVal(Columna4);
236.      Filas47PutVal(Fila6);
237.      Filas03PutVal(0);
238.      GananciaClrVal();
239.      break;
240.
241.    case21 :
242.      Contador2Enable();
243.      MultiplexorPutVal(3);
244.      SalidaMuxPutVal(4);
245.      ColumnasPutVal(Columna5);
246.      Filas47PutVal(Fila6);
247.      Filas03PutVal(0);
248.      GananciaClrVal();
249.      break;
250.
251.    case22 :
252.      Contador2Enable();
253.      MultiplexorPutVal(3);
254.      SalidaMuxPutVal(5);
255.      ColumnasPutVal(Columna6);
256.      Filas47PutVal(Fila6);
257.      Filas03PutVal(0);
258.      GananciaClrVal();
259.      break;
260.
261.    case23 :
262.      Contador2Enable();
263.      MultiplexorPutVal(3);
264.      SalidaMuxPutVal(6);
265.      ColumnasPutVal(Columna7);
266.      Filas47PutVal(Fila6);
267.      Filas03PutVal(0);
268.      GananciaClrVal();
269.      break;
270.
271.    case24 :
272.      Contador2Enable();
```

```
273.      MultiplexorPutVal(3);
274.      SalidaMuxPutVal(7);
275.      ColumnasPutVal(Columna1);
276.      Filas47PutVal(Fila7);
277.      Filas03PutVal(0);
278.      GananciaClrVal();
279.      break;
280.
281.   case25 :
282.      Contador2Enable();
283.      MultiplexorPutVal(4);
284.      SalidaMuxPutVal(0);
285.      ColumnasPutVal(Columna2);
286.      Filas47PutVal(Fila7);
287.      Filas03PutVal(0);
288.      GananciaClrVal();
289.      break;
290.
291.   case26 :
292.      Contador2Enable();
293.      MultiplexorPutVal(4);
294.      SalidaMuxPutVal(1);
295.      ColumnasPutVal(Columna3);
296.      Filas47PutVal(Fila7);
297.      Filas03PutVal(0);
298.      GananciaClrVal();
299.      break;
300.
301.   case27 :
302.      Contador2Enable();
303.      MultiplexorPutVal(4);
304.      SalidaMuxPutVal(2);
305.      ColumnasPutVal(Columna4);
306.      Filas47PutVal(Fila7);
307.      Filas03PutVal(0);
308.      GananciaClrVal();
309.      break;
310.
311.   case28 :
312.      Contador2Enable();
313.      MultiplexorPutVal(4);
314.      SalidaMuxPutVal(3);
315.      ColumnasPutVal(Columna5);
316.      Filas47PutVal(Fila7);
317.      Filas03PutVal(0);
318.      GananciaClrVal();
319.      break;
```

```
320.
321.    case 29 :
322.       Contador2Enable();
323.       MultiplexorPutVal(4);
324.       SalidaMuxPutVal(4);
325.       ColumnasPutVal(Columna6);
326.       Filas47PutVal(Fila7);
327.       Filas03PutVal(0);
328.       GananciaClrVal();
329.       break;
330.
331.    case 30 :
332.       Contador2Enable();
333.       MultiplexorPutVal(4);
334.       SalidaMuxPutVal(5);
335.       ColumnasPutVal(Columna7);
336.       Filas47PutVal(Fila7);
337.       Filas03PutVal(0);
338.       GananciaClrVal();
339.       break;
340.
341.    case 31 :
342.       Contador2Enable();
343.       MultiplexorPutVal(4);
344.       SalidaMuxPutVal(6);
345.       ColumnasPutVal(Columna1);
346.       Filas47PutVal(Fila8);
347.       Filas03PutVal(0);
348.       GananciaClrVal();
349.       break;
350.
351.    case 32 :
352.       Contador2Enable();
353.       MultiplexorPutVal(4);
354.       SalidaMuxPutVal(7);
355.       ColumnasPutVal(Columna2);
356.       Filas47PutVal(Fila8);
357.       Filas03PutVal(0);
358.       GananciaClrVal();
359.       break;
360.
361.    case 33 :
362.       Contador2Enable();
363.       MultiplexorPutVal(5);
364.       SalidaMuxPutVal(0);
365.       ColumnasPutVal(Columna3);
366.       Filas47PutVal(Fila8);
```

```
367.        Filas03PutVal(0);
368.        GananciaClrVal();
369.        break;
370.
371.    case 34 :
372.        Contador2Enable();
373.        MultiplexorPutVal(5);
374.        SalidaMuxPutVal(1);
375.        ColumnasPutVal(Columna4);
376.        Filas47PutVal(Fila8);
377.        Filas03PutVal(0);
378.        GananciaClrVal();
379.        break;
380.
381.  //————————-
382.  // Leds del 1 al 19
383.  //————————
384.
385.    case 35 :
386.        Contador2Enable();
387.        MultiplexorPutVal(5);
388.        SalidaMuxPutVal(2);
389.        ColumnasPutVal(Columna1);
390.        Filas03PutVal(Fila1);
391.        Filas47PutVal(0);
392.        GananciaClrVal();
393.        break;
394.
395.    case 36 :
396.        Contador2Enable();
397.        MultiplexorPutVal(5);
398.        SalidaMuxPutVal(3);
399.        ColumnasPutVal(Columna2);
400.        Filas03PutVal(Fila1);
401.        Filas47PutVal(0);
402.        GananciaClrVal();
403.        break;
404.
405.    case 37 :
406.        Contador2Enable();
407.        MultiplexorPutVal(5);
408.        SalidaMuxPutVal(4);
409.        ColumnasPutVal(Columna3);
410.        Filas03PutVal(Fila1);
411.        Filas47PutVal(0);
412.        GananciaClrVal();
413.        break;
```

```
414.
415.    case 38 :
416.       Contador2Enable();
417.       MultiplexorPutVal(5);
418.       SalidaMuxPutVal(5);
419.       ColumnasPutVal(Columna4);
420.       Filas03PutVal(Fila1);
421.       Filas47PutVal(0);
422.       GananciaClrVal();
423.       break;
424.
425.    case 39 :
426.       Contador2Enable();
427.       MultiplexorPutVal(5);
428.       SalidaMuxPutVal(6);
429.       ColumnasPutVal(Columna5);
430.       Filas03PutVal(Fila1);
431.       Filas47PutVal(0);
432.       GananciaClrVal();
433.       break;
434.
435.    case 40 :
436.       Contador2Enable();
437.       MultiplexorPutVal(5);
438.       SalidaMuxPutVal(7);
439.       ColumnasPutVal(Columna6);
440.       Filas03PutVal(Fila1);
441.       Filas47PutVal(0);
442.       GananciaClrVal();
443.       break;
444.
445.    case 41 :
446.       Contador2Enable();
447.       MultiplexorPutVal(6);
448.       SalidaMuxPutVal(0);
449.       ColumnasPutVal(Columna7);
450.       Filas03PutVal(Fila1);
451.       Filas47PutVal(0);
452.       GananciaClrVal();
453.       break;
454.
455.    case 42 :
456.       Contador2Enable();
457.       MultiplexorPutVal(6);
458.       SalidaMuxPutVal(1);
459.       ColumnasPutVal(Columna1);
460.       Filas03PutVal(Fila2);
```

```
461.        Filas47PutVal(0);
462.        GananciaClrVal();
463.        break;
464.
465.    case 43 :
466.        Contador2Enable();
467.        MultiplexorPutVal(6);
468.        SalidaMuxPutVal(2);
469.        ColumnasPutVal(Columna2);
470.        Filas03PutVal(Fila2);
471.        Filas47PutVal(0);
472.        GananciaClrVal();
473.        break;
474.
475.    case 44 :
476.        Contador2Enable();
477.        MultiplexorPutVal(6);
478.        SalidaMuxPutVal(3);
479.        ColumnasPutVal(Columna3);
480.        Filas03PutVal(Fila2);
481.        Filas47PutVal(0);
482.        GananciaClrVal();
483.        break;
484.
485.    case 45 :
486.        Contador2Enable();
487.        MultiplexorPutVal(6);
488.        SalidaMuxPutVal(4);
489.        ColumnasPutVal(Columna4);
490.        Filas03PutVal(Fila2);
491.        Filas47PutVal(0);
492.        GananciaClrVal();
493.        break;
494.
495.    case 46 :
496.        Contador2Enable();
497.        MultiplexorPutVal(6);
498.        SalidaMuxPutVal(5);
499.        ColumnasPutVal(Columna5);
500.        Filas03PutVal(Fila2);
501.        Filas47PutVal(0);
502.        GananciaClrVal();
503.        break;
504.
505.    case 47 :
506.        Contador2Enable();
507.        MultiplexorPutVal(6);
```

```
508.      SalidaMuxPutVal(6);
509.      ColumnasPutVal(Columna6);
510.      Filas03PutVal(Fila2);
511.      Filas47PutVal(0);
512.      GananciaClrVal();
513.      break;
514.
515.  case48 :
516.      Contador2Enable();
517.      MultiplexorPutVal(6);
518.      SalidaMuxPutVal(7);
519.      ColumnasPutVal(Columna7);
520.      Filas03PutVal(Fila2);
521.      Filas47PutVal(0);
522.      GananciaClrVal();
523.      break;
524.
525.  case49 :
526.      Contador2Enable();
527.      MultiplexorPutVal(7);
528.      SalidaMuxPutVal(0);
529.      ColumnasPutVal(Columna1);
530.      Filas03PutVal(Fila3);
531.      Filas47PutVal(0);
532.      GananciaClrVal();
533.      break;
534.
535.  case50 :
536.      Contador2Enable();
537.      MultiplexorPutVal(7);
538.      SalidaMuxPutVal(1);
539.      ColumnasPutVal(Columna2);
540.      Filas03PutVal(Fila3);
541.      Filas47PutVal(0);
542.      GananciaClrVal();
543.      break;
544.
545.  case51 :
546.      Contador2Enable();
547.      MultiplexorPutVal(7);
548.      SalidaMuxPutVal(2);
549.      ColumnasPutVal(Columna3);
550.      Filas03PutVal(Fila3);
551.      Filas47PutVal(0);
552.      GananciaClrVal();
553.      break;
554.
```

```
555.    case 52 :
556.       Contador2Enable();
557.       MultiplexorPutVal(7);
558.       SalidaMuxPutVal(3);
559.       ColumnasPutVal(Columna4);
560.       Filas03PutVal(Fila3);
561.       Filas47PutVal(0);
562.       GananciaClrVal();
563.       break;
564.
565.    case 53 :
566.       Contador2Enable();
567.       MultiplexorPutVal(7);
568.       SalidaMuxPutVal(4);
569.       ColumnasPutVal(Columna5);
570.       Filas03PutVal(Fila3);
571.       Filas47PutVal(0);
572.       GananciaClrVal();
573.       $break;
574. }
575. }
576.
577. //----------------
578. //Segundaparte del codigo
579. //----------------
580.
581. if(Flag == 0){
582. Contador2Enable();
583. ColumnasPutVal(0);
584. Cuenta++;
585. }
586.
587. //----------------
588.
589. Flag=1;
590.
591. }
```

A.5 Código fuente de la interrupción del contador 2

```
1.
2. voidContador2OnInterrupt(void)
```

```
3. {
4. //----------------------
5. //VariablesGlobales
6. //----------------
7.
8. externerr;
9. externFoto;
10. externCuenta;
11. externPromedio[5];
12. externBandera;
13. externvalor1;
14. externvalor2;
15. //----------------------
16. //VariablesLocales
17. //----------------------
18.
19. wordLectura;
20. intLectura2;
21. //----------------------
22. //IniciodelaISRdelContador2
23. //----------------
24.
25. if(Bandera)
```

```
26. {
27.     err = AD1GetValue(&Lectura);
28.     Lectura2 = Lectura >> 3;
29.     valor1 = Lectura * 330/4095;
30.     }
31.
32. if(!Bandera){
33.     err = AD1GetValue(&Lectura);
34.     Lectura2 = Lectura >> 3;
35.     valor2 = Lectura * 330/4095;
36.     Promedio[Cuenta?1] = valor1?valor2;
37.     valor1 = valor2 = 0;
38.     }
39.
40. Contador2 Disable();
41. Bandera=1;
42.
43.     }//Termina la ISR
```

A.6 Código fuente de la Interfaz gráfica de usuario

```
1. functionvarargout = quarccreatingmatlabguisdemo(varargin)
2. %QUARCCREATINGMATLABGUISDEMOM - file
   for_quarc_creating_matlab_guisdemo.fig
```

3. %QUARCCREATINGMATLABGUISDEMO, byitself, createsanew QUARCCREATINGMATLABGUISDEMOorraisestheexisting
4. %singleton.
5. %
6. %H = QUARCCREATINGMATLABGUISDEMOreturnsthehandletoanew QUARCCREATINGMATLABGUISDEMOorthehandleto
7. %theexistingsingleton.
8. %
9. %QUARCCREATINGMATLABGUISDEMO('CALLBACK', hObject, eventData, handles, ...)callsthelocal
10. %functionnamedCALLBACKinQUARCCREATINGMATLABGUISDEMO.M withthegiveninputarguments.
11. %
12. %QUARCCREATINGMATLABGUISDEMO('Property','Value', ...) createsanewQUARCCREATINGMATLABGUISDEMOorraisesthe
13. %existingsingleton.Startingfromtheleft, property valuepairsare
14. %appliedtotheGUIbeforequarccreatingmatlabguis demoOpeningFunctiongetscalled.An
15. %unrecognizedpropertynameorinvalidvaluemakes propertyapplication
16. %stop.Allinputsarepassedtoquarccreatingmatlab guisdemoOpeningFcnviavarargin.
17. %
18. % ∗ SeeGUIOptionsonGUIDE'sToolsmenu.Choose"GUIallowsonlyone
19. %instancetorun(singleton)".

20. %
21. %Seealso : GUIDE, GUIDATA, GUIHANDLES
22.
23. %Edittheabovetexttomodifytheresponsetohelpquar ccreatingmatlabguisdemo
24.
25. %LastModifiedbyGUIDEv2.517 − Feb − 201212 : 46 : 41
26.
27. %Begininitializationcode − DONOTEDIT
28. guiSingleton = 1;
29. guiState = struct('guiName', mfilename, ...
30. 'guiSingleton', guiSingleton, ...
31. 'guiOpeningFcn', @quarccreatingmatlabguisdemoOpeningFcn, ...
32. 'guiOutputFcn', @quarccreatingmatlabguisdemoOutputFcn,
33. 'guiLayoutFcn', [], ...
34. 'guiCallback', []);
35. if nargin&&ischar(varargin{1})
36. guiState.guiCallback = str2func(varargin{1});
37. end
38.
39. if nargout
40. [varargout{1 : nargout}] = guimainfcn(guiState, varargin{:});
41. else

42. $guimainfcn(guiState, varargin f : g);$
43. end
44. $\%Endinitializationcode - DONOTEDIT$
45.
46.
47. $\%---Executesjustbeforequarccreatingmatlabguis demoismadevisible.$
48. $functionquarccreatingmatlabguisdemoOpeningFcn (hObject, eventdata, handles, varargin)$
49. $\%Thisfunctionhasnooutputargs, seeOutputFcn.$
50. $\%hObjecthandletofigure$
51. $\%eventdatareserved - tobedefinedinafuture versionofMATLAB$
52. $\%handlesstructurewithhandlesanduserdata (seeGUIDATA)$
53. $\%varargincommandlineargumentstoquarccreating matlabguisdemo(seeVARARGIN)$
54.
55. $globalthandles1$
56.
57. $handles1 = handles;$
58.
59. $\%Choosedefaultcommandlineoutputforquarc creatingmatlabguisdemo$
60. $handles.output = hObject;$

61. $handles.model = ' quarcmatlabguidemo1';$
62. $loadsystem(handles.model);$
63.
64. $\%SEDECLARALARUTINAQUEDEBEDEEJECUTARSE$
    $ALCERRARLAAPLICACION$
65. $\%set(handles.figure1,'CloseRequestFcn',$
    $@figure1CloseRequestFcn);$
66. $\%CREAMOSELOBJETOTIMERYSEARRANCA$
67. $t = timer('TimerFcn', @tiempo,' ExecutionMode$
    $',' fixedSpacing',' BusyMode',' queue',' Period', 0.001);$
68. $start(t);$
69.
70. $\%Updatehandlesstructure$
71. $guidata(hObject, handles);$
72.
73. $\%UIWAITmakesquarccreatingmatlabguisdemo$
    $waitforuserresponse(seeUIRESUME)$
74. $\%uiwait(handles.figure1);$
75.
76.
77. $\%---Outputsfromthisfunctionarereturnedtothecommandline.$
78. $functionvarargout = quarccreatingmatlabguis$
    $demoOutputFcn(hObject, eventdata, handles)$
79. $\%varargoutcellarrayforreturningoutputargs(seeVARARGOUT);$
80. $\%hObjecthandletofigure$

```
81. %eventdatareserved - tobedefinedinafutureversionofMATLAB
82. %handlesstructurewithhandlesanduserdata(seeGUIDATA)
83.
84. %Getdefaultcommandlineoutputfromhandlesstructure
85. varargoutf1g = handles.output;
86.
87.
88. %---ExecutesonbuttonpressinpushbuttonBuild.
89. functionpushbuttonBuildCallback(hObject,eventdata,handles)
90. %hObjecthandletopushbuttonBuild(seeGCBO)
91. %eventdatareserved - tobedefinedinafutureversionofMATLAB
92. %handlesstructurewithhandlesanduserdata(seeGUIDATA)
93. qcbuildmodel(handles.model);
94.
95. %---ExecutesonbuttonpressinpushbuttonStart.
96. functionpushbuttonStartCallback(hObject,eventdata,handles)
97. %hObjecthandletopushbuttonStart(seeGCBO)
98. %eventdatareserved - tobedefinedinafutureversionofMATLAB
99. %handlesstructurewithhandlesanduserdata(seeGUIDATA)
100. qcstartmodel(handles.model);
101.
102. %---ExecutesonbuttonpressinpushbuttonStop.
103. functionpushbuttonStopCallback(hObject,eventdata,handles)
```

```
104. %hObject handle to pushbutton Stop (see GCBO)
105. %eventdata reserved - to be defined in a future version of MATLAB
106. %handles structure with handles and user data (see GUIDATA)
107. qcstopmodel(handles.model);
108.
109. %--- Executes on slider movement.
110. function sliderFrequency Callback(hObject, eventdata, handles)
111. %hObject handle to sliderFrequency (see GCBO)
112. %eventdata reserved - to be defined in a future version of MATLAB
113. %handles structure with handles and user data (see GUIDATA)
114.
115. %Hints: get(hObject,'Value') returns position of slider
116. %get(hObject,'Min') and get(hObject,'Max')
     to determine range of slider
117. value = get(hObject,'Value');
118. set param([handles.model,'/K'],'Gain',num2str(value));
119. set(handles.showV,'String',num2str(value));
120.
121. %--- Executes during object creation, after
     setting all properties.
122. function sliderFrequency CreateFcn(hObject, eventdata, handles)
123. %hObject handle to sliderFrequency (see GCBO)
124. %eventdata reserved - to be defined in a future version of MATLAB
```

```
125. %handlesempty - handlesnotcreateduntil
     afterallCreateFcnscalled
126.
127. %Hint : slidercontrolsusuallyhavealightgraybackground.
128. ifisequal(get(hObject,'BackgroundColor'),
     get(0,'defaultUicontrolBackgroundColor'))
129. set(hObject,'BackgroundColor',[.9.9.9]);
130. end
131.
132.
133.
134.
135. %Executeswhenuserattemptstoclosefigure1.
136. functionfigure1CloseRequestFcn(hObject,eventdata,handles)
137. %hObjecthandletofigure1(seeGCBO)
138. %eventdatareserved - tobedefinedinafutureversionofMATLAB
139. %handlesstructurewithhandlesanduserdata(seeGUIDATA)
140. closesystem(handles.model,0);
141. %Hint : delete(hObject)closesthefigure
142.
143. %-------------------------
144. %Cerramos el puerto COM1
145. %-------------------------
146. globalt
```

147.

148. $stop(t)$;

149. $closeall$

150. $clearall$

151. $INSTRFIND$

152.

153. $\%---$

154. $\%Fin\ de\ cerrar\ el\ puerto\ COM1$

155. $\%---$

156. $delete(hObject)$;

157.

158.

159.

160.

161. $\%---Executesonslidermovement.$

162. $functionedKPCallback(hObject, eventdata, handles)$

163. $\%hObjecthandletoedKP(seeGCBO)$

164. $\%eventdatareserved-tobedefinedinafutureversionofMATLAB$

165. $\%handlesstructurewithhandlesanduserdata(seeGUIDATA)$

166.

167. $\%Hints: get(hObject,'Value')returnspositionofslider$

168. $\%get(hObject,'Min')andget(hObject,'Max')$
$todeterminerangeofslider$

```
value = get(hObject,'Value');
setparam([handles.model,'/KP'],'Gain',num2str(value));
set(handles.showKP,'String',num2str(value));

%---Executesduringobjectcreation,
aftersettingallproperties.
functionedKPCreateFcn(hObject,eventdata,handles)
%hObjecthandletoedKP(seeGCBO)
%eventdatareserved-tobedefinedinafutureversionofMATLAB
%handlesempty-handlesnotcreateduntil
afterallCreateFcnscalled

%Hint:slidercontrolsusuallyhavealightgraybackground.
ifisequal(get(hObject,'BackgroundColor')
,get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor',[.9.9.9]);
end


%---Executesonslidermovement.
functionedKDCallback(hObject,eventdata,handles)
%hObjecthandletoedKD(seeGCBO)
%eventdatareserved-tobedefinedinafutureversionofMATLAB
%handlesstructurewithhandlesanduserdata(seeGUIDATA)
```

```
190.
191. %Hints: get(hObject,'Value')returnspositionofslider
192. %get(hObject,'Min')andget(hObject,'Max')
     todeterminerangeofslider
193. value = get(hObject,'Value');
194. setparam([handles.model,'/KD'],'Gain',num2str(value));
195. set(handles.showKD,'String',num2str(value));
196.
197. %---Executesduringobjectcreation,aftersettingallproperties.
198. functionedKDCreateFcn(hObject,eventdata,handles)
199. %hObjecthandletoedKD(seeGCBO)
200. %eventdatareserved-tobedefinedinafutureversionofMATLAB
201. %handlesempty-handlesnotcreateduntil
     afterallCreateFcnscalled
202.
203. %Hint: slidercontrolsusuallyhavealightgraybackground.
204. ifisequal(get(hObject,'BackgroundColor'),
     get(0,'defaultUicontrolBackgroundColor'))
205. set(hObject,'BackgroundColor',[.9.9.9]);
206. end
207.
208.
209. %---Executesonslidermovement.
210. functionedKICallback(hObject,eventdata,handles)
```

```
211. %hObjecthandletoedKI(seeGCBO)
212. %eventdatareserved-tobedefinedinafutureversionofMATLAB
213. %handlesstructurewithhandlesanduserdata(seeGUIDATA)
214.
215. %Hints:get(hObject,'Value')returnspositionofslider
216. %get(hObject,'Min')andget(hObject,'Max')
     todeterminerangeofslider
217. value=get(hObject,'Value');
218. setparam([handles.model,'/KI'],'Gain',num2str(value));
219. set(handles.showKI,'String',num2str(value));
220.
221. %---Executesduringobjectcreation,aftersettingallproperties.
222. functionedKICreateFcn(hObject,eventdata,handles)
223. %hObjecthandletoedKI(seeGCBO)
224. %eventdatareserved-tobedefinedinafutureversionofMATLAB
225. %handlesempty-handlesnotcreateduntilafterallCreateFcnscalled
226.
227. %Hint:slidercontrolsusuallyhavealightgraybackground.
228. ifisequal(get(hObject,'BackgroundColor'),
     get(0,'defaultUicontrolBackgroundColor'))
229. set(hObject,'BackgroundColor',[.9.9.9]);
230. end
231.
232. %-----------------------------------------------
```

233. $\%----------$

234.

235. $\%TIMEREMPIEZAAREALIZARTAREAASIGNADA$

236. $functiontiempo(hObject,eventdata)\%(source,eventdata)$

237.

238. $globalconectadomyoutpipemyinpipedataindataouthandles1$

239. $int16data;$

240. $\%----------$

241. $PS = serial('COM2');$

242. $set(PS,'Baudrate',38400);\%seconfiguralavelocidada9600Baudios$

243. $set(PS,'StopBits',1);\%seconfigurabitdeparadaauno$

244. $set(PS,'DataBits',8);\%seconfiguraqueeldatoesde8bits,$
$debeestarentre5y8$

245. $set(PS,'Parity','none');\%seconfigurasinparidad$

246. $set(PS,'Terminator','CR/LF');\%ccaracterconquefinalizaenvio$

247. $set(PS,'OutputBufferSize',1);\%neselnumerodebytesaenviar$

248. $set(PS,'InputBufferSize',2);\%neselnumerodebytesarecibir$

249. $set(PS,'Timeout',5);\%5segundosdetiempodeespera$

250.

251. $fopen(PS);$

252. $i = 105;$

253. $f = 102;$

254.

```
xy = [];
Boton196 = [105495754102];%i196f
Boton249 = [105505257102];%i249f
Boton294 = [105505752102];%i294f
xy = fread(PS,1,'uchar');
if xy == i
while xy  = f
xy = [xyfread(PS,1,'uchar')];
end;
xy = char(xy);
xy
end
%- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%   BarraKP
%- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

if xy =='i0812f'
    set(handles1.edKP,'value',0);
    set(handles1.showKP,'String','0');
elseif xy =='i0912f'
    set(handles1.edKP,'value',5);
    set(handles1.showKP,'String','5');
elseif xy =='i1012f'
```

```
278.    set(handles1.edKP,'value',10);
279.    set(handles1.showKP,'String','10');
280. elseifxy =='i1112f'
281.    set(handles1.edKP,'value',15);
282.    set(handles1.showKP,'String','15');
283. elseifxy =='i1212f'
284.    set(handles1.edKP,'value',20);
285.    set(handles1.showKP,'String','20');
286. elseifxy =='i1312f'
287.    set(handles1.edKP,'value',25);
288.    set(handles1.showKP,'String','25');
289. elseifxy =='i1412f'
290.    set(handles1.edKP,'value',30);
291.    set(handles1.showKP,'String','30');
292. elseifxy =='i1512f'
293.    set(handles1.edKP,'value',35);
294.    set(handles1.showKP,'String','35');
295. elseifxy =='i1612f'
296.    set(handles1.edKP,'value',40);
297.    set(handles1.showKP,'String','40');
298. elseifxy =='i1712f'
299.    set(handles1.edKP,'value',45);
300.    set(handles1.showKP,'String','45');
```

```
301. elseif xy ==' i1812f'
302.     set(handles1.edKP,'value',50);
303.     set(handles1.showKP,'String','50');
304. elseif xy ==' i1912f'
305.     set(handles1.edKP,'value',55);
306.     set(handles1.showKP,'String','55');
307. elseif xy ==' i2012f'
308.     set(handles1.edKP,'value',60);
309.     set(handles1.showKP,'String','60');
310. elseif xy ==' i2112f'
311.     set(handles1.edKP,'value',65);
312.     set(handles1.showKP,'String','65');
313. elseif xy ==' i2212f'
314.     set(handles1.edKP,'value',70);
315.     set(handles1.showKP,'String','70');
316. elseif xy ==' i2312f'
317.     set(handles1.edKP,'value',75);
318.     set(handles1.showKP,'String','75');
319. elseif xy ==' i2412f'
320.     set(handles1.edKP,'value',80);
321.     set(handles1.showKP,'String','80');
322. elseif xy ==' i2512f'
323.     set(handles1.edKP,'value',85);
```

```
    set(handles1.showKP,'String','85');
elseif xy=='i2612f'
    set(handles1.edKP,'value',90);
    set(handles1.showKP,'String','90');
elseif xy=='i2712f'
    set(handles1.edKP,'value',95);
    set(handles1.showKP,'String','95');
elseif xy=='i2812f'
    set(handles1.edKP,'value',100);
    set(handles1.showKP,'String','100');


% ---------------------------------------
%  BarraKD
% ---------------------------------------
elseif xy=='i0814f'
    set(handles1.edKD,'value',0);
    set(handles1.showKD,'String','0');
elseif xy=='i0914f'
    set(handles1.edKD,'value',0.05);
    set(handles1.showKD,'String','0.05');
elseif xy=='i1014f'
    set(handles1.edKD,'value',0.1);
```

```
347.    set(handles1.showKD,'String','0.1');
348. elseif xy =='i1114f'
349.    set(handles1.edKD,'value',0.15);
350.    set(handles1.showKD,'String','0.15');
351. elseif xy =='i1214f'
352.    set(handles1.edKD,'value',0.2);
353.    set(handles1.showKD,'String','0.2');
354. elseif xy =='i1314f'
355.    set(handles1.edKD,'value',0.25);
356.    set(handles1.showKD,'String','0.25');
357. elseif xy =='i1414f'
358.    set(handles1.edKD,'value',0.3);
359.    set(handles1.showKD,'String','0.3');
360. elseif xy =='i1514f'
361.    set(handles1.edKD,'value',0.35);
362.    set(handles1.showKD,'String','0.35');
363. elseif xy =='i1614f'
364.    set(handles1.edKD,'value',0.35);
365.    set(handles1.showKD,'String','0.35');
366. elseif xy =='i1714f'
367.    set(handles1.edKD,'value',0.4);
368.    set(handles1.showKD,'String','0.4');
369. elseif xy =='i1814f'
```

```
370.    set(handles1.edKD,'value',0.45);
371.    set(handles1.showKD,'String','0.45');
372. elseifxy =='i1914f'
373.    set(handles1.edKD,'value',0.5);
374.    set(handles1.showKD,'String','0.5');
375. elseifxy =='i2014f'
376.    set(handles1.edKD,'value',0.55);
377.    set(handles1.showKD,'String','0.55');
378. elseifxy =='i2114f'
379.    set(handles1.edKD,'value',0.6);
380.    set(handles1.showKD,'String','0.6');
381. elseifxy =='i2214f'
382.    set(handles1.edKD,'value',0.65);
383.    set(handles1.showKD,'String','0.65');
384. elseifxy =='i2314f'
385.    set(handles1.edKD,'value',0.7);
386.    set(handles1.showKD,'String','0.7');
387. elseifxy =='i2414f'
388.    set(handles1.edKD,'value',0.75);
389.    set(handles1.showKD,'String','0.75');
390. elseifxy =='i2514f'
391.    set(handles1.edKD,'value',0.8);
392.    set(handles1.showKD,'String','0.8');
```

```
393. elseifxy ==' i2614f'
394.     set(handles1.edKD,' value',0.85);
395.     set(handles1.showKD,' String',' 0.85');
396. elseifxy ==' i2714f'
397.     set(handles1.edKD,' value',0.9);
398.     set(handles1.showKD,' String',' 0.9');
399. celseifxy ==' i2814f'
400.     set(handles1.edKD,' value',0.95);
401.     set(handles1.showKD,' String',' 0.95');
402. elseifxy ==' i2914f'
403.     set(handles1.edKD,' value',1);
404.     set(handles1.showKD,' String',' 1');
405.
406.
407. %-------------------------------------------------
408. %   BarraKI
409. %-------------------------------------------------
410. elseifxy ==' i0816f'
411.     set(handles1.edKI,' value',0);
412.     set(handles1.showKI,' String',' 0');
413. elseifxy ==' i0916f'
414.     set(handles1.edKI,' value',0.25);
415.     set(handles1.showKI,' String',' 0.25');
```

```
elseif xy ==' i1016f'
    set(handles1.edKI,'value',0.5);
    set(handles1.showKI,'String','0.5');
elseif xy ==' i1116f'
    set(handles1.edKI,'value',0.75);
    set(handles1.showKI,'String','0.75');
elseif xy ==' i1216f'
    set(handles1.edKI,'value',1);
    set(handles1.showKI,'String','1');
elseif xy ==' i1316f'
    set(handles1.edKI,'value',1.25);
    set(handles1.showKI,'String','1.25');
elseif xy ==' i1416f'
    set(handles1.edKI,'value',1.5);
    set(handles1.showKI,'String','1.5');
elseif xy ==' i1516f'
    set(handles1.edKI,'value',1.75);
    set(handles1.showKI,'String','1.75');
elseif xy ==' i1616f'
    set(handles1.edKI,'value',2);
    set(handles1.showKI,'String','2');
elseif xy ==' i1716f'
    set(handles1.edKI,'value',2.25);
```

```
439.     set(handles1.showKI,'String','2.25');
440.  elseif xy =='i1816f'
441.     set(handles1.edKI,'value',2.5);
442.     set(handles1.showKI,'String','2.5');
443.  elseif xy =='i1916f'
444.     set(handles1.edKI,'value',2.75);
445.     set(handles1.showKI,'String','2.75');
446.  elseif xy =='i2016f'
447.     set(handles1.edKI,'value',3);
448.     set(handles1.showKI,'String','3');
449.  elseif xy =='i2116f'
450.     set(handles1.edKI,'value',3.25);
451.     set(handles1.showKI,'String','3.25');
452.  elseif xy =='i2216f'
453.     set(handles1.edKI,'value',3.5);
454.     set(handles1.showKI,'String','3.5');
455.  elseif xy =='i2316f'
456.     set(handles1.edKI,'value',3.75);
457.     set(handles1.showKI,'String','3.75');
458.  elseif xy =='i2416f'
459.     set(handles1.edKI,'value',4);
460.     set(handles1.showKI,'String','4');
461.  elseif xy =='i2516f'
```

```
462.     set(handles1.edKI,'value',4.25);
463.     set(handles1.showKI,'String','4.25');
464. elseif xy =='i2616f'
465.     set(handles1.edKI,'value',4.5);
466.     set(handles1.showKI,'String','4.5');
467. elseif xy =='i2716f'
468.     set(handles1.edKI,'value',4.75);
469.     set(handles1.showKI,'String','4.75');
470. elseif xy =='i2816f'
471.     set(handles1.edKI,'value',5);
472.     set(handles1.showKI,'String','5');
473.
474. %end
475. %-------------------------------------------
476. %   BotonConstruir
477. %-------------------------------------------
478. %Tag = pushbuttonBuild
479. %Max = 1
480. %Min = 0
481. %-------------------------------------------
482. %if xy =='i0213f'
483. %%set(handles1.pushbuttonBuild,'value',1);
```

```
%quarccreatingmatlabguisdemo('pushbuttonBuildCallback'
,hObject,eventdata,guidata(hObject))
%set(handles1.showKI,'String','5');
%end

%-----------------------------------------
%BarradePosicion
%-----------------------------------------
elseifxy=='i0202f'
    set(handles1.sliderFrequency,'value',60);
    set(handles1.showV,'String','60');
elseifxy=='i0203f'
    set(handles1.sliderFrequency,'value',52.5);
    set(handles1.showV,'String','52.5');
elseifxy=='i0204f'
    set(handles1.sliderFrequency,'value',45);
    set(handles1.showV,'String','45');
elseifxy=='i0205f'
    set(handles1.sliderFrequency,'value',37.5);
    set(handles1.showV,'String','37.5');
elseifxy=='i0206f'
    set(handles1.sliderFrequency,'value',30);
    set(handles1.showV,'String','30');
```

```
506. elseif xy ==' i0207f'
507.     set(handles1.sliderFrequency,'value',22.5);
508.     set(handles1.showV,'String','22.5');
509. elseif xy ==' i0208f'
510.     set(handles1.sliderFrequency,'value',15);
511.     set(handles1.showV,'String','15');
512. elseif xy ==' i0209f'
513.     set(handles1.sliderFrequency,'value',7.5);
514.     set(handles1.showV,'String','7.5');
515. elseif xy ==' i0210f'
516.     set(handles1.sliderFrequency,'value',0);
517.     set(handles1.showV,'String','0');
518. elseif xy ==' i0211f'
519.     set(handles1.sliderFrequency,'value',-7.5);
520.     set(handles1.showV,'String','-7.5');
521. elseif xy ==' i0212f'
522.     set(handles1.sliderFrequency,'value',-15);
523.     set(handles1.showV,'String','-15');
524. elseif xy ==' i0213f'
525.     set(handles1.sliderFrequency,'value',-22.5);
526.     set(handles1.showV,'String','-22.5');
527. elseif xy ==' i0214f'
528.     set(handles1.sliderFrequency,'value',-30);
```

```
529.    set(handles1.showV,'String','-30');
530. elseif xy =='i0215f'
531.    set(handles1.sliderFrequency,'value',-37.5);
532.    set(handles1.showV,'String','-37.5');
533. elseif xy =='i0216f'
534.    set(handles1.sliderFrequency,'value',-45);
535.    set(handles1.showV,'String','-45');
536. elseif xy =='i0217f'
537.    set(handles1.sliderFrequency,'value',-52.5);
538.    set(handles1.showV,'String','-52.5');
539. elseif xy =='i0218f'
540.    set(handles1.sliderFrequency,'value',-60);
541.    set(handles1.showV,'String','-60');
542.
543. end
544. %-------------------------------------------
545. fclose(PS);
546. delete(PS);
547. clearPS;
548. INSTRFIND
```

Bibliografía

From, A. (2015). History of elo. *Elo's Touching Moments*.

Golnaraghi, M. F. (2010). *Automatic control systems / Farid Golnaraghi, Benjamin C. Kuo.* Wiley, Hoboken, NJ, 9th ed. edition.

Han, J. (2006). The radical promise of the multi-touch interface. *TED*, 27:41.

Hawkins, W. J. (1983). Bits and bytes. *Popular Science*, 223:78.

Johnson, E. and Establishment, R. R. (1966). *The Touch Display: A Novel Input/output Device for Computers*. RRE technical note: Royal Radar Establishment. Royal Radar Establishment.

Johnson, E. A. (1967). Touch displays: A programmed man-machine interface. *Ergonomics*, 10(2):271–277.

Korhonen, T. O. and Ainamo, A. (2013). Handbook of product and service development in communication and information technology.

Ramírez, L., Jiménez, G., Carreño, J., and e libro, C. (2014). *Sensores y actuadores: aplicaciones con Arduino*. Elibro Catedra. Larousse - Grupo Editorial Patria.

Roebuck, K. (2012). *Mobile Robot: High-impact Emerging Technology*. Emereo Publishing.

Service, I. N. (1993). Pdas competeat cebit'93. *Computer World, The newspaper of Information System Management*, 27:41.