ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Grado

DISEÑO DE ENTRENADOR DE ARDUINO CON APLICACIONES INDUSTRIALES

(Design of an Arduino trainer with industrial applications)

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Raúl Sainz-Maza Serna

Septiembre – 2017

ÍNDICE

1. **MEMORIA**
2. **ANEXO – CÓDIGOS PROGRAMACIÓN**
3. **PLANOS**
4. **PLIEGO DE CONDICIONES**
5. **PRESUPUESTO**

DOCUMENTO Nº1:

MEMORIA

Índice de la Memoria

1. INTRODUCCIÓN 1
1.1. OBJETIVO 1
1.2. MOTIVACIÓN 2
2. ARDUINO 3
2.1. INTRODUCCIÓN A ARDUINO 3
2.2. ENTORNO DE DESARROLLO 3
2.3. MANEJO DE LIBRERÍAS 5
2.4. JUSTIFICACIÓN DE LA ELECCIÓN 7
3. ¿QUÉ ES UN ENTRENADOR? 8
4. COMPONENTES PANELES 9
4.1. PANEL FRONTAL 9
4.1.1. BOTONES. 9
4.1.2. POTENCIÓMETROS. 10
4.1.3. LED 11
4.1.4. JOYSICK 12
4.1.5. CODIFICADOR GIRATORIO 13
4.1.6. PANTALLA LCD 15
4.1.7. PANTALLA TFT 16
4.2. PANEL TRASERO 17
4.2.1. ALIMENTACIÓN 17
4.2.2. CONECTOR PROGRAMACIÓN 18
4.2.3. CONECTORES DB 19
5. DESARROLLO 20
5.1. METODOLOGÍA 20
5.2. ETAPAS 20
6. DISEÑO Y FABRICACIÓN MECÁNICA 22
6.1. FRESADO 23
6.2. PANEL FRONTAL 24
6.3. PANEL TRASERO 27
7. DISEÑO ELECTRÓNICO 29
7.1. BUS I2C 29
7.2. MÓDULO I/O 24 V 30

7.2.1. DISEÑO.....33
7.2.2. FUNCIONAMIENTO.....34
7.2.3. MONTAJE.....37
7.3. MÓDULO LCD.....39
7.3.1. FUNCIONAMIENTO.....42
7.4. PANTALLA TFT.....44
7.4.1. BUS SPI.....45
7.4.2. FUNCIONAMIENTO.....47
7.5. MÓDULO CONTROL MOTORES.....48
7.6. COMUNICACIÓN RS232.....52
7.7. ALIMENTACIÓN.....53
8. MONTAJE FINAL.....55
9. CONCLUSIONES Y LÍNEAS FUTURAS.....57
10. DISPOSICIONES LEGALES Y NORMAS APLICADAS.....58
11. BIBLIOGRAFÍA.....58
11.1. LIBROS.....58
11.2. PÁGINAS WEB.....58
11.3. SOFTWARE.....58
11.4. REFERENCIAS.....59

Índice de Figuras

Figura 2.1: Interfaz Arduino IDE (MacOS) 4
Figura 2.2: Gestor de librerías. 6
Figura 2.3: Menu para Incluir librería IDE 6
Figura 2.4: Placa Arduino Mega 7
Figura 3.1: Entrenador de PLCs 8
Figura 4.1: Botón RS Pro. 9
Figura 4.2: Potenciómetro con mando 10
Figura 4.3: Conexión Potenciómetro. 10
Figura 4.4: Vista frontal LED 11
Figura 4.5: Vista trasera LED 12
Figura 4.6: Vista frontal Joystick. 12
Figura 4.7: Codificador giratorio. 13
Figura 4.8: Representación canales encoder. 14
Figura 4.9: Mando embellecedor 14
Figura 4.10: Display LCD 15
Figura 4.11: Pantalla TFT 16
Figura 4.12: Conector Macho Alimentación 17
Figura 4.13: Interruptor Alimentación 18
Figura 4.14: Adaptador USB 18
Figura 4.15: Cable adaptador USB 19
Figura 4.16: Conector DB9 19
Figura 6.1: Caja vista superior. 22
Figura 6.2: Caja vista trasera. 22
Figura 6.3: Ejemplo fresadora. [19] 23
Figura 6.4: Palpador para fresadora. 23
Figura 6.5: Panel frontal. 24
Figura 6.6: Esquema panel frontal. 25
Figura 6.7: Tornillo M2 x16 mm 26
Figura 6.8: Vista superior panel frontal terminado. 26
Figura 6.9: Alzado panel frontal terminado. 27
Figura 6.10: Panel trasero. 27
Figura 6.11: Esquema panel trasero. 28
Figura 6.12: Panel trasero montado. 28
Figura 7.1: Esquema conexión Bus I2C. 29
Figura 7.2: Conexión optoacoplador 30
Figura 7.3: Integrado MCP23017 31
Figura 7.4: Pinout MCP23017 [24] 31
Figura 7.5: Registro dirección de esclavo I2C. 32
Figura 7.6: Esquema conexión optoacoplador 5V-24V 33
Figura 7.7: Esquema conexión optoacoplador 24V-5V 34
Figura 7.8: Registros control MCP23017 35
Figura 7.9 Placa de prototipos. 37
Figura 7.10: Modulo 24V montado y fijado. 38
Figura 7.11: Módulo LCD montado. 39
Figura 7.12: Módulo LCD desconectado de pantalla. 40
Figura 7.13: Placa control LCD 41
Figura 7.14: Vista posterior del módulo LCD 42

Figura 7.15: Vista frontal pantalla TFT ..... **¡Error! Marcador no definido.**
Figura 7.16: Vista frontal y conexiones pantalla TFT 44
Figura 7.17: Esquema conexiones Bus SPI 45
Figura 7.18: Secuencia Bits protocolo SPI 46
Figura 7.19: Vista trasera pantalla TFT 47
Figura 7.20: Esquema funcionamiento motor DC con escobillas 48
Figura 7.21: Servomotor 49
Figura 7.22: Diagrama de bloque del servomotor 49
Figura 7.23: Motores paso a paso. 50
Figura 7.24: Módulo control motores. 51
Figura 7.25: Adaptador MAX3232 52
Figura 7.26: Convertidor dc/dc aislado 24V-05V 53
Figura 7.27: Convertidor dc/dc aislado 24V-12V 53
Figura 8.1: Elementos ensamblados en panel. 55
Figura 8.2: Interior en etapa media de montaje. 55
Figura 8.3: Cableado final. 56

MEMORIA

1. INTRODUCCIÓN

El presente trabajo se ha realizado con el objeto de poner en práctica algunos de los conocimientos adquiridos en el Grado en Ingeniería Electrónica Industrial y Automática.

La aparición de la plataforma Arduino en el mercado de los microcontroladores, ha despertado un gran interés por todos los proyectos que se pueden realizar con estos dispositivos. Aunque la mayor parte de estos proyectos se realizan en un ámbito no industrial, existen infinidad de aplicaciones para mejora de procesos o sustitución de elementos obsoletos en los que es muy útil contar con una herramienta de estas características.

La idea de crear un entrenador de Arduino enfocado a la industria, surgió durante mi periodo de prácticas en Robert Bosch Treto. En la empresa existían entrenadores de PLCs, mediante los cuales se podían comprobar algunas modificaciones antes de implementarlas en los puestos de trabajo reales, ahorrando tiempos de parada muy costosos.

En los seis meses que estuve como becario, trabajé con un ingeniero electrónico en el departamento de mantenimiento. Dicho ingeniero, hacía uso de placas de Arduino para muchos de sus proyectos, por lo que consideré conveniente la fabricación de un entrenador donde poder testear las aplicaciones pensadas rápidamente y sin la necesidad de realizar ningún prototipo nuevo. Además, existía un muy buen grupo de técnicos eléctricos e ingenieros especializados en otros ámbitos, los cuales podrían aprender a programar utilizando el entrenador descrito.

1.1. OBJETIVO

El principal objetivo de este proyecto es diseñar e implementar un entrenador de Arduino con herramientas útiles para la industria. Así pues, el entrenador constará de una caja cerrada con una serie de módulos que he considerado adecuados para aplicaciones industriales.

La caja se compondrá de dos paneles, uno frontal inclinado, donde se alojarán los botones, leds, pantallas y demás elementos que funcionarán como interfaz con el usuario, y otro trasero, con conectores industriales, donde se encuentran las comunicaciones con el exterior.

Se pretende acercar de esta manera las posibilidades que brinda Arduino al entorno industrial, proporcionar un prototipo donde poder testear las aplicaciones creadas así como ofrecer un entrenador donde poder aprender.

Además, se deja espacio libre para ampliar el prototipo con más módulos como por ejemplo añadir conectividad wifi para la futura industria 4.0.

1.2. MOTIVACIÓN

Uno de los aspectos que más me motivan para realizar este trabajo es el poner en práctica los conocimientos adquiridos para desarrollar un proyecto completo desde el diseño puramente estético hasta la fabricación, pasando por el diseño electrónico. El hecho de tener que elegir y comprar los componentes en proveedores oficiales de electrónica como RS también ha supuesto un reto extra al que no estaba acostumbrado.

Por otra parte, descubrir el potencial que tiene Arduino y aprender de otros diseños existentes en la web es interesante dentro del proceso de formación continua, siempre necesario para cualquier ingeniero. Además es muy gratificante la rapidez con la que se obtienen resultados con este sistema.

Una de las cuestiones que más me motivan personalmente es el hecho de que las ideas aportadas en el diseño de cada módulo, pueden ser utilizadas en futuros prototipos. Así como el poder acercar esta plataforma a los compañeros que conocí que mostraron gran interés por aprender.

2. ARDUINO

2.1. INTRODUCCIÓN A ARDUINO

Arduino es una plataforma que aúna hardware y software libre, fácil de usar y flexible. Además alrededor de Arduino gira una comunidad de desarrolladores muy extensa en continuo movimiento, que desarrollan librerías y demás recursos que todo el mundo puede utilizar.

El objetivo de Arduino es simplificar el uso de electrónica y programación de microcontroladores para hacerla accesible a un mayor número de personas. Todos los productos, tanto de hardware como de software, son liberados con licencia de código abierto, lo que permite su uso, copia y modificación gratuita.

El hardware en sus placas de desarrollo se compone de una PCB con un microcontrolador, pines para acceder fácilmente a los puertos de entrada y salida (tanto digitales como analógicos), los cuales se utilizan para conectarse con el exterior, e incluso se pueden conectar a placas de expansión (shields), que amplían las características de funcionamiento de la placa Arduino. Además, posee un puerto de conexión USB por donde se puede alimentar la placa y comunicarse con un ordenador para programar el microcontrolador.

Debido a su gran acogida en el mercado, arduino ha crecido exponencialmente en los últimos años y actualmente ofrece una amplia gama de productos con diferentes características, desde placas de desarrollo de diferentes tamaños y funcionalidades hasta Shields de expansión pasando por impresoras 3D. Todos los productos ofertados se pueden encontrar en la página oficial de Arduino [1].

2.2. ENTORNO DE DESARROLLO

Cuando se habla de entorno de desarrollo, se refiere a una aplicación o programa que facilita al desarrollador o programador el desarrollo de un software determinado. En el caso de Arduino por tanto, el entorno de desarrollo facilita la edición, compilación y carga a la placa determinada del código o programa que se suele denominar “sketch”.

Como ya se ha comentado anteriormente, una de las grandes ventajas de la plataforma Arduino es que trabaja con licencias libres, por tanto, ofrece su propio entorno de desarrollo (IDE) de forma totalmente gratuita y facilita su descarga a través de la propia página web oficial.

El software está disponible para varios sistemas operativos entre los que se encuentran Windows, MacOS y Linux. Recientemente se ha desarrollado además una herramienta de edición web, con la

que no es necesario descargar ningún programa, ya que toda la edición se realiza online y es posible guardar los sketches en la "nube". [2]

El código del programa se escribe en el editor de texto. A la hora de escribir los sketches, estos siguen un esquema con tres partes claramente diferenciadas:

- En la parte superior del editor y fuera de cualquier función, se llama a las librerías y se definen las variables y constantes globales.
- void setup(): Es una función que se utiliza para configurar la placa, así como los posibles módulos. Esta parte del programa solo se ejecuta una vez cuando el microcontrolador arranca.
- void loop(): Es la función principal que ejecuta el código presente en ella de forma cíclica a modo de bucle. En ella se estructura la lógica del programa, y se llama a las demás funciones que deben de estar definidas fuera de esta.

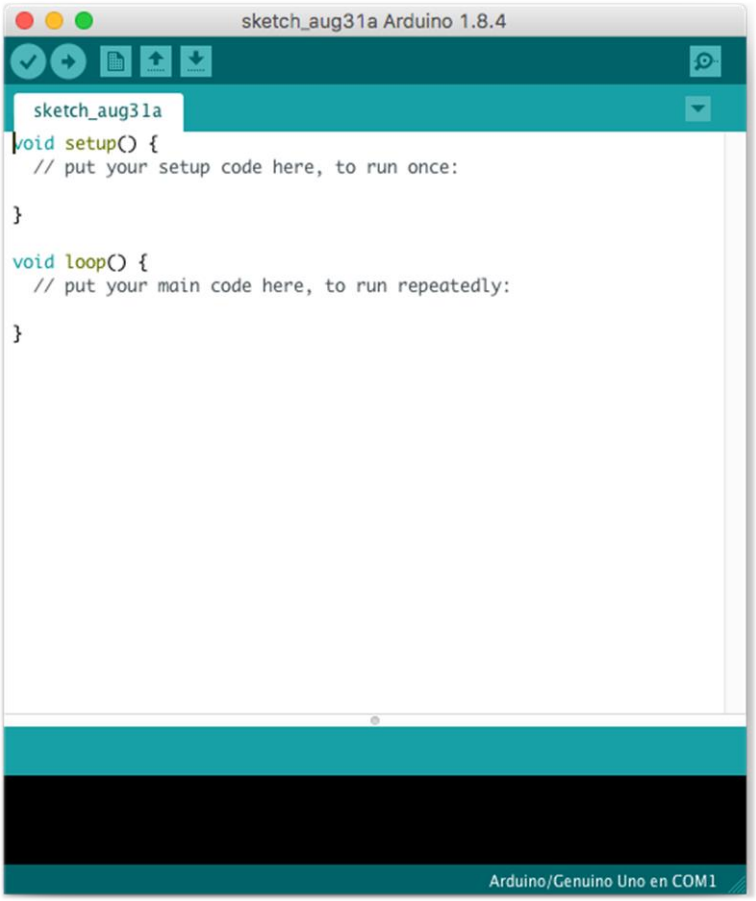


Figura 2.1: Interfaz Arduino IDE (MacOS)

Además del editor, la interfaz se compone por otras partes como la barra de herramientas de acceso rápido, o el área de mensajes, donde aparece información relevante para el usuario como mensajes de error durante la carga del programa a la placa o detalles sobre el proceso de compilación.

Dentro de la barra de herramientas de acceso rápido, podemos encontrar seis botones que de izquierda a derecha desempeñan las siguientes funciones:

- Verificar: compila el código y muestra en el área de mensajes una confirmación si está todo correcto o un mensaje de error si por el contrario no se ha podido compilar correctamente.
- Subir: compila y si no hay errores, carga el programa a la placa.
- Nuevo: crea un nuevo sketch vacío.
- Abrir: despliega un submenú en el que se pueden abrir sketches de ejemplos de utilización de diferentes tecnologías, o un sketch previamente guardado.
- Salvar: guarda el sketch actual.
- Monitor Serie: abre el monitor serie, herramienta que permite la comunicación con la placa a través del puerto USB. Es posible tanto leer datos como enviar datos a la placa.

2.3. MANEJO DE LIBRERÍAS

Uno de los puntos más fuertes de Arduino es la cantidad de información y librerías en constante crecimiento que existen. Las librerías facilitan mucho las tareas de programación simplificando los códigos empleados. Además, la inmensa mayoría de las librerías son de código abierto para que cualquiera pueda tanto usarlas como mejorarlas.

Existen dos tipos generales de librerías:

- **Librerías estándar:** cuando se descarga el IDE de Arduino, vienen incluidas una serie de librerías denominadas estándar. Con estas librerías se pueden ejecutar códigos de ejemplo que también se incluyen en la descarga. Estas librerías incluyen funciones para el manejo de comunicaciones básicas o algún tipo de hardware muy usado como son los servo motores o las pantallas LCD.
- **Librerías instaladas por el usuario:** Existen muchas más librerías con funciones útiles o drivers para dispositivos pertenecientes a todos los tipos de hardware. Estas librerías estan disponibles para su descarga en sitios como Arduino Playground, GitHub o Google Code. [3]

En este apartado se introduce de forma breve la forma de descargar y utilizar librerías, ya que será necesario para el manejo de algunos de los módulos utilizar librerías instaladas por el usuario.

Existen dos maneras fáciles implementadas en el IDE, de incluir librerías no estándar.

- **Gestor de librerías**: Mediante esta opción se abre una ventana, en la que aparece una lista de librerías ya instaladas o que están listas para instalar. Se puede filtrar la búsqueda, o buscar por nombre. Para instalar una nueva librería basta con seleccionar la librería deseada, elegir la versión (si hay más de una disponible) y pinchar en instalar. El gestor de librerías está disponible desde la versión de IDE 1.6.2. [4]

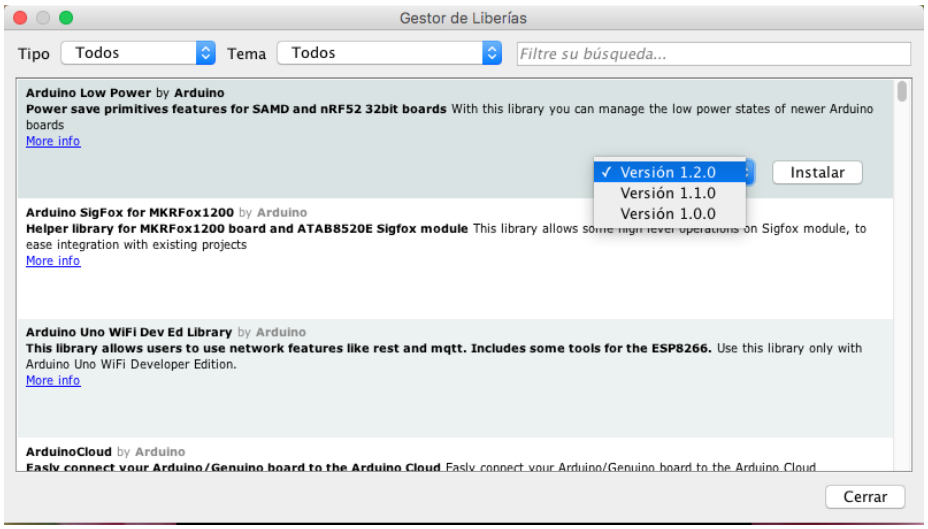


Figura 2.2: Gestor de librerías.

- **Añadir librería .ZIP:** Mediante este método, se selecciona una librería previamente descargada de algún repositorio en formato .ZIP y se añade directamente.

Para acceder a estas opciones, basta con pinchar en el menú Programa > Incluir Librería

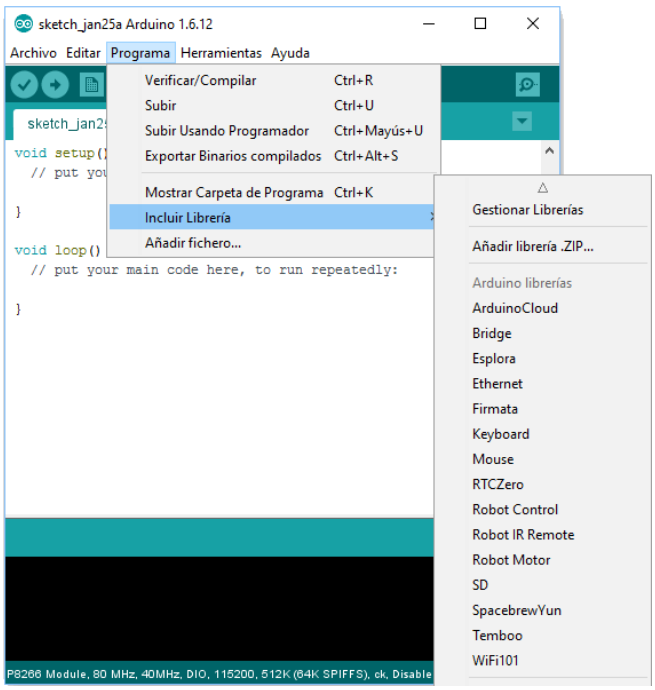


Figura 2.3: Menu para Incluir librería IDE

En el menú aparecen tanto las opciones antes descritas, como todas las librerías instaladas. Si se pincha en una de estas librerías, se incluyen automáticamente en el sketch, para poder hacer uso de sus funciones en el mismo.

2.4. JUSTIFICACIÓN DE LA ELECCIÓN

Dentro de todos los productos ofertados [1], la placa elegida para desarrollar este proyecto es la Arduino Mega 2560 [5]. Antes de argumentar el porqué de la elección, resulta conveniente enumerar algunas de las características fundamentales de esta placa.

- Microcontrolador: ATmega2560
- Tensión de operación: 5V
- Tensión de entrada (recomendada): 7-12V
- Tensión de entrada (límite): 6-20V
- Pines de E/S digitales: 54 (de los cuales 15 soportan salidas PWM)
- Entradas analógicas: 16
- Corriente soportada en los pines de E/S: 20mA
- Memoria flash: 256KB de los cuales 8KB son usados por el bootloader
- Memoria SRAM: 8KB
- Memoria EEPROM: 4KB
- Frecuencia del reloj: 16MHz
- Largo: 101.52mm
- Ancho: 53.3mm
- Peso: 37g

Figura 2.4: Placa Arduino Mega

Se estudió la posibilidad de utilizar un Arduino UNO, pero aparecieron varios inconvenientes que hicieron que se descartara esta opción.

Por un lado, la escasez de pines de entrada y salida en la placa UNO, limita mucho las opciones y periféricos que se pueden añadir al entrenador.

Por otro lado, el Arduino UNO solo posee un puerto de comunicación serie. Existe una librería llamada “SoftwareSerial” que permite usar cualquier par de pines de E/S para crear un puerto serie virtual, pero no siempre funciona correctamente y las velocidades de trasmisión de datos son menores. [6] Estos factores unidos al hecho de que la diferencia de precio no es muy grande y tampoco la diferencia en tamaño, hicieron más recomendable la opción del Arduino MEGA.

3. ¿QUÉ ES UN ENTRENADOR?

Se entiende como entrenador a un sistema mediante el cual se aprende o se practica a utilizar una cierta tecnología. Así pues, los más comunes son los entrenadores de PLCs, como el que se puede observar en la Figura 3.1 mediante los cuales se facilita el aprendizaje en la programación de PLCs, permitiendo simular y testear ciertas aplicaciones mediante el uso de periféricos como son indicadores Leds conectados a las salidas o interruptores o pulsadores en las entradas.

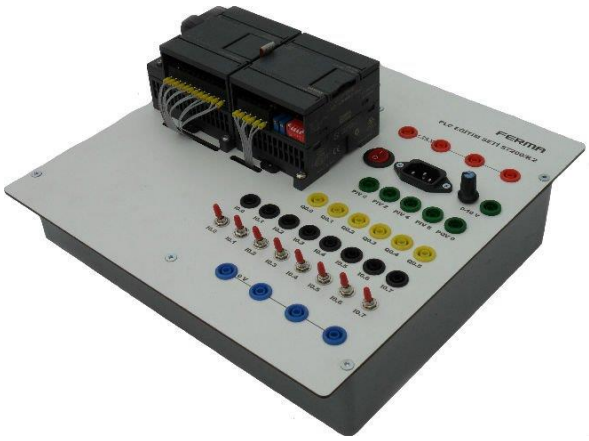


Figura 3.1: Entrenador de PLCs

Los entrenadores por tanto constan de dos elementos básicos. Por un lado el dispositivo o tecnología que se pretende estudiar, que en el caso del entrenador descrito en este apartado, sería un PLC de la marca correspondiente al software que se desea aprender. Y por otro lado los periféricos que resulten necesarios para el testeo de las aplicaciones, que en el supuesto de un entrenador de PLCs suelen ser varios interruptores simulando entradas y varios indicadores simulando salidas.

Para el caso que nos atañe en este proyecto, contaremos por tanto con una placa de desarrollo Arduino y los periféricos que se consideran oportunos aprender a controlar, los cuales serán descritos en capítulos posteriores.

4. COMPONENTES PANELES

En este apartado se describirán desde un punto de vista físico y de funcionamiento, los componentes que se han utilizado como interfaz con el mundo externo, y que por tanto se instalan en los paneles. Se incluyen también las cotas físicas de cada elemento, que luego serán utilizadas en el diseño mecánico del panel.

Los componentes utilizados internamente por cada módulo, se describen en apartados posteriores, por lo que no resulta necesario incluirlos en este capítulo.

Uno de los atributos comunes que deben tener todos los componentes elegidos es que sea apto para montaje en panel. Esta característica restringe bastante las opciones, ya que algunos elementos como las pantallas son difíciles de encontrar con este tipo de montaje.

4.1. PANEL FRONTAL

En el panel frontal se encuentran todos los dispositivos que funcionan como interfaz con el usuario. Tenemos por tanto elementos utilizados para enviar información al sistema como botones o potenciómetros, y elementos que el sistema utiliza para enviar información al usuario como son las pantallas. A continuación se describe cada componente utilizado.

4.1.1. BOTONES.

Los botones elegidos son los que se muestran en la Figura3.1, son interruptores pulsadores normalmente abiertos fabricados por RS Pro.

Figura 4.1: Botón RS Pro.

Estos botones fueron seleccionados por varias razones. La primera de ellas es que están preparados para montaje en panel, se suministran con una tuerca que facilita el montaje y el acabado es bueno, la segunda razón de importancia es la comodidad de uso, debido a su tamaño óptimo y su

fuerza de funcionamiento que hace muy cómodo su uso. Por último, el hecho de que sus terminales faciliten la soldadura también fue un factor determinante.

Las especificaciones obtenidas del datasheet del producto son las siguientes:

- Fuerza de funcionamiento: 2 N - 5 N
- Rebote de contacto: 10 ms
- Vida mecánica: 1.000.000 ciclos
- Soldadura: 350 °C máximo durante 5 segundos
- Rosca: M12

De este elemento nos interesa especialmente la métrica de la rosca, dato que utilizaremos posteriormente en el diseño mecánico del panel frontal. [7]

4.1.2. POTENCIÓMETROS.

Los potenciómetros son elementos que varían su resistencia eléctrica en función de una variable mecánica. En nuestro caso los potenciómetros serán rotativos, esto significa que la resistencia varía en función de un giro mecánico.

Figura 4.2: Potenciómetro con mando

Como se puede observar en la Figura 3.3 conexión eléctrica es simple. Poseen tres terminales, dos de longitud similar, que se conectarán entre alimentación y tierra y un tercero más largo, que corresponde al valor variable.

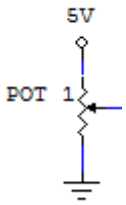


Figura 4.3: Conexión Potenciómetro.

Los potenciómetros utilizados están fabricados por Vishay y el aspecto exterior se puede observar en la Figura 3.2. Estos potenciómetros fueron elegidos por varios factores, el más determinante es el diseño exterior y la facilidad de uso y de montaje en panel.

Las características más relevantes se muestran a continuación:

- Resistencia: 22kΩ
- Potencia nominal: 1W
- Abertura en panel: Ø10mm

Al igual que sucede con los botones, resulta especialmente interesante el dato de abertura en panel, que luego será utilizado en el diseño mecánico. [8]

4.1.3. LED

Los ledes son un tipo especial de diodo con la particularidad de que cuando es polarizado directamente, emite luz. Como todo diodo, constan de un ánodo, parte positiva, y un cátodo, parte negativa.

Figura 4.4: Vista frontal LED

Este tipo de diodos trabaja con un nivel de tensión nominal aproximado de 2V y con una corriente nominal de 20 mA, por lo que resulta necesario añadir una resistencia en serie para limitar la tensión e imponer la corriente necesaria. Para calcular la resistencia sabemos que la tensión de trabajo de nuestro Arduino es de 5V, por lo que necesitaremos que caigan 3V en la resistencia y que la atraviesen 20mA. Aplicando la ley de ohm obtenemos que necesitamos una resistencia de 150Ω sin embargo experimentalmente se observa que con una resistencia de 330Ω la intensidad de luz es suficiente y el consumo se reduce a la mitad.

Los LED escogidos, fabricados por RS Pro, presentan 3 contactos, apreciables ligeramente en la Figura 3.5. Esto se debe a que internamente se componen de dos ledes, uno que emite luz roja y otro que emite luz verde. También es posible activar ambos ledes, obteniendo una luz amarilla. La

configuración de conexiones es cátodo común, lo que significa que los dos leds comparten el mismo terminal negativo. Los dos contactos iguales corresponden por tanto a cada uno de los ánodos y el tercer contacto al cátodo.

Figura 4.5: Vista trasera LED

Las especificaciones más relevantes de estos leds son las siguientes:

- Tensión Nominal: 2Vdc
- Color de la luz: Rojo/Verde Amarillo
- Tamaño del orificio de fijación: 8mm
- Corriente Nominal: 20mA

Como en los anteriores elementos, cabe destacar el tamaño del orificio de fijación, dato que será necesario posteriormente. [9]

4.1.4. JOYSICK

Una palanca de mando o joystick, es un dispositivo de control de dos o tres ejes que se ha popularizado con la industria de los videojuegos. Una de las razones por la que se decidió incluir un joystick en el entrenador es la facilidad de uso a la hora de navegar por menús.

Figura 4.6: Vista frontal Joystick.

El joystick elegido se compone de una palanca con un dispositivo de retorno de manera que se mantiene en la posición central. El rango de funcionamiento es de dos ejes, lo que significa que tenemos 4 posiciones además de la posición neutral.

El funcionamiento es simple, cuando se desplaza la palanca en alguna de las direcciones posibles, se activa un micro interruptor. Así pues, tendremos 4 micro interruptores que deberemos leer como entradas digitales.

El joystick seleccionado está fabricado por RS Pro y las características más relevantes son las siguientes:

- Número de Ejes: 2
- Altura del mango: 37mm
- Tamaño del orificio de fijación: 22mm

Se eligió este joystick en concreto por las dimensiones óptimas, ya que el resto de dispositivos similares, eran demasiado grandes. Como en anteriores apartados, cabe destacar el dato del tamaño del orificio de fijación. [10]

4.1.5. CODIFICADOR GIRATORIO

Un codificador giratorio es un dispositivo generador de pulsos, muy utilizado en sistemas electrónicos para realizar tareas tales como desplazarse por menús o subir y bajar el volumen en una radio por ejemplo.

Figura 4.7: Codificador giratorio.

El funcionamiento de estos dispositivos se basa en la generación de dos señales desfasadas 90º eléctricos, mediante la lectura de estas señales se determina si se está girando hacia la derecha o hacia la izquierda y con qué rapidez.

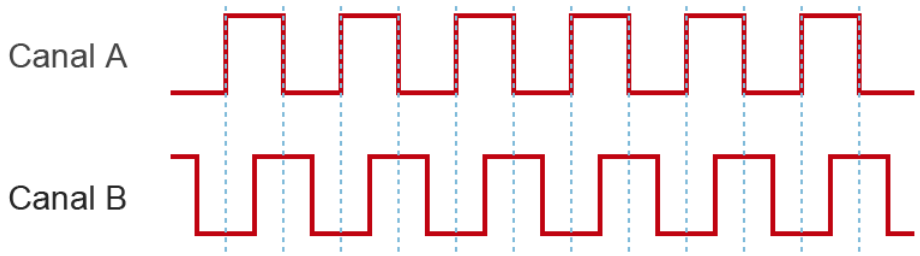


Figura 4.8: Representación canales encoder.

Los codificadores seleccionados están fabricados por Bourns y las características más relevantes son las siguientes:

- Pulsos por revolución: 24
- Estilo del eje: Plano
- Tamaño orificio fijación: 7mm

Cabe destacar de estos dispositivos en concreto que además de los dos canales también poseen un interruptor que se acciona al presionar el mando hacia abajo. [11]

Figura 4.9: Mando embellecedor.

A modo de embellecedor se utilizó un mando de potenciómetro como el de la figura 4.10, que proporciona mejor agarre y un diseño más elegante. [12]

Como observación decir que se podría utilizar el mismo programa utilizado para leer este tipo de codificador giratorio, para leer encoders relativos, dispositivos muy utilizados en la industria para determinar posiciones de ejes mecánicos. [13]

4.1.6. PANTALLA LCD

Con el fin de tener la posibilidad de mostrar información al usuario se incluye una pantalla alfanumérica LCD monocromo como la que se muestra en la Figura 4.11

Este tipo de pantalla es muy utilizada en dispositivos electrónicos, especialmente en aquellos equipos que se alimenten con baterías, ya que requieren una cantidad muy pequeña de energía eléctrica.

En la página distribuidora de electrónica RS, en la cual han sido pedidos todos los elementos que conforman este proyecto, existe una gran variedad de modelos con diferentes características.

Figura 4.10: Display LCD

En concreto el modelo elegido está fabricado por Displaytech y las características principales son las siguientes:

- Número de caracteres: 4 filas x 20 columnas.
- Color: Blanco sobre azul.
- Tipo de display: Transflectivo STN [14]
- Tamaño del display: 76 x 25.2 mm
- Tamaño matriz de carácter: 5 x 8 mm
- Interfaz : paralela
- Intervalo de temperaturas de funcionamiento: de -10ºC a 60ºC

Las dos principales razones por las que se eligió este modelo, son el tamaño de la pantalla y el color y contraste que posee la misma.

Por un lado, el tamaño de la pantalla, está relacionado con el número de caracteres que se pueden mostrar. Existen diferentes configuraciones, aunque las más comunes son las de 2 filas x 16 columnas

y las de 4 x 20. En este caso, teniendo en cuenta que el espacio disponible en el panel donde se montan los elementos, es suficiente, se optó por la segunda configuración, ya que aporta más opciones y mejores visualizaciones.

Por otro lado, RS dispone de varios tipos de pantallas LCD según el color tanto de fondo de pantalla como de caracteres. Se barajó la posibilidad de elegir un modelo RGB, mediante el cual se puede cambiar de color tanto las letras como el fondo de pantalla, pero la falta de disponibilidad de este modelo en la configuración de tamaño deseada, hizo decantarse por la opción de letras blancas sobre fondo azul, que es una de las combinaciones que mejor resultan a la vista y mejor contraste tiene.

Las dimensiones de la pantalla, así como del a PCB de control acoplada a ella, son un dato también relevante para la posterior fijación en el panel. Esta y más características se pueden encontrar en el datasheet del producto. [15]

4.1.7. PANTALLA TFT

En este apartado a diferencia de los anteriores, se describirá con menos detalles el elemento, ya que existe un apartado en el **capítulo 7: Diseño electrónico** donde se explica más detalladamente las características y el funcionamiento de este display.

Figura 4.11: Pantalla TFT

Esta segunda pantalla se incluye fundamentalmente con dos finalidades. Por un lado, desde el punto de vista funcional, tener la posibilidad de mostrar al usuario figuras dibujos o imágenes, como pueden ser gráficas. Y por otro lado desde el punto de vista pedagógico, aprender a utilizar el BUS SPI, del cual hablaremos más adelante.

Las características más relevantes de esta pantalla son las siguientes:

- Tamaño pantalla: 1,77 pulgadas
- Resolución de 160 x 128 píxeles
- Colores de la pantalla: 262.000
- Área activa de 28 x 35 mm

La pantalla junto con la electrónica asociada que hace posible la comunicación por SPI, constituye un módulo diseñado por Arduino, y por tanto de hardware libre. Es por eso que se puede encontrar mucha información acerca de esta pantalla y sus posibilidades. [16]

4.2. PANEL TRASERO

En el panel trasero se encuentran los elementos que hacen de interfaz con el mundo externo, desde la alimentación hasta los diferentes puertos de conexiones o de programación.

4.2.1. ALIMENTACIÓN

La alimentación del entrenador se realiza mediante un conector como el de la figura 4.13 que se utiliza comúnmente en equipos de automatización y detección. La idea es que no solo se pueda alimentar con una fuente, sino que también sea posible ir al taller donde abundan este tipo de conectores y alimentar el sistema.

Figura 4.12: Conector Macho Alimentación

A pesar de que tiene 5 conexiones, solo se utilizarán las dos dedicadas a la alimentación.

De este conector cabe destacar que está diseñado para incluir una tuerca de abrazadera PG9 en su parte posterior. Teniendo en cuenta este último dato, se obtiene que el tamaño del orificio de fijación en panel debe de ser de 16 mm. [17]

Figura 4.13: Interruptor Alimentación

Se incluye también un interruptor que corta la corriente de alimentación de todo el sistema. El único dato que resulta relevante de este componente son sus dimensiones que serán utilizadas en el diseño del panel trasero. Del datasheet se obtiene que se necesita un tamaño del agujero de 30x11mm

4.2.2. CONECTOR PROGRAMACIÓN

Como se ha comentado en el **Capítulo 2**, la programación de Arduino se realiza mediante un puerto USB que, en el caso del Arduino mega, es de tipo B. Para tener accesible este puerto desde fuera del sistema, se utiliza el adaptador que se muestra en la Figura 4.15.

Figura 4.14: Adaptador USB

El adaptador elegido, está fabricado por RS Pro y se eligió por la facilidad de montaje en panel además de por el diseño elegante. Este elemento se compone de una entrada USB de tipo B, y una salida USB tipo A, por lo que para llevar este puerto hasta la situación física del Arduino en el sistema, se utiliza un cable como el de la figura, que conecta la salida USB tipo A del adaptador, a la entrada tipo B del Arduino Mega.

Figura 4.15: Cable adaptador USB

4.2.3. CONECTORES DB

Se decidió que la mayor parte de las conexiones con el mundo exterior se realizaran mediante estos conectores, ya que son ampliamente utilizados en la industria y es relativamente fácil fabricar adaptadores caseros para llevar a cabo las conexiones pertinentes en las situaciones dadas.

Figura 4.16: Conector DB9

Existen conectores de varios tamaños y cantidad de pines. Los conectores elegidos tienen terminación en soldadura, que facilita la acción de soldar los cables de conexión.

Se utilizan conectores hembra debido a que son más duraderos y sufren menos daños.

5. DESARROLLO

5.1. METODOLOGÍA

Primeramente se propuso la fabricación de un sistema de entrenamiento para proyectos basados en Arduino con el fin de probar posibles programas antes de implantarlos en un prototipo final, además de poder ser utilizado para aprender el lenguaje y las posibilidades que oferta arduino.

Una vez determinado el fin, fue necesario determinar los elementos y funciones a incorporar en dicho sistema. Desde el punto de vista del aprendizaje, se propuso que hubiese diferentes niveles de dificultad, desde encender un led hasta controlar una pantalla o un sistema externo.

Posteriormente se realiza el diseño eléctrico y se piden los componentes necesarios.

Por último se diseña la parte mecánica y se ensambla el prototipo, comprobando la funcionalidad del mismo.

5.2. ETAPAS

El proyecto descrito en este documento se ha desarrollado siguiendo cronológicamente las etapas que se numeran a continuación:

1. Investigación, comparación y selección de los posibles elementos a incorporar en el entrenador (pantallas, botones, leds…).
2. Investigación de la oferta Arduino y selección de la placa que cumpla con las necesidades del proyecto.
3. Investigación sobre el funcionamiento de los elementos de control y periféricos seleccionados.
4. Creación de los primeros esquemas eléctricos.
5. Documentación sobre las medidas físicas de los elementos a incorporar, así como de su modo de fijación.
6. Creación de primeros bocetos mecánicos a escala y elección del lugar físico que ocupará cada periférico.
7. Solicitud de pedido de componentes.
8. Revisión, comprobación y testeo de componentes recibidos.
9. Creación y comprobación de planos finales, tanto mecánicos como eléctricos.
10. Mecanizado de paneles según planos mecánicos.

11. Montaje y fijación de componentes en paneles.
12. Fabricación de las placas soldadas utilizadas en los diferentes módulos.
13. Testeo individual del funcionamiento de cada módulo.
14. Conexionado interno mediante soldadura de componentes y periféricos según planos eléctricos.
15. Ensamblaje final del sistema.
16. Testeo del funcionamiento de cada uno de los componentes utilizados tras el ensamblaje final.
17. Redacción de este documento.

6. DISEÑO Y FABRICACIÓN MECÁNICA

En este capítulo se tratarán todos los factores relacionados con el exterior, desde el primer diseño con “AutoCad” hasta los detalles estéticos.

Uno de los objetivos de diseño del entrenador es que sea estéticamente simple, sin demasiados componentes externos, pero a la vez debe de ser funcional.

Con el fin de ocultar todo el cableado y parte electrónica, se decide contar con una caja como la que se muestra en la Figura 6.1. El principal criterio de elección de dicha caja, es el tamaño, ya que resulta conveniente para incluir los módulos necesarios y deja espacio para futuras ampliaciones. Sin embargo, también fue un factor determinante el hecho de que el panel frontal no presente tornillería externa, en el apartado 6.2 se detallará la forma de anclaje del panel. [18]

Figura 6.1: Caja vista superior.

Para la fijación de los elementos internos, se utiliza una placa de metacrilato cortada con las dimensiones apropiadas, que se atornilla a la base de la caja. De esta manera, se puede fijar a cualquier zona de la base los componentes necesarios mediante tornillos u otros elementos de fijación.

Figura 6.2: Caja vista trasera.

Antes de pasar a describir el diseño de cada panel, resulta conveniente introducir los principios básicos del fresado, ya que las aperturas en los paneles serán realizadas mediante una fresa.

6.1. FRESADO

El fresado es un proceso mediante el cual se elimina parte del material que se va a mecanizar con una herramienta rotativa de varios filos, que ejecuta movimientos en varios ejes. La pieza a mecanizar permanece fija a una bancada móvil.

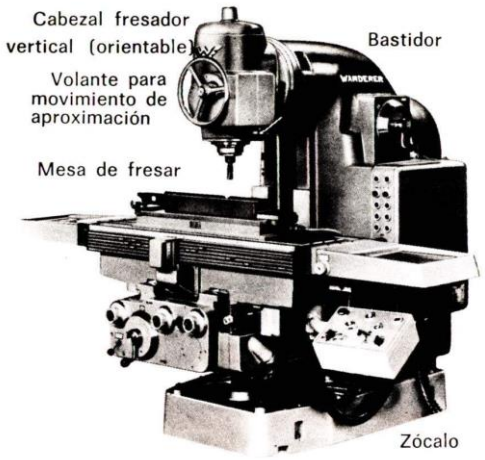


Figura 6.3: Ejemplo fresadora. [19]

La máquina herramienta que se utilizó en este proyecto, fue una fresadora de tres ejes. Los desplazamientos en Z y en X se realizan mediante mandos que mueven la bancada, y el desplazamiento en el eje Y se ejecuta con un mando que mueve la herramienta. La posición se controla mediante una pantalla que muestra en todo momento las coordenadas con una precisión de milésimas de milímetro.

Así pues, el primer paso que se debe dar es el de obtener la posición de origen. Para ello el monitor de la fresa cuenta con unos botones mediante los cuales se resetea cada una de las coordenadas. Con ayuda de un palpador como el de la Figura 6.4, es posible fijar la posición en X Y deseada.

Figura 6.4: Palpador para fresadora.

Antes del proceso de fresado, se debe forrar la chapa de aluminio para protegerla de posibles daños como rayones o pequeños golpes, para dicho propósito se utilizó cinta de carrocero.

Resulta conveniente utilizar avellanadores para mejorar el acabado de los agujeros y limar los bordes de los cajetines donde van las pantallas o los conectores de esta forma se evitarán las rebabas.

6.2. PANEL FRONTAL

El panel frontal se utiliza para cerrar la caja descrita anteriormente, por la parte delantera, así como para fijar todos los dispositivos que se utilizan como interfaz con el usuario. A pesar de que se vendan por separado, tanto el panel frontal como el trasero, están diseñados para montarse con la caja utilizada en este proyecto.

Físicamente, es un rectángulo de aluminio de 257.35 mm de largo por 167.8 mm de ancho con un grosor de 2 mm. [20]

Figura 6.5: Panel frontal.

La particularidad de este panel es que no se necesita añadir tornillería externa para su fijación a la caja, por lo que resulta muy atractivo visualmente. Para la correcta fijación el panel incorpora en su parte posterior seis salientes roscados con métrica M3 como se puede observar en la Figura 6.5. Estos salientes se acoplan en la caja y se fijan mediante tuercas.

Para el diseño, se realizó primeramente un esquema con las dimensiones reales a escala de los elementos a introducir, de esta manera, se puede elegir el lugar óptimo de fijación asegurando que las partes externas de los elementos no se toquen.

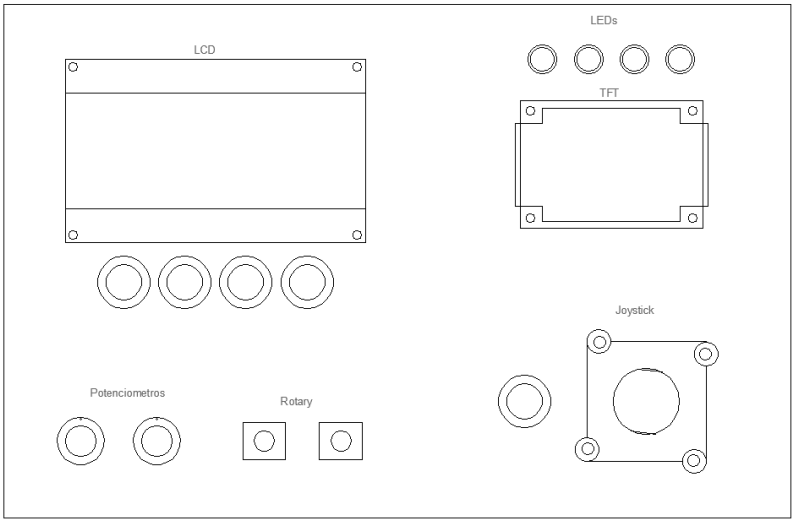


Figura 6.6: Esquema panel frontal.

El esquema se puede observar en la Figura 6.6. Los elementos indicadores de información, como son las pantallas o los LEDs, se sitúan en la parte superior, mientras que los dispositivos manipulables como son botones, potenciómetros o joystick, ocupan los lugares inferiores. Esta configuración resultó la más óptima, entre otras ventajas, permite utilizar el entrenador sin tapar con las manos los elementos de visualización y la manipulación del joystick resulta cómoda debido a la altura y la posición de agarre correcta.

Una vez seleccionado el lugar donde se alojarán los elementos, se realiza el plano pertinente que define las aberturas de sujeción de los mismos en el panel. Para la realización de dicho plano hay que tener en cuenta el tamaño exacto de los elementos, estos datos se pueden en los datasheet de los componentes, que han sido referenciados en el Capítulo 4 COMPONENTES PANELES. El plano descrito se puede encontrar en el documento Nº3 PLANOS.

Una vez acotado correctamente el plano, se realiza el proceso de fresado y se insertan los componentes en las aperturas realizadas.

Tal y como se comenta en el Capítulo 4 COMPONENTES PANELES, todos los elementos seleccionados están pensados para montaje en panel y por tanto vienen con sus consiguientes roscas y tuercas utilizadas pata la fijación de los mismos. Sin embargo las pantallas, tanto la LCD como la TFT, no están destinadas a montaje en panel, por lo tanto es necesario incluir elementos de fijación externos. Se utilizan tornillos y tuercas de métrica dos para satisfacer dicho propósito.

Figura 6.7: Tornillo M2 x16 mm

Además con fines estéticos, se pretende que las pantallas queden al mismo nivel que la superficie del panel, por lo tanto se utilizan tornillos de longitud 16mm como los de la Figura 6.7, con las arandelas necesarias para ajustar las pantallas a la altura deseada.

Por último para dar un acabado más elegante, se realizan unos marcos para las pantallas. Se utiliza para ello una lámina de plástico negro cortada con las dimensiones adecuadas y una placa de metacrilato de unos 3 mm de espesor. Mediante un taladro se realizan los agujeros al metacrilato donde irán los tornillos de sujeción.

El resultado final tras el proceso descrito anteriormente, se puede apreciar en la Figura 6.8 y Figura 6.9.

Figura 6.8: Vista superior panel frontal terminado.

Figura 6.9: Alzado panel frontal terminado.

6.3. PANEL TRASERO

El panel trasero se encarga de cerrar la caja por la parte posterior, así como de alojar diferentes elementos de conexiones externas. A diferencia del panel frontal, este no posee roscas, sino que la sujeción a la caja se realiza mediante topes mecánicos. El panel encaja en unos railes en la parte posterior de la caja diseñados para alojar el mismo.

El panel trasero al igual que el panel frontal, es físicamente un rectángulo de aluminio, pero en este caso con unas dimensiones de 290.9mm x 120mm. [21]

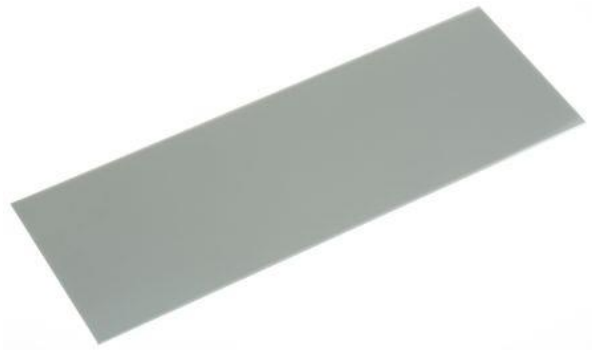


Figura 6.10: Panel trasero.

Siguiendo la misma metodología que en el caso del panel frontal, se diseña el plano a escala para determinar la colocación de los elementos.

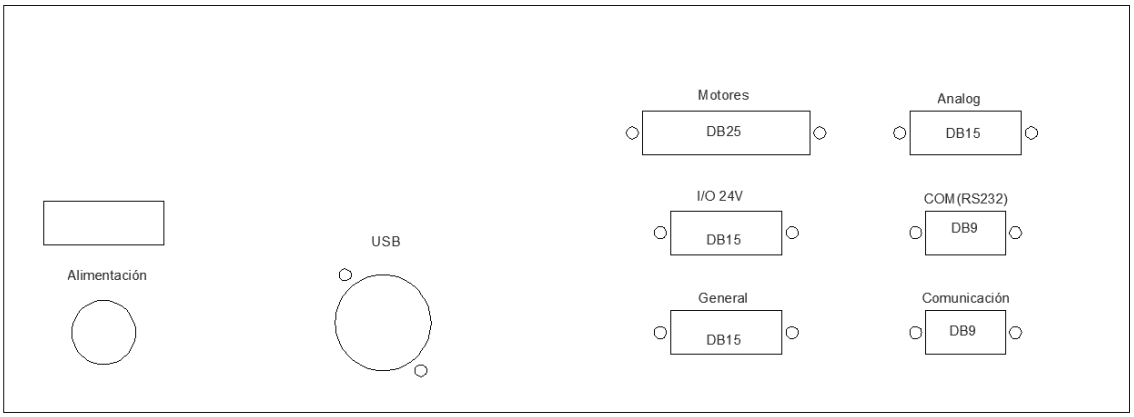


Figura 6.11: Esquema panel trasero.

Para el diseño se han tenido en cuenta una serie de factores tanto estéticos como funcionales. La separación entre conectores DB, debe ser suficiente para que sea posible utilizar dos conectores al mismo tiempo. Por otro lado, los conectores de alimentación y programación, se sitúan lo más cerca posible del lugar donde se colocará interiormente el controlador Arduino.

El plano descrito se puede encontrar en el documento Nº3 PLANOS.

Al igual que el en el apartado anterior, el resultado final tras el proceso descrito, se muestra en la Figura 6.12.

Figura 6.12: Panel trasero montado.

En este panel cabe reseñar que es necesario incluir etiquetas encima de cada conector de manera informativa, para diferenciar que módulos o señales están presentes en cada uno de los conectores.

Primeramente, este proceso se pensó realizar mediante una grabadora laser, pero por falta de medios y tiempo, se llevó a cabo mediante una etiquetadora.

7. DISEÑO ELECTRÓNICO

Este capítulo tratará sobre los diferentes módulos electrónicos que componen el entrenador, como se han diseñado y sus funcionalidades.

Sin embargo ya que varios de los módulos se comunican por el bus I2C, resulta conveniente incluir un apartado explicando sus características.

7.1. BUS I2C

El Bus I2C (del inglés Inter-Integrated Circuit) también llamado I^2C o IIC, es un protocolo de comunicación serie multi-master (admite más de un maestro) y multi-slave (se puede conectar al bus más de un esclavo).

Es ampliamente utilizado en el mundo de la electrónica por su sencillez, ya que solo utiliza 2 hilos para conseguir la comunicación: **SCL** por dónde van los pulsos del reloj asíncrono que indica cuando leer los datos y **SDA** que contiene los datos. [22]

Sigue un método de comunicación denominada Packet switching, (conmutación por paquetes) mediante el cual los datos son enviados en paquetes con cabecera y carga útil. La información que contiene la cabecera se utiliza para dirigir el paquete a su destino.

En el bus los diferentes nodos pueden jugar dos roles: maestro y esclavo.

- Maestro: se encarga de generar el reloj e iniciar la comunicación con los esclavos. Puede haber varios maestros conectados al bus, pero solo uno puede estar activo al mismo tiempo.
- Esclavo: recibe el reloj y responde con la dirección asignada.

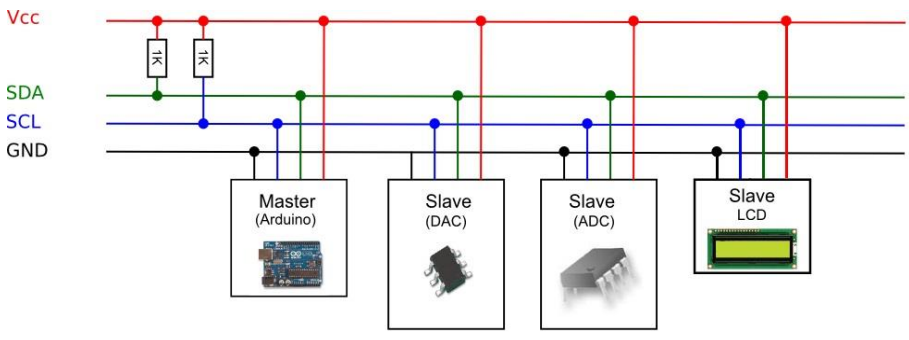


Figura 7.1: Esquema conexión Bus I2C.

Cada esclavo conectado en el bus debe tener una dirección diferente, ya que si no el maestro no sabría a qué esclavo dirigirse. Estas direcciones puede que vengan configuradas de serie o que se puedan modificar de algún modo. Es habitual que los integrados que funcionan con esta tecnología, reserven algunas de sus patas para poder cambiar esta dirección.

Como se muestra en la Figura 7.1 la conexión es sencilla, simplemente los dos hilos del bus y la alimentación de cada integrado. Cabe destacar que el bus es activo bajo, por ello se necesitan resistencias de pull up conectadas a cada línea para su correcto funcionamiento. En muchos módulos comerciales vienen integradas las resistencias.

Arduino ofrece una librería exclusivamente para manejar este bus y se llama Wire. [23]

7.2. MÓDULO I/O 24 V

Debido a que los autómatas industriales trabajan con niveles de tension de 24V, no es posibe la comunicación con nuestro sistema que trabaja a niveles de tensión de 5V. Este módulo se ha diseñado para ofrecer esta posibilidad, de esta manera se pueden mandar por ejemplo señales de consigna a un automata o leer señales que vengan de este.

En concreto se podrá escribir seis señales y leer otras seis. Como ya se menciona en la introducción de la memoria, se ha seguido una filosofía modular, con la idea de que el diseño se pueda utilizar en posibles sistemas desarrollados a partir de este.

El módulo se compone básicamente de dos elementos: MCP23017 (integrado expansor de entradas y salidas) y optoacopladores. A continuación se describe el funcionamiento básico de los elementos mencionados.

- **Optoacoplador**: es un dispositivo que funciona como un interruptor. Está compuesto por un diodo LED y un fototransistor. Cuando al diodo le llega una cierta intensidad, este emite una luz que satura el fototransistor poniéndose en conducción.

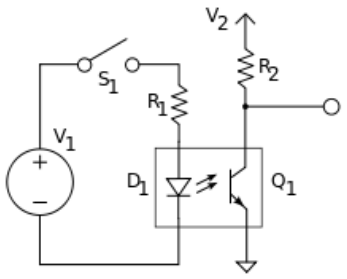


Figura 7.2: Conexión optoacoplador.

Tienen múltiples usos, como por ejemplo el aislar eléctricamente un circuito de otro, que pueda funcionar con unos niveles de tensión diferente. En este caso se usan para adaptar el nivel de tensión al que trabaja el PLC (24V), a los 5V con los que trabaja nuestro control. Además, se protegen ambas partes de posibles sobretensiones o sobre intensidades que puedan ocurrir.

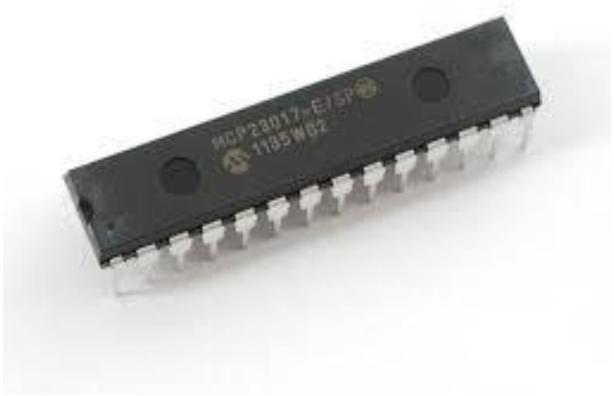


Figura 7.3: Integrado MCP23017

- **MCP23017:** este integrado que se conoce coloquialmente como expansor de entradas y salidas, utiliza el Bus I2C, para comunicarse con Arduino. Tiene 28 pines, de los cuales 16 corresponden a los dos puertos GPA y GPB. Cada puerto consta de 8 señales que se pueden configurar como entradas o salidas digitales. A continuación se detallan todos los pines del integrado y su función:

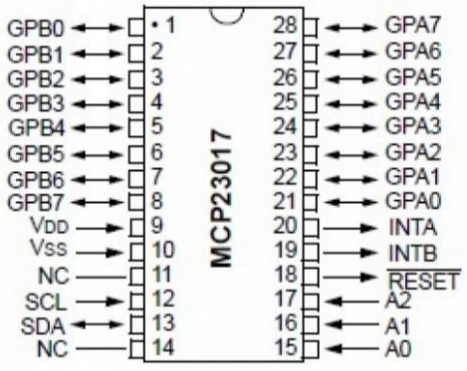


Figura 7.4: Pinout MCP23017 [24]

- **GPB0-GPB7**: Pines correspondientes al puerto de entradas y salidas digitales GPB. Se puede configurar para que la entrada tenga una resistencia de pull up interna.
- **GPA0-GPA7:** Pines correspondientes al puerto GPA de las mismas características que el GPB.
- **VDD:** Pin de alimentación del integrado, se alimenta con 5V.
- **Vss:** Pin de referencia eléctrica a 0V, también llamado GND.
- **NC**: No conectado.
- **SCL**: Línea de reloj del bus I2C.
- **SDA:** Línea de datos del bus I2C.
- **INTA:** Pin de interrupción en el puerto A, esta señal se activa alta si se detecta una interrupción en el puerto GPA.
- **INTB:** Pin de interrupción en el puerto B, esta señal se activa alta si se detecta una interrupción en el puerto GPB.
- **RESET:** Pin para hacer un reset del integrado por hardware, es activo bajo, es decir, se activa llevando el pin a tierra.
- **A0-A2:** Pines para la configuración de la dirección del bus I2C

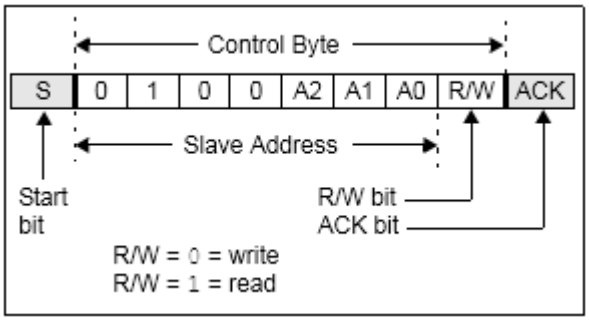


Figura 7.5: Registro dirección de esclavo I2C.

En la Figura 7.5 se muestra como se selecciona la dirección de esclavo que tendrá este integrado. Los cuatro primeros bits son fijos he iguales en todos los integrados de este tipo, sin embargo, se pueden modificar tres bits que están accesibles en las patas A2, A1 y A0, por lo tanto, se podrían llegar a conectar 8 integrados iguales. De esta manera si conectamos dicha pata a 5V tendremos un 1 lógico y si la conectamos a tierra tendremos un 0, formando así las direcciones deseadas. [24]

Una vez descritos los componentes utilizados, se explica el diseño seguido.

7.2.1. DISEÑO.

Para calcular las resistencias que se encargan de administrar la corriente necesaria a los optoacopladores para que saturen correctamente hay que valerse de las especificaciones del dispositivo utilizado, en este caso se trata de un FOD817 fabricado por FAIRCHILD. [25]

Utilizando el datasheet del componente se obtiene que la corriente máxima que puede atravesar el diodo (I_F) es 50mA, y que para una corriente de 20mA, el valor típico de caída de tensión en el diodo es 1.2V.

La tensión que cae en la resistencia será por tanto la tensión a la que estará trabajando, que en el caso de las salidas serán 5V y en el caso de las entradas 24V, menos la tensión que cae en el diodo, que del datasheet se obtiene que son 1.2V.

Con estos datos resulta sencillo obtener una primera aproximación de la resistencia deseada.

Para los optoacopladores colocados como salidas la conexión será como la de la Figura 7.6

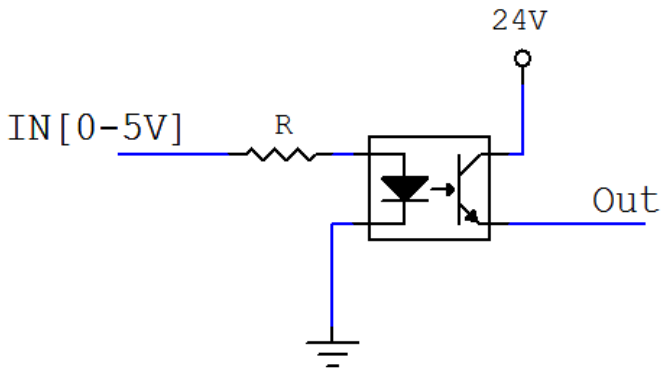


Figura 7.6: Esquema conexión optoacoplador 5V-24V.

Y la resistencia R se calculará de la siguiente manera:

$$(5 - 1.2) = 20 * 10^{-3} * R$$

Despejando se obtiene que sería necesaria una resitencia R de 190Ω

Sin embargo en la práctica con una intensidad a través del diodo de unos 5mA es suficiente para que el optoacoplador conmute, además el consumo será menor. Por lo tanto, se elige una resistencia normalizada de **330 Ω**, que resulta en una intensidad a través del diodo de 11.5mA.

Para los optoacopladores conectados como entradas la conexión será como en la figura 7.7.

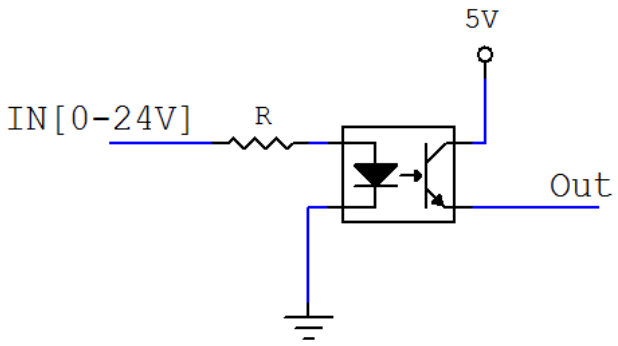


Figura 7.7: Esquema conexión optoacoplador 24V-5V.

Análogamente al proceso anterior, se obtiene una resitencia normalizada de **3300 Ω** con la que obtenemos una intensidad a través del diodo de 7mA

En el esquema electrico de este módulo que se encuentra en el **Documento Nº3 Planos** se puede observar que el bus I2C lleva sus resistencias de pull up externas. También se puede observar la direccion seleccionada en el integrado, los tres bits que se pueden modificar estan puestos a 5V por lo tanto la **dirección seleccionada para este módulo será 0100111.**

7.2.2. FUNCIONAMIENTO.

Para poder operar con este módulo, se debe conocer el funcionamiento del MCP23017, ya que el usuario únicamente tendrá que manejar este integrado y serán los optoacopladores los que se encarguen de acoplar la señal a 24V.

Existe una librería desarrollada por Adafruit que facilita mucho el manejo del integrado, pero debido a que no es excesivamente complicado trabajar directamente con el Bus I2C y uno de los objetivos del proyecto es aprender a programar este bus, resulta más efectivo describir el funcionamiento sin utilizar dicha librería. A pesar de ello, se incluirá una breve explicación de esta librería al final del capítulo.

Como ya se comentó anteriormente, este integrado utiliza una interfaz I2C, por lo que deberemos utilizar la librería Wire para el control del mismo.

La mayoría de dispositivos que trabajan con el bus I2C tienen varios registros con los que se puede trabajar, primero se debe de mandar el dato de la dirección del registro a la que se quiere acceder, y posteriormente el valor que se quiere cargar al registro.

A continuación se muestra una tabla obtenida del datasheet con los diferentes registros a los que se puede acceder:

Address IOCON.BANK = 1	Address IOCON.BANK = 0	Access to:
00h	00h	IODIRA
10h	01h	IODIRB
01h	02h	IPOLA
11h	03h	IPOLB
02h	04h	GPINTENA
12h	05h	GPINTENB
03h	06h	DEFVALA
13h	07h	DEFVALB
04h	08h	INTCONA
14h	09h	INTCONB
05h	0Ah	IOCON
15h	0Bh	IOCON
06h	0Ch	GPPUA
16h	0Dh	GPPUB
07h	0Eh	INTFA
17h	0Fh	INTFB
08h	10h	INTCAPA
18h	11h	INTCAPB
09h	12h	GPIOA
19h	13h	GPIOB
0Ah	14h	OLATA
1Ah	15h	OLATB

Figura 7.8: Registros control MCP23017

Cabe recordar que la dirección de esclavo del módulo es 0100111, normalemente se trabaja en hexadecimal, por lo que la dirección será 0x27.

Siguiendo esta metodología, primero configuraremos el puerto A como entrada, y el puerto B como salida, por defecto los puertos vienen configurados como entradas, por lo que solo necesitamos modificar el registro del puerto B que será el puerto de salidas. A continuación, se muestra el código necesario:

```
Wire.beginTransmission(0x27); //  Iniciamos la comunicación con el módulo.
Wire.write(0x01); // Accedemos al registro de configuración del puerto B (IODIRB)
Wire.write(0x00); //  Configuramos todo el puerto como salida
Wire.endTransmission(); // Terminamos la transmisión
```

Siguiendo la misma filosofía, para escribir un determinado valor se utiliza el siguiente código:

```
Wire.beginTransmission(0x27);
Wire.write(0x13); // Dirección del registro del puerto B (GPIOB)
Wire.write(¿?);  // Valor que se desea enviar
Wire.endTransmission();
```

El valor escrito en el puerto representará los bits que se activan, así pues si se escribe 001, se activará la salida 0 del puerto y si se escribe 002 la salida 1 y se desactivará la salida 0.

Y para leer de las entradas se utilizará la siguiente secuencia de comandos:

```
Wire.beginTransmission(0x27);
// Colocar el puntero interno del módulo en el registro del  puerto A
Wire.write(0x12);
Wire.endTransmission();
// Pedir al módulo el valor del registro al que apunta.
Wire.requestFrom(0x27, 1);
Entradas=Wire.read();
```

El valor de las entradas se guarda en una variable que hay que definir previamente.

Como se ha comentado anteriormente, existe una librería desarrollada por Adafruit que maneja este integrado y hace la tarea de programación un poco más amena.

La librería se llama Adafruit_MCP23017 y sigue los principios del lenguaje Arduino. A continuación se explica cómo usarla.

Primero se debe instanciar la librería dándole el nombre que queramos, por ejemplo “Mcp”:

Adafruit_MCP23017 Mcp;

Una vez instanciada la librería ya se pueden usar las funciones que implementa, para ello debemos escribir el nombre de la instancia seguido de un punto y la función que queramos utilizar.

Las funciones que nos brinda la librería son las siguientes:

- Mcp.begin(7): Inicia la comunicación con la dirección asignada, por defecto es 0, pero como ya se ha comentado anteriormente este módulo tiene asignada la dirección 0x27. En esta función solo hay que señalar los tres últimos bits, que son los que se pueden modificar por hardware. Por lo tanto le pasaremos como argumento un 7.

- Mcp.pinMode(N,Modo): Configura un pin N de un puerto como entrada o como salida. Deberemos indicarle como argumentos el número del pin que se quiere configurar y el modo de funcionamiento como INPUT u OUTPUT.
- Mcp.pullup(N,estado): activa las resistencias de pull-up internas. Los argumentos son el número de pin N, y el estado HIGH para activar las resistencias y LOW para desactivarlas
- Mcp.digitalWrite(N,estado): Escribe un estado HIGH o LOW en el pin N.
- Mcp.digitalRead(N): Lee el estado del pin N.

Por último, se deja un ejemplo completo de un sketch en el **Anexo Códigos de programación**

7.2.3. MONTAJE.

Para el montaje de este módulo, se utilizó una placa de prototipos similar a la que se muestra en la Figura 7.9. En ella se insertan los componnetes utilizados y se sueldan las pistas por debajo creando las conexiones pertinentes.

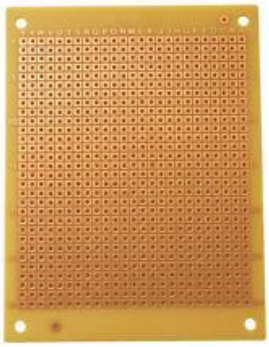


Figura 7.9 Placa de prototipos.

Antes de soldar los componentes, es recomendable realizar un pequeño boceto de cómo irán colocados y por donde pasarán las pistas, para posteriormente ir soldando los componentes con mayor facilidad.

El resultado de la placa ya montada en el entrenador se puede observar en la Figura 7.10 en la cual se resaltan con óvalos los elementos más relevantes y serán enumerados y descritos posteriormente.

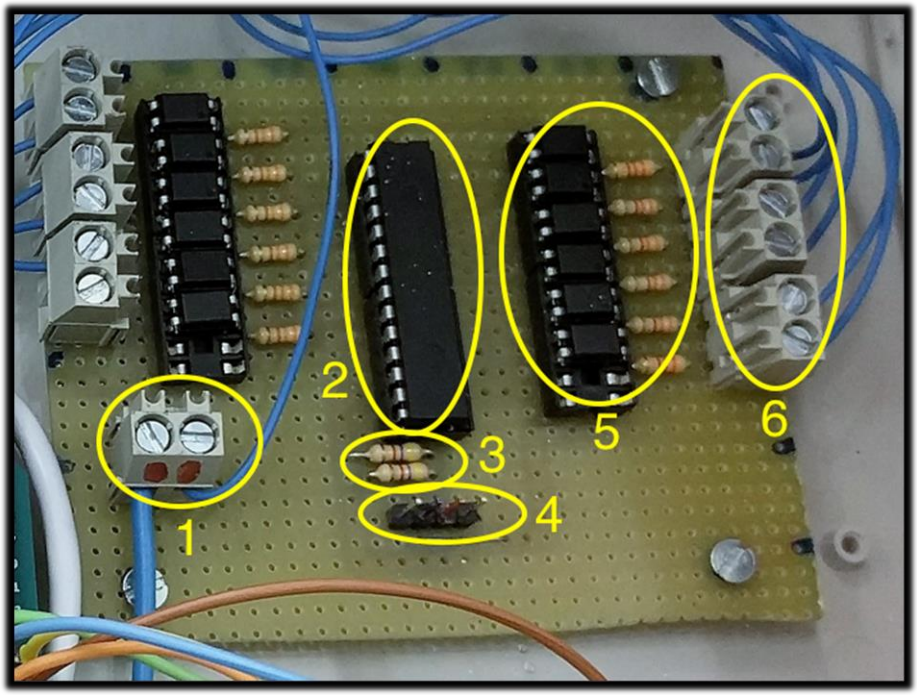


Figura 7.10: Modulo 24V montado y fijado.

Los elementos resaltados son los siguientes:

1. Alimentación a 24V.
2. MCP23017.
3. Resistencias de pull-up.
4. Conector BUS I2C
5. Optoacopladores
6. Conectores para E/S

7.3. MÓDULO LCD

La idea de este módulo surge de observar la configuración de muchos de los dispositivos utilizados en la industria, los cuales, con cuatro botones debajo de una pantalla se controla todo un menú con funciones a desarrollar.

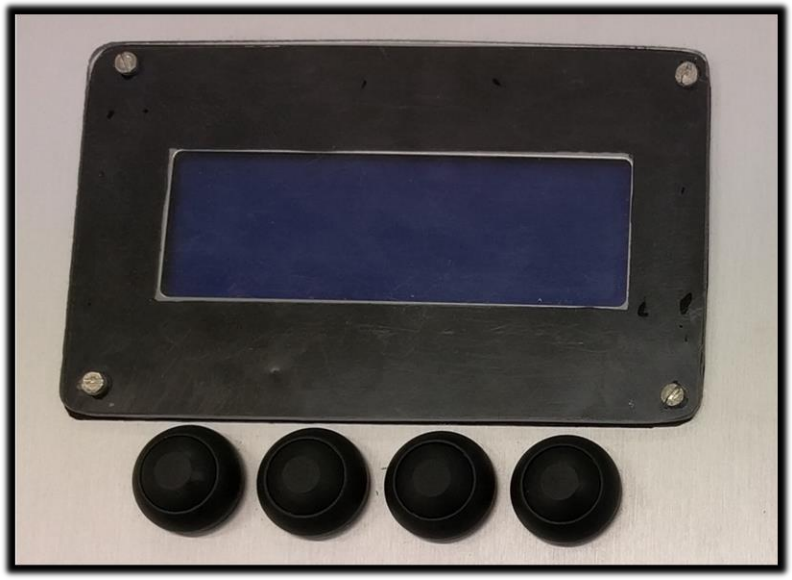


Figura 7.11: Módulo LCD montado.

Este módulo está basado en un diseño de la empresa “Adafruit”. La idease basa en que mediante el bus I2C se pueda manejar todo el módulo, es decir la pantalla completa y los cuatro botones. [26]

Los componentes utilizados son el ya mencionado MCP23017, los botones descritos en el capítulo 4 y una pantalla LCD que se podrá intercambiar. En este proyecto se utilizará una pantalla como la que se describe en el capítulo 4.1.6 de este documento.

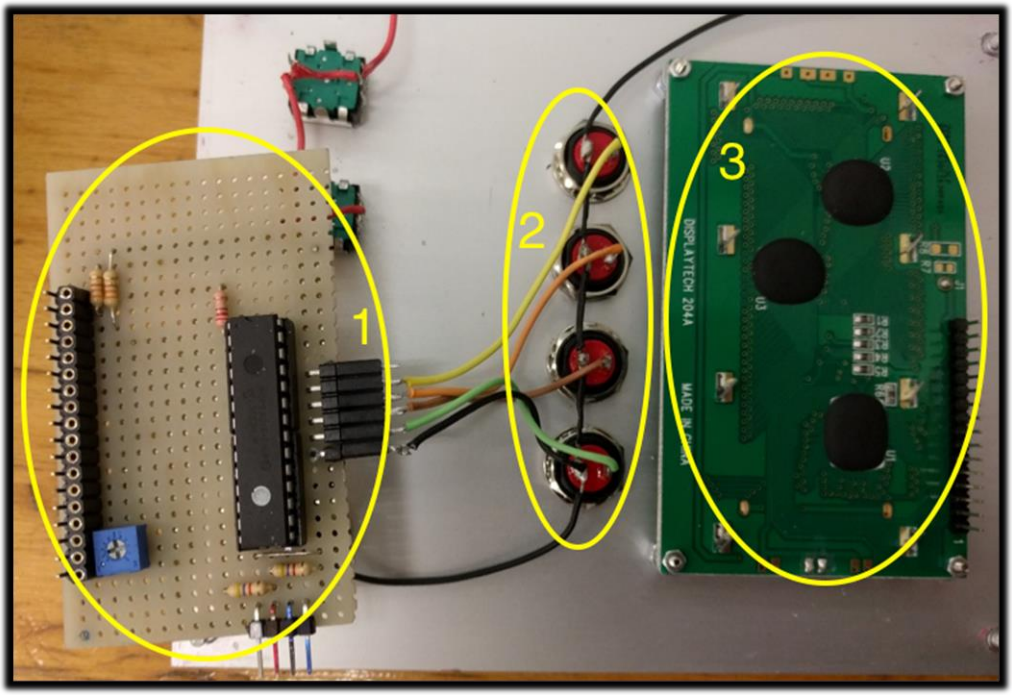


Figura 7.12: Módulo LCD desconectado de pantalla.

En la figura 7.12, se puede observar la configuración física del módulo desconectado de la pantalla, este se compone de 3 elementos básicos.

1. Placa de control: En ella se alojan los componentes encargados del control de la pantalla y los botones. Este elemento se explica en detalle seguidamente.
2. Botones: Cuatro botones pulsadores conectados a la placa de control.
3. Pantalla LCD.

Una vez comentadas las partes generales de este módulo, se explican los componentes que constituyen la placa de control. La figura 7.13 muestra y enumera los elementos más importantes de esta placa.

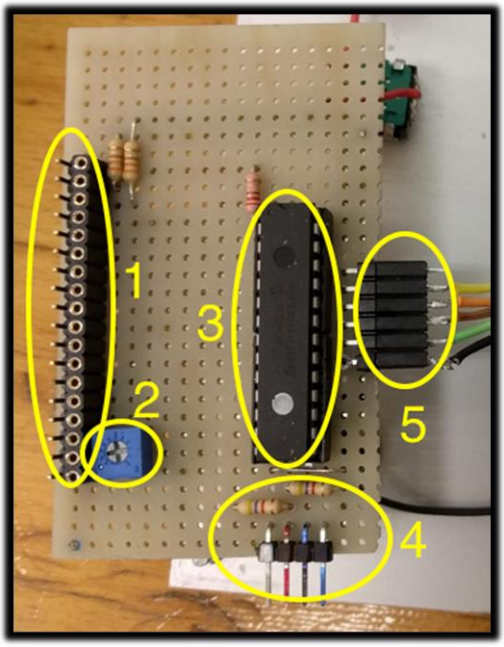


Figura 7.13: Placa control LCD

1. Conector para pantallas LCD: dos conjuntos de pines soldados de manera que sea cómodo el acople de cualquier tipo de pantalla LCD con una conexión convencional.
2. Potenciómetro: resistencia de 10KΩ variable. Utilizada para ajustar el brillo de la pantalla. Una vez configurado el brillo deseado, no es necesario modificarla.
3. MCP23017
4. BUS I2C: conector ya visto en el módulo de 24V, incluyendo las resistencias de pull-up.
5. Conector botones: Pines utilizados para poder conectar y desconectar fácilmente hasta 5 señales de entrada, en este caso solo se utilizan las 4 de los botones.

Se puede consultar el detalle de las conexiones en el plano Nº6: "Módulo LCD" que se encuentra en el **Documento Nº3 Planos**.

Al igual que en el módulo de 24V, los elementos de la placa de control se alojan sobre una placa de prototipos, en la que se sueldan las pistas pertinentes por la parte posterior para crear las conexiones necesarias. El resultado de las pistas soldadas se puede apreciar en la Figura 7.14.

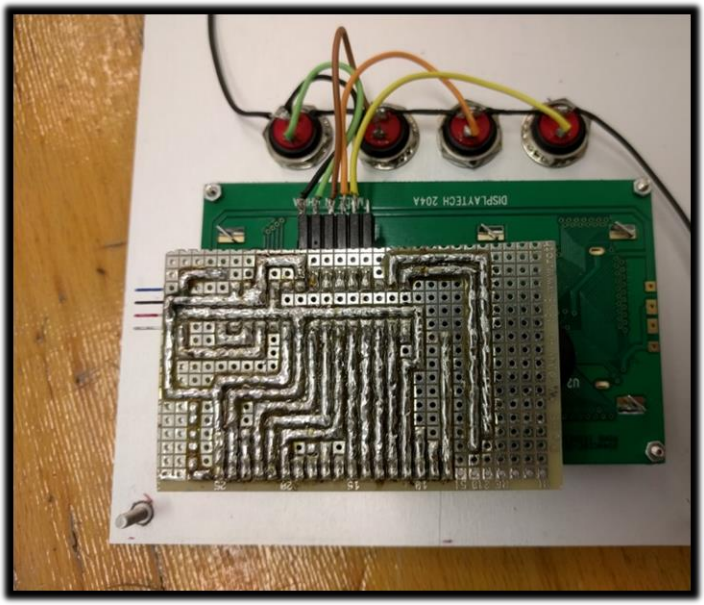


Figura 7.14: Vista posterior del módulo LCD

Una vez determinados todos los elementos que componen este módulo, se explica el funcionamiento del mismo así como la librería que se debe utilizar.

7.3.1. FUNCIONAMIENTO

Para manejar este módulo con facilidad, se utilizará una librería desarrollada por Adafruit llamada "**Adafruit_RGBLCDShield**" que se puede descargar gratuitamente desde un repositorio de GitHub [27]. Es necesario descargar e instalar previamente esta librería para poder utilizarla, ya que no es una librería estándar de Arduino y por tanto no viene por defecto cuando se instala el IDE.

Esta librería es una expansión de la librería estándar desarrollada por Arduino para el control de pantallas LCD, llamada **"LiquidCrystal"** [28]. Es por ello que se pueden utilizar cualquiera de las funciones que implementa y funcionará exactamente igual. [29]

Se implementan dos funciones extra interesantes:

- **lcd.setBacklight();** se utiliza para cambiar el color la pantalla. Es una función útil cuando se manejan pantallas LCD RGB. Como ya se ha comentado anteriormente, en este proyecto se utiliza una pantalla monocroma, por lo que esta función solo encendería o apagaría la luz trasera de la pantalla.

- **lcd.readButtons();** los botones se leen todos al mismo tiempo cuando se llama a esta función, que devuelve una variable que contiene la información de si el botón está pulsado o no pulsado. Esta variable contiene una serie de bits, los cuales se ponen a 0 o a 1 dependiendo del estado del botón. Cabe destacar, que la librería maneja los rebotes mecánicos producidos en la conmutación internamente, por lo que no es necesario añadir código para suprimir este efecto.

El módulo original de Adafruit utiliza una configuración de botones en estrella, similar a como se encuentran en un mando de PlayStation, por ejemplo. Es por ello que las variables utilizadas en la lectura de los botones y que se definen internamente en la librería, tengan nombres que no se ajusten a la configuración física de los botones de este proyecto, que según se aprecia en la Figura 7.11 es una configuración en línea. Para que quede claro, se denomina F1 al primer botón, F2 al botón de su derecha, F3 al siguiente y F4 al último botón. Y la relación con los nombres utilizados por la librería es la siguiente:

- F1 → BUTTON_SELECT
- F2 → BUTTON_RIGHT
- F3 → BUTTON_DOWN
- F4 → BUTTON_UP

Por último, se deja un ejemplo de utilización del Módulo en el DOCUMENTO Nº2 ANEXO CODIGOS DE PROGRAMACIÓN.

7.4. PANTALLA TFT

Este módulo, ya ha sido brevemente introducido en el APARTADO 4.1.7, explicando algunas de sus características más importantes, sin embargo, aún resta por detallar el conexionado y el funcionamiento. Estos y algunos detalles más se explican en este apartado.

Figura 7.15: Vista frontal pantalla TFT

Como ya se comentó anteriormente, con la idea de poder representar alguna gráfica o dibujo sencillo se incluye este módulo que consta de una pantalla TFT de 1.77" en la que se puede escribir texto, imágenes o formas de diferentes colores. También tiene incluido en la parte de atrás un slot para leer tarjetas micro-SD aunque este no se utilizará. Este módulo se comunica por el bus SPI descrito en el siguiente apartado.

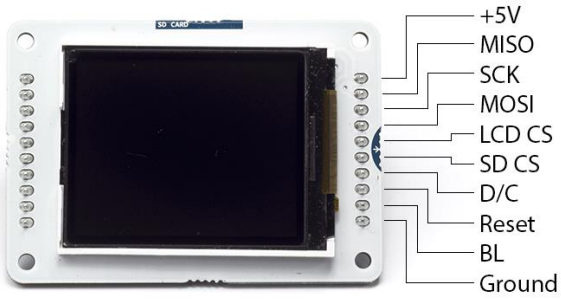


Figura 7.16: Vista frontal y conexiones pantalla TFT

A continuación se explican las conexiones de este módulo:

- **MISO**: 'Master Input Slave Output' utilizada por la tarjeta SD para la trasmisión de datos por SPI. *Si solo utilizas la pantalla, se puede omitir este pin.*
- **SCK**: entrada del reloj SPI.
- **MOSI**: 'Master Output Slave Input' salida de datos del Arduino al módulo, tanto para la SD como para la pantalla.
- **LCD CS**: Chip Select, selecciona la pantalla LCD para la comunicación con Arduino.
- **SD CS**: selecciona la tarjeta SD. Se puede omitir si no se utiliza la SD.
- **D/C**: define si los datos enviados son data o comandos.

- **BL**: BackLight, luz de fondo, dos leds que se pueden alimentar por PWM. (controlado mediante BJT, ver esquema de conexiones en PFD "Adafruit_18TFT_Guide"). *Se puede conectar a 5 V directo.*

Cabe reseñar como observación, que el pin de reset del módulo, se puede conectar al pin de reset del controlador Arduino, suprimiendo de esta manera la necesidad de utilizar otro pin de la placa de desarrollo Arduino. Esta característica es especialmente útil en el caso de utilizar una placa de control con pocos pines de E/S, como podría ser el Arduino UNO. Por tanto, y en resumen, para el correcto funcionamiento de la pantalla, sin utilizar el slot de tarjetas SD, son obligatorios 4 pines además de VCC y GND, que son: SCK, MOSI, LCD CS y D/C.

7.4.1. BUS SPI

A pesar de que será una librería la que se encargue de manejar este bus resulta conveniente explicar los principios y características básicos de este bus.

El bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. Al igual que sucede con el I2C, en este bus la comunicación también es síncrona, por lo que es necesario una línea de reloj.[31]

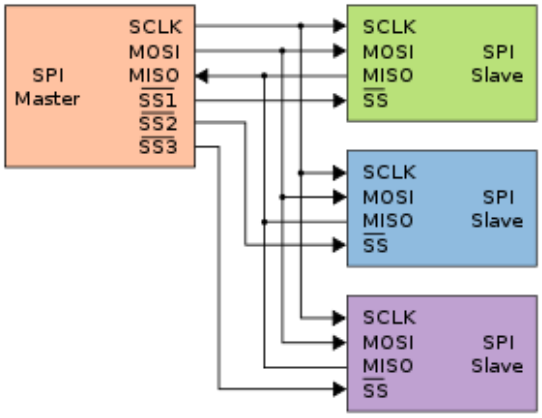


Figura 7.17: Esquema conexiones Bus SPI

El bus se compone básicamente de cuatro líneas:

- **SCLK (Clock):** Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit. También llamado TAKT (en alemán).
- **MOSI (Master Output Slave Input):** Salida de datos del Master y entrada de datos al Slave. También llamada SIMO.

- **MISO (Master Input Slave Output):** Salida de datos del Slave y entrada al Master. También conocida por SOMI.
- **SS/Select**: Para seleccionar un Slave, o para que el Master le diga al Slave que se active. También es llamada SSTE.

Algunos pros y contras de este bus se enumeran a continuación.

Ventajas:

- Comunicación Full Duplex
- Mayor velocidad de transmisión que con I2C
- No está limitado a trasferencia de bloques de 8 bits
- Consume menos energía que el Bus I2C

Desventajas:

- Consume más pines ya que por **cada esclavo conectado, se necesita una línea más de SS**
- No permite tener varios *servidores* conectados al mismo bus

El funcionamiento de este bus se basa en una secuencia sencilla. Cuando el master desea enviar o recibir información de un esclavo, le selecciona dejando baja la línea ss correspondiente y activando el reloj a una frecuencia útil tanto para el master como para el slave. El maestro genera información en la línea MOSI mientras lee de MISO. (Observar Figura 7.18)

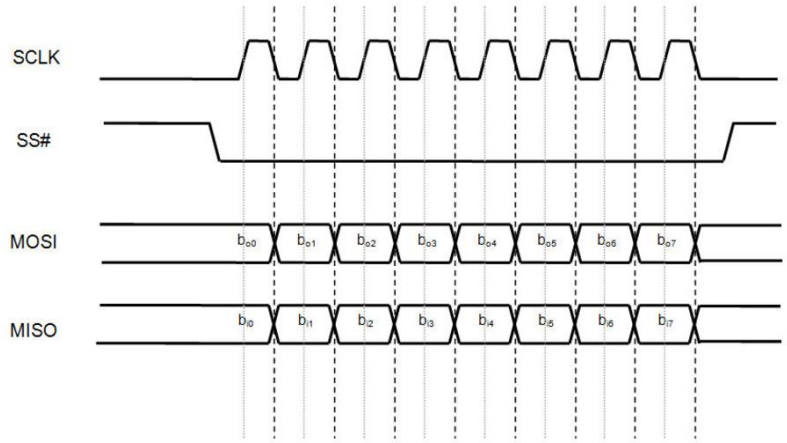


Figura 7.18: Secuencia Bits protocolo SPI

7.4.2. FUNCIONAMIENTO.

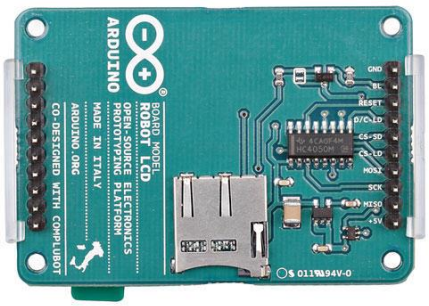


Figura 7.19: Vista trasera pantalla TFT

Primeramente, es importante conocer la configuración física de la pantalla para posteriormente saber cómo y dónde dibujar.

La pantalla está orientada horizontalmente, tiene 160 píxeles de largo y 128 de alto, teniendo esto en cuenta, la esquina superior izquierda se considera el pixel de coordenadas (0,0), la esquina superior derecha tendría unas coordenadas (0,159), la esquina inferior izquierda (127,0) y la esquina inferior derecha (127,159). [31]

Otro punto importante es el color, este display cuenta con un registro de 16 bits para manejar este aspecto. El color rojo y el azul tienen 5 bits de resolución cada uno, es decir 32 niveles de intensidad de cada color, los otros 6 bits restantes son utilizados para el color verde. Este último color tiene un bit más de resolución que el resto porque es el color que más aprecia el ojo humano.

La librería, sin embargo, maneja datos de 8 bits para cada canal, por lo tanto, se debe escribir un valor entre 0 y 255. Estos datos son escalados apropiadamente para que ocupen exactamente la longitud de bits que tiene asignado cada color.

La librería utilizada se llama “TFT library” y es una extensión de las librerías “AdafruitGFX” y “AdafruitST7735”. Utiliza comandos simples y tiene la ventaja de que se pueden seguir utilizando las funciones que implementaban las librerías de Adafruit.

Por último, se deja un ejemplo de utilización del Módulo en el DOCUMENTO Nº2 ANEXO CODIGOS DE PROGRAMACIÓN.

7.5. MÓDULO CONTROL MOTORES

Este módulo se enfoca en introducir algunas de las técnicas de control de motores eléctricos, y brinda la posibilidad de controlar fácilmente varios tipos de motores ampliamente utilizados. El control de motores tiene especial importancia en campos como la robótica o la impresión 3D.

Existen diferentes tipos de motores y para cada uno se debe usar una técnica de control diferente. A continuación, se explican algunos de los motores más utilizados. [33]

- **Motores DC**: existen dos tipos de motores de corriente directa o continua, los motores con escobillas (**Brushed DC Motors**) y los motores sin escobillas (**Brushless DC Motors**). Los motores de corriente continua con escobillas, son probablemente el tipo de motor eléctrico más utilizado. Se encuentran en todo tipo de aplicaciones, desde ventiladores hasta taladros inalámbricos, pasando por zumbadores de móviles o motores para coches o trenes. Estos motores utilizan escobillas que rozan con un anillo segmentado induciendo corriente en las bobinas del rotor que se alterna a medida que el motor gira. [34]

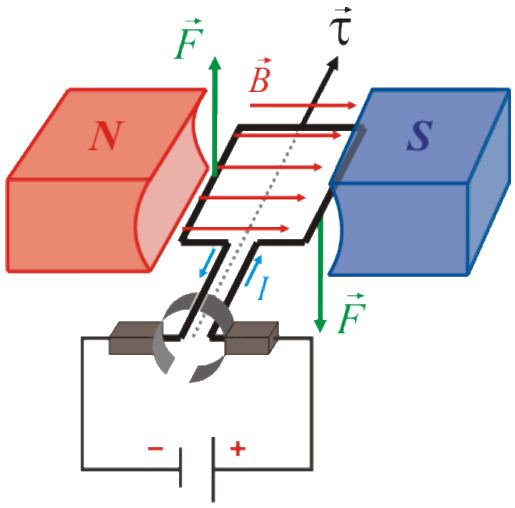


Figura 7.20: Esquema funcionamiento motor DC con escobillas

Los motores sin escobillas, son mecánicamente más simples. Reemplazan las escobillas por electrónica para realizar las conmutaciones. Eliminan así el rozamiento mecánico y el ruido eléctrico introducido por la conmutación brusca en los motores con escobillas. Son motores más silenciosos, por lo que se pueden encontrar en discos duros o ventiladores de ordenador. También se utilizan en cuadcopteros y vehículos en los que se necesitan mecanismos de alta precisión. [35]

- **Servomotor**: es un tipo de motor eléctrico con características especiales de control que tiene la capacidad de situarse en cualquier posición dentro de su rango de operación y mantenerse estable. Un servomotor puede ser controlado tanto en posición como en velocidad. Existen diferentes tipos de servomotores en función de su uso, por ahora se hablará de los servos de modelismo, utilizados principalmente en robótica. Estos servos constan de un motor de corriente continua, una caja reductora y un circuito de control.

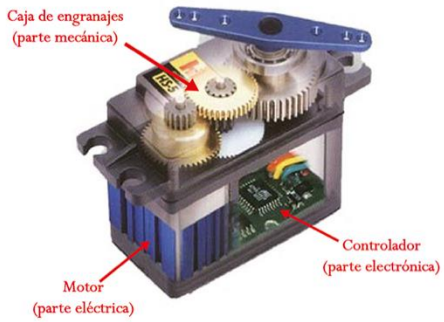


Figura 7.21: Servomotor

Uno de los engranajes de la caja reductora, está conectado a un potenciómetro, el cual determina la posición actual del motor. Mediante una señal PWM se define la nueva posición que debe adquirir el motor, la señal debe repetirse periódicamente para que el motor mantenga la posición deseada.

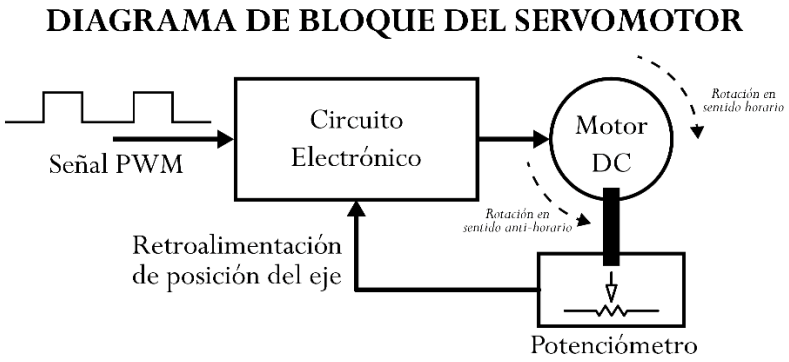


Figura 7.22: Diagrama de bloque del servomotor

El control de este tipo de motores se reduce, por tanto, a indicar el ángulo de posicionamiento en función del ancho de pulso de la señal de control.

Para manejar este tipo de motores Arduino cuenta con una librería propia llamada *Servo.h*

- **Motores paso a paso**: es un tipo de motor de corriente continua sin escobillas, el cual divide una vuelta completa del eje en un número determinado de pasos iguales. A diferencia del servomotor, la posición del eje se puede controlar sin necesidad de ningún tipo de realimentación. [36]

  Existen varios tipos de motores en función del tamaño, forma o características eléctricas, por lo que es imprescindible conocer bien las características del motor que se va a utilizar para no dañar el controlador o el propio motor.

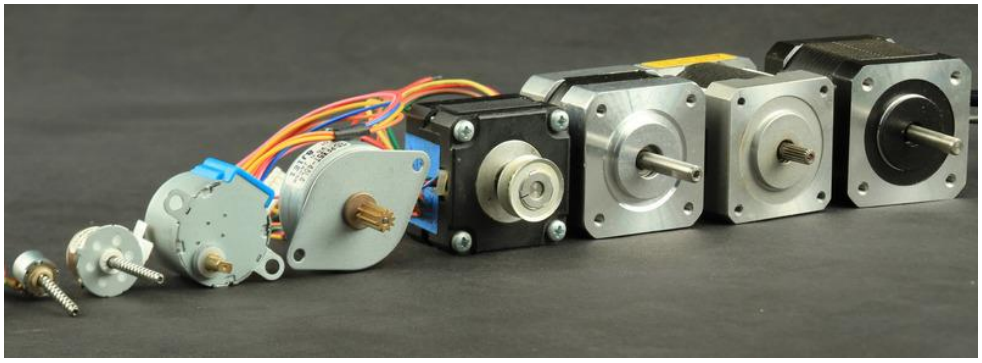


Figura 7.23: Motores paso a paso.

  Este tipo de motores se utilizan en aplicaciones que requieran un control preciso de la posición y la velocidad del eje, o cuando se requiere un buen par a bajas velocidades. Es por ello que suelen ser utilizados en impresoras 3D. Por el contrario, son motores poco eficientes, con poco par a altas velocidades, y sin retroalimentación, por lo que muchas veces es necesario la incursión de detectores para establecer una posición de referencia.

Tanto los motores descritos anteriormente, como el resto de tipo de motores eléctricos existentes, suelen requerir bastante potencia para ser manejados. Es por esto que se necesitan drivers de potencia.

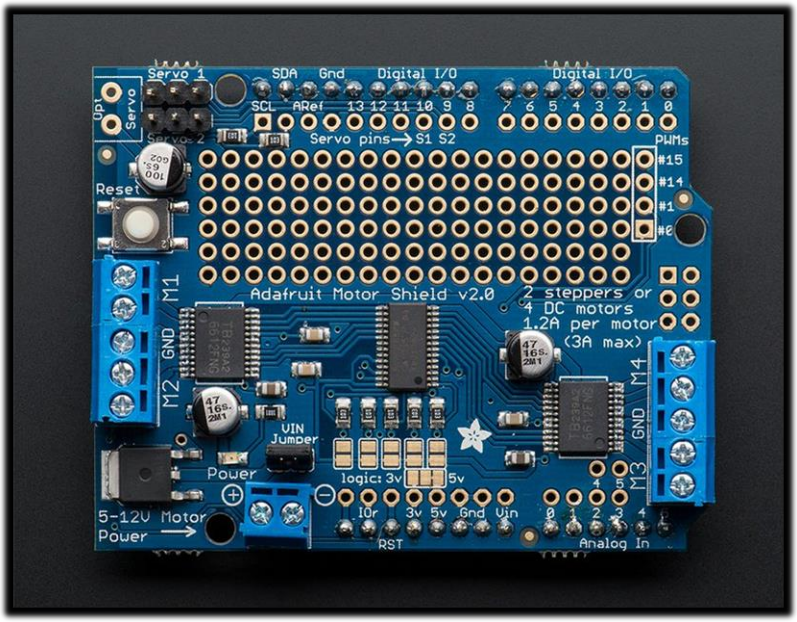


Figura 7.24: Módulo control motores.

El módulo que se incluye en este proyecto, el cual se puede observar en la figura 7.24, está formado por dos componentes fundamentales:

- **PCA9685**: es un integrado controlado mediante el bus I2C que genera hasta 16 señales PWM, en un principio está pensado para aplicaciones de control de LED, pero puede servir perfectamente como 'expansor' de salidas PWM.[37]
- **TB6612FNG**: es un integrado que funciona como driver para motores con transitores MOSFET como salida. Estos integrados son capaces de suministrar 1.2A por canal y tienen incluidos diodos flyback para evitar daños en la electrónica producidos cuando se manejan motores. [38]

La dirección I2C de este módulo es seleccionable desde la 0X60 hasta la 0x80, en el caso de este proyecto se selecciona la **dirección 0x60**.

El funcionamiento de este módulo se basa por tanto en controlar el integrado PCA9685 que es el encargado de generar las señales PWM para realizar las diferentes técnicas de control de motores. El driver TB6612, se encarga de adecuar la potencia de las señales generadas para que los motores funcionen. Existe una librería, desarrollada por Adafruit, con diversas funciones y ejemplos que ayudan al control de este módulo.

Se estudió la posibilidad de diseñar e implementar un módulo propio para abordar este tema, pero surgieron varios problemas que hicieron más factible la opción de comprar el modulo diseñado por adafruit. El obstáculo más importante fue la incapacidad para encontrar en RS integrados que

funcionasen como driver con un montaje en orificio pasante. En contraposición, resultó muy útil la facilidad de montaje del módulo de Adafruit y la disponibilidad del mismo en RS.

Además, Adafruit pone a disposición de los usuarios de este módulo, un tutorial muy completo donde se puede aprender los métodos para el control de motores.

7.6. COMUNICACIÓN RS232

Con el fin de dotar al entrenador de más conectividad aún, se incluye el adaptador MAX3232. Esta placa se encarga de agregar el estándar de comunicación serie RS232 al diseño.[40]

Aunque el número de dispositivos que utilizan la conexión RS232 está disminuyendo, aun se pueden encontrar numerosos elementos en la industria que utilizan este tipo de comunicación. Es por ello que se decide contar con este módulo que convierte la señal RS232 a niveles TTL para que el microcontrolador Arduino sea capaz de comunicarse a través de uno de los puertos serie disponibles.

Para utilizar este módulo, basta con abrir el puerto serie al que está conectado, y leer o escribir datos en él.

El rango de tensiones de funcionamiento de la placa es de 3V a 5V y la velocidad máxima de trasmisión de datos es 250kbps [41]

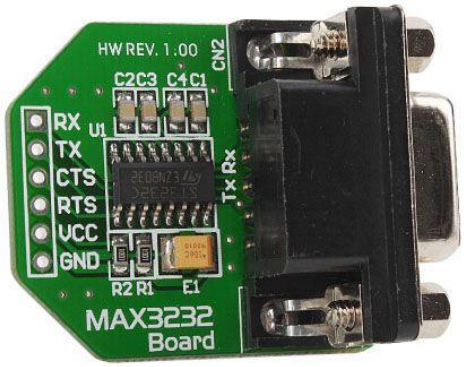


Figura 7.25: Adaptador MAX3232

7.7. ALIMENTACIÓN

Como se ha comentado en el apartado 4.2.1, la alimentación del sistema completo se realiza mediante un conector M12 que se utiliza habitualmente en equipos de automatización y detección. Estas conexiones funcionan por estándar a 24V, por lo que es necesario convertir estos 24V a los niveles de trabajo utilizados por los diferentes módulos que compone el entrenador.

Con este objetivo se incluyen dos convertidores dc-dc aislados:

- **REC7.5-2405**: este elemento convierte los 24V de entrada en **5V**. Es capaz de suministrar hasta 1.5A, lo que se traduce en una potencia máxima de 7.5W con una eficiencia de hasta el 86%. Se encarga de alimentar casi todos los elementos del sistema, desde el propio Arduino hasta las pantallas, ya que como se ha comentario en varias ocasiones se trabaja con niveles de tensión de 5V. [42]

Figura 7.26: Convertidor dc/dc aislado 24V-05V

- **REC8-2412**: este otro convertidor transforma los 24V que le llegan a la entrada a **12V**. Es capaz de suministrar hasta 666mA lo que se traduce en una potencia de 8W con una eficiencia de entre 84 y 86 %. Se encarga de alimentar exclusivamente el módulo de control de motores descrito anteriormente, ya que este módulo necesita más potencia para poder manejar correctamente algún tipo de motor.[43]

Figura 7.27: Convertidor dc/dc aislado 24V-12V

Como observación cabe reseñar que en el apartado de la alimentación se pensó en diseñar una alimentación propia mediante un convertidor de potencia controlado mediante una señal PWM del Arduino, y que así se pudiera optar al nivel de tensión deseado en cada momento, pero se descartó esta opción entre otras cosas por la facilidad de montaje y el reducido espacio de los convertidores descritos anteriormente.

8. MONTAJE FINAL

Por último y tras haber descrito los elementos que componen el sistema. Se procede a describir el montaje final. En esta última fase del proyecto primeramente se ensamblan los elementos en los paneles como se puede observar en la figura 8.1.

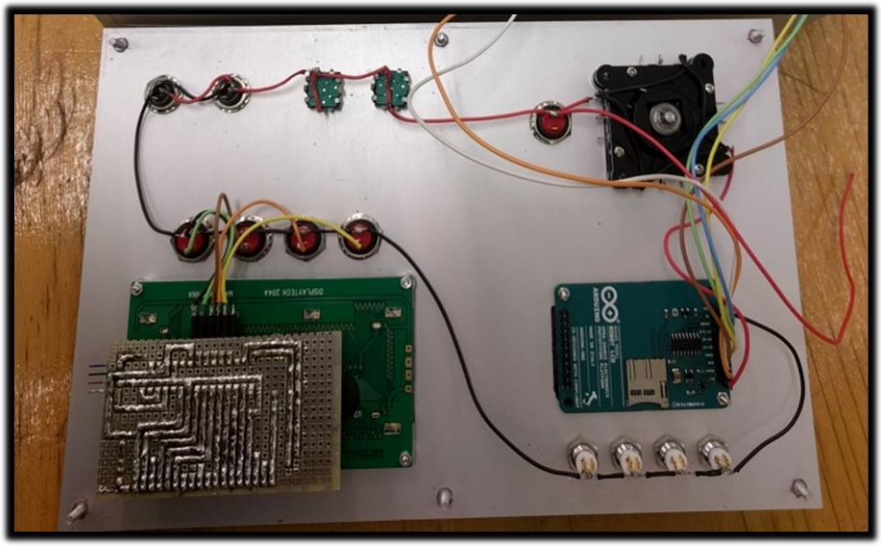


Figura 8.1: Elementos ensamblados en panel.

Posteriormente se fijan los elementos situados internamente y se realizan las conexiones de los módulos internos.

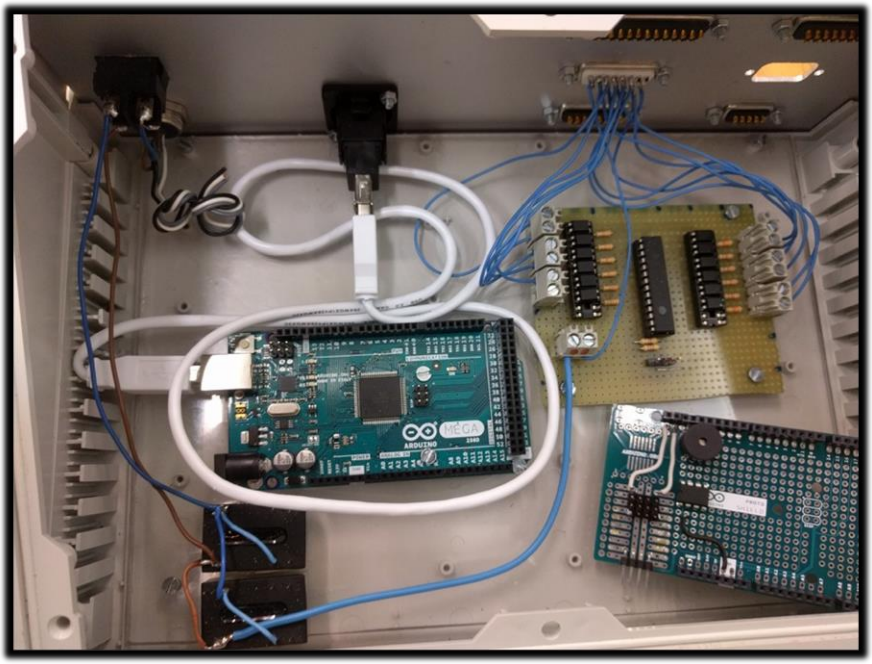


Figura 8.2: Interior en etapa media de montaje.

Por último se realiza todo el cableado necesario y se cierra debidamente la caja para ocultar toda la electrónica.

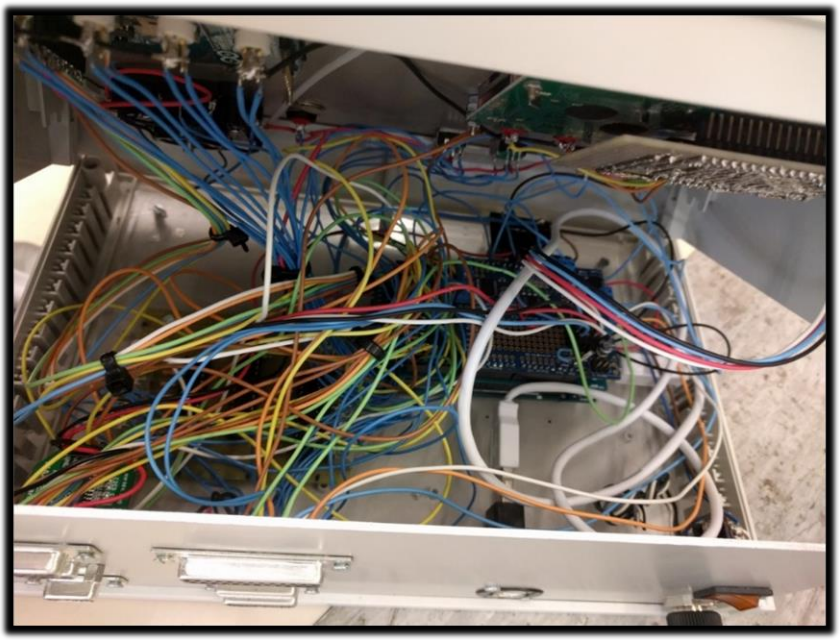


Figura 8.3: Cableado final.

El resultado final exterior se puede observar en las figuras 6.8 y 6.9 del capítulo 6 Diseño y Fabricación Mecánica.

9. CONCLUSIONES Y LÍNEAS FUTURAS

Tras la finalización del proyecto, se realiza un balance entre los objetivos planteados al iniciar el proyecto y las metas conseguidas.

Primeramente, cabe recordar que el objetivo primordial de este proyecto era el diseño y fabricación de un sistema donde poder aprender y desarrollar capacidades relacionadas con la programación de microcontroladores y en especial de la plataforma Arduino. Dicho sistema contiene elementos útiles para la su aplicación en la industria.

Una vez finalizado el montaje y el testeo de los diferentes módulos de los que se compone el sistema, se puede afirmar que se ha diseñado y fabricado un entorno donde aprender y entrenar las habilidades antes descritas y que todos elementos introducidos funcionan satisfactoriamente.

Además, se deja bastante espacio tanto físico como en los conectores que comunican con el mundo exterior, para ampliar el sistema con más funcionalidades. Se puede afirmar por tanto que el sistema aún tiene margen de mejora introduciendo más módulos como puede ser la comunicación wifi para poder trabajar sobre el internet de las cosas.

Paralelamente a este proyecto se pretende realizar una serie de maquetas físicas, con cilindros neumáticos y compuertas simulando un puesto de trabajo, para que tanto mediante el sistema descrito en este documento como los entrenadores de PLC existentes en la fábrica, se pueda aprender de una manera visual, añadiendo sensores y actuadores y trabajando sobre la maqueta.

También se pretende fabricar conectores y tener una serie de dispositivos como motores a los cuales se pueda conectar el entrenador fácilmente y poder aprender con ellos.

10. DISPOSICIONES LEGALES Y NORMAS APLICADAS

- "*Normativa del Proyecto Fin de Grado en Ingeniería Mecánica, Eléctrica y Electrónica Industrial y Automática*". -Universidad de Cantabria, 2013.
- "*Normativa del Trabajo de Fin de Grado*". -Universidad de Cantabria, 2013.
- Norma UNE 157001-2014

11. BIBLIOGRAFÍA

11.1. Libros.

- "Introducción a Arduino"
    - Autor: Massimo Banzi
    - Editorial: ANAYA
    - Edición: 2015

11.2. Páginas Web.

- https://es.wikipedia.org/wiki/Wikipedia:Portada
- https://programarfacil.com/blog/arduino-blog/adafruit-motor-shield-arduino/
- https://www.staticboards.es/blog/motores-paso-paso/#Tamano_y_Estandard_NEMA
- https://www.adafruit.com/
- https://www.prometec.net/
- https://www.youtube.com/?hl=es&gl=ES
- http://www.rae.es/
- http://ieeexplore.ieee.org/Xplore/home.jsp?reload=true
- https://scholar.google.es/

11.3. Software.

- Circuit Maker: utilizado para la generación de esquemas eléctricos.
- AutoCad: utilizado para el diseño mecánico.
- Arduino IDE: utilizado para la programación del microcontrolador.

11.4. Referencias.

[1] "Products – Arduino". [Online]. Available: https://www.arduino.cc/en/Main/Products

[2] "IDE- Arduino". [Online]. Available: https://www.arduino.cc/en/main/software

[3] "Tutorial librerías en arduino". [Online]. Available: https://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

[4] "Librerías – Arduino". [Online]. Available: https://www.arduino.cc/en/Guide/Libraries

[5] "Mega 2560 Documentation – Arduino". [Online]. Available: https://store.arduino.cc/arduino-mega-2560-rev3

[6] "SoftwareSerial – Arduino". [Online]. Available: https://www.arduino.cc/en/Reference/SoftwareSerial

[7] "Datasheet SPTS Panel Mount Off-(On) Pusch Button Switch". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1587/0900766b8158753b.pdf

[8] "Datasheet Potenciometer P16". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/0ab9/0900766b80ab9da2.pdf

[9] "Datasheet RS Pro Prominent Indicator Panel Mount". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1579/0900766b81579053.pdf

[10] "Datasheet Joystick". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1586/0900766b81586b75.pdf

[11] "Datasheet PEC11 Series - 12 mm Incremental Encoder". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/13ec/0900766b813ecfbc.pdf

[12] "Datasheet Black 13 mm potentiometer knob, with a black white indicator". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1580/0900766b81580752.pdf

[13] "Basics of Rotary Encoders: Overvew and New Technologies". [Online]. Available: http://www.machinedesign.com/sensors/basics-rotary-encoders-overview-and-new-technologies-0

[14] "Flat Panel Displays: Advanced Organic Materials" [Online]. Available: https://books.google.co.uk/books?id=LMW0gofBH7sC&pg=PA115&dq=Super-twisted+nematic+display&client=firefox-a&sig=ACfU3U3TfpkrO7WuFCIgbOXyRLMRSd3_TQ&hl=en#v=onepage&q=Super-twisted%20nematic%20display&f=false

[15] "Datasheet LCD Module 204A Series". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/06dd/0900766b806dda1c.pdf

[16] "LCD Screen- Arduino". [Online]. Available: https://store.arduino.cc/arduino-lcd-screen

[17] "Datasheet connector assembly". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1587/0900766b8158713d.pdf

[18] "Datasheet Carcasa de sobremesa bopla 45620185" [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1273/0900766b812738c6.pdf

[19] "Fresadora- EcuRed". [Online]. Available: https://www.ecured.cu/Fresadora

[20] "Datasheet panel frontal bopla". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1117/0900766b811177a0.pdf

[21] "Datasheet panel trasero". [Online]. Available: http://es.rs-online.com/web/p/products/1199139/

[22] "I2C Bus". [Online]. Available: http://www.i2c-bus.org/i2c-bus/

[23] "Librería wire I2C". [Online]. Available: https://www.arduino.cc/en/Reference/Wire

[24] "MCP23017 Datasheet". [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf

[25] "FOD814 Datasheet". [Online]. Available: http://www.mouser.com/ds/2/149/FOD817-194951.pdf

[26] "RGB LCD Shield Kit- Adafruit". [Online]. Available: https://www.adafruit.com/product/714

[27] "Librería LCD Adafruit". [Online]. Available: https://github.com/adafruit/Adafruit-RGB-LCD-Shield-Library

[28] "LiquidCrystal Library". [Online]. Available: https://www.arduino.cc/en/Reference/LiquidCrystal

[29] "Tutorial de manejo de librería LCD Adafruit". [Online]. Available: https://learn.adafruit.com/rgb-lcd-shield/using-the-rgb-lcd-shield

[30] "SPI Introduction". [Online]. Available: http://www.mct.net/faq/spi.html

[31] "TFT Guide - Arduino". [Online]. Available: https://www.arduino.cc/en/Guide/TFT

[32] "TFT Library - Arduino". [Online]. Available: https://www.arduino.cc/en/Reference/TFTLibrary

[33] "Adafruit motor selection guide". [Online]. Available: https://learn.adafruit.com/adafruit-motor-selection-guide?view=all

[34] "Motor Serie - EcuRed". [Online]. Available: https://www.ecured.cu/Motor_serie

[35] "Motor sin escobillas- EcuRed". [Online]. Available: https://www.ecured.cu/Motores_sin_escobillas

[36] "Motores paso a paso. Características básicas". [Online]. Available: http://robots-argentina.com.ar/MotorPP_basico.htm

[37] "PCA9685 datasheet". [Online]. Available: https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf

[38] "TB6621 datasheet". [Online]. Available: https://cdn.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf

[39] "Adafruit motor shield v2 Tutorial". [Online]. Available: https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino?view=all#install-software

[40] "Puerto Serie - EcuRed". [Online]. Available: https://www.ecured.cu/Puerto_serie

[41] "Datasheet MAX3232". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/1407/0900766b81407400.pdf

[42] "Datasheet REC7.5". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/135e/0900766b8135ee03.pdf

[43] "Datasheet REC8". [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/0f8d/0900766b80f8dda4.pdf

DOCUMENTO Nº2:

ANEXO

CÓDIGOS PROGRAMACIÓN

Índice del Anexo

1. CONTENIDO..... 1

2. MÓDULO 24V 2

3. MÓDULO LCD. 3

4. PANTALLA TFT. 4

1. CONTENIDO

En este anexo se recogen los códigos necesarios para testear cada uno de los módulos que contiene el entrenador. Estos códigos de ejemplo pueden ser utilizados como base para la creación de futuros proyectos.

Así pues, para testear la pantalla TFT, se ha creado un pequeño menú que se controla mediante el Joystick.

Para testear la Pantalla LCD y los botones, se utiliza un pequeño sketch mediante el cual se leen los botones y se escribe en la pantalla LCD que botón se ha pulsado.

Además se utilizan los Leds y potenciómetros y demás elementos, para aprovechar y comprobar su correcto funcionamiento.

2. Módulo 24V

```
/*
Ejemplo de utilización del MCP23017 de Microchip.
Utilizado en el módulo de 24V
En el ejemplo se leen entradas del puerto A y se escriben en el B.
La dirección I2C del módulo es 0x27.
*/

#include "Wire.h" // se incluye la librería
byte lectura=0; // se crea la variable donde se guardaran las lecturas


void setup()
{
// Se activa el BUS I2C
  Wire.begin();
// Se configura el puerto B como salidas
  Wire.beginTransmission(0x27);
  Wire.write(0x01);
  Wire.write(0x00);
  Wire.endTransmission();
}

void loop()
{
  // lee las entradas del puerto A
  Wire.beginTransmission(0x27);
  Wire.write(0x12);
  Wire.endTransmission();
  Wire.requestFrom(0x27, 1);
  lectura=Wire.read();

  // Escribe las entradas en el puerto de salidas
  Wire.beginTransmission(0x27);
  Wire.write(0x13);
  Wire.write(lectura);
  Wire.endTransmission();
// Espera un tiempo para volver a leer
  delay(200);
}
```

3. Módulo LCD.

```
/*********************

Prueba del correcto funcionamiento de la pantalla LCD y los botones
que forman el 'Módulo LCD'
Además enciende los Leds con la pulsación de cada boton.
**********************/

// Incluir las librerias
#include <Wire.h>
#include <Adafruit_RGBLCDShield.h>
#include <utility/Adafruit_MCP23017.h>



//Instancia de LCD de la libreria.
Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();

// Estas constantes se utilizan para cambiar el color del fondo de pantalla
.
// (la pantalla del entrenador es de solo un color, elegiremos blanco siemp
re)
#define RED 0x1
#define YELLOW 0x3
#define GREEN 0x2
#define TEAL 0x6
#define BLUE 0x4
#define VIOLET 0x5
#define WHITE 0x7

void setup() {

  //Configurar los pines de los LEDs como salidas:
  pinMode(26, OUTPUT); pinMode(30, OUTPUT); pinMode(34, OUTPUT);
pinMode(38, OUTPUT);


  // Configura el numero de columnas y filas de la pantalla:
  lcd.begin(20, 4);
  lcd.print("Hola Mundo");
  lcd.setBacklight(WHITE);
}

void loop() {

  // Posiciona el cursor en la cuarta fila,(la primera es la 0)
  lcd.setCursor(0, 3);
  // Escribe los nombres de los botones encima de estos.
  lcd.print("F1    F2   F3   F4");
  // Se lee el estado de los botones.
  uint8_t botones = lcd.readButtons();
  // si se ha leido algo en los botones
  if (botones) {
    // se borra la pantalla
    lcd.clear();
    // se sitúa el cursor en el origen.
```

```
    lcd.setCursor(0, 0);

    // Dependiendo del boton que se pulse se escribe el correspondiente
texto en pabtalla, y se enciende el led determinado.
    if (botones & BUTTON_UP) {
      lcd.print("Has pulsado F4");
      // invertir estado del LED
      digitalWrite( 26,!digitalRead(26)) ;
    }

    if (botones & BUTTON_DOWN) {
      lcd.print("Has pulsado F3 ");
      digitalWrite( 30,!digitalRead(30)) ;
    }

    if (botones & BUTTON_RIGHT) {
      lcd.print("Has pulsado F2 ");
      digitalWrite( 34,!digitalRead(34)) ;
    }

    if (botones & BUTTON_SELECT) {
      lcd.print("Has pulsado F1 ");
      digitalWrite( 38,!digitalRead(38)) ;
    }
  }
}
```

4. Pantalla TFT.

```
/*
  Menú en TFT Robot LCD

  Programa que despliega un menú en un display TFT modelo ROBOT LCD y te pe
rmite moverte
  por él mediente un joystick.

  Configurado con una placa Arduino Mega

  El programa consta de 4 funciones principales:

    > pintarMenus

    > pintarflecha

    > selectMenuPrincipal

    > selectSubmenu


  Autor: Raúl Sainz-Maza Serna.
 */


// Las librerías deben cargarse en este orden.
```

```
#include <SPI.h>  // Comunicación SPI.
#include <SD.h>   // Librería manejo SD.
#include <TFT.h>  // Librería LCD de Arduino.



// Definicion de constantes globales
#define CS_LD   49
#define DC_LD   46
#define rst     48
#define MISO    50
#define SCK     52
#define MOSI    51

#define subir   23      // Boton movimiento menús subir.
#define bajar   27      // Boton movimiento menús bajar.
#define select  31      // Boton seleccion de opción.
#define LED     38      // LED externo para encender o apagar desde
submenu.

// Definición variables globales:
char sensorPrintout[4];  // array para la lectura de señal analógiaca
int flecha=1;            // variable selección en menú
bool submenu=false;      // variable determina si estás en submenú o en
menú principal
int colR=255;            // variable color rojo
int colG=255;            // variable color verde
int colB=255;            // variable color azul

// Crear instancia de la libreria TFT.
TFT TFTscreen = TFT(CS_LD, DC_LD, rst);

//Configuración Inicial
void setup() {


  //Setup I/O
  pinMode(subir,INPUT_PULLUP);                  //Entrada pulsador subir
  pinMode(bajar,INPUT_PULLUP);                  //Entrada pulsador bajar
  pinMode(select,INPUT_PULLUP);                 //Entarda pulsador
seleccionar opcion
  pinMode(LED,OUTPUT);digitalWrite(LED,LOW);    //Salida LED

  //probar LED:

  digitalWrite(LED,HIGH);delay(2000);digitalWrite(LED,LOW);


  //Iniciar la pantalla.
  TFTscreen.begin();

  // Poner fondo en negro
  TFTscreen.background(0, 0, 0);
  // Seleccionar el color de letra en blanco.
  TFTscreen.stroke(colR, colG, colB);

  // Pinta el menu principal y la flecha de selección.
  pintarMenus(submenu);
  pintarflecha();
}
```

```
// ******** EMPIEZA LOOP **********
void loop() {

  // Mover flecha menú principal:
  if (!digitalRead(bajar) && flecha<5 && submenu==false){flecha++;
pintarflecha(); delay(150);}
  if (!digitalRead(subir) && flecha>1 && submenu==false){flecha--;
pintarflecha(); delay(150);}
  // Mover flecha menú secundario:
  if (!digitalRead(bajar) && flecha<3 && submenu==true){flecha++;
pintarflecha(); delay(150);}
  if (!digitalRead(subir) && flecha>1 && submenu==true){flecha--;
pintarflecha(); delay(150);}


  // Ejecutar selección Menú principal
  if (!digitalRead(select) && (submenu==false)){
    selectMenuPrincipal();
  }

  // Ejecutar selección Submenú
  if (!digitalRead(select) && (submenu==true)){
    selectSubmenu();
  }

}
// ******* FIN LOOP ********


// ***Pinta el menú o submenú en función de la variable submenú***
void pintarMenus(bool sub){

  TFTscreen.stroke(colR, colG, colB);
  // Menú principal.
  if (sub==false){

    // Poner fondo en negro.
    TFTscreen.background(0, 0, 0);

    //TÍTULO.
    TFTscreen.setTextSize(3);         // selecciona el tamaño de la fuente.
(3 --> 30 pixeles cada letra)
    TFTscreen.text("Menu", 50, 5);    // escribir el título del menu
principal en la esquina superior izquierda.
    TFTscreen.line(0,30,160,30);      // dibujar una raya debajo del título
a modo de subrayado. "screen.line(xStart, yStart, xEnd, yEnd); "
    TFTscreen.setTextSize(1);         // Bajar el tamaño de la fuente.

    //Mostrar menú principal:
    TFTscreen.text("1.Escribir 'HOLA MUNDO' ",20, 40);
    TFTscreen.text("2.Escribir A0",20,55);
    TFTscreen.text("3.Entrar Sub-menu",20,70);
    TFTscreen.text("4.Vacio",20,85);
    TFTscreen.text("5.Sobre este dispositivo",20,100);
  }

  // Sub menú.
   else if (sub==true){

    TFTscreen.background(0, 0, 0);     // Poner fondo en negro
```

```
    TFTscreen.setTextSize(3);          // Selecciona el tamaño de la
fuente. (3 --> 30 pixeles cada letra)
    TFTscreen.text("Submenu", 20, 5);  // Escribir el título del menu
principal en la esquina superior izquierda.
    TFTscreen.line(0,30,160,30);       // Dibujar una raya debajo del
título a modo de subrayado. "screen.line(xStart, yStart, xEnd, yEnd); "
    TFTscreen.setTextSize(1);          // Bajar el tamaño de la fuente.


    //Mostrar submenú:
    TFTscreen.text("1.Encender/Apagar LED ",20, 45);
    TFTscreen.text("2.Cambiar color letras",20,60);
    TFTscreen.text("3.SALIR",20,75);
   }

  // Si no es ninguna de los dos, muestro señal de error
  else {

    TFTscreen.background(0, 0, 0);
    TFTscreen.setTextSize(3);
    TFTscreen.text("ERROR \n ", 20, 0);
    TFTscreen.line(0,35,160,40);

  }
}



//*** Pinta la flecha de selección***

void pintarflecha(void){

  TFTscreen.setTextSize(1);
  // Selecciona negro para borrar flechas anteriores.
  TFTscreen.stroke(0, 0, 0);
  // Borra flechas anteriores:
  TFTscreen.text(">",5, 40);
  TFTscreen.text(">",5, 55);
  TFTscreen.text(">",5, 70);
  TFTscreen.text(">",5, 85);
  TFTscreen.text(">",5, 100);
  // Selecciona el color deseado para pintar flecha:
  TFTscreen.stroke(colR, colG, colB);

  // Pintar flecha de seleccion de menús:
  switch(flecha){
    //Pinta flecha:
    case 1:
      TFTscreen.text(">",5, 40);
    break;
    //Pinta flecha:
    case 2:
      TFTscreen.text(">",5, 55);
    break;
    //Pinta flecha:
    case 3:
      TFTscreen.text(">",5, 70);
    break;
    //Pinta flecha:
    case 4:
      TFTscreen.text(">",5, 85);
```

```
    break;
    //Pinta flecha:
    case 5:
      TFTscreen.text(">",5, 100);
    break;
  }
}



// ****Ejecuta la selección en el menu principal***

void selectMenuPrincipal(void){

    String sensorVal;
    int i=0;
    switch (flecha){

      // Escribir hola mundo.
      case 1:
        TFTscreen.background(0, 0, 0);       // poner fondo en negro
        TFTscreen.stroke(colR, colG, colB); // seleccionar el col de letra
        TFTscreen.setTextSize(3);                 // selecciona el tamaño de
la fuente. (3 --> 30 pixeles cada letra)
        TFTscreen.text("HOLA MUNDO ", 0, 0);    // escribir el título del
menu principal en la esquina superior izquierda.
        TFTscreen.setTextSize(1);                 // Bajar el tamaño de la
fuente.

        // no hacer nada hasta que no se pulse select
        TFTscreen.text("*Pulsa select para salir", 0, 110);
        while(digitalRead(select)){};
        pintarMenus(submenu);
        pintarflecha();
      break;

      //Escribir valor de A0
      case 2:

        delay(250);         //esperar para no coger rebotes del pulsador

        TFTscreen.background(0, 0, 0);
        // repetir la muestra del valor analógico hasta que se pulse select
        while(digitalRead(select)){

        // escribir valor de entrada A0
        sensorVal = String(analogRead(A0));
        // convert the reading to a char array
        sensorVal.toCharArray(sensorPrintout, 4);

        // col de fuente
        TFTscreen.stroke(colR, colG, colB);
        // print the sensor value
        TFTscreen.text("A0 vale:\n ", 0, 0);
        TFTscreen.setTextSize(3);
        TFTscreen.text(sensorPrintout, 0, 20);
        delay(200);
        // Erase the text you just wrote
        TFTscreen.stroke(0, 0, 0);
        TFTscreen.text(sensorPrintout, 0, 20);
        TFTscreen.setTextSize(1);
```

```
        TFTscreen.stroke(colR, colG, colB);
        TFTscreen.text("*Pulsa select para salir", 0, 110);
        }
        pintarMenus(submenu);
        pintarflecha();
      break;

      // Entrar en submenú
      case 3:
        submenu=true; flecha=1;
        pintarMenus(submenu);
        pintarflecha();
      break;

      // Leer TarjetaSD
      case 4:

        // volvemos a pintar los menús.
        pintarMenus(submenu);
        pintarflecha();
      break;

      // Mostrar información de la pantalla
      case 5:
       // poner fondo en negro
        TFTscreen.background(0, 0, 0);
        // col de fuente
        TFTscreen.stroke(colR, colG, colB);
        // tamaño
        TFTscreen.setTextSize(1);
        // Escribir algo sobre el dispositivo
        TFTscreen.text(" Pantalla: ROBOT LCD \n Tamanio: 160 x 128 pixeles
\n Made in Italy \n ARDUINO.ORG \n ", 0, 20);

        TFTscreen.text("*Pulsa select para salir", 0, 110);
        // no hacer nada hasta que no se pulse select
        while(digitalRead(select)){};
        pintarMenus(submenu);
        pintarflecha();
      break;

    }
}


//*** Ejecuta la seleccion del sumbmenu***
void selectSubmenu(void){

    switch (flecha){
      case 1:
        // invertir estado del LED
        digitalWrite( LED,!digitalRead(LED)) ;
        pintarMenus(submenu);
        pintarflecha();
      break;
      case 2:
        // cambiar de color
        // si estaba en blnco, poner rojo
        if (colG>0){colR=255;colG=0;colB=0;}
        // si estaba en rojo, poner blanco:
```

```
        else{colR=255;colG=255;colB=255;}

        pintarMenus(submenu);
        pintarflecha();
      break;
      case 3:
      //Salir al menú principal.
        submenu=false; flecha=1;
        // Pinta los menus
        pintarMenus(submenu);
        pintarflecha();
      break;
    }
}



/*
 // Esta funcion dibuja las flechas mediante líneas. por si se quisiera dar
 otra forma a la flecha de selección.
void pintarflecha(void){

   TFTscreen.stroke(colR, colG, colB);
  // Pintar flecha de seleccion de menús:


    // "screen.line(xStart, yStart, xEnd, yEnd); "
    // Borra flechas anteriores:
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.line(5,40,15,45);
    TFTscreen.line(5,50,15,45);
    TFTscreen.line(5,55,15,60);
    TFTscreen.line(5,65,15,60);
    TFTscreen.line(5,70,15,75);
    TFTscreen.line(5,80,15,75);
    TFTscreen.line(5,85,15,90);
    TFTscreen.line(5,95,15,90);
    TFTscreen.line(5,100,15,105);
    TFTscreen.line(5,110,15,105);

    // Pinta flecha con el color deseado:
    TFTscreen.stroke(colR, colG, colB);

    switch(flecha){
      case 1:

        TFTscreen.line(5,40,15,45);
        TFTscreen.line(5,50,15,45);
      break;
      case 2:

        TFTscreen.line(5,55,15,60);
        TFTscreen.line(5,65,15,60);
      break;
      case 3:

        TFTscreen.line(5,70,15,75);
        TFTscreen.line(5,80,15,75);
      break;
      case 4:
```

```
        TFTscreen.line(5,85,15,90);
        TFTscreen.line(5,95,15,90);
      break;
      case 5:

        TFTscreen.line(5,100,15,105);
        TFTscreen.line(5,110,15,105);
      break;
  }
}
*/
```

DOCUMENTO Nº3:

PLANOS

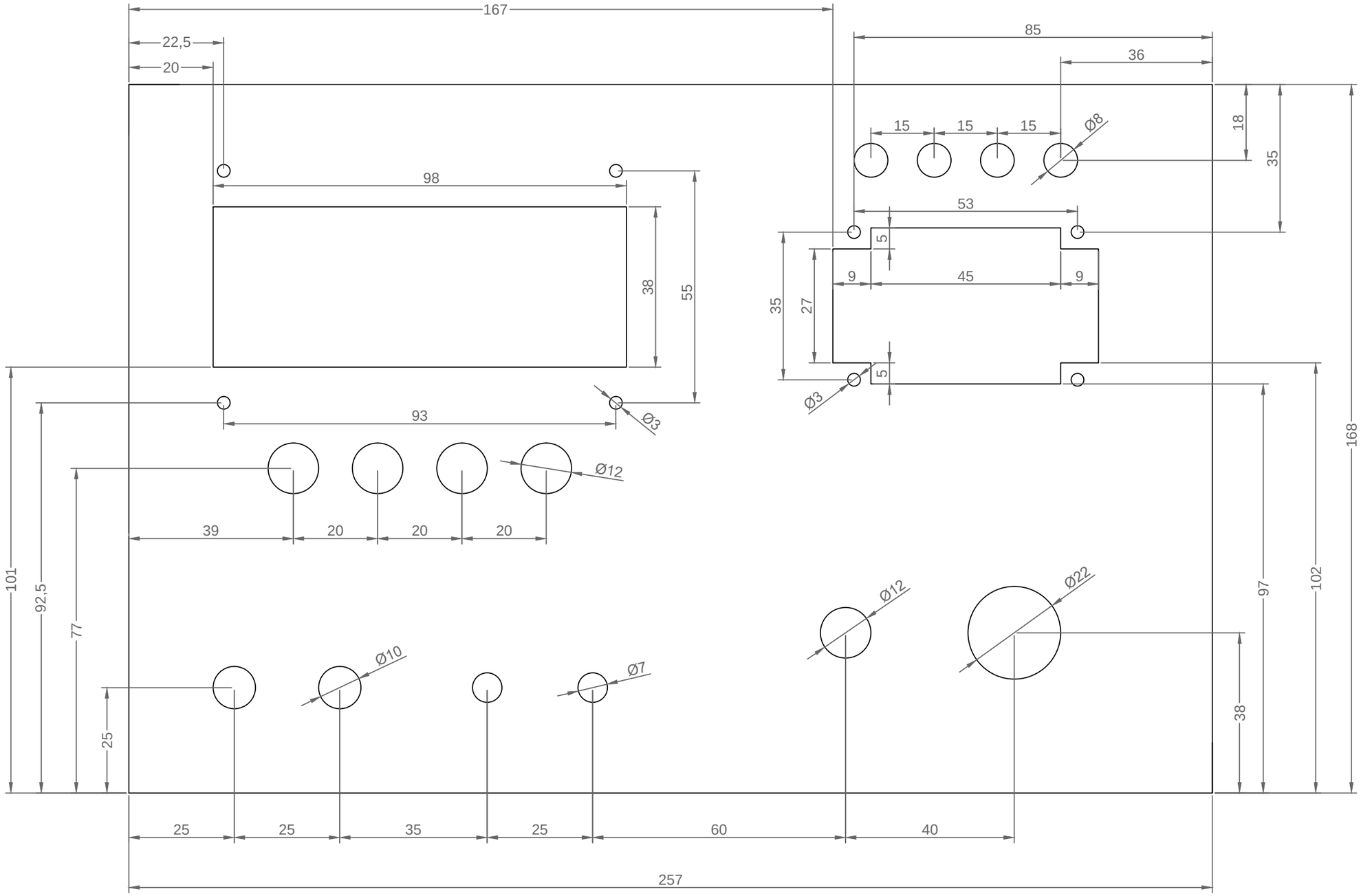
Índice de Planos

Planos Mecánicos:

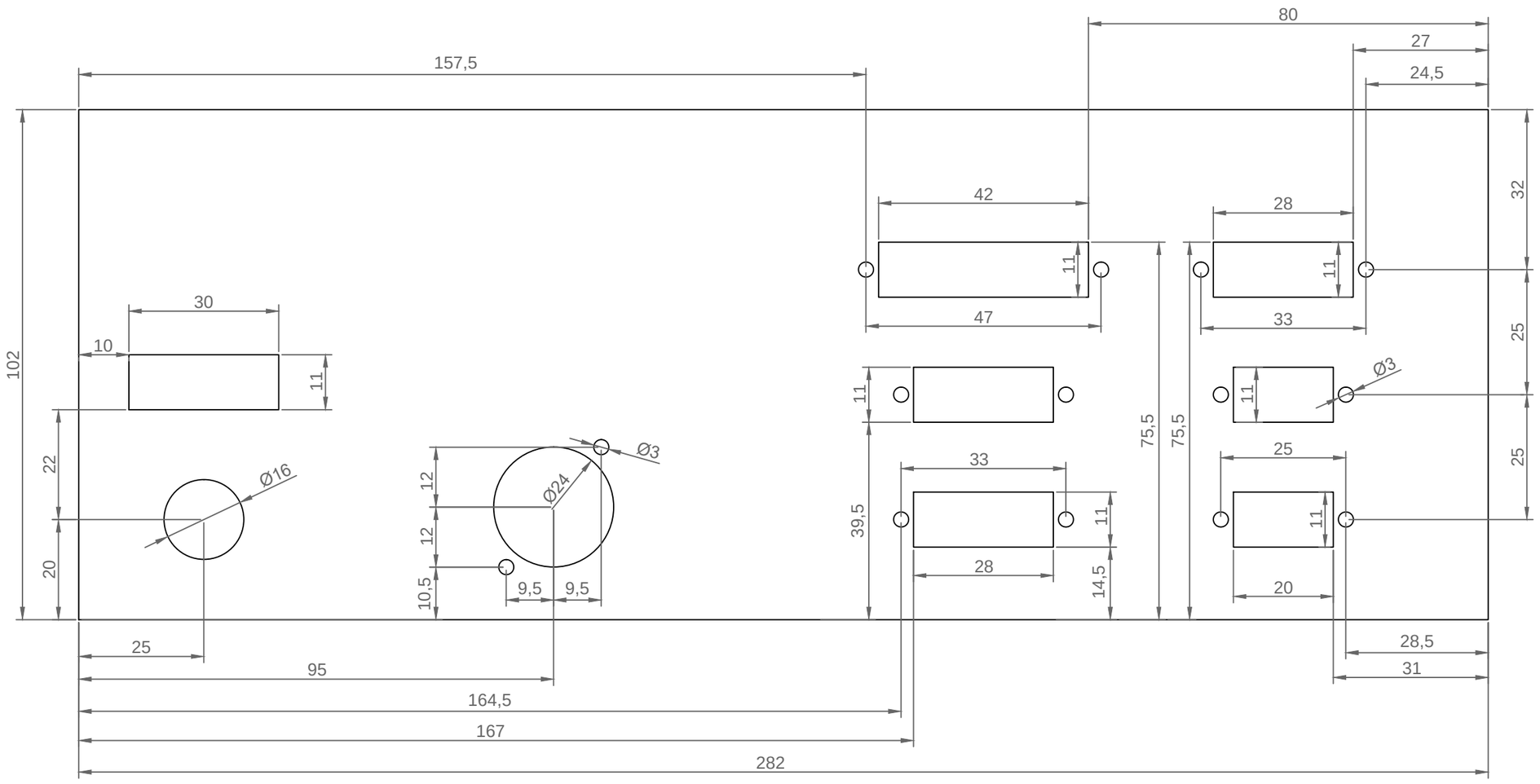
Plano 1: Panel Frontal…… 1
Plano 2: Panel Trasero…… 2

Esquemáticos:

Plano 3: General 1…… 3
Plano 4: General 2…… 4
Plano 5: Módulo 24V…… 5
Plano 6: Módulo LCD…… 6
Plano 7: Módulo Motores…… 7
Plano 8: Alimentación…… 8

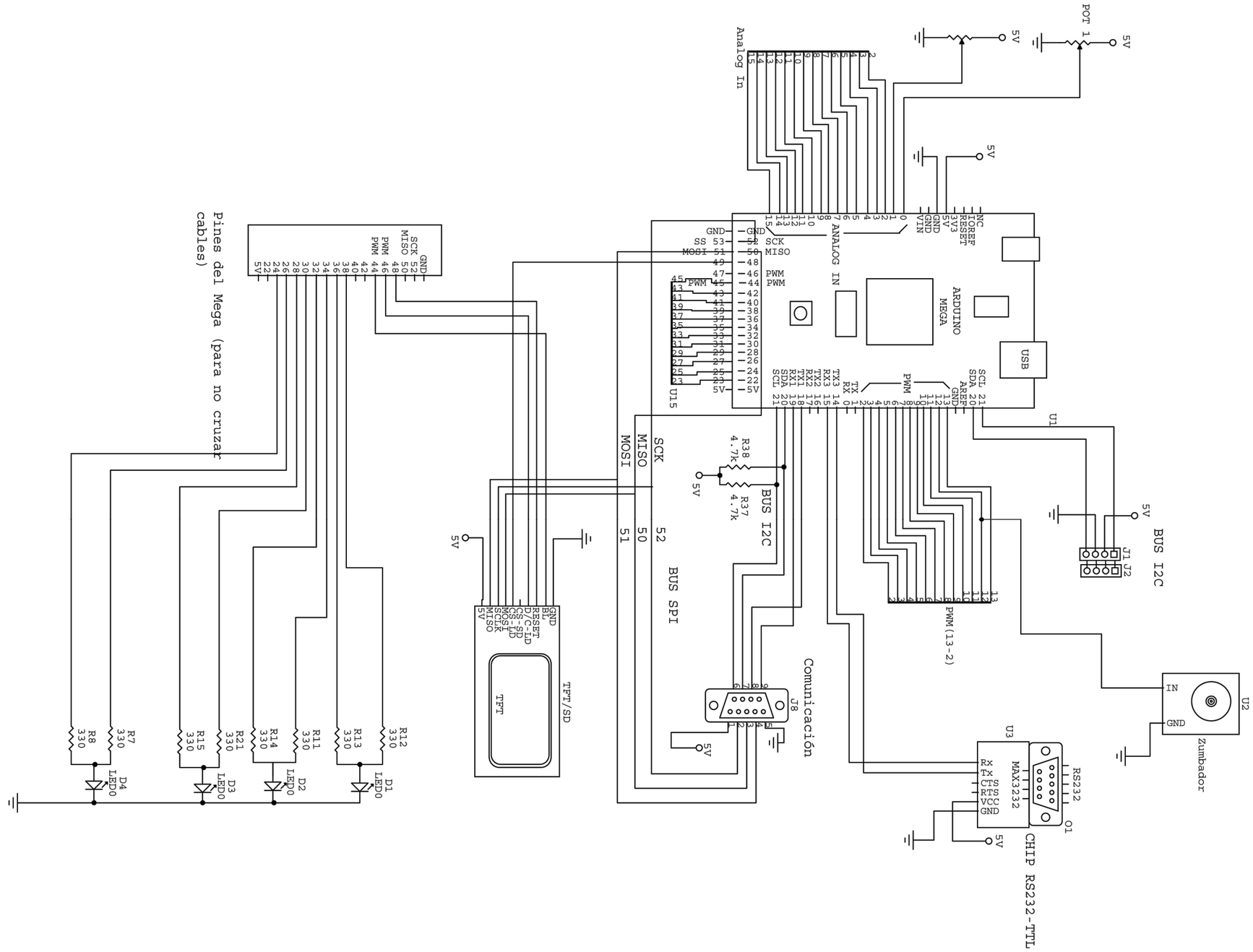


Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Plano de detalle	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: Panel Frontal	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
Escala 1:1		Fecha 3 Junio 2017 / Nº de Plano 01	Hoja 1/8

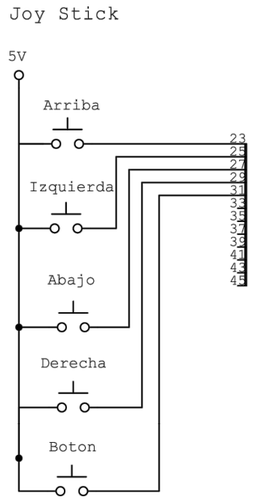


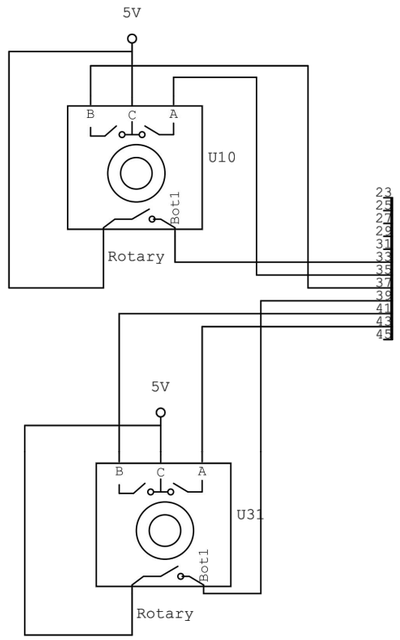
Escala 1:1

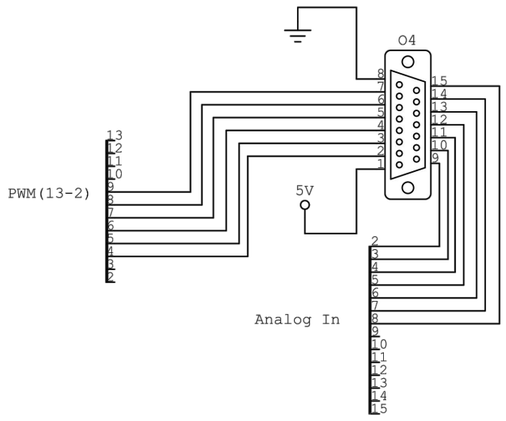
Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Plano de detalle	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: Panel Trasero	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
		Fecha 3 Junio 2017 / Nº de Plano 02	Hoja 2/8



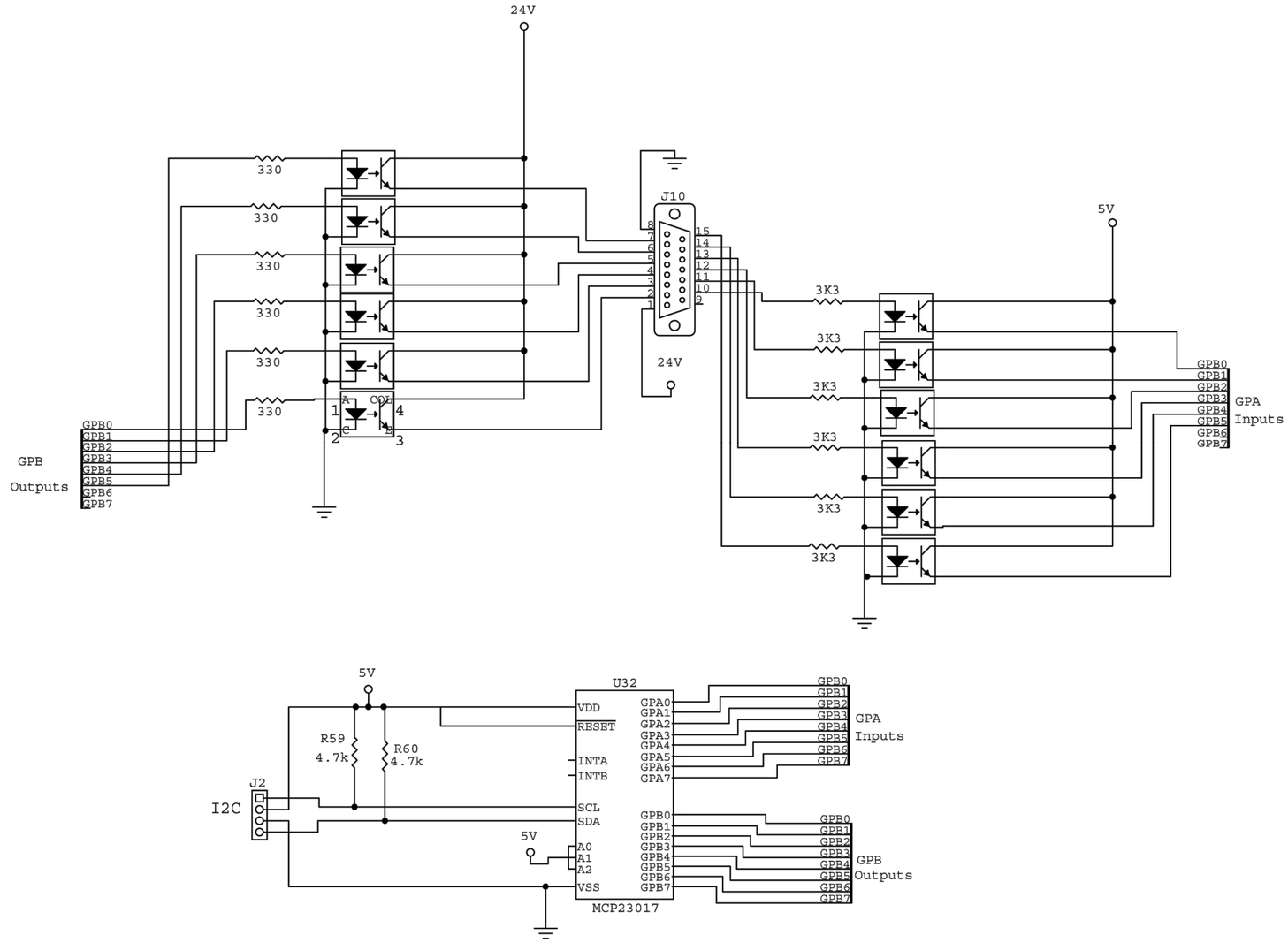
Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Esquemático	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: General 1	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
		Fecha 3 Junio 2017 / Nº de Plano 03	Hoja 3/8



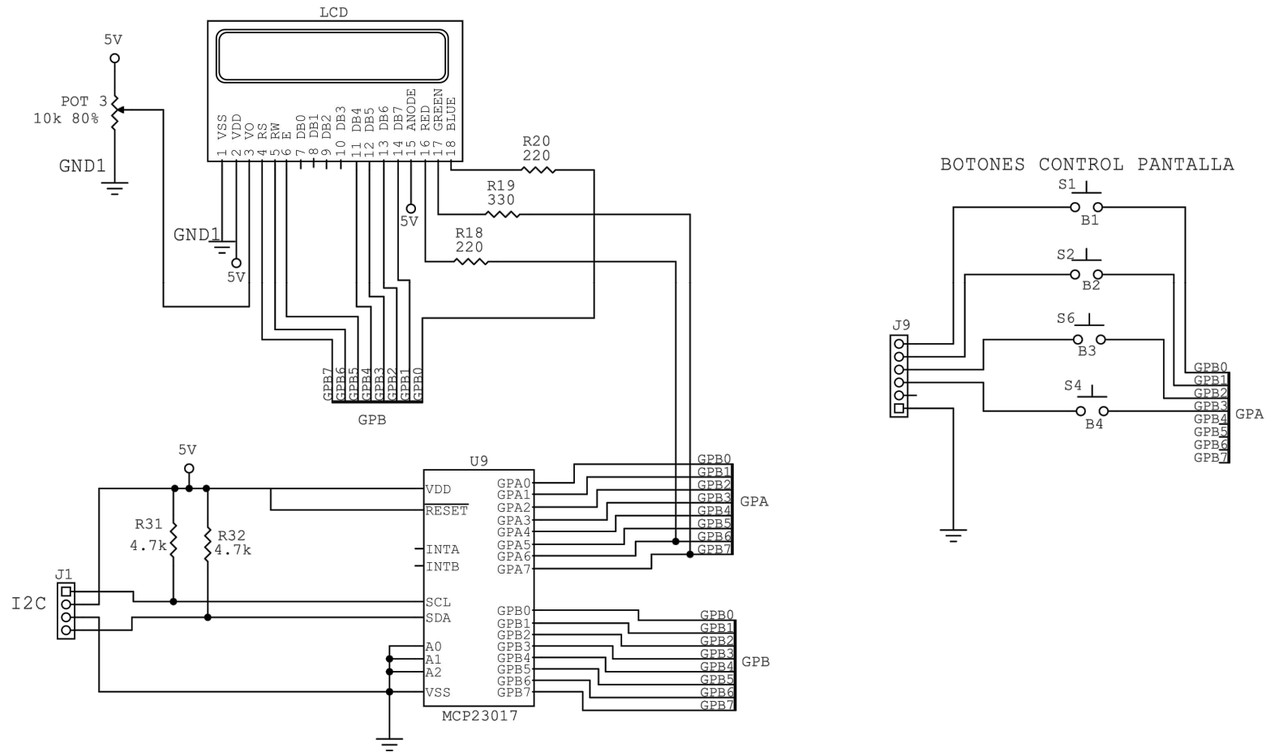




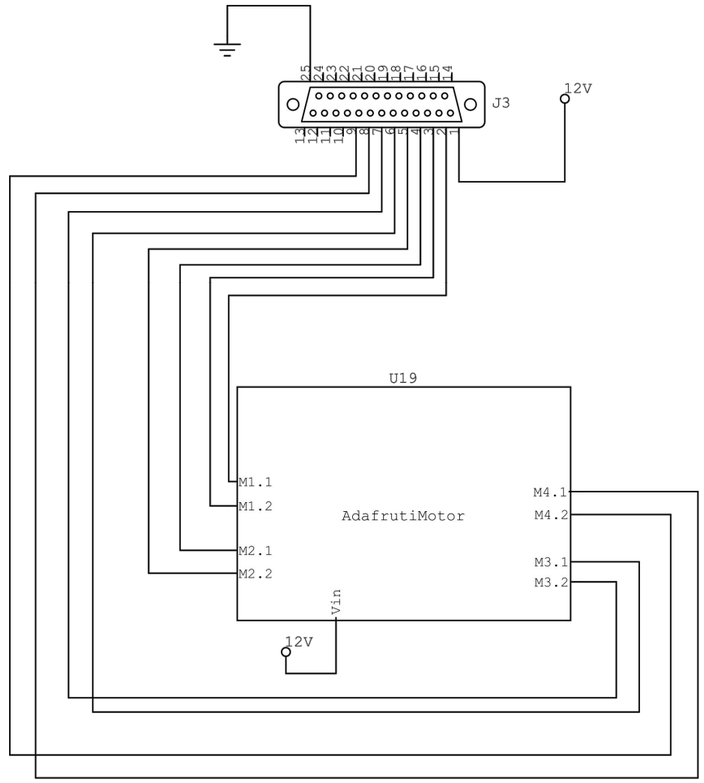
Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Esquemático	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: General 2	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
		Fecha 3 Junio 2017 / Nº de Plano 04	Hoja 4/8



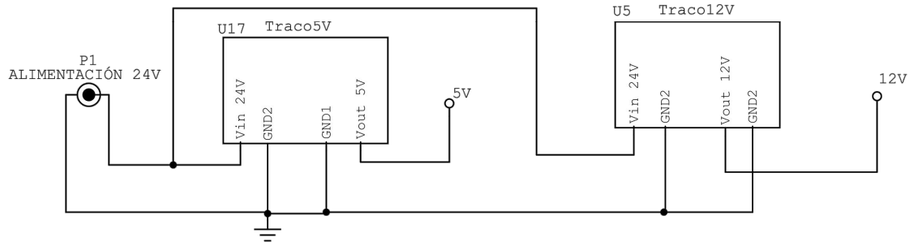
Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Esquemático	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: Módulo 24V	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
		Fecha 3 Junio 2017 / Nº de Plano 05	Hoja 5/8



Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Esquemático	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: Módulo LCD	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
		Fecha 3 Junio 2017 / Nº de Plano 06	Hoja 6/8



Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Esquemático	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: Módulo Motores	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
		Fecha 3 Junio 2017 / Nº de Plano 07	Hoja 7/8



Grado en Ing.Electrónica Indus. y Aut.	Tipo de documento: Esquemático	Creado por: Raúl Sainz-Maza Serna	
E.T.S.I Industriales y T.	Título. Título suplementario: Alimentación	Aprobado por: Miguel Ángel Allende	Rev. 02
		Referencia técnica: Miguel Ángel Allende	Idioma Es
		Fecha 3 Junio 2017 / Nº de Plano 08	Hoja 8/8

DOCUMENTO Nº4:

PLIEGO DE CONDICIONES

ÍNDICE DEL PLIEGO DE CONDICIONES

1 DISPOSICIONES GENERALES 1

1.1 RESUMEN DEL PROYECTO 1

1.2 ALCANCE Y APLICABILIDAD DEL PLIEGO DE CONDICIONES 1

2 CONDICIONES TÉCNICAS 2

2.1 CARACTERISTICAS DEL ENTRENADOR DE ARDUINO 2

2.2 CONDICIONES DE EJECUCIÓN 2

2.3 CONDICIONES DE MONTAJE 2

2.4 PRECAUCIONES DE USO 2

3 CONDICIONES LEGALES 3

3.1 USOS PERMITIDOS 3

3.2 PROPIEDAD INTELECTUAL 3

3.3 SEGURIDAD E HIGIENE 3

4 CONDICIONES ECONOMICAS. 4

1 DISPOSICIONES GENERALES

1.1 RESUMEN DEL PROYECTO

El proyecto consiste en el diseño y fabricación de un sistema de entrenamiento mediante el cual se facilite la tarea de enseñanza y de testeo de aplicaciones basadas en la tecnología Arduino. El campo de aplicación de dicha tecnología no suele ser la industria, sin embargo, existen muchas situaciones de índole industrial, en las que el uso de esta tecnología si es conveniente y realmente útil. Es por ello que el entrenador constará de una serie de módulos con funciones aplicables a la industria.

El sistema se compone de elementos básicos como botones, pulsadores o leds, con los cuales se puede empezar a aprender a programar desde los niveles más bajos. Pero también recoge pantallas y módulos más complicados de controlar, con los que se puede aprender niveles más avanzados.

La mayor parte de los módulos se comunican entre sí con el estándar I2C, el cual cada vez se utiliza más para la comunicación de dispositivos electrónicos en cortas distancias. Sin embargo, también se utilizan buses con otro tipo de estándar de comunicación como el SPI o el RS232.

El sistema se alimenta a 24V, ya que es un estándar de tensión muy utilizado en la industria. Se puede alimentar tanto desde una fuente de tensión, como desde cualquier toma de 24V.

1.2 ALCANCE Y APLICABILIDAD DEL PLIEGO DE CONDICIONES

El Pliego de Condiciones establece los mínimos legales que se deben satisfacer para la ejecución del proyecto.

Las condiciones recogidas en este documento se entenderán aplicables al ámbito del diseño, fabricación y verificación del sistema, así como de todas las tareas internas o externas derivadas de la ejecución.

2 CONDICIONES TÉCNICAS

2.1 CARACTERISTICAS DEL ENTRENADOR DE ARDUINO

El sistema está diseñado para que se pueda alimentar mediante la conexión a un conector de sensor industrial estándar el cual se alimenta a 24V. Así pues, también se puede realizar la alimentación del mismo mediante una fuente de alimentación que simule estas condiciones.

Los materiales, a su vez, podrán ser sometidos a diversas pruebas y ensayos para asegurar su calidad y su correcto funcionamiento. Si son reemplazados por unos nuevos, estos deberán tener las mismas características que los originales.

2.2 CONDICIONES DE EJECUCIÓN

El proyecto debe ejecutarse de acuerdo al diseño descrito anteriormente tanto en la Memoria como en los Planos. Cualquier variación parcial puede conducir a un funcionamiento erróneo del sistema o alguno de sus módulos.

2.3 CONDICIONES DE MONTAJE

La dirección del montaje estará realizada en su totalidad por el ingeniero o proyectante, atendiendo a los documentos y planos presentes en apartados previos.

Si es necesario realizar una modificación, se realizará bajo el pertinente consentimiento del propio ingeniero o proyectante.

2.4 PRECAUCIONES DE USO

Tanto la placa Arduino como los componentes conectados a ella, no pueden recibir tensiones por encima de las establecidas como máximas para la alimentación. En caso contrario, podría acortar la vida útil o incluso destruir el sistema o alguno de sus módulos

Si para el montaje y puesta en marcha de los sistemas y dispositivos que componen el entrenador se siguen todas las indicaciones, recomendaciones y especificaciones que se dan a lo largo de los diferentes apartados de este documento, la vida útil de los elementos estará supeditada a aquella que dé el fabricante siguiendo sus recomendaciones de mantenimiento.

El deterioro de los diferentes componentes puede ser debido al propio envejecimiento del material con el paso del tiempo.

3 CONDICIONES LEGALES

3.1 USOS PERMITIDOS

El proyecto establece el diseño de un sistema de entrenamiento basado en la tecnología Arduino. Su carácter es de prototipo, por lo que carece de los correspondientes permisos para ser utilizado de manera formal en aplicaciones industriales finales.

El sistema podrá ser utilizado en pruebas experimentales en un entorno controlado de laboratorio técnico. En todo caso, se deberán respetar sus características técnicas y satisfacer las precauciones de uso establecidas anteriormente.

El uso indebido del producto será responsabilidad única y directa del usuario, careciendo de responsabilidad civil ni penal el proyectante.

3.2 PROPIEDAD INTELECTUAL

La propiedad intelectual corresponde tanto al proyectante como a su institución. Asimismo, las partes que integran el diseño y que son origen de trabajos externos, serán propiedad intelectual de su autor original.

3.3 SEGURIDAD E HIGIENE

Las disposiciones mínimas de Seguridad e Higiene para el uso de los diferentes equipos de trabajo por los trabajadores se establecen en el real Decreto 1215/1997 y es un componente fundamental de la normativa de Seguridad y Salud en el trabajo, la cual es encabezada por la ley de Prevención de Riesgos Laborales.

4 CONDICIONES ECONOMICAS.

El producto resultante del presente proyecto tiene carácter de prototipo, por lo que no está disponible para su comercialización.

En cambio, se podrá valorar una futura explotación del diseño, siempre y cuando sea con fines de investigación y para mejorar el producto. Las partidas presupuestarias que pudieran surgir de la misma se definirán dentro de ese futuro marco de investigación.

DOCUMENTO Nº5:

PRESUPUESTO

Índice del Presupuesto

1 COSTES DIRECTOS 1

1.1 MANO DE OBRA DIRECTA 1

1.2 MATERIAS PRIMAS 1

1.3 PUESTO DE TRABAJO 2

2 COSTES INDIRECTOS 3

2.1 MANO DE OBRA INDIRECTA 3

3 COSTE TOTAL DEL PROYECTO 3

Índice de cuadros

Cuadro 1.1: Mano de obra directa..... 1
Cuadro 1.2: Materias Primas. 2
Cuadro 1.3: Software 2
Cuadro 1.4: Herramientas 2
Cuadro 2.1: Mano de obra indirecta..... 3
Cuadro 3.1: Coste total. 3

1 COSTES DIRECTOS

1.1 MANO DE OBRA DIRECTA

Los costes de mano de obra directa corresponden al salario de todas aquellas personas relacionadas directamente con la elaboración del proyecto.

	Sueldo/hora	Nº Horas	Total
Ingeniero	16 €	320	5120 €
Coste total de mano de obra directa			**5120 €**

Cuadro 1.1: Mano de obra directa

1.2 MATERIAS PRIMAS

Para la fabricación del prototipo es necesario utilizar materias primas, las cuales se obtuvieron del distribuidor de electrónica RS. Se aporta el código RS, para facilitar la localización de los mismos.

Referencia RS	Descripción	Coste/Unidad	Unidades	Total
877-1154	Conector alimentación	10,30€	1	**10,3 €**
815-8466	Latiguillo USB	1,36€	1	**1,36 €**
916-0227	Adaptador USB	9,44€	1	**9,44 €**
733-1146	Convertidor dc/dc 24-12V	23,94€	1	**23,94**
494-4128	Convertidor dc/dc 24-5V	20,98€	1	**20,98**
734-6704	Interruptor de Botón	2,84€	5	**14,2**
210-765	Indicador LED	4,44€	4	**17,76**
707-7622	Resistencia 330	0,121€	10	**1,21**
739-0265	Optoacoplador FOD817A300	0,338€	10	**3,38**
850-9643	Resistencia variable	0,47€	1	**0,47**
403-806	MCP23017	1,118€	5	**5,59**
303-0161	Zumbador magnético	0,58€	1	**0,58**
882-9060	MAX3232	9,15€	1	**9,15**
769-7399	Placa Protoshild	4,17€	1	**4,17**
715-4084	Arduino Mega	43,51€	1	**43,51**
905-4618	Adafruit Motor shield	21,59€	1	**21,59**
681-2938	Pines 90º paso 2.54mm	0,682€	5	**3,41**
156-049	Pines rectos paso 2.54mm	0,394€	10	**3,94**
549-0026	Conector hembra paso 2.54mm	3,76€	5	**18,8**
161-9330	Caja de sobremesa ABS	34,62€	1	**34,62**
161-9346	Panel frontal	22,25€	1	**22,25**
119-9139	Panel trasero	11,23€	1	**11,23**
825-2596	Interruptor de joystick	15,87€	1	**15,87**
790-4417	Potenciómetro con mando	10,45€	1	**10,45**
782-4585	Pantalla TFT	20,33€	1	**20,33**
532-6818	Display LCD	13,24€	1	**13,24**

737-7760	Rotary giratorio	1,17€	2	**2,34**
560-271	Tuerca M2	5,81€	1	**5,81**
482-9120	Tornillo M2	6,56€	1	**6,56**
287-7262	Interruptor Balancín	1,39€	1	**1,39**
467-6106	Mando potenciómetro	0,60€	10	**6**
748-2115	Cable para equipos 0.5mm2	15,82€	1	**15,82**
			Coste total de las materias primas	**379,69 €**

Cuadro 1.2: Materias Primas.

1.3 PUESTO DE TRABAJO

En este apartado se recogen los costes asociados al espacio de trabajo donde se realizan las diferentes actividades necesarias para llevar a cabo el proyecto.

Software	Coste
AutoCad	459,80
Circuit Maker 2000	0
Arduino IDE	0
	459,80 €

Cuadro 1.3: Software

Herramienta	Coste
Fresadora	4981
Multímetro	19,70
Estación soldadura	155
Maletín Herramientas	30,95
	5186,65 €

Cuadro 1.4: Herramientas

Se estima que la vida útil del proyecto es de 6 años, y que el tiempo de uso del puesto de trabajo es de 3 meses, con estos datos se obtendrá un coste de amortización calculado a continuación:

$$\boldsymbol{Coste\ amortización} = (5186{,}65 + 459{,}80) * \left(\frac{3\ meses}{72\ meses}\right) = \mathbf{235{,}27\ €}$$

2 COSTES INDIRECTOS

2.1 MANO DE OBRA INDIRECTA

Los costes de mano de obra indirecta corresponden al trabajo empleado en la áreas administrativas de la empresa, por el personal que no participa directamente en la transformación de la materia prima y que sirven de apoyo a la producción y el comercio consistente.

Se ha considerado un coste de un 3% respecto al de la mano de obra directa.

Descripción	Coste
Mano de obra indirecta	153,06
Total Coste de mano de obra indirecta	**153,06 €**

Cuadro 2.1: Mano de obra indirecta

3 COSTE TOTAL DEL PROYECTO

Costes	Coste
Costes Directos	**5734,96**
Mano de obra directa	5120,00€
Materias primas	379,69 €
Puesto de trabajo	235,27€
Costes indirectos	**153,06 €**
Mano de obra indirecta	153,06 €
COSTE TOTAL DEL PROYECTO	**5888,02 €**

Cuadro 3.1: Coste total.

El Coste Total del Proyecto asciende a **cinco mil ochocientos ochenta y ocho euros con dos céntimos.**