UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universitat Politècnica de València

Escuela Técnica Superior de Diseño Industrial

DISEÑO PLACA DE COMUNICACIÓN POR BUSES INDUSTRIALES PARA ARDUINO®

Autor: Pedro José Ante Espada

Tutor: Roberto Capilla Lladró

Co-tutor: Carlos Sánchez Díaz

Grado en Ingeniería Electrónica Industrial y Automática

Curso académico 2015/2016

Valencia, septiembre de 2016

Contenido

Índice de Figuras

Índice de Tablas

1. Resumen

El trabajo trata del desarrollo de una pasarela de comunicaciones industriales, donde la trama de tres datos se recibe por un tipo de comunicación y se envía por otro completamente diferente, la cual se va a transmitir a una maqueta de un almacén automatizado que mueve los palets o bultos a diferentes posiciones dependiendo de la orden enviada. Para el envío de la trama se utilizará tanto el puerto serie del ordenador como una SCADA creado a partir de una pantalla LCD.

2. Introducción y objetivos

2.1. Introducción.

En este trabajo se verán los diferentes tipos de comunicaciones más utilizados en el sector industrial, además de los protocolos de comunicación y sus características.

Por otra parte, se detallará el microcontrolador seleccionado para este trabajo y sus especificaciones, con el que se creará una pasarela que permita la unión entre dos tipos de comunicaciones diferentes. Para ello se definirá una trama y un protocolo a usar.

Así mismo se va a programar un SCADA mediante una pantalla LCD para el envío de las tramas al microcontrolador.

Para concluir se transmitirá la trama hasta una maqueta de un almacén inteligente, el cual realizará las órdenes dadas por el microcontrolador.

2.2. Objetivos.

- Se pretenderá ampliar el conocimiento sobre comunicaciones obtenido en la carrera.
- Crear una pasarela que permita unir dos tipos de comunicaciones.
- Programar un SCADA.

3. Comunicaciones

Para conectar dos partes de una maquinaria es necesario tener una comunicación entre ellas. La comunicación está compuesta de dos partes, una parte física o *HARDWARE* y una parte de programación o *FIRMWARE*.

El *HARDWARE* está compuesto por los materiales necesarios para la comunicación, ya sea el cableado, el tipo de conector o el conversor, mientras que el *FIRMWARE* está compuesto por las instrucciones necesarias o tramas para el funcionamiento de la máquina.

3.1. Tipos de comunicación en la industria.

En este apartado se van a estudiar los tipos de conexión a la hora de hablar de comunicación en el entorno industrial, como por ejemplo:

- **Rs232**

Este tipo de comunicación fue diseñado para la comunicación punto a punto, donde un equipo hace la función de **maestro** transmitiendo hacia un equipo **esclavo** ubicado como máximo a una distancia de 15 metros y a una velocidad máxima de 19,200 bps. El tipo de comunicación está configurado como "*single ended*" ya que usa la misma masa (GND). Este método es vulnerable al ruido y por ello se emplea en comunicaciones de distancias cortas.

En la trasmisión RS232 normalmente las tramas se envían como un conjunto de caracteres ASCII, incluyendo letras, números y caracteres especiales.

Esta comunicación consta de la señal de trasmisión TX, recepción RX y tierra GND, En la siguiente imagen se observa el conector DB9 y la asignación de señales, como se muestra en la *Figura 1*.

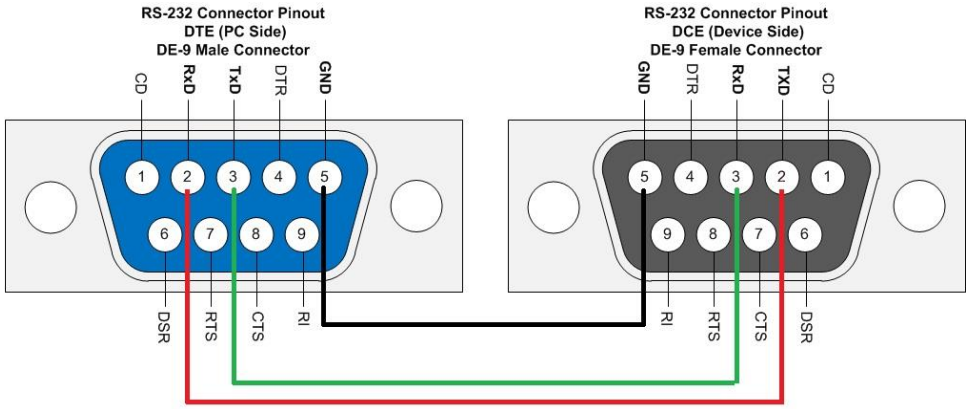


Figura 1. Conexión DB9 Rs232

La trasmisión de datos mediante TX y RX se trata de una señal serial bipolar, normalmente entre +10 y -10 volts. Normalmente va acompañado del circuito MAX232 que permite descomponer el carácter ASCII en un byte para su envío, usando el modo inverso para su recepción. Por ejemplo para la transmisión del numero ASCII "1" se trasmitirá el byte 00110001, Además incluye un bit de Inicio y un bit de Paro.

El bit de inicio (*Start bit*) se encarga de indicar al receptor cuándo debe empezar a leer la trama mediante un flanco ascendente. Por otro lado se añade el bit de Paro al final de la trama, para indicar al receptor que ha terminado la trama, esto se efectúa mediante un flanco descendente. Como la comunicación no necesita una señal de sincronización se trata de una transmisión "Asíncrona".

- **Rs485**

Para una comunicación a mayores distancias y velocidades de transmisión se utiliza la norma RS485. Esta norma también permite la trasmisión multipunto, es decir un ordenador central conectado con varias

UTR. Este tipo de transmisión se denomina trasmisión diferencial y permite velocidades de hasta 10 Mbps, sobre distancias de hasta 1.3 kms.

En este tipo de transmisión se usan dos señales para trasmitir y dos para recibir, además de la tierra, la cual se suele conectar a la malla del cable. En la transmisión se envía la señal de trasmisión y su complemento, mientras que en el receptor, la señal original se obtiene restando las dos señales, lo cual ayuda a reducir notablemente el ruido generado en la línea, ya que éste se adhiere en ambas líneas por igual, con lo que se consigue cancelarlo al restar ambas señales.

Para la comunicación por RS485 se usan 2 señales y el GND y es necesario un protocolo "*half dúplex*", ya que las líneas son usadas tanto para la trasmisión como para la recepción, a continuación se detalla en la *Figura 2*.

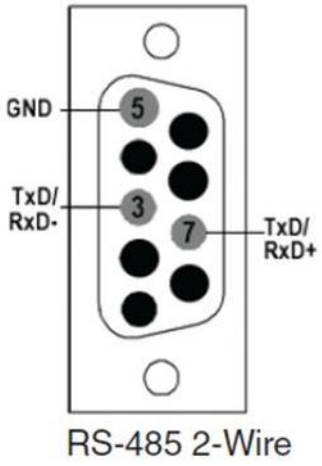


Figura 2. Conexión DB9 Rs485

Todos los dispositivos RS485 deben desconectarse de la red una vez acaben de transmitir la trama para que puedan usarse como recepción. Normalmente, para este fin se usa un circuito temporizador automático habilitado por el flanco ascendente de la señal de trasmisión, aunque en este caso se usará una salida del microcontrolador. El temporizador habilita

el circuito trasmisor durante el tiempo que dura el mensaje y lo deshabilita al terminar éste.

3.2. Protocolos

Un protocolo de comunicación describe un estándar a seguir para que la comunicación y el entendimiento entre dos o más equipos sea favorable, dentro de un protocolo se define tanto la velocidad de comunicación, como la longitud de la trama. Depende de cada protocolo esta trama puede variar de longitud, ya que se pueden aplicar bits especiales como el de inicio o el de *Stop*. A continuación se detallan diferentes tipos de protocolos industriales.

3.2.1. Profibus

El protocolo de comunicación Profibus consiste en un bus de campo de alta velocidad para el control de procesos. Éste puede trabajar hasta 12 Mb/sg, garantizado según los estándares EN 50170 y EN 50254. Este tipo de comunicacion trabaja con unas ordenes de tipo **maestro-esclavo,** donde los equipos **esclavos** se conectan a una línea central sobre el que actúa el **maestro**. Este protocolo consta de una sincronización asíncrona con una trama de 11 bits, la cual está orientada a caracteres. Además, incluye un bit de espera entre dos tramas, el cual debe de ser un valor lógico alto.

El protocolo Profibus actúa sobre tres capas no consecutivas del modelo **OSI** de comunicaciones, el cual se explica a continuación.

MODELO OSI

El modelo OSI consiste en un estándar que regula la comunicación entre distintos sistemas. Está compuesto por siete capas distintas, como se muestra en la *Figura 3*. El modelo OSI

consiste en un estándar que regula la comunicación entre distintos sistemas. Está compuesto por siete capas distintas, como se muestra en la *Figura 3*.

Figura 3. Capas del modelo OSI

Cada capa tiene su función y debe prestar unos servicios que serán usados por la capa superior. Existen dos tipos de comunicación dentro del modelo OSI, la comunicación horizontal entre dos equipos en la misma capa y la comunicación vertical entre una capa y su superior.

Las principales funciones de las capas de aplicación son:

- Capa Física: transmite la información entre dos equipos.
- Capa Enlace de Datos: estructura la información bajo una trama.
- Capa de Red: recibe y envía las tramas.
- Capa de Transporte: controla el flujo de comunicaciones.
- Capa de Sesión: sincroniza los equipos.
- Capa de Presentación: descodifica los datos recibidos.
- Capa de Aplicación: transfiere los archivos.

El protocolo Profibus solo actúa sobre la capa física, de enlace y de aplicación. Además, es un tipo de comunicación **maestro-esclavo**, lo que consiste en un equipo que inicia una solicitud y posteriormente espera la respuesta, siendo el maestro el encargado de iniciar siempre cada comunicación. Usualmente, el maestro suele ser un sistema SCADA mientras que el esclavo suele estar a un nivel inferior pudiendo ser tanto un PLC como un sensor o una unidad de control. Este protocolo también incluye paso de testigo, lo que consiste en que se va pasando un *torken* o marcador entre los esclavos, y solo aquel que tenga el *torken* puede comunicar. Como capa física, lo más utilizado en la industria para este protocolo es un par trenzado con conexión RS485.

La arquitectura de este protocolo de comunicación está dividida en tres tipos, dependiendo del trabajo a realizar:

- **Profibus DP:** es un perfil orientado a sensores o actuadores enlazados a procesadores (PLCs) o terminales. Ofrece mayor velocidad y eficiencia que el resto de perfiles. Está diseñado para la comunicación entre los sistemas de automatización y los equipos periféricos, los cuales pueden alcanzar velocidades de transmisión de hasta 12 Mb/sg. Se puede utilizar para la sustitución de los sistemas de comunicación con el estándar transmisión de 4 a 20 mA.

- **Profibus FMS:** es un perfil utilizado para la comunicación entre células de proceso o equipos de automatización. Se trata de un perfil especializado en la transmisión de datos comunes en las comunicaciones de entorno industrial. Es utilizado para la comunicación a nivel de control y además utiliza un sistema multimaestro, el cual puede alcanzar velocidades de transmisión de hasta 1.5 Mb/sg.

- **Profibus PA:** es un perfil orientado para el control de los procesos de automatización. Este perfil permite una transmisión síncrona a una

velocidad de 31.2 Kbits/sg, y es utilizado para la tecnología de transmisión en IEC 1158-2 usada en las industrias químicas y petrolíferas. Se puede utilizar para la sustitución de los sistemas de comunicación con el estándar transmisión de 4 a 20 mA. Permite la comunicación a través de dos hilos simples trenzados y apantallados mediante la capa física RS485.

3.2.2. Ethernet

Ethernet o IEE 802.3 es uno de los estándares más usados en comunicaciones de Red de Área Local (LAN), donde los equipos están conectados dentro de un área geográfica a través de una red, lo que le permite tener entre 100 y 1000 usuarios y una velocidad de transmisión de entre 10 Mbps y 10 Gbps, dependiendo de la tecnología de transmisión utilizada.

Este protocolo usa un método de transmisión CSMA o acceso múltiple con detección de portadora y colisiones. Lo primero que hace es escuchar y comprobar que no esté transfiriendo otro equipo, si está libre la red, manda la trama y el resto de equipos escuchan la trama pero solo el seleccionado analiza la información. Si dos equipos intentan mandar la trama a la vez, paran de enviar y esperan un tiempo aleatorio antes de volver a intentar enviar la trama.

La trama está compuesta por la dirección del equipo receptor, la dirección del equipo transmisor y el mensaje a transmitir. El mensaje puede variar entre 64 y 1500 bytes, lo que hace que el tiempo de transmisión trabaje entre 50 y 1200 microsegundos. Además, contiene una zona de preámbulo al inicio del envío, compuesto por 64 bits que ayudan a sincronizar con el receptor deseado y una zona de redundancia cíclica o CRC, compuesto por 32 bits donde se comprueba que la trama enviada es correcta, se puede observar la trama de Ethernet en la *Figura 4*.

Preámbulo	Dirección destino	Dirección fuente	Tipo	Datos	CRC
8 bytes	6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes

Figura 4. Trama Ethernet

También puede incluir una zona de tipo de trama, que permite identificar el tipo de dato a transferir en el mensaje, lo que permite que las tramas se puedan auto identificar. El receptor de la trama utiliza esta información para determinar qué protocolo utilizar para el procesamiento de la trama. Esto permite que un equipo pueda utilizar diferentes protocolos con una misma red sin interferencias.

TCP/IP

El protocolo TCP/IP se compone de una trama Ethernet auto identificable para diferenciar entre diferentes protocolos. Este conjunto de protocolos se corresponde con el modelo de comunicaciones definido por la norma ISO, que define un sistema de redes de interconexión de sistemas abiertos u OSI, que permite la comunicación entre procesos con diferentes capas, las capas dan servicio a las capas superiores y reciben servicio de capas inferiores.

3.2.3. Modbus

Modbus consiste en un protocolo de solicitud-respuesta entre dos dispositivos. Éste funciona con una relación de maestro-esclavo, ya definido anteriormente. La capa física del protocolo Modbus puede ser a través de RS232, RS422 o RS485.

Inicialmente, el protocolo Modbus no podía estar dividido en múltiples capas de comunicación, ya que, estaba hecho en base al protocolo serial.

Actualmente el protocolo Modbus ha evolucionado permitiendo el uso de redes TCP/IP, explicadas previamente en el apartado anterior, o UDP. Este protocolo está dividido en dos partes:

- **PDU,** es la unidad de datos del protocolo. Los datos enviados a través de Modbus son almacenados en uno de los cuatro bloques de memoria. Dependiendo de la aplicación a usar, estos bancos de memoria determinan el tipo y el acceso de los datos, donde el esclavo tiene acceso a estos datos, mientras que el maestro debe pedir el acceso a estos datos a través de funciones. Cada bloque tiene un espacio máximo de 65.536 celdas, donde cada celda va desde 0 a 65.535, aunque cada elemento de datos esta numerado de 1 a n, siendo n como máximo 65.536. Para simplificar la ubicación de cada bloque de memoria se introdujo un prefijo añadido a la dirección de los datos, este prefijo está compuesto entre 4 y 6 valores (YXXX, YXXXX, YXXXXX), donde Y corresponde con el valor del bloque, mientras que XXXX corresponde con la dirección del registro variando el número de X dependiendo de las celdas ocupadas. Por ejemplo el prefijo 3025 corresponde al bloque 3 celda 25, por otro lado el prefijo 165536 corresponde al bloque 1 celda 65536.

- **ADU**, es la unidad de datos de aplicación. Por otro lado, el equipo **maestro** debe conocer el límite de la trama por lo que además del prefijo comentado anteriormente, se añade la longitud de la trama y el orden de bytes. Por ejemplo el prefijo 3025.2H corresponde al bloque 3 celda 25 seguido de una trama de 2 caracteres donde el byte alto es el primer carácter, así se evita que el equipo maestro tenga que buscar la longitud de la trama. Algunos equipos **esclavos** pueden almacenar la información invirtiendo el orden de bytes, con lo que es trabajo del equipo **maestro** saber cómo esta almacenando el equipo

esclavo y decodificar la información almacenada. A diferencia del modelo PDU, el modelo ADU sigue un patrón marcado, donde el **esclavo** primero valida el prefijo, incluyendo el rango de datos, y a continuación ejecuta la orden proveniente del equipo **maestro** y envía la respuesta necesaria al código. Si una parte del proceso falla, se responde una excepción al equipo maestro.

3.2.4. DeviceNet

Este protocolo es utilizado en la interconexión de equipos industriales y dispositivos de entrada/salida. Puede estar configurado tanto como un sistema **maestro/esclavo** como en un sistema distribuido de punto a punto. Además, puede tener una comunicación hasta con 64 equipos, ordenándolos de 0 a 63.

Este protocolo usa un tipo de manguera de doble-par. Ésta está compuesta por dos cables trenzados; uno por el que se transmite la comunicación y otro por el que va la alimentación del dispositivo. La alimentación viene conectada directamente de una fuente conectada a la red, siendo esta de 24 Vdc.

La trama de datos viene definida por las especificaciones del estándar CAN. El equipo transmisor primero debe escuchar y comprobar que no esté transfiriendo otro equipo, si está libre la red manda la trama. Además, está provisto de detección de colisión por lo que si dos equipos intentan mandar la trama a la vez paran de enviarla y el que tenga prioridad mandará la trama en primer lugar. La trama enviada por DeviceNet utiliza el protocolo CIP (*Common Industrial Protocol*), lo que lo hace independiente de la capa física.

CIP (*Common Industrial Protocol*)

CIP es un protocolo orientado al envío de objetos, donde cada objeto es una representación abstracta de un componente. Así mismo dentro de cada objeto se puede encontrar varias clases y dentro de cada clase se puede encontrar diferentes casos que corresponden con las especificaciones del objeto. Por último dentro de cada caso es posible encontrar uno o varios atributos correspondientes a la descripción de una característica del objeto, en la *Figura 5* se muestra un esquema del protocolo CIP.

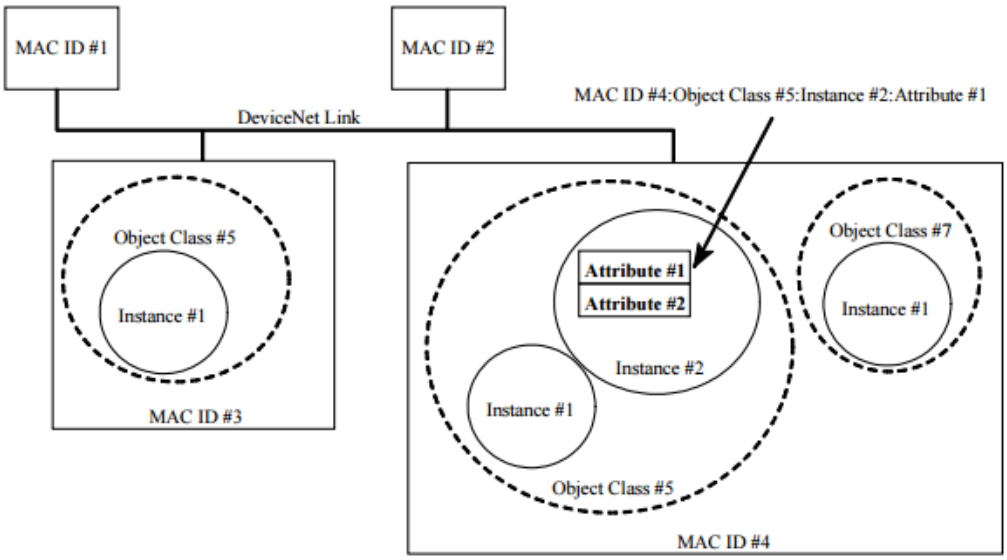


Figura 5. Esquema protocolo CIP

La trama enviada por DeviceNet está compuesta por 1 bit de inicio de trama, 11 bits que conforman el identificador del equipo receptor y la prioridad, 1 bit RTR el que indica si es una trama remota o de datos, 6 bits para el campo de control donde indica el número de datos por el que está computa la información a enviar, de 0 a 8 bytes de datos de información, 2 bytes de CRC donde se detecta si hay error de transmisión, a continuación envía 2 bits de ACK siendo este el acuse de recibo, seguidamente envía 7

bits de final de trama y por ultimo 3 bits de espacio entre tramas. A continuación se muestra la composición de la trama en la *Figura 6*.

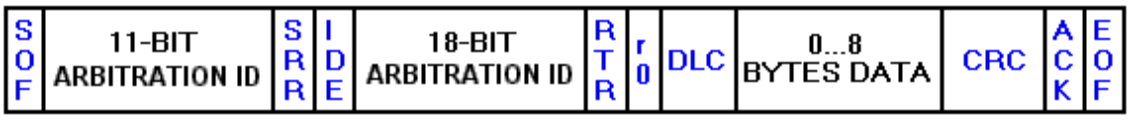


Figura 6. Trama DeviceNet

4. Arduino®

4.1. IDE Arduino®

Entorno de Desarrollo Integrado (de ahora en adelante IDE) se trata de un programa informático compuesto por un conjunto de herramientas de programación. Dicho programa puede utilizar uno o varios lenguajes de programación.

El IDE se compone de un editor de texto, un compilador, un *debug* o depurador y un GUI (constructor de interfaz gráfica) en un mismo entorno de programación, además de poder incluir herramientas para poder cargar un programa ya compilado, como es nuestro caso.

Arduino® trabaja con ficheros con la extensión "ino", aunque es necesario que el programa principal este dentro de una carpeta con el mismo nombre, se pueden añadir programas secundarios donde incluir llamadas del programa principal.

Al tratarse de un software "*OpenSource*" facilita el auto aprendizaje ya que es posible encontrar por internet gran cantidad de ejemplos de diferentes aplicaciones compartidas por otros usuarios, además de facilitar la localización de un gran abanico de *shields d*e sensores compatibles con Arduino®.

4.2. Tipos de Arduino®

Debido a que Arduino® es una de las primeras plataformas microcontroladoras del mundo *OpenSource* se han desarrollado varias versiones sobre la aplicación. A continuación se muestra las versiones más relevantes de la misma en la *Figura 7*.

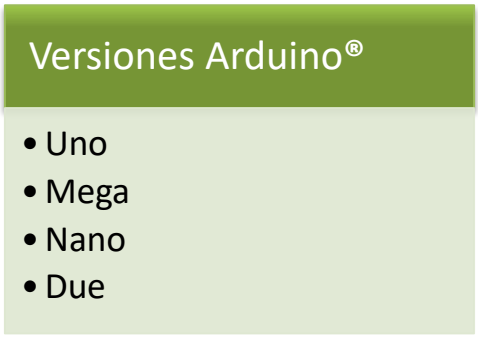


Figura 7. Tipos de Arduino®

Dependiendo del tipo de Arduino® que se elija las características del propio Arduino® varían, con lo que es necesario elegir el Arduino® más adecuado para el tipo de aplicación que se desea programar. Seguidamente se muestran las características más relevantes de cuatro ejemplos de tipos de Arduino®, se pueden ver las imágenes en la *Figura 8, Figura 9, Figura 10 y Figura 11*.

Arduino® Uno:

- Pines I/O digitales: 14 I/O, 6 de las I/O pueden usarse con salida PWM
- Entradas Analógicas: 6
- Memoria Flash: 32 KB
- Velocidad del reloj: 16 MHz
- Corriente DC: 40-50 mA

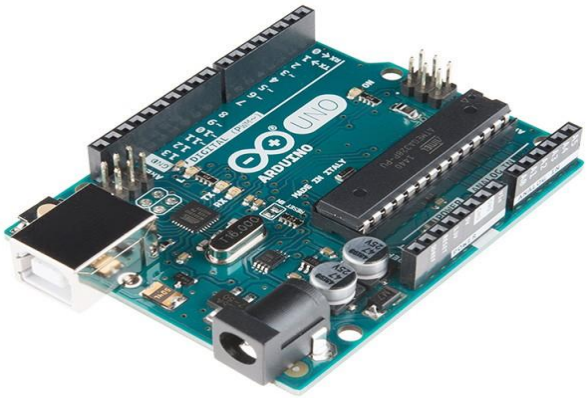


Figura 8. Arduino® UNO

Arduino® Mega 2560:

- Pines I/O digitales: 54 I/O, 14 de las I/O pueden usarse con salida PWM
- Entradas Analógicas: 16
- Memoria Flash: 256 KB
- Velocidad del reloj: 16 MHz
- Corriente DC: 40-50 mA

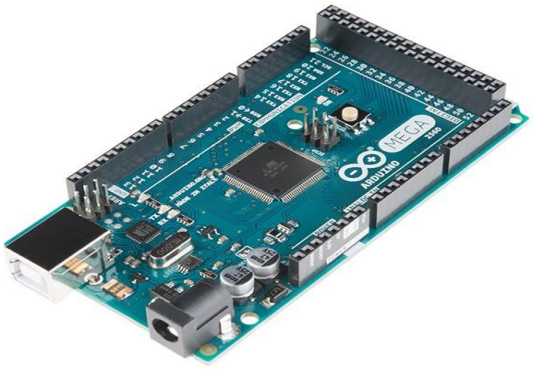


Figura 9. Arduino® Mega

Arduino® Nano:

- Pines I/O digitales: 14 I/O, 6 de las I/O pueden usarse con salida PWM
- Entradas Analógicas: 8
- Memoria Flash: 16 KB
- Velocidad del reloj: 16 MHz
- Corriente DC: 40 mA

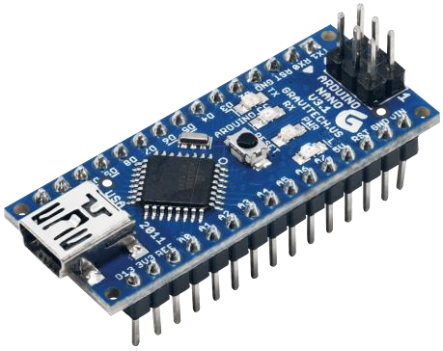


Figura 10. Arduino® Nano

Arduino® Due:

- Pines I/O digitales: 54 I/O, 12 de las I/O pueden usarse con salida PWM
- Entradas Analógicas: 12
- Memoria Flash: 512 KB
- Velocidad del reloj: 84 MHz
- Corriente DC: 800 mA

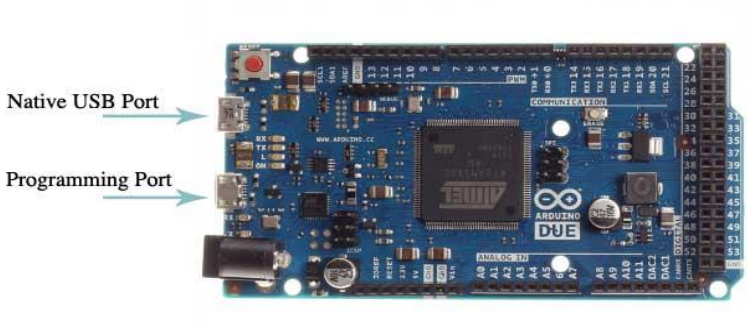


Figura 11. Arduino® Due

5. SCADA

Supervisory Control And Data Adquisition (de ahora en adelante SCADA) es un sistema basado en equipos que permiten supervisar y controlar a distancia un proceso, instalación o sistemas con características variadas. Es necesario un control en lazo cerrado generalmente dado por el operador, todo lo contrario a lo que ocurre en los Sistemas de Control Distribuido, los cuales realizan acciones de control de forma automática. El SCADA permite trabajar mediante protocolos de comunicación como el ModBus, OPC, etc.

A continuación se describe el flujo de información en este tipo de sistemas. En primer lugar aclarar que el fenómeno físico lo constituye la variable que se desea medir. Esta variable puede ser muy diversa dependiendo de la magnitud a medir, el cual podría ser la presión, flujo de potencia, temperatura, voltaje, etc. Es decir que esta magnitud se transformará en una variable eléctrica para su posterior lectura, mediante sensores, los cuales convierten una variación de una magnitud física en variaciones proporcionales de una variable eléctrica.

A continuación es necesario procesar estas variaciones eléctricas para poder analizarlas mediante un equipo. El proceso de las variaciones se hace a través de amplificadores cuya función es amplificar la señal proveniente del sensor, mediante una ganancia configurada en el amplificador, ya que generalmente las señales provenientes del sensor suelen ser del orden de miliVoltios. Seguidamente es necesario convertir las señales en señales digitales, a través de un convertidor analógico/digital, para su posterior análisis y visualización en el equipo.

El SCADA permite a un operario actuar sobre el proceso en un momento deseado sin tener la máquina delante, como si estuviese actuando sobre el mismo actuador de la máquina, lo que permite un control más rápido y eficaz.

6. Desarrollo del TFG

En este apartado se va a analizar el desarrollo de este trabajo, compuesto principalmente por dos partes, la parte mecánica de la que se ha ocupado Alexander Rosales en el proyecto titulado *Diseño del Sistema Automatizado de Almacén Paletizado*, y la parte de comunicaciones. Aunque se habla de dos trabajos diferentes, estos se complementan para el funcionamiento de la maqueta, ya que en el proyecto de Alexander Rosales se encuentra la fabricación de la maqueta de un almacén automatizado y la programación de los motores y detectores, mientras que en este proyecto se estudian las diferentes formas de comunicación y una pasarela para unir dos comunicaciones diferentes.

La idea de realizar este trabajo surge de la necesidad de mejorar un proyecto previamente fabricado en la asignatura de la Electrónica Digital, el cual fue donado a la Universidad Politécnica de Valencia una vez acabada la asignatura y en la actualidad se encuentra expuesto en la Escuela Técnica Superior de Ingeniería de Diseño, en la *Figura 12* se muestra la maqueta original junto con el equipo que la fabrico.

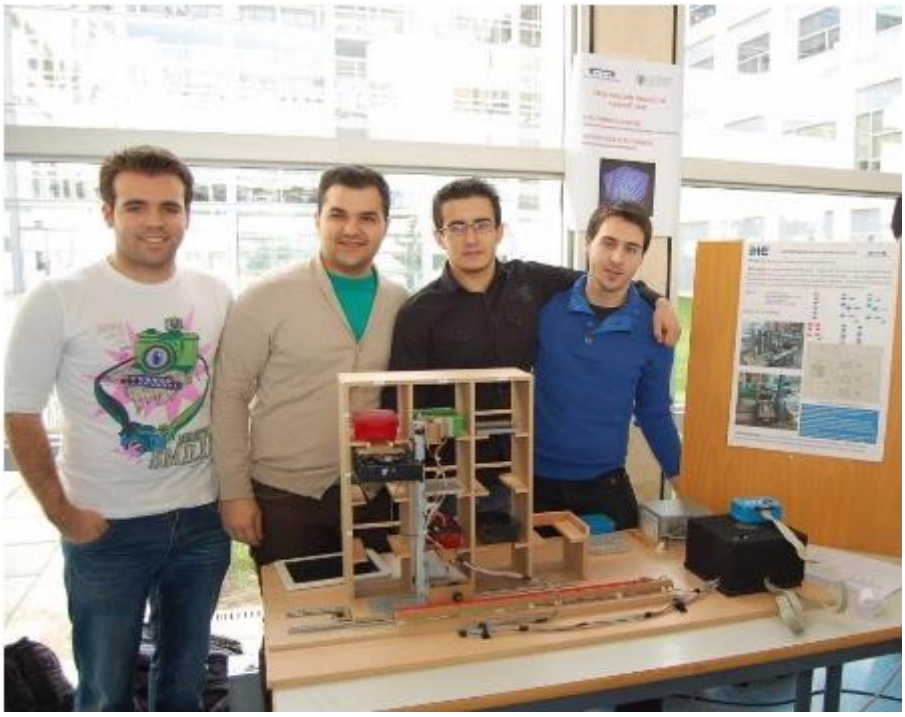


Figura 12. Proyecto original

En este proyecto se ha fabricado una placa donde se encuentran los circuitos electrónicos necesarios para la comunicación. Además se ha creado una trama específica para el envío de datos y se ha mandado de dos formas diferentes, una por caracteres mientras que la otra bit a bit. Por otra parte también está programación de un SCADA desde el que controlar la paletizadora automática.

6.1. Desarrollo Eléctrico.

En este apartado se estudia la parte eléctrica del trabajo, donde se detallaran los componentes utilizados para la transmisión de datos y sus características. En la *Figura 13* se muestra el montaje de final del desarrollo eléctrico.

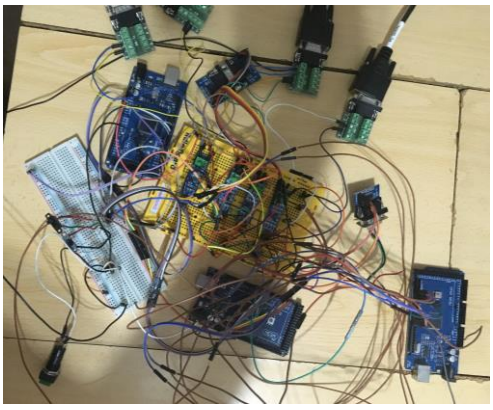


Figura 13. Desarrollo eléctrico

Además se adjuntarán los planos eléctricos seguidos para su implementación, estos planos se pueden consultar en la sección de planos dentro de Anexos. A continuación se detallan los componentes utilizados para la fabricación de este proyecto:

6.1.1. Arduino®

Anteriormente se han especificado las características de este microcontrolador, para este proyecto se ha elegido el Arduino® Mega. Este controlador ha sido elegido porque consta de 54 entradas/salidas digitales, 16 entradas analógicas, pero lo más importante para este proyecto es que tiene 4 pares de pines reservados para el envío de comunicaciones. Para este proyecto se han usado 3 ArduinoS® mega: 1 para el maestro, 1 para la pasarela y el último para el esclavo, el cual se comparte con el control de la maqueta a escala. Además se utiliza un Arduino® Mega para el control del sistema SCADA, el cual se conecta con el Arduino® maestro para el envío de la trama.

6.1.2. MAX 232

Para el envío de datos por RS232 se utiliza el circuito integrado MAX232, el cual trabaja a 5V. Este componente transforma los niveles de transmisión a niveles TTL. El pin Tx se utiliza para el envío de datos mientras que el pin Rx se utiliza para la recepción de datos. En la *Figura 14* se muestra lo mencionado.

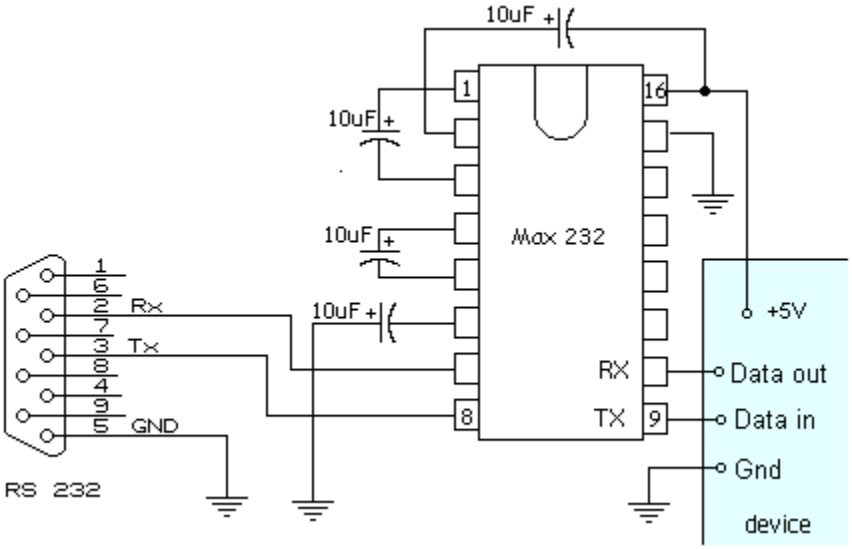


Figura 14. Conexión MAX 232

Para la comunicación entre dos equipos por RS232 es necesario cruzar las señales de envío y recepción, lo que significa que el pin Tx del master tiene que ir conectado con el pin Rx del esclavo, mientras que el pin Rx del master a conectado al pin Tx del esclavo, lo que permite el envío y recepción de cada Arduino®.

6.1.3. MAX 485

Para el envío de datos por RS485 se utiliza el circuito integrado MAX485, el cual trabaja a 5V, transformando los niveles de transmisión a niveles TTL. El pin Di se utiliza para el envío de datos mientras que el pin Ro se utiliza para la recepción de datos. Además de los pines De y Re, los cuales son los enables del circuito, los que habilitan la transmisión o la recepción de datos, estos pines están conectado a una salida de Arduino®, la cual los activa o desactiva según las necesidades. En la *Figura 15* se muestra lo mencionado anteriormente.

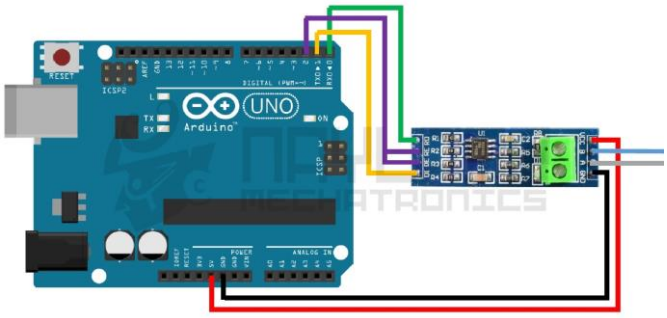


Figura 15. Conexión MAX 485

Para la comunicación entre dos equipos por RS485 es necesario tenerlos en configuración **"*half duplex*"**, lo que significa interconectar los pines **A** de los dos equipos. Por otro lado también tienen que ir interconectados los pines **B** de los dos equipos, o lo que es lo mismo, no cruzar las señales a diferencia del RS232. Además los pines De y Re deberán tener una valor lógico alto para la transmisión de datos y un

valor lógico bajo para la recepción de los mismos, por lo que si se quiere iniciar una comunicación entre el maestro y el esclavo, el maestro tendrá que tener un nivel lógico alto en los pines De y Re, mientras que el esclavo deberá tener un nivel lógico bajo para estar configurado como recepctor. En la *Figura 16* se muestra la conexión entre dos equipos.

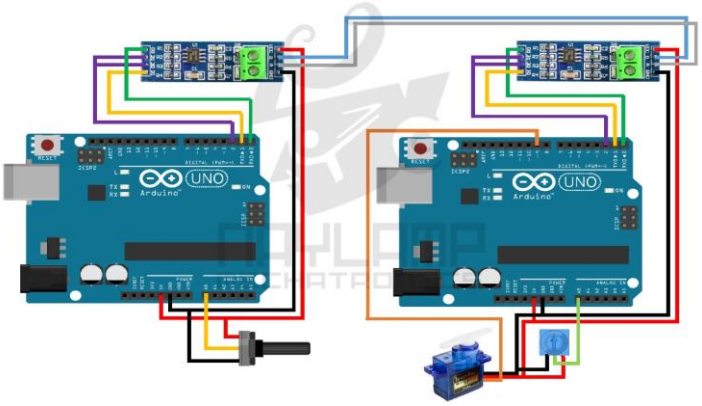


Figura 16. Conexión MAX 485 *half duplex*

6.1.4. Pantalla TFT

Para la parte del sistema SCADA se ha utilizado una pantalla TFT de 5 pulgadas conectada a un Arduino® Mega, además incluye dos filas de 4 botones cada fila con los que poder moverse a través de los menús. Por ultimo este sistema dispone de un teclado numérico con el cual se podrán introducir los números necesarios para la selección de la posición de la maqueta o el valor numérico que se quiera enviar en la trama. En la *Figura 17* se muestra la pantalla TFT.

Figura 17. Pantalla TFT

El Arduino® utiliza 5 entradas para cada una de las botoneras 4x1 (*keypad* 2 y *keypad* 3), también utiliza 7 entradas para el keypad 4x4 y 20 salidas que configura la pantalla, A continuación se muestra en la *tabla* 1 las conexiones necesarias entre la pantalla y el Arduino®.

keypad 1	Fila 1	A0
	Fila 2	A1
	Fila 3	A2
	Fila 4	A3
	Columna 1	A4
	Columna 2	A5
	Columna 3	A6
Keypad 2	Columna 1	8
	Columna 2	9
	Columna 3	10
	Columna 4	11
	Fila 1	12
Keypad 3	Columna 1	A8
	Columna 2	A9
	Columna 3	A10
	Columna 4	A11
	Fila 1	A12

Pantalla	DB8	22
	DB9	23
	DB10	24
	DB11	25
	DB12	26
	DB13	27
	DB14	28
	DB15	29
	DB7	30
	DB6	31
	DB5	32
	DB4	33
	DB3	34
	DB2	35
	DB1	36
	DB0	37
	LCD_RS	38
	LCD_WR	39
	LCD_CS	40
	LCD_RST	41

Tabla 1. Conexiones Arduino® pantalla

6.2. Desarrollo Mecánico

Como ya se ha comentado anteriormente, de la parte mecánica del proyecto se encarga mi compañero Alexander Rosales, construyendo una maqueta a escala de una Paletizadora Automática. Dicha maqueta consta de tres motores para el movimiento del carro en x, y, z, además de los circuitos de potencia necesarios para su funcionamiento y la lectura de los sensores que indican el posicionamiento del carro. En la *Figura 18* se muestra la maqueta a escala.

Figura 18. Maqueta a escala

6.3. Programación.

En este apartado se muestra la programación necesaria para el funcionamiento del envío y recepción de tramas tanto por RS232 como por RS485, además del funcionamiento de la pasarela que actúa como intermediario entre el envío y la recepción. Por otra parte se verá la programación necesaria para el funcionamiento de la pantalla TFT y también se mostrará el protocolo usado para la comunicación entre el mando de control y la maqueta a escala.

6.3.1. Protocolo

En este proyecto de fin de grado se ha cogido como referencia el estándar del modelo OSI para la creación del protocolo a seguir. Como ya se ha explicado anteriormente el modelo OSI consta de siete capas de aplicación, de las cuales para este proyecto interesan principalmente las tres primeras. Estas capas son:

- **Capa Física:** Esta capa está compuesta de la parte física necesaria para la comunicación, en concreto consta de los conectores Db9, los cables y los circuitos max 232 y max

485 utilizados en este proyecto, además también se podría incluir en esta capa los Arduino® necesarios.

- **Capa de Enlace de Datos:** Esta capa está compuesta de la programación necesaria para el envío y recepción de datos entre los Arduino®, en concreto la trama usada, la cual se explicará a continuación.
- **Capa de Red:** Esta capa se encarga de analizar la trama recibida y actuar en consecuencia.

Una vez ya definidas las capas del modelo OSI que utiliza este protocolo, y la parte física definida previamente en la parte de comunicaciones y desarrollo eléctrico, a continuación se define la trama utilizada en este proyecto.

- **Trama**

Como trama para este proyecto se ha utilizado un conjunto de números y letras que indican a la maqueta a escala lo que el usuario quiere que haga. Esta trama está compuesta por **cuatro** valores, **dos** números, **una** letra y el último valor que puede ser tanto un número o una letra, ya que se trata del ***checksum*** y depende de los tres valores anteriores. A continuación se da una explicación del significado de los distintos valores:

- **Primer valor:** Este valor se trata de un número, tiene un rango entre 1 y 8, e indica la columna a la que se tiene que desplazar la maqueta. Originalmente este valor tenía un rango entre 1 y 4, teniendo otro valor para indicar si era el lado de la derecha o de la izquierda, pero se decidió ampliar este valor para reducir el tamaño de la trama, con lo que ahora el rango de 1 a 4 indica que se trata del lado derecho y el rango de 5 a 8 que se tata del lado izquierdo.
- **Segundo valor:** Este valor se trata de un número, tiene un rango entre 1 y 5, e indica la fila a la que se tiene que

desplazar la maqueta. Junto con el primer valor se define exactamente la posición en la que tiene que trabajar la maqueta.

- **Tercer valor:** Este valor se trata de una letra y tiene un rago entre la A y la F, e indica la acción que debe hacer la maqueta, siendo; A la opción de **coger**, B la opción de **dejar**, C la opción de **leer posición**, D la opción de **borrar posición**, E la opción de **borrar toda la memoria** y F la opción de **escribir en una posición**.
- **Cuarto valor:** Este valor puede tratarse tanto de un número como de una letra, ya que se trata del ***checksum*.** Este valor se conforma de de los tres valores anteriores ya que se trata de una suma **XOR** de los tres valores previos. Además, este valor sirve para la comprobación de que la trama ha sido recibida correctamente ya que este valor cuando se envía debe coincidir con el valor calculado en la recepción.

6.3.2. Envío

En este apartado se muestran las funciones necesarias para el envío de la trama tanto por RS232 como por RS485a Además se detallará lo que realiza cada función y también se explicará la forma de enviar de los dos tipos de comunicación ya que envían de forma diferente, en la *Figura 19* se muestra la conexión del sistema de envío.

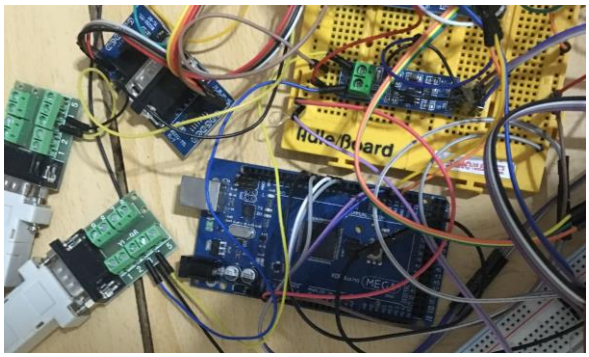


Figura 19. Sistema de Envío

- **RS232**

El envío de la trama por RS232 se hace a través del circuito integrado MAX232. Esta comunicación se hace a una velocidad de 9600 baudios y se hace a través del puerto Serie 1 del Arduino®. Este método de comunicación coge cada valor de la trama y lo envía directamente como un valor ASCII. A continuación se explican las funciones que utiliza para llevar a cabo la transmisión.

- **Calculo_Chk_In_232()** → Calcula el ***checksum***, el cual corresponde al cuarto valor de la trama, este ***checksum*** se obtiene con una suma **XOR** de los tres primeros valores de la trama.
- **Comprobacion_trama_envio_232()** → Comprueba que los valores de la trama estén dentro de los rangos previamente definidos, si no coinciden se pide que se vuelvan a introducir.
- **Envio_Rs232()** → Envía la trama por el puerto Serie 1 que corresponde al RS232.

Aunque se trate del envío de la trama, el Arduino® **maestro** también espera a recibir un byte proveniente del Arduino® **esclavo** que le indica si la trama ha sido entregada correctamente. Una vez recibida la trama correctamente, el Arduino® **maestro** espera a que la maqueta a escala acabe de trabajar y le indique tanto si ha acabado

como si ha ocurrido algún error en el transcurso de la actividad. Hasta que la maqueta no le indica que ha acabado de hacer su actividad, el Arduino® **maestro** no sigue mandando ordenes. Estas funciones son las siguientes:

- **Recepcion_trama_ok_232()** → El Arduino® **maestro** espera a que el Arduino® **esclavo** le devuelva un byte indicando si ha recibido correctamente la trama o no.
- **Comprobar_trama_ok_232()** → Comprueba si la trama recibida por el Arduino® **esclavo** es correcta, en el caso de que no sea correcta el Arduino® **maestro** vuelve a enviar la trama. Esta función incluye la función anterior "Recepcion_trama_ok_232".
- **Esperar_Respuesta_232()** → Espera y recibe un byte proveniente de la maqueta a escala que indica si ha acabado su actividad o ha ocurrido cualquier error.
- **Comprobar_errores_232()** → Analiza la respuesta proveniente de la maqueta a escala. Si le lleva el byte de que ha finalizado satisfactoriamente, el Arduino® **maestro** sigue enviando órdenes, en caso contrario muestra un mensaje con el error correspondiente.

- **RS485**

El envío de la trama por RS485 se hace a través del circuito integrado MAX485. Esta comunicación se hace a una velocidad de 9600 baudios y se hace a través del puerto Serie 2 del Arduino®. Para este método de comunicación se coge cada valor de la trama y se descompone en los ocho bits del que se compone y seguidamente se adaptan para el envío. Antes de enviar la trama se envía un bit para indicar al Arduino® **esclavo** que va a comenzar la transmisión. A continuación se explican las funciones que utiliza para llevar a cabo la transmisión:

- **Conversion_485()** → Separa el valor ASCII en los ocho bits que lo componen para su posterior envío y los almacena en una variable para su posterior envío.
- **Calculo_Chk_In_485()** → Calcula el ***checksum***, el cual corresponde al cuarto valor de la trama. Este ***checksum*** se obtiene con una suma **XOR** de los tres primeros valores de la trama.
- **Comprobacion_trama_envio_485()** → Comprueba que los valores de la trama estén dentro de los rangos previamente definidos, si no coinciden se pide que se vuelvan a introducir.
- **Envio_inicio()** → Envía un bit de inicio al **esclavo** para indicar que se empieza la transmisión de la trama.
- **Adaptacion_Envio_485(int a)** → Sustituye los bits correspondientes a los valores de la trama en la variable de envío, esta función tiene un argumento que corresponde a la posición del valor a enviar.
- **Envio_Rs485()** → Envía los ocho bits correspondientes a cada valor de la trama a través del puerto serie 2 que corresponde al Rs485.
- **Envio_Trama_Rs485()** → En esta función se incluyen las funciones "Adaptacion_Envio_485, Envio_inicio, Envio_Rs485", y es la que se encarga de ir sustituyendo los valores almacenados en la variable de envío, además de enviar el bit de inicio y la trama.

El Arduino® **maestro** espera a recibir si la trama ha sido entregada correctamente y los mensajes provenientes de la maqueta a escala, al igual que sucede con el Rs232. Estas son las funciones utilizadas para su comprobación:

- **Recepcion_trama_ok_485()** → El Arduino® **maestro** espera a que el Arduino® **esclavo** le devuelva un byte indicando si ha recibido correctamente la trama o no.

- **Conversion_recep_485()** → Agrupa y desplaza a su posición los ocho bits recibidos para transformarlos en un valor ASCII con el que poder trabajar, además debe invertir el valor recibido ya que lo recibe invertido.
- **Comprobar_trama_ok_485()** → Comprueba si la trama recibida por el Arduino® **esclavo** es correcta, en el caso de que no sea correcta el Arduino® **maestro** vuelve a enviar la trama, esta función incluye las funciones anteriores "Recepcion_trama_ok_485 y Conversion_recep_485".
- **Esperar_Respuesta_485()** → Espera y recibe un byte proveniente de la maqueta a escala que indica si ha acabado su actividad o ha ocurrido cualquier error. Esta función incluye las funciones anteriores "Recepcion_trama_ok_485 y Conversion_recep_485".
- **Comprobar_errores_485()** → Analiza la respuesta proveniente de la maqueta a escala, si le lleva el byte de que ha finalizado satisfactoriamente, el Arduino® **maestro** sigue enviando órdenes, en caso contrario muestra un mensaje con el error correspondiente.

6.3.3. Recepción

En este apartado se muestran las funciones necesarias para la recepción de la trama tanto por RS232 como por RS485. Además, se detallará lo que realiza cada función y por último se explicará la forma de recepción de los dos tipos de comunicación ya que reciben de forma diferente, en la *Figura 20* se muestra la conexión del sistema de Recepción.

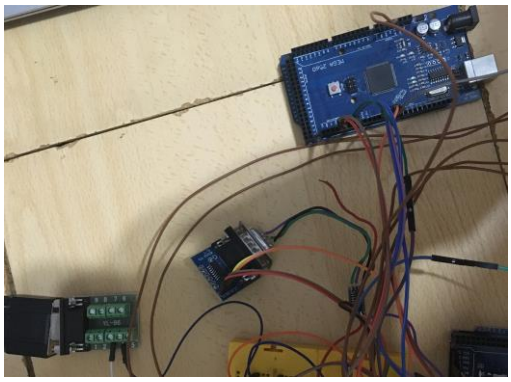


Figura 20. Sistema de Recepción

- **RS232**

La recepción de la trama por RS232 se hace a través del circuito integrado MAX232. Esta comunicación se hace a una velocidad de 9600 baudios y se hace a través del puerto Serie 1 del Arduino®. Este método de comunicación recibe cada valor de la trama en valor ASCII y comprueba que la trama recibida es correcta calculando el ***checksum*** y comparándolo con el ***checksum*** recibido. A continuación se explican las funciones que utiliza para llevar a cabo la recepción.

- **Recepcion_Rs232(int a)** → El Arduino® **esclavo** recibe la trama mediante el puerto serie 1 que corresponde al Rs232, esta función tiene un argumento que indica el número de valores a leer.
- **Calculo_Chk_Out_232()** → Calcula el ***checksum*** con los tres primeros valores de la trama, el cual se utilizara seguidamente para comprobar si la trama recibida es correcta.
- **Comprobar_trama_232()** → Comprueba que la trama recibida es correcta comparando el ***checksum*** calculado y el cuarto valor de la trama que corresponde al ***checksum*** enviado. Si la trama es correcta devuelve un valor indicando al Arduino® **maestro** que es correcto, en caso contrario

indica que no es correcta y se prepara para volver a recibir la trama.

- **Esperar_Respuesta()** → Espera a que la maqueta le indique que ha acabado o le indique un byte de error para transmitirlo al Arduino® **maestro**.
- **Respuesta_232()** → Espera indicaciones de la maqueta a escala en función de la acción enviada. Esta función incluye la función anterior "Esperar_Respuesta".
- **Respuesta_trama_232()** → Devuelve al Arduino® **maestro** las indicaciones de la maqueta a escala.

- **RS485**

La recepción de la trama por RS485 se hace a través del circuito integrado MAX485. Esta comunicación se hace a una velocidad de 9600 baudios y se hace a través del puerto Serie 2 del Arduino®. Para este método de comunicación se reciben los 8 bits que conforman cada valor de la trama, y se desplazan a su posición para volver a juntarlos en un valor ASCII. Seguidamente se calcula que la trama recibida es correcta calculando el ***checksum*** y comparándolo con el ***checksum*** recibido, a continuación se explican las funciones que utiliza para llevar a cabo la recepción:

- **Recepcion_Rs485()** → El Arduino® esclavo recibe la trama mediante el puerto serie 2 que corresponde al Rs485, recibe tanto el bit de inicio como los cuatro bytes de la trama.
- **Conversion_485()** → Agrupa y desplaza a su posición los ocho bits recibidos para transformarlos en un valor ASCII con el que poder trabajar, además debe invertir el valor recibido ya que lo recibe invertido.
- **Calculo_Chk_Out_485()** → Calcula el ***checksum*** con los tres primeros valores de la trama, el cual se utilizara seguidamente para comprobar si la trama recibida es correcta.

- **Comprobar_trama_485()** → Comprueba que la trama recibida es correcta comparando el ***checksum*** calculado y el cuarto valor de la trama que corresponde al ***checksum*** enviado. Si la trama es correcta devuelve un valor indicando al Arduino® **maestro** que es correcto, en caso contrario indica que no es correcta y se prepara para volver a recibir la trama.
- **Envio_inicio()** → Envía un bit de inicio al **esclavo** para indicar que se empieza la transmisión de la trama.
- **Adaptacion_Envio_485(int a)** → Sustituye los bits correspondientes a los valores de la trama en la variable de envío, esta función tiene un argumento que corresponde a la posición del valor a enviar.
- **Envio_Rs485()** → Envía los ocho bits correspondientes a cada valor de la trama a través del puerto serie 2 que corresponde al Rs485.
- **Respuesta_trama_485()** → En esta función se incluyen las funciones "Adaptacion_Envio_485, Envio_inicio, Envio_Rs485", y es la que se encarga de ir sustituyendo los valores almacenados en la variable de envío, además de enviar el bit de inicio y la respuesta de la maqueta a escala.
- **Esperar_Respuesta()** → Espera a que la maqueta le indique que ha acabado o le indique un byte de error para transmitirlo al Arduino® **maestro**.
- **Respuesta_485()** → Espera indicaciones de la maqueta a escala en función de la acción enviada. Esta función incluye la función anterior "Esperar_Respuesta".

6.3.4. Pasarela

En este apartado se muestran las funciones necesarias para la conmutación entre la forma de transmisión de la trama por RS232 y por RS485, incluyendo las funciones necesarias para la transmisión y

la recepción de la trama, además se detallar lo que realiza cada función, en la *Figura 21* se muestra la conexión del sistema de intercomunicación.

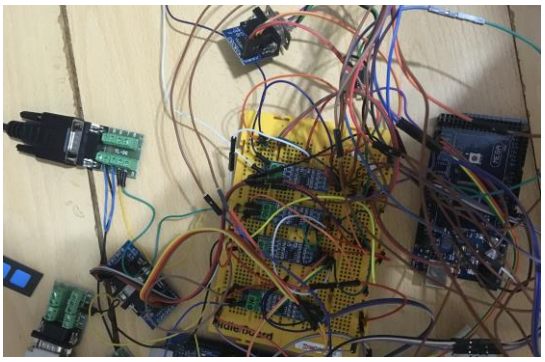


Figura 21. Sistema de intercomunicación

- **Conversión de RS232 a Rs485**

Para la conversión de RS232 a RS485 es necesario que el programa reciba las tramas correctas en valor ASCII, el cual es descompuesto en los 8 bits que conforman cada valor y se almacenan hasta el momento de enviarlos. Tanto la transmisión como la recepción han sido explicadas en los apartados anteriores. A continuación se explican las funciones que utiliza para llevar a cabo la conversión:

  - **Rs485_to_Rs232()** → Transforma el valor proveniente del Rs485 a un tipo de variable **char** para poder trabajar con el valor.

- **Conversión de RS485 a RS232**

Para la conversión de RS485 a RS232 es necesario que el programa reciba los 8 bits de cada valor de la trama correctamente, los cuales son desplazados a la posición correspondiente y transformados nuevamente al valor ASCII que le corresponde, seguidamente son almacenados hasta el momento de su transmisión. Tanto la transmisión como la recepción han sido explicadas en los

apartados anteriores. A continuación se explican las funciones que utiliza para llevar a cabo la conversión:

- **Rs232_to_Rs485()** → Separa en 8 bits los valores de la trama recibidos por Rs232.

6.3.5. SCADA

En este apartado se muestran las funciones necesarias para el funcionamiento correcto de la pantalla TFT que hace el efecto de SCADA en este proyecto. A continuación se explican las funciones que utiliza para llevar a cabo el SCADA.

- **inicio()** → Configura la apariencia de la pantalla y el título de inicio.
- **menu()** → Configura la pantalla de menú de la pantalla TFT.
- **P01()** → Corresponde a la opción dejar. Incluye las funciones necesarias para configurar las diferentes pantallas para adquirir los datos necesarios para la opción de dejar, incluyendo la lectura de las botoneras y el keypad.
- **P02()** → Corresponde a la opción coger. Incluye las funciones necesarias para configurar las diferentes pantallas para adquirir los datos necesarios para la opción de coger, incluyendo la lectura de las botoneras y el keypad.
- **P03()** → Corresponde a la opción leer memoria. Incluye las funciones necesarias para configurar las diferentes pantallas para adquirir los datos necesarios para la opción de leer memoria, incluyendo la lectura de las botoneras y el keypad.
- **P04()** → Corresponde a la opción borrar memoria. Incluye las funciones necesarias para configurar las diferentes pantallas para adquirir los datos necesarios para la opción de borrar memoria, incluyendo la lectura de las botoneras y el keypad.
- **P05()** → Corresponde a la opción borrar toda la EEPROM. Incluye las funciones necesarias para configurar las diferentes

pantallas para adquirir los datos necesarios para la opción de borrar toda la EEPROM, incluyendo la lectura de las botoneras y el keypad.

- **P06()** → Corresponde a la opción escribir en memoria. Incluye las funciones necesarias para configurar las diferentes pantallas para adquirir los datos necesarios para la opción de escribir en memoria, incluyendo la lectura de las botoneras y el keypad.
- **colum()** → Espera a que se escriba un valor de columna y comprueba que este dentro del rango correspondiente.
- **fil()** → Espera a que se escriba un valor de fila y comprueba que este dentro del rango correspondiente.

Además de las funciones principales anteriormente descritas, también incluye funciones secundarias que ayudan a adquirir los datos necesarios y a moverse entre las diferentes pantallas de una misma opción, en la *Tabla 2* se muestran los subprogramas e cada uno de los programas principales:

P01()	**P11()** **P12()** **P13()**
P02()	P21() P22() P23()
P03()	P31() P32() P33() P34()
P04()	P41() P42() P43()
P05()	P51()
P06()	P61() P62() P63()

Tabla 2. Funciones pantalla

7. Presupuesto

- Materiales

MATERIAL	Unidades	Precio/ud (€)	Coste (€)
Arduino® Mega	4	35	140
Pantalla TFT	1	12,66	12,66
Max232	4	3,48	13,92
Max485	4	4,54	18,16
DB9 Hembra	4	0,43	1,72
DB9 Macho	4	0,43	1,72
Cables varios	1	19	19
Pulsador	1	0,91	0,91
Interruptor 2 posiciones	2	0,46	0,92
Botonera 4	2	1	2
KeyPad	1	1,6	1,6
		TOTAL	**212,61**

Tabla 3. Presupuesto materiales

- Mano de Obra

PERSONAL	Horas	€/Hora	Coste (€)
Mano de obra de un Graduado en Ingeniería	**300**	**11,50**	**3.450**
Envío	*50*	*11,50*	*575*
Recepción	*50*	*11,50*	*575*
Pasarela	*100*	*11,50*	*1.150*
SCADA	*100*	*11,50*	*1.150*
Técnico de laboratorio	**100**	**9,5**	**950**
		TOTAL	**4.400**

Tabla 4. Presupuesto mano de obra

- Coste total del proyecto

	Coste (€)
Materiales	212,61
Mano de Obra	4.400
TOTAL	**4.612,61**

Tabla 5. Presupuesto total

8. Conclusiones

En primer lugar se ha de decir que se partía casi desde cero con respecto a los conocimientos de comunicación. Durante el período de duración de la carrera es una rama de conocimiento que solo se ha estudiado en una asignatura y aplicado a los microcontroladores de la familia **C2000**. A pesar de ello y con diversas herramientas como internet y referencias de otros proyectos se han podido adquirir los conocimientos básicos para la realización de este proyecto.

Los aspectos positivos de no tener conocimientos previos ha sido el aprendizaje. Se ha adquirido conocimientos en varias áreas aparte de la comunicación, tampoco vistas en la carrera, como son SCADA (controlador de visualización) y Arduino® (microcontrolador).

Uno de los grandes retos del proyecto era unir las 2 partes, la parte de la comunicación y la parte mecánica. A medida que se iba realizando el proyecto era necesario el feedback con el compañero ya que había que coordinar ambos proyectos para que funcionaran una vez juntos. Ambos proyectos se han podido unir finalmente sin inconvenientes por lo que se ha podido realizar el almacén automatizado planeado inicialmente.

Adicionalmente a lo anteriormente mencionado, otra de las mayores dificultades a superar en el trabajo, ha sido la realización de la programación que conmuta entre 2 comunicaciones diferentes.

Finalmente, con un presupuesto más elevado y más tiempo dedicado al perfeccionamiento de su diseño este almacén automatizado a gran escala podría competir con los que actualmente se encuentran instalados en numerosas factorías multinacionales.

9. Líneas de investigación futura

A continuación se exponen las posibles líneas de investigación en un futuro del proyecto:

- Se podría estudiar el cambio del microcontrolador a uno más potente, como una tarjeta de adquisición de datos con un microcontrolador STM o un PLC, que es lo que se utiliza en la industria ya que el Arduino® no es suficiente en cuanto a velocidad y memoria en el caso de ampliar el proyecto.

- Existe la posibilidad de ampliar la cantidad de comunicaciones a conmutar en la zona de la pasarela con lo que se podría fabricar un conmutador universal de protocolos de comunicaciones industriales.

- Se podría implementar una mejora en el sistema SCADA aplicándolo a ordenadores o pantallas táctiles en vez de a pantallas TFT, ya que el problema de la pantalla TFT es que necesita una botonera externa. Esta mejora se podría realizar a través de programas como Labview o similares.

10. Valoración personal

Este proyecto ha significado un gran reto para mí ya que inicialmente no poseía conocimientos de comunicación, SCADA o Arduino®. Durante el proyecto he podido enriquecer mis conocimientos más allá de los aprendidos en la carrera.

En mis circunstancias particulares ha sido difícil compaginar el trabajo con la realización del proyecto ya que mi puesto de trabajo requiere de una gran cantidad de horas de dedicación diarias pero a pesar de ello y con la ayuda de mi compañero he podido terminar el proyecto y culminar la idea principal del mismo.

Valoro muy positivamente el haber trabajado con Alexander Rosales en el proyecto ya que entre ambos hemos ido aprendiendo nuevas ramas de conocimiento y hemos compartido los conocimientos que ambos poseíamos previamente al proyecto.

En definitiva, creo que este proyecto me ha enriquecido tanto personal como académicamente, y gracias a estos sistemas automatizados hemos podido tener una guía a seguir, la cual nos ha ayudado a la fabricación de este proyecto.

11. Bibliografía

Rs232 y Rs485

http://www.puntoflotante.net/RS485.htm [Consulta: 2 de agosto de 2016]

Profibus

http://www.aie.cl/files/file/comites/ca/articulos/agosto-06.pdf [Consulta: 3 de agosto de 2016]

http://www.santiagoapostol.net/srca/buses/profibus.pdf [Consulta: 2 de agosto de 2016]

http://www.smar.com/espanol/profibus [Consulta: 2 de agosto de 2016]

http://www.rtaautomation.com/technologies/profibus/ [Consulta: 4 de agosto de 2016]

http://docente.ucol.mx/al950441/public_html/osi1hec_B.htm [Consulta: 27 de julio de 2016]

http://www.etitudela.com/celula/downloads/2profibus.pdf [Consulta: 3 de agosto de 2016]

Modbus

http://www.aie.cl/files/file/comites/ca/articulos/agosto-06.pdf [Consulta: 8 de agosto de 2016]

http://www.ni.com/white-paper/52134/en/ [Consulta: 5 de agosto de 2016]

DeviceNet

http://www.aie.cl/files/file/comites/ca/articulos/agosto-06.pdf [Consulta: 9 de agosto de 2016]

http://ecatalog.weg.net/files/wegnet/WEG-ssw08-manual-de-comunicacion-devicenet-10000046974-manual-espanol.pdf [Consulta: 9 de agosto de 2016]

http://www.emb.cl/electroindustria/articulo.mvc?xid=117 [Consulta: 9 de agosto de 2016]

http://read.pudn.com/downloads166/ebook/763211/EIP-CIP-V1-1.0.pdf [Consulta: 8 de agosto de 2016]

http://www.ni.com/white-paper/2732/en/ [Consulta: 12 de agosto de 2016]

Ethernet

http://mixteco.utm.mx/~resdi/historial/materias/IPv4.pdf [Consulta: 11 de agosto de 2016]

http://es.ccm.net/contents/253-lan-red-de-area-local [Consulta: 11 de agosto de 2016]

Arduino®

<https://aprendiendoarduino®.wordpress.com/2016/03/29/entorno-de-programacion-de-arduino®-ide/> [Consulta: 13 de agosto de 2016]

<http://hacedores.com/cuantos-tipos-diferentes-de-arduino®-hay/> [Consulta: 15 de agosto de 2016]

SCADA

http://control-accesos.es/scada/%C2%BFque-es-un-sistema-scada [Consulta: 12 de agosto de 2016]

MAX 232

http://www.ti.com/lit/ds/symlink/max232.pdf [Consulta: 14 de agosto de 2016]

MAX 485

<http://www.naylampmechatronics.com/blog/37_Comunicaci%C3%B3n-RS485-con-Arduino®.html> [Consulta: 13 de agosto de 2016]

12. Anexos

12.1. Código.

12.1.1. Envío.

```
////////////////////////////////////////////// MAIN //////////////////////////////////////////////
/*Variables del programa*/
int i = 0;
int n = 0;
int x = 0;
int z = 0;
int a = 0;
int b = 0;
int c = 0;
int d = 0;
int Tipo_In = 3;
int In = 0;
int Inicio = 0;
int Fin = 0;
int TT = 79;
int Res = 0;
int Res_Ok = 0;
int Rx_res = ' ';

/*Variables de comprobacion*/
int Ac[6] = {'A', 'B', 'C', 'D', 'E', 'F'}; //C = Coger   - D = Dejar
int row = '5';
int column = '8';

/*Variables de Recepcion*/
int cont = 0;
int cont1 = 0;
int Chk_In = 0;
int C_Trama = 0;

int Rx_232[4] = {' ', ' ', ' ', ' '};

int num = 0;
int num1 = 0;
int num2 = 0;

int E[8] = {247, 251, 253, 254, 247, 251, 253, 254};
int E0[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E1[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E2[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E3[8] = {0, 0, 0, 0, 0, 0, 0, 0};

int Rx_485[8] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};
int Rxt[4] = {255, 255, 255, 255};
```

```
int RxtT[4] = {255, 255, 255, 255};

int S = 0;

/*Variables de Envio*/
char Tx[3] = {' ', ' ', ' '};
int D = 0;
int Tx_485[8] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};

/*Entradas del switch*/
int In_Switch[2] = {2, 3};
int enable = 6;
int boton = 7;

void setup() {
  pinMode(enable, OUTPUT);
  pinMode(boton, INPUT);
  for (i = 0; i < 2; i++) {
    pinMode(In_Switch[i], INPUT);
  }
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(9600);
  Serial3.begin(9600);
  Serial1.setTimeout(1000);
  Serial2.setTimeout(1000);
}

void loop() {
  Serial1.flush();
  Serial2.flush();
  while (b == 0) {
    if (Fin == 0) {
      Serial.println("Configura el tipo de entrada y salida y pulsa el boton");
      Fin = 1;
    }

    Inicio = digitalRead(boton);
    if (Inicio == LOW) {
      Serial.print("Inicio ");
      Tipo_In = entrada();
      Serial.print(Tipo_In);
      Serial.println(" ");
      b = 1;
    }
  }

  switch (Tipo_In) {
    case 0:
      //Entrada Rs232
```

```
    Serial.println("Tipo In Rs232");
    Recepcion_Rs232();
    Comprobacion_trama_envio_232();
    Chk_In = Calculo_Chk_In_232();
    Envio_Rs232();
    Comprobar_trama_ok_232();
    Esperar_Respuesta_232();
    Comprobar_errores_232();
    break;

   case 1:
    //Entrada Rs485
    Serial.println("Tipo In Rs485");
    Recepcion_Rs485();
    Comprobacion_trama_envio_485();
    Chk_In = Calculo_Chk_In_485();
    Conversion_485();
    Envio_Trama_Rs485();
    Comprobar_trama_ok_485();
    Esperar_Respuesta_485();
    Comprobar_errores_485();
    break;

   default:
    break;
  }
  Reset_var();
}

//////////////////////////////////////////////// Rs 232 ////////////////////////////////////////////
//-------------------------Recepcion Rs232-----------------------------//
void Recepcion_Rs232() {
  for (i = 0; i < 4; i++) {
   Rx_232[i] = ' ';
  }
  cont = 0;

  while (cont <= 2) {
   while (Serial.available()) {
    //Serial.print("si");
    Rx_232[cont] = Serial.read();
    //Serial.write(Rx_232[cont]);
    delayMicroseconds(1000);
    cont++;
   }
  }
}
```

```
//-----------------Calculo del checksun de entrada----------------------
int Calculo_Chk_In_232() {
  for (i = 0; i < 3; i++) {
    x = x ^ Rx_232[i];
  }

  Rx_232[3] = char(x);
  Rx_res = Rx_232[2];

  return x;
}

//-----------------Comprobacion Trama----------------------------------
void Comprobacion_trama_envio_232() {
  for (i = 0; i < 3; i++) {
    switch (i) {

      case 0:
        if (((Rx_232[i]) <= '0') || ((Rx_232[i]) > column)) {
          Serial.print("El valor de columna no coincide, introduce otro(del 1 al ");
          Serial.write(column);
          Serial.print("):");
          while ((Rx_232[i] <= '0') || (Rx_232[i] > column)) {
            while (Serial.available()) {
              Rx_232[i] = Serial.read();
              Serial.write(Rx_232[i]);
              Serial.println(' ');
              if ((Rx_232[i] <= '0') || (Rx_232[i] > column)) {
                Serial.print("El valor de columna no coincide, introduce otro(del 1 al ");
                Serial.write(column);
                Serial.print("):");
              }
            }
          }
        }
        break;

      case 1:
        if ((Rx_232[i] <= '0') || (Rx_232[i] > row)) {
          Serial.print("El valor de fila no coincide, introduce otro(del 1 al ");
          Serial.write(row);
          Serial.print("):");
          while ((Rx_232[i] <= '0') || (Rx_232[i] > row)) {
            while (Serial.available()) {
              Rx_232[i] = Serial.read();
              Serial.write(Rx_232[i]);
              Serial.println(' ');
              if ((Rx_232[i] <= '0') || (Rx_232[i] > row)) {
                Serial.print("El valor de fila no coincide, introduce otro(del 1 al ");
                Serial.write(row);
```

```
        Serial.print("):");
       }
      }
     }
    }
    break;

  case 2:
    d = 0;
    for (n = 0; n < 6; n++) {
     if (Rx_232[i] != Ac[n]) {
      d++;
     } else {
      d = 0;
      n = 6;
     }
    }
    if (d > 0) {
     Serial.print("El valor de accion no coincide, introduce otro( ");
     for (n = 0; n < 6; n++) {
      Serial.write(Ac[n]);
      if (n != 5) {
       Serial.print(", ");
      }
     }
     Serial.print(" ): ");
     while (d > 0) {
      while (Serial.available()) {
       Rx_232[i] = Serial.read();
       Serial.write(Rx_232[i]);
       Serial.println(' ');
       d = 0;
       for (n = 0; n < 6; n++) {
        if (Rx_232[i] != Ac[n]) {
         d++;
        } else {
         d = 0;
         n = 6;
        }
       }
       if (d > 0) {
        Serial.print("El valor de accion no coincide, introduce otro( ");
        for (n = 0; n < 6; n++) {
         Serial.write(Ac[n]);
         if (n != 5) {
          Serial.print(", ");
         }
        }
        Serial.print(" ): ");
       }
```

```
        }
       }
      }
      break;

    default:
      break;
   }
  }
}

//----------------------------Envio Rs232-------------------------------
void Envio_Rs232() {
  Rx_232[0] = char(Rx_232[0]);
  Rx_232[1] = char(Rx_232[1]);
  Rx_232[2] = char(Rx_232[2]);

  for (i = 0; i < 4; i++) {
    Serial.write(Rx_232[i]);
    Serial1.write(Rx_232[i]);
    Serial1.flush();
  }
}

//-----------------Comprobacion Trama enviada correctamente---------------------------------
void Comprobar_trama_ok_232() {

  C_Trama = Recepcion_trama_ok_232();

  if (C_Trama == 79) {
    //Trama correcta
    Serial.println("OK");
  } else {
    //Trama incorrecta
    Serial.println("NOK");
    cont1 = 0;
    while ((C_Trama != 79) && (cont1 < 2)) {
      Envio_Rs232();
      C_Trama = Recepcion_trama_ok_232();
      if (C_Trama == 79) {
        Serial.println("OK");
      } else {
        Serial.println("NOK");
      }
      cont1++;
    }
    if (cont1 >= 2) {
      Serial.println("Trama imposible de enviar, consulte con el tecnico.");
    }
  }
```

```
}

//-------------------------Recepcion Comprobacion trama----------------------------
int Recepcion_trama_ok_232() {

  cont = 0;
  while (cont < 1) {
    while (Serial1.available()) {
      c = Serial1.read();
      delayMicroseconds(1000);
      cont++;
    }
  }
  return c;
}

//-----------------Espera la respuesta---------------------
void Esperar_Respuesta_232() {
  cont = 0;
  Serial.print("Esperando respuesta: ");
  while (cont < 1) {
    while (Serial1.available()) {
      Res = Serial1.read();
      Serial.write(Res);
      delayMicroseconds(1000);
      cont++;
    }
  }
  Serial.println(" ");
  Res_Ok = 1;
}

//-----------------Comprobacion errores ---------------------------------
void Comprobar_errores_232() {

  switch (Rx_res) {
    case 'A': /*Coger*/
      //Mandar 0 cuando acaba, sino mandar 1 para errores
      if (Res == '0') {
        Serial.println("Finalizado con exito");
      } else if (Res == '1') {
        Serial.println("error de posición ocupada, elija otra posicion");
      }

      break;

    case 'B': /*Dejar*/
      //Mandar 0 cuando acaba, sino mandar 1 para errores
      if (Res == '0') {
        Serial.println("Finalizado con exito");
```

```
    } else if (Res == '1') {
      Serial.println("error de posición libre, elija otra posicion");
    }
    break;

  case 'C': /*Leer*/
    //Mandar 0 para libre y 1 para ocupado
    if (Res == '0') {
      Serial.println("Posición libre");
    } else if (Res == '1') {
      Serial.println("Posición ocupada");
    }
    break;

  case 'D': /*Borrar*/
    //Mandar 0 cuando acaba de borrar
    if (Res == '0') {
      Serial.println("Borrado completado");
    }
    break;

  case 'E': /*Borrar todo*/
    //Mandar 0 cuando acaba de borrar
    if (Res == '0') {
      Serial.println("Borrado completado");
    }
    break;

  case 'F': /*Escribir*/
    //Mandar 0 cuando escribe
    if (Res == '0') {
      Serial.println("Escritura completada");
    }
    break;

  default:
    Serial.println("No se conoce el error");
    break;
 }
}

//////////////////////////////////////////////// Rs 485 ////////////////////////////////////////////
//--------------------------Recepcion Rs485-----------------------------
void Recepcion_Rs485() {

 Serial.flush();

 while (cont < 3) {
  while (Serial.available()) {
    if (cont == 0) {
```

```
      num = Serial.read();
      Serial.write(num);
    } else if (cont == 1) {
      num1 = Serial.read();
      Serial.write(num1);
    } else if (cont == 2) {
      num2 = Serial.read();
      Serial.write(num2);
    }
    cont++;
    delay(10);
  }
 }
}

//-----------------Conversión valor de entrada----------------------
void Conversion_485() {
 for (n = 0; n < 3; n++) {
  switch (n) {
    case 0:
      for (i = 0; i < 8; i++) {
        S = bitRead(num, (7 - i));
        E0[i] = S;
      }
      break;

    case 1:
      for (i = 0; i < 8; i++) {
        S = bitRead(num1, (7 - i));
        E1[i] = S;
      }
      break;

    case 2:
      Serial.println(" ");
      for (i = 0; i < 8; i++) {
        S = bitRead(num2, (7 - i));
        E2[i] = S;
      }
      break;

    default:
      break;
  }
 }
}
```

```
//-----------------Calculo del checksun de entrada---------------------
int Calculo_Chk_In_485() {
  x = x ^ num;
  x = x ^ num1;
  x = x ^ num2;
  Serial.write(x);
  Serial.println(' ');

  for (i = 0; i < 8; i++) {
    S = bitRead(x, (7 - i));
    E3[i] = S;
  }

  Rx_res = num2;

  return x;
}

//-----------------Muestra Trama---------------------------------
void Mostrar_trama_485(int a) {
  Serial.write(num);
  Serial.write(num1);
  Serial.write(num2);
  Serial.write(a);
}

//-----------------Comprobacion Trama----------------------------------
void Comprobacion_trama_envio_485() {

  for (i = 0; i < 3; i++) {
    switch (i) {

      case 0:
        if ((num <= '0') || (num > column)) {
          Serial.println(" ");
          Serial.print("El valor de columna no coincide, introduce otro(del 1 al ");
          Serial.write(column);
          Serial.print("):");
          while ((num <= '0') || (num > column)) {
            while (Serial.available()) {
              num = Serial.read();
              Serial.write(num);
              if ((num <= '0') || (num > column)) {
                Serial.print("El valor de columna no coincide, introduce otro(del 1 al ");
                Serial.write(column);
                Serial.print("):");
              }
            }
          }
        }
```

```
  break;

 case 1:
  if ((num1 <= '0') || (num1 > row)) {
   Serial.println(" ");
   Serial.print("El valor de fila no coincide, introduce otro(del 1 al ");
   Serial.write(row);
   Serial.print("):");
   while ((num1 <= '0') || (num1 > row)) {
    while (Serial.available()) {
     num1 = Serial.read();
     Serial.write(num1);
     if ((num1 <= '0') || (num1 > row)) {
      Serial.print("El valor de fila no coincide, introduce otro(del 1 al ");
      Serial.write(row);
      Serial.print("):");
     }
    }
   }
  }
  break;

 case 2:
  d = 0;
  for (n = 0; n < 6; n++) {
   if (num2 != Ac[n]) {
    d++;
   } else {
    d = 0;
    n = 6;
   }
  }
  if (d > 0) {
   Serial.println(" ");
   Serial.print("El valor de accion no coincide, introduce otro( ");
   for (n = 0; n < 6; n++) {
    Serial.write(Ac[n]);
    if (n != 5) {
     Serial.print(", ");
    }
   }
   Serial.print(" ): ");
   while (d > 0) {
    while (Serial.available()) {
     num2 = Serial.read();
     Serial.write(num2);
     Serial.println(" ");
     d = 0;
     for (n = 0; n < 6; n++) {
      if (num2 != Ac[n]) {
```

```
            d++;
           } else {
            d = 0;
            n = 6;
           }
          }
          if (d > 0) {
           Serial.print("El valor de accion no coincide, introduce otro( ");
           for (n = 0; n < 6; n++) {
            Serial.write(Ac[n]);
            if (n != 5) {
             Serial.print(", ");
            }
           }
           Serial.print(" ): ");
          }
         }
        }
       }
       break;

     default:
       break;
   }
  }
}

//----------------Adaptacion Envio Rs485-------------------------------
void Adaptacion_Envio_485(int a) {

  for (i = 0; i < 8; i++) {
    switch (i) {
      case 0:
        if (a == 0) {
          Tx_485[i] = E0[i];
        } else if (a == 1) {
          Tx_485[i] = E1[i];
        } else if (a == 2) {
          Tx_485[i] = E2[i];
        } else if (a == 3) {
          Tx_485[i] = E3[i];
        }
        break;

      case 1:
        if (a == 0) {
          Tx_485[i] = E0[i];
        } else if (a == 1) {
          Tx_485[i] = E1[i];
        } else if (a == 2) {
```

```
    Tx_485[i] = E2[i];
  } else if (a == 3) {
    Tx_485[i] = E3[i];
  }
  break;

case 2:
  if (a == 0) {
    Tx_485[i] = E0[i];
  } else if (a == 1) {
    Tx_485[i] = E1[i];
  } else if (a == 2) {
    Tx_485[i] = E2[i];
  } else if (a == 3) {
    Tx_485[i] = E3[i];
  }
  break;

case 3:
  if (a == 0) {
    Tx_485[i] = E0[i];
  } else if (a == 1) {
    Tx_485[i] = E1[i];
  } else if (a == 2) {
    Tx_485[i] = E2[i];
  } else if (a == 3) {
    Tx_485[i] = E3[i];
  }
  break;

case 4:
  if (a == 0) {
    Tx_485[i] = E0[i];
  } else if (a == 1) {
    Tx_485[i] = E1[i];
  } else if (a == 2) {
    Tx_485[i] = E2[i];
  } else if (a == 3) {
    Tx_485[i] = E3[i];
  }
  break;

case 5:
  if (a == 0) {
    Tx_485[i] = E0[i];
  } else if (a == 1) {
    Tx_485[i] = E1[i];
  } else if (a == 2) {
    Tx_485[i] = E2[i];
  } else if (a == 3) {
```

```
        Tx_485[i] = E3[i];
       }
       break;

     case 6:
       if (a == 0) {
         Tx_485[i] = E0[i];
       } else if (a == 1) {
         Tx_485[i] = E1[i];
       } else if (a == 2) {
         Tx_485[i] = E2[i];
       } else if (a == 3) {
         Tx_485[i] = E3[i];
       }
       break;

     case 7:
       if (a == 0) {
         Tx_485[i] = E0[i];
       } else if (a == 1) {
         Tx_485[i] = E1[i];
       } else if (a == 2) {
         Tx_485[i] = E2[i];
       } else if (a == 3) {
         Tx_485[i] = E3[i];
       }
       break;

     default:
       break;
    }
  }
}

//---------------------------Envio Trama Rs485---------------------------
void Envio_Trama_Rs485() {
  Envio_inicio();
  a = 0;
  while (a < 4) {
    Adaptacion_Envio_485(a);
    Envio_Rs485();
  }
}

//---------------------------Envio Inicio Rs485---------------------------
void Envio_inicio() {

  digitalWrite(enable, HIGH);
  Serial2.print(D);
  Serial2.flush();
```

```
}

//----------------------------Envio Rs485-------------------------------
int Envio_Rs485() {

  for (i = 0; i < 8; i++) {
    digitalWrite(enable, HIGH);
    Serial2.print(Tx_485[i]);
    Serial2.flush();
  }
  a = a + 1;
  return a;
}

//-----------------Comprobacion Trama enviada correctamente---------------------------------
void Comprobar_trama_ok_485() {

  digitalWrite(enable, LOW);
  Recepcion_trama_ok_485();
  Conversion_recep_485();

  if (RxtT[0] == 79) {
    //Trama correcta
    Serial.println(" ");
    Serial.println("OK");
  } else {
    //Trama incorrecta
    Serial.println(" ");
    Serial.println("NOK");
    cont1 = 0;
    while ((RxtT[0] != 79) && (cont1 < 2)) {
      Envio_Trama_Rs485();
      Recepcion_trama_ok_485();
      Conversion_recep_485();
      if (RxtT[0] == 79) {
        Serial.println(" ");
        Serial.println("OK");
      } else {
        Serial.println(" ");
        Serial.println("NOK");
      }
      cont1++;
    }
    if (cont1 >= 2) {
      Serial.println("Trama imposible de enviar, consulte con el tecnico.");
    }
  }
}
```

```
//-------------------------Recepcion Comprobacion trama-----------------------------
void Recepcion_trama_ok_485() {

  cont = 0;
  z = 0;

  while (z < 2) {

    if ((Serial2.available()) && (z == 0)) {
      D = Serial2.read();
      z = z + 1;
    }

    if ((Serial2.available()) && (cont < 8) && (z == 1)) {
      Rx_485[cont] = Serial2.read();
      cont++;
      if (cont == 8) {
        z = z + 1;
        cont = 0;
      }
    }
  }
}

//-----------------Conversión valor de entrada---------------------
void Conversion_recep_485() {
  for (i = 0; i < 8; i++) {
    Rx_485[i] = Rx_485[i] & 1;
    Rx_485[i] = Rx_485[i] << (7 - i);
    Rxt[0] = Rxt[0] ^ Rx_485[i];
  }
  RxtT[0] = Rxt[0];
  RxtT[0] = RxtT[0] ^ 255;
  Serial.write(RxtT[0]);
}

//-----------------Espera la respuesta---------------------
void Esperar_Respuesta_485() {
  cont = 0;
  Rxt[0] = 255;
  digitalWrite(enable, LOW);
  Serial.print("Esperando respuesta: ");
  while (cont < 1) {
    Recepcion_trama_ok_485();
    Conversion_recep_485();
    Res = RxtT[0];
    cont++;
  }
  Serial.println(" ");
}
```

```
//-----------------Comprobacion errores ---------------------------------
void Comprobar_errores_485() {

  switch (Rx_res) {
    case 'A': /*Coger*/
      //Mandar 0 cuando acaba, sino mandar 1 para errores
      if (Res == '0') {
        Serial.println("Finalizado con exito");
      } else if (Res == '1') {
        Serial.println("error de posición ocupada, elija otra posicion");
      }
      break;

    case 'B': /*Dejar*/
      //Mandar 0 cuando acaba, sino mandar 1 para errores
      if (Res == '0') {
        Serial.println("Finalizado con exito");
      } else if (Res == '1') {
        Serial.println("error de posición libre, elija otra posicion");
      }
      break;

    case 'C': /*Leer*/
      //Mandar 0 para libre y 1 para ocupado
      if (Res == '0') {
        Serial.println("Posición libre");
      } else if (Res == '1') {
        Serial.println("Posición ocupada");
      }
      break;

    case 'D': /*Borrar*/
      //Mandar 0 cuando acaba de borrar
      if (Res == '0') {
        Serial.println("Borrado completado");
      }
      break;

    case 'E': /*Borrar todo*/
      //Mandar 0 cuando acaba de borrar
      if (Res == '0') {
        Serial.println("Borrado completado");
      }
      break;

    case 'F': /*Escribir*/
      //Mandar 0 cuando acaba de leer
      if (Res == '0') {
        Serial.println("Escritura completada");
      }
```

```
    break;
    default:
    Serial.println("No se conoce el error");
    break;
  }
}

//////////////////////////////////////////////// funciones //////////////////////////////////////////
//----------------------------Tipo de entrada----------------------------------------------
int entrada() {
  for (i = 0; i < 2; i++) {
    In = digitalRead(In_Switch[i]);
    if (In == 1) {
      return i;
    }
  }
}

//----------------------------Resetea Variables--------------------------------------------
void Reset_var() {
  n = 0;
  x = 0;
  z = 0;
  a = 0;
  b = 0;
  c = 0;
  d = 0;
  Tipo_In = 3;
  In = 0;
  Inicio = 0;
  Fin = 0;
  TT = 79;
  Res = 0;
  Res_Ok = 0;
  Rx_res= ' ';

  /*Variables de Recepcion*/
  cont = 0;
  cont1 = 0;
  Chk_In = 0;
  C_Trama = 0;

  num = 0;
  num1 = 0;
  num2 = 0;
  S = 0;
  D = 0;

  for (i = 0; i < 3; i++) {
    Tx[i] = ' ';
```

```
 }

 for (i = 0; i < 4; i++) {
  Rx_232[i] = ' ';
  Rxt[i] = 255;
  RxtT[i] = 255;
 }

 for (i = 0; i < 8; i++) {
  E0[i] = 0;
  E1[i] = 0;
  E2[i] = 0;
  E3[i] = 0;
  Rx_485[i] = ' ';
  Tx_485[i] = ' ';
 }

 i = 0;
 digitalWrite(enable, HIGH);
}
```

12.1.2. Pasarela

```
////////////////////////////////////////////// MAIN //////////////////////////////////////////////
/*Variables del programa*/
int i = 0;
int n = 0;
int x = 0;
int z = 0;
int a = 0;
int b = 0;
int c = 0;
int Tipo_In = 0;
int In = 0;
int Tipo_Out = 0;
int Out = 0;
int Inicio = 0;

/*Variables de Recepcion*/
int cont = 0;
int cont1 = 0;
int Chk_In = 0;
int Chk_Out = 0;
int Ok = 0;
int Ok_C = 0;
int C_Trama = 0;
```

```
int T_OK = ' ';
int Res_Ok = 0;
char Res = ' ';

int Rx_232[4] = {' ', ' ', ' ', ' '};

int E[8] = {247, 251, 253, 254, 247, 251, 253, 254};
int E0[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E1[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E2[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E3[8] = {0, 0, 0, 0, 0, 0, 0, 0};

int S = 0;
int S1 = 0;
int S2 = 0;
int S3 = 0;

int Rx_485[8] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};
int Rx1_485[8] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};
int Rx2_485[8] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};
int Rx3_485[8] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};

int Rxt[4] = {255, 255, 255, 255};
int RxtT[4] = {255, 255, 255, 255};

/*Variables de Envio*/
char Tx[3] = {' ', ' ', ' '};
int D = 0;
int Tx_485[8] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};


/*Entradas del switch*/
int In_Switch[2] = {2, 3};
int Out_Switch[2] = {4, 5};
int enable = 6;
int boton = 7;

void setup() {
  pinMode(enable, OUTPUT);
  pinMode(boton, INPUT);
  for (i = 0; i < 2; i++) {
    pinMode(In_Switch[i], INPUT);
    pinMode(Out_Switch[i], INPUT);
  }
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(9600);
  Serial3.begin(9600);
  digitalWrite(enable, LOW);
  Serial1.setTimeout(1000);
```

```
 Serial2.setTimeout(1000);
}

void loop() {
 Serial1.flush();
 Serial2.flush();

 while (b == 0) {
  Inicio = digitalRead(boton);
  if (Inicio == LOW) {
   Serial.print("Inicio ");
   Tipo_In = entrada();
   Serial.print(Tipo_In);
   Serial.print(" ");
   Tipo_Out = salida();
   Serial.println(Tipo_Out);
   Serial.println(" ");
   b = 1;
  }
 }

 switch (Tipo_In) {
  case 0:
   //Entrada Rs232
   Serial.println("Tipo In Rs232");
   Recepcion_Rs232();
   Chk_In = Calculo_Chk_In_232();
   Comprobar_trama_232();
   break;

  case 1:
   //Entrada Rs485
   Serial.println("Tipo In Rs485");
   Recepcion_Rs485();
   Conversion_485();
   Chk_In = Calculo_Chk_In_485();
   //delay(1000);
   Comprobar_trama_485();
   break;

  default:
   break;
 }

 if (Ok == 1) {
  switch (Tipo_Out) {
   case 0:
    //Salida Rs232
    Serial.println("Tipo Out Rs232");
    if (Tipo_In == 1) {
```

```
        Rs485_to_Rs232();
       }
       Envio_Rs232();
       Comprobar_trama_ok_232();
       if (Ok_C == 1) {
        Esperar_Respuesta_232();
       }
       break;

     case 1:
       //Salida Rs485
       Serial.println("Tipo Out Rs485");
       if (Tipo_In == 0) {
        Rs232_to_Rs485();
       }
       Envio_Trama_Rs485();
       Comprobar_trama_ok_485();
       if (Ok_C == 1) {
        Esperar_Respuesta_485();
       }
       break;
     default:
       break;
    }
  }

  if (Res_Ok == 1) {
    switch (Tipo_In) {
     case 0:
       Respuesta_trama_232();
       break;

     case 1:
       Respuesta_trama_485();
       break;

     default:
       break;
    }
  }
  Reset_var();
}

//////////////////////////////////////////// Rs 232 ////////////////////////////////////////
//-------------------------Recepcion Rs232----------------------------
void Recepcion_Rs232() {

  for (i = 0; i < 4; i++) {
    Rx_232[i] = ' ';
  }
```

```
  while (cont < 4) {
    while (Serial1.available()) {
      Rx_232[cont] = Serial1.read();
      delayMicroseconds(1000);
      Serial.write(Rx_232[cont]);
      cont++;
    }
  }
}

//-----------------Calculo del checksun de entrada----------------------
int Calculo_Chk_In_232() {
  for (i = 0; i < 3; i++) {
    x = x ^ Rx_232[i];
  }
  return x;
}

//-----------------Comprobacion Trama---------------------------------
void Comprobar_trama_232() {

  if (Chk_In == Rx_232[3]) {
    //Trama correcta
    Serial.println(" OK");
    Serial1.print('O');
    Ok = 1;
  } else {
    //Trama incorrecta
    Serial.println(" NOK");
    Serial1.print('N');
    cont1 = 0;
    while ((Chk_In != Rx_232[3]) && (cont1 <= 2)) {
      Recepcion_Rs232();
      Chk_In = Calculo_Chk_In_232();
      if (Chk_In == Rx_232[3]) {
        Serial.println(" OK");
        Serial1.print('O');
        Ok = 1;
      } else {
        Serial.println(" NOK");
        Serial1.print('N');
      }
      cont1++;
    }
    if (cont1 >= 2) {
      Serial.println("Trama imposible de enviar, consulte con el tecnico.");
    }
  }
}
```

```
//-----------------Conversión valor de entrada----------------------
int  Rs485_to_Rs232() {
  for (i = 0; i < 4; i++) {
    Tx[i] = char(RxtT[i]);
  }
}

//----------------------------Envio Rs232-------------------------------
void Envio_Rs232() {

  Chk_Out = 0;

  for (i = 0; i < 3; i++) {
    Chk_Out = Chk_Out ^ Tx[i];
  }
  Tx[3] = char(Chk_Out);

  for (i = 0; i < 4; i++) {
    Serial.write(Tx[i]);
    Serial1.write(Tx[i]);
    Serial1.flush();
  }
  Serial.println(" ");
}

//-----------------Comprobacion Trama enviada correctamente----------------------------------
void Comprobar_trama_ok_232() {

  C_Trama = Recepcion_trama_ok_232();
  if (C_Trama == 79) {
    //Trama correcta
    Serial.println("OK");
    Ok_C = 1;
  } else {
    //Trama incorrecta
    Serial.println("NOK");
    cont1 = 0;
    while ((C_Trama != 79) && (cont1 <= 2)) {
      Envio_Rs232();
      C_Trama = Recepcion_trama_ok_232();
      if (C_Trama == 79) {
        Serial.println("OK");
        Ok_C = 1;
      } else {
        Serial.println("NOK");
      }
      cont1++;
    }
    if (cont1 >= 2) {
      Serial.println("Trama imposible de enviar, consulte con el tecnico.");
```

```
   }
  }
}

//--------------------------Recepcion Comprobacion trama----------------------------
int Recepcion_trama_ok_232() {

  cont = 0;
  while (cont < 1) {
    while (Serial1.available()) {
      //Serial.print("si");
      c = Serial1.read();
      delayMicroseconds(1000);
      cont++;
    }
  }
  return c;
}

//-----------------Espera la respuesta---------------------
void Esperar_Respuesta_232() {

  cont = 0;
  Serial.print("Esperando respuesta: ");
  while (cont < 1) {
    while (Serial1.available()) {
      //Serial.print("si");
      Res = Serial1.read();
      Serial.println(Res);
      delayMicroseconds(1000);
      //Serial.write(Rx_232[cont]);
      cont++;
    }
  }
  Serial.println(" ");
  Res_Ok = 1;
}

//---------------------------Respuesta Rs232-------------------------------
void Respuesta_trama_232() {

  Serial1.print(Res);
  Serial1.flush();
}

////////////////////////////////////////////// Rs 485 //////////////////////////////////////////
//--------------------------Recepcion Rs485----------------------------
void Recepcion_Rs485(){

  for(i=0;i<8;i++){
```

```
   Rx_485[i]=' ';
   Rx1_485[i]=' ';
   Rx2_485[i]=' ';
   Rx3_485[i]=' ';
  }

  while(z<5){
   if((Serial2.available())&&(z==0)){
    D=Serial2.read();
    z=z+1;
   }

   if((Serial2.available())&&(cont<8)&&(z>=1)&&(z<5)){
    if(z==1){
     Rx_485[cont]=Serial2.read();
     cont++;
     if(cont==8){
      z=z+1;
      cont=0;
     }
    }else if(z==2){
     Rx1_485[cont]=Serial2.read();
     cont++;
     if(cont==8){
      z=z+1;
      cont=0;
     }
    }else if(z==3){
     Rx2_485[cont]=Serial2.read();
     cont++;
     if(cont==8){
      z=z+1;
      cont=0;
     }
    }else if(z==4){
     Rx3_485[cont]=Serial2.read();
     cont++;
     if(cont==8){
      z=5;
      cont=0;
     }
    }
   }
  }
}

//-----------------Conversión valor de entrada----------------------
int Conversion_485(){
  for(z=0;z<4;z++){
   for(i=0;i<8;i++){
```

```
    if(z==0){
      Rx_485[i]=Rx_485[i]&1;
      Rx_485[i]=Rx_485[i]<<(7-i);
      Rxt[z]=Rxt[z]^Rx_485[i];
    }else if(z==1){
      Rx1_485[i]=Rx1_485[i]&1;
      Rx1_485[i]=Rx1_485[i]<<(7-i);
      Rxt[z]=Rxt[z]^Rx1_485[i];
    }else if(z==2){
      Rx2_485[i]=Rx2_485[i]&1;
      Rx2_485[i]=Rx2_485[i]<<(7-i);
      Rxt[z]=Rxt[z]^Rx2_485[i];
    }else if(z==3){
      Rx3_485[i]=Rx3_485[i]&1;
      Rx3_485[i]=Rx3_485[i]<<(7-i);
      Rxt[z]=Rxt[z]^Rx3_485[i];
    }
   }
   RxtT[z]=Rxt[z];
   RxtT[z]=RxtT[z]^255;
   Serial.write(RxtT[z]);
  }
}

//-----------------Calculo del checksun de entrada---------------------
int Calculo_Chk_In_485(){
  for(i=0;i<3;i++){
    x = x^RxtT[i];
  }
    return x;
}

//-----------------Comprobacion Trama----------------------------------
void Comprobar_trama_485(){

  if(Chk_In == RxtT[3]){
    //Trama correcta
    Serial.println(" OK");
    Envio_trama_Ok('O');
    Ok=1;
  }else{
    //Trama incorrecta
    Serial.println(" NOK");
    Envio_trama_Ok('N');
    cont1=0;
    while((Chk_In != RxtT[3])&&(cont1<=2)){
      Recepcion_Rs485();
      Conversion_485();
      Chk_In = Calculo_Chk_In_485();
```

```
    if(Chk_In != RxtT[3]){
      Serial.println(" OK");
      Envio_trama_Ok('O');
      Ok=1;
    }else{
      Serial.println(" NOK");
      Envio_trama_Ok('N');
    }
    cont1++;
  }
  if(cont1>=2){
    Serial.println("Trama imposible de enviar, consulte con el tecnico.");
  }
 }
}

//-----------------Rs232 to Rs485----------------------
int Rs232_to_Rs485(){
 Serial.write(Rx_232[0]);
 Serial.write(Rx_232[1]);
 Serial.write(Rx_232[2]);
 Serial.write(Rx_232[3]);

 for(i=0;i<8;i++){
  E0[i] = bitRead(Rx_232[0],(7-i));
  E1[i] = bitRead(Rx_232[1],(7-i));
  E2[i] = bitRead(Rx_232[2],(7-i));
  E3[i] = bitRead(Chk_In,(7-i));
 }
}

//----------------Adaptacion Envio Rs485-------------------------------
void Adaptacion_Envio_485(int a){

 for(i=0;i<8;i++){
  switch (i){
    case 0:
     if(a==0){
      Tx_485[i]=E0[i];
     }else if(a==1){
      Tx_485[i]=E1[i];
     }else if(a==2){
      Tx_485[i]=E2[i];
     }else if(a==3){
      Tx_485[i]=E3[i];
     }
     break;

    case 1:
     if(a==0){
```

```
   Tx_485[i]=E0[i];
  }else if(a==1){
   Tx_485[i]=E1[i];
  }else if(a==2){
   Tx_485[i]=E2[i];
  }else if(a==3){
   Tx_485[i]=E3[i];
  }
  break;

case 2:
  if(a==0){
   Tx_485[i]=E0[i];
  }else if(a==1){
   Tx_485[i]=E1[i];
  }else if(a==2){
   Tx_485[i]=E2[i];
  }else if(a==3){
   Tx_485[i]=E3[i];
  }
  break;

case 3:
  if(a==0){
   Tx_485[i]=E0[i];
  }else if(a==1){
   Tx_485[i]=E1[i];
  }else if(a==2){
   Tx_485[i]=E2[i];
  }else if(a==3){
   Tx_485[i]=E3[i];
  }
  break;

case 4
  if(a==0){
   Tx_485[i]=E0[i];
  }else if(a==1){
   Tx_485[i]=E1[i];
  }else if(a==2){
   Tx_485[i]=E2[i];
  }else if(a==3){
   Tx_485[i]=E3[i];
  }
  break;

case 5:
  if(a==0){
   Tx_485[i]=E0[i];
  }else if(a==1){
```

```
          Tx_485[i]=E1[i];
        }else if(a==2){
          Tx_485[i]=E2[i];
        }else if(a==3){
          Tx_485[i]=E3[i];
        }
        break;

      case 6:
        if(a==0){
          Tx_485[i]=E0[i];
        }else if(a==1){
          Tx_485[i]=E1[i];
        }else if(a==2){
          Tx_485[i]=E2[i];
        }else if(a==3){
          Tx_485[i]=E3[i];
        }
        break;

      case 7:
        if(a==0){
          Tx_485[i]=E0[i];
        }else if(a==1){
          Tx_485[i]=E1[i];
        }else if(a==2){
          Tx_485[i]=E2[i];
        }else if(a==3){
          Tx_485[i]=E3[i];
        }
        break;

      default:
        break;
    }
  }
}

//----------------------------Envio Trama Rs485----------------------------
void Envio_Trama_Rs485(){

  Envio_inicio();
  a=0;
  while(a<4){
    Adaptacion_Envio_485(a);
    Envio_Rs485();
  }
}
```

```
//---------------------------Envio Inicio Rs485--------------------------
void Envio_inicio(){

  D=0;
  digitalWrite(enable,HIGH);
  Serial2.print(D);
  Serial2.flush();

}

//---------------------------Envio Rs485-------------------------------
int Envio_Rs485(){

    for(i=0;i<8;i++){
      digitalWrite(enable,HIGH);
      Serial2.print(Tx_485[i]);
      Serial2.flush();
    }
    a=a+1;
    return a;
}

//-----------------Comprobacion Trama enviada correctamente---------------------------------
void Comprobar_trama_ok_485(){

  digitalWrite(enable,LOW);
  Recepcion_trama_ok_485();
  Conversion_recep_485();

  if(RxtT[0] == 79){
    //Trama correcta
    Serial.println(" OK");
    Ok_C=1;
  }else{
    //Trama incorrecta
    Serial.println(" NOK");
    cont1=0;
    while((RxtT[0] != 79)&&(cont1<2)){
      Envio_Trama_Rs485();
      Recepcion_trama_ok_485();
      Conversion_recep_485();
      if(RxtT[0] == 79){
        Serial.println(" OK");
        Ok_C=1;
      }else{
        Serial.println(" NOK");
      }
      cont1++;
    }
    if(cont1>=2){
```

```
    Serial.println("Trama imposible de enviar, consulte con el tecnico.");
   }
  }
}

//-----------------Conversión valor de entrada---------------------
void Conversion_recep_485(){

 for(i=0;i<8;i++){
  Rx_485[i]=Rx_485[i]&1;
  Rx_485[i]=Rx_485[i]<<(7-i);
  Rxt[0]=Rxt[0]^Rx_485[i];
 }
 RxtT[0]=Rxt[0];
 RxtT[0]=RxtT[0]^255;
 Serial.write(RxtT[0]);
}

//-------------------------Recepcion Comprobacion trama----------------------------
void Recepcion_trama_ok_485(){

 cont=0;
 z=0;
 while(z<2){

  if((Serial2.available())&&(z==0)){
   D=Serial2.read();
   z=z+1;
  }

  if((Serial2.available())&&(cont<8)&&(z==1)){
   Rx_485[cont]=Serial2.read();
   cont++;
   if(cont==8){
    z=z+1;
    cont=0;
   }
  }
 }
}

//---------------------------Envio Trama Ok---------------------------
void Envio_trama_Ok(int T_OK){

 digitalWrite(enable,HIGH);
 for(i=0;i<8;i++){
  E0[i] = bitRead(T_OK,(7-i));
 }

 Envio_inicio();
```

```
  Adaptacion_Envio_485(0);
  Envio_Rs485();
  digitalWrite(enable,LOW);
}

//-----------------Espera la respuesta---------------------
void Esperar_Respuesta_485(){

  cont=0;
  Rxt[0]=255;
  digitalWrite(enable,LOW);
  Serial.print("Esperando respuesta: ");
  while(cont < 1){
    Recepcion_trama_ok_485();
    Conversion_recep_485();
    Res=RxtT[0];
    cont++;
  }
  Serial.println(" ");
  Res_Ok=1;
}

//---------------------------Respuesta Rs485-------------------------------
void Respuesta_trama_485(){

  for(i=0;i<8;i++){
    E0[i] = bitRead(Res,(7-i));
  }

  digitalWrite(enable,HIGH);
  Envio_inicio();
  Adaptacion_Envio_485(0);
  Envio_Rs485();
  digitalWrite(enable,LOW);
}

////////////////////////////////////////////// funciones ////////////////////////////////////////
//---------------------------Tipo de entrada-----------------------------------------------
int entrada(){
  for(i=0;i<2;i++){
    In=digitalRead(In_Switch[i]);
    if(In==1){
      return i;
    }
  }
}

//---------------------------Tipo de salida-----------------------------------------------
int salida(){
  for(i=0;i<2;i++){
```

```
   Out=digitalRead(Out_Switch[i]);
   if(Out==1){
    return i;
   }
  }
 }

//----------------------------Resetea Variables-------------------------------------------
void Reset_var(){
  Res=' ';
  Res_Ok=0;
  b=0;
  n=0;
  x=0;
  z=0;
  a=0;
  Tipo_In=0;
  In=0;
  Tipo_Out=0;
  Out=0;

  Inicio=0;
  cont=0;
  Chk_In=0;
  Chk_Out=0;
  Ok=0;
  C_Trama=0;

  S=0;
  S1=0;
  S2=0;
  S3=0;

  for(i=0;i<3;i++){
     Tx[i]=' ';
  }

  for(i=0;i<4;i++){
     Rxt[i]= 255;
     RxtT[i]= 255;
     Rx_232[i]=' ';
  }

  for(i=0;i<8;i++){

   E0[i] = 0;
   E1[i] = 0;
   E2[i] = 0;
   E3[i] = 0;
```

```
   Rx_485[i]=' ';
   Rx1_485[i]=' ';
   Rx2_485[i]=' ';
   Rx3_485[i]=' ';

   Tx_485[i]=' ';
  }
  i=0;
}
```

12.1.3. Recepción

```
////////////////////////////////////////////// MAIN ////////////////////////////////////////////
/*Variables del programa*/
int i=0;
int x=0;
int a=0;
int z=0;
int b=0;
int Tipo_Out=0;
int Out=0;
int Inicio=0;
int I=0;

/*Variables de Recepcion*/
int Ac[6]={'A','B','C','D','E','F'};
          /*A = Coger
            B = Dejar
            C = Leer posición
            D = Borrar posición
            E = Borrar toda la memoria
            F = Escribir una posición*/

int cont=0;
int cont1=0;
int Chk_Out=0;
int Ok=0;

int Rx_232[4]={' ',' ',' ',' '};
char Res=' ';

int E[8]={247, 251, 253, 254, 247, 251, 253, 254};
int E0[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E1[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E2[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int E3[8] = {0, 0, 0, 0, 0, 0, 0, 0};
```

```
int Rx_485[8]={' ',' ',' ',' ',' ',' ',' ',' '};
int Rx1_485[8]={' ',' ',' ',' ',' ',' ',' ',' '};
int Rx2_485[8]={' ',' ',' ',' ',' ',' ',' ',' '};
int Rx3_485[8]={' ',' ',' ',' ',' ',' ',' ',' '};

int Rxt[4]={255,255,255,255};
int RxtT[4]={255,255,255,255};

/*Variables de Envio*/
int D=0;
int Tx_485[8]={' ',' ',' ',' ',' ',' ',' ',' '};

/*Entradas del switch*/
int Out_Switch[2]={4,5};
int enable=6;
int boton=7;

void setup() {
  pinMode(enable,OUTPUT);
  pinMode(boton,INPUT);
  for(i=0;i<2;i++){
    pinMode(Out_Switch[i],INPUT);
  }
  Serial.begin(9600);  //Inicia el puerto Serie 0 //ok
  //Serial1.begin(9600); //Inicia el puerto Serie 1 //OJO coinciden interrupciones
  Serial2.begin(9600); //Inicia el puerto Serie 2//ok
  Serial3.begin(9600); //Inicia el puerto Serie 3 // ok
  Serial2.setTimeout(1000);
  Serial3.setTimeout(1000);
}

void loop() {
  Serial2.flush();
  Serial3.flush();
  /*Espera a que se pulse el boton de inicio*/
  while(b==0){
    Inicio=digitalRead(boton);
    if(Inicio == HIGH){
      Serial.print("Inicio ");
      Tipo_Out = salida(); //Lee el tipo de salida
      Serial.println(Tipo_Out);
      Serial.println(" ");
      b = 1;
    }
  }

  while (Serial2.available()) {
    D = Serial2.read();
  }
  while (Serial3.available()) {
```

```
   D = Serial3.read();
  }

  switch(Tipo_Out){ //Escoge entre salida por RS232 y RS485
   case 0:
    //Salida Rs232
    Serial.println("Tipo Out Rs232");
    Recepcion_Rs232(4); //Lee la trama
    Chk_Out = Calculo_Chk_Out_232(); //Calcula el Checksum
    Comprobar_trama_232(); //Comprueba que la trama recibida es correcta
    if(Ok==1){
     Mostrar_trama_232(); //Muestra la trama
     Respuesta_232(); //Espera la respuesta
     Respuesta_trama_232(); //Devuelve la respuesta
    }
    break;

   case 1:
    //Salida Rs485
    Serial.println("Tipo Out Rs485");
    Recepcion_Rs485(); //Lee la trama
    Conversion_485();  //Ordena la trama en Bytes
    Chk_Out = Calculo_Chk_Out_485(); //Cakcula el Checksum
    Comprobar_trama_485(); //Comprueba que la trama recibida es correcta
    if(Ok==1){
     Mostrar_trama_485(); //Muestra la trama
     Respuesta_485(); //Espera la respuesta
     Respuesta_trama_485(); //Devuelve la respuesta
    }

    break;

   default:
    break;
  }
  Reset_var(); //Resetea las variables
}

//////////////////////////////////////////////// Rs 232 //////////////////////////////////////////////
//--------------------------Recepcion Rs232-----------------------------
void Recepcion_Rs232(int a){
  cont=0;
  while(cont < a){
   while(Serial2.available()){
    Rx_232[cont]=Serial2.read();
    delayMicroseconds(1000);
    Serial.write(Rx_232[cont]);
    cont++;
   }
  }
```

```
}

//-----------------Calculo del checksun de entrada---------------------
int Calculo_Chk_Out_232(){
  for(i=0;i<3;i++){
    x = x^Rx_232[i];
  }
  return x;
}

//-----------------Comprobacion Trama---------------------------------
void Comprobar_trama_232(){
  if(Chk_Out == Rx_232[3]){
    //Trama correcta
    Serial.println(" OK");
    Serial2.print('O');
    Ok=1;
  }else{
    //Trama incorrecta
    Serial.println(" NOK");
    Serial2.print('N');
    cont1=0;
    while((Chk_Out != Rx_232[3])&&(cont1<=2)){
      Recepcion_Rs232(4);
      Chk_Out = Calculo_Chk_Out_232();
      if(Chk_Out == Rx_232[3]){
        Serial.println(" OK");
        Serial2.print('O');
        Ok=1;
      }else{
        Serial.println(" NOK");
        Serial2.print('N');
      }
      cont1++;
    }
    if(cont1>=2){
      Serial.println("Trama imposible de enviar, consulte con el tecnico.");
    }
  }
}

//-----------------Mostrar la trama----------------------
int  Mostrar_trama_232(){
  for(i=0;i<4;i++){
    Serial.write(Rx_232[i]);
  }
  Serial.println(' ');
}
```

```
//-----------------Espera la respuesta---------------------
void Esperar_Respuesta(){
 cont=0;
 Serial.print("Esperando respuesta: ");
 while(cont < 1){
  while(Serial.available()){
   Res=Serial.read();
   Serial.println(Res);
   delayMicroseconds(1000);
   cont++;
  }
 }
 Serial.println(" ");
}

//-----------------Asigna Respuesta---------------------
void Respuesta_232(){
 switch(Rx_232[2]){
  case 'A': /*Coger*/
   //Mandar 0 cuando acaba, sino mandar 1 para errores
   Esperar_Respuesta();
   break;

  case 'B': /*Dejar*/
   //Mandar 0 cuando acaba, sino mandar 1 para errores
   Esperar_Respuesta();
   break;

  case 'C': /*Leer*/
   //Mandar 0 para libre y 1 para ocupado
   Esperar_Respuesta();
   break;

  case 'D': /*Borrar*/
   //Mandar 0 cuando acaba de borrar
   Esperar_Respuesta();
   break;

  case 'E': /*Borrar todo*/
   //Mandar 0 cuando acaba de borrar
   Esperar_Respuesta();
   break;

  case 'F': /*Escribir*/
   //Mandar 0 cuando lea y espera el valor de escritura
   Esperar_Respuesta();
   break;
 }
}
```

```
//---------------------------Respuesta Rs232-------------------------------
void Respuesta_trama_232(){
 Serial2.print(Res);
}

////////////////////////////////////////////// Rs 485 ////////////////////////////////////////////
//---------------------------Recepcion Rs485-----------------------------
void Recepcion_Rs485() {

 for (i = 0; i < 8; i++) {
  Rx_485[i] = ' ';
  Rx1_485[i] = ' ';
  Rx2_485[i] = ' ';
  Rx3_485[i] = ' ';
 }
 i = 1;
 while (z < 5) {
  if ((Serial3.available()) && (z == 0) && (i == 1)) {
   D = Serial3.read();
   Serial3.flush();
   z = z + 1;
  }

  if ((Serial3.available()) && (cont < 8) && (z >= 1) && (z < 5)) {
   if (z == 1) {
    Rx_485[cont] = Serial3.read();
    Serial3.flush();
    delayMicroseconds(1500);
    cont++;
    if (cont == 8) {
     z = z + 1;
     cont = 0;
    }
   } else if (z == 2) {
    Rx1_485[cont] = Serial3.read();
    Serial3.flush();
    cont++;
    if (cont == 8) {
     z = z + 1;
     cont = 0;
    }
   } else if (z == 3) {
    Rx2_485[cont] = Serial3.read();
    Serial3.flush();
    cont++;
    if (cont == 8) {
     z = z + 1;
     cont = 0;
    }
   } else if (z == 4) {
```

```
      Rx3_485[cont] = Serial3.read();
      Serial3.flush();
      cont++;
      if (cont == 8) {
       z = 5;
       cont = 0;
      }
     }
    }
   }
  l++;
}

//-----------------Conversión valor de entrada----------------------
int Conversion_485() {

  for (z = 0; z < 4; z++) {
    for (i = 0; i < 8; i++) {
     if (z == 0) {
      /*Valor 1 de la trama*/
      Rx_485[i] = Rx_485[i] & 1;     //Convierte en numero binario
      Rx_485[i] = Rx_485[i] << (7 - i); //Desplaza a la posición correspondiente
      Rxt[z] = Rxt[z] ^ Rx_485[i];   //Agrupa los bits en un byte

     } else if (z == 1) {
      /*Valor 2 de la trama*/
      Rx1_485[i] = Rx1_485[i] & 1;   //Convierte en numero binario
      Rx1_485[i] = Rx1_485[i] << (7 - i); //Desplaza a la posición correspondiente
      Rxt[z] = Rxt[z] ^ Rx1_485[i];  //Agrupa los bits en un byte

     } else if (z == 2) {
      /*Valor 3 de la trama*/
      Rx2_485[i] = Rx2_485[i] & 1;   //Convierte en numero binario
      Rx2_485[i] = Rx2_485[i] << (7 - i); //Desplaza a la posición correspondiente
      Rxt[z] = Rxt[z] ^ Rx2_485[i];  //Agrupa los bits en un byte

     } else if (z == 3) {
      /*Valor 4 de la trama*/
      Rx3_485[i] = Rx3_485[i] & 1;   //Convierte en numero binario
      Rx3_485[i] = Rx3_485[i] << (7 - i); //Desplaza a la posición correspondiente
      Rxt[z] = Rxt[z] ^ Rx3_485[i];  //Agrupa los bits en un byte

     }
    }

    RxtT[z] = Rxt[z];     //Almacena el valor de la trama en una array
    RxtT[z] = RxtT[z] ^ 255; //Invierte el valor recibido
    Serial.write(RxtT[z]); //Transformar a ASCII
  }
}
```

```
//-----------------Calculo del checksun de entrada----------------------
int Calculo_Chk_Out_485() {
  for (i = 0; i < 3; i++) {
    x = x ^ RxtT[i];
  }
  return x;

}

//-----------------Comprobacion Trama---------------------------------
void Comprobar_trama_485() {
  if (Chk_Out == RxtT[3]) {
    //Trama correcta
    Serial.println(" OK");
    Envio_trama_Ok('O');
    Ok = 1;
  } else {
    //Trama incorrecta
    Serial.println(" NOK");
    Envio_trama_Ok('N');
    cont1 = 0;
    while ((Chk_Out != RxtT[3]) && (cont1 <= 2)) {
      Recepcion_Rs485();
      Conversion_485();
      Chk_Out = Calculo_Chk_Out_485();
      if (Chk_Out != RxtT[3]) {
        Serial.println(" OK");
        Envio_trama_Ok('O');
        Ok = 1;
      } else {
        Serial.println(" NOK");
        Envio_trama_Ok('N');
      }
      cont1++;
    }
    if (cont1 >= 2) {
      Serial.println("Trama imposible de enviar, consulte con el tecnico.");
    }
  }
}

//----------------------------Envio Trama Ok----------------------------
void Envio_trama_Ok(int T_OK) {
  digitalWrite(enable, HIGH);
  for (i = 0; i < 8; i++) {
    E0[i] = bitRead(T_OK, (7 - i));
  }
  Envio_inicio();
  Adaptacion_Envio_485(0);
  Envio_Rs485();
```

```
  digitalWrite(enable, LOW);
}

//-----------------Mostrar la trama---------------------
void  Mostrar_trama_485() {
  for (i = 0; i < 4; i++) {
    Serial.write(RxtT[i]);
  }
  Serial.println(' ');
}

//----------------------------Respuesta Rs485-------------------------------
void Respuesta_trama_485() {
  for (i = 0; i < 8; i++) {
    E0[i] = bitRead(Res, (7 - i)); //Descompone el valor en un 8 bits
  }
  digitalWrite(enable, HIGH);  //Habilita la transmisión de datos
  Envio_inicio();            //Envia el inicio de señal
  Adaptacion_Envio_485(0);    //Adapta la trama para su envío
  Envio_Rs485();             //Envia la trama
  digitalWrite(enable, LOW);  //Deshabilita la transmisión y habilita la recepción
}

//----------------------------Envio Inicio Rs485--------------------------
void Envio_inicio() {
  D = 0;
  Serial3.print(D);
  Serial3.flush();
}

//----------------------------Envio Rs485-------------------------------
void Envio_Rs485() {
  for (i = 0; i < 8; i++) {
    Serial3.print(Tx_485[i]);
    Serial3.flush();
  }
  a = a + 1;
}

//----------------Adaptacion Envio Rs485-------------------------------
void Adaptacion_Envio_485(int a) {
  /*Adapta la trama para su envío, tiene como argumento la posición de la trama a enviar*/
  for (i = 0; i < 8; i++) {
    switch (i) {
      case 0:
        if (a == 0) {
          Tx_485[i] = E0[i];
        } else if (a == 1) {
          Tx_485[i] = E1[i];
        } else if (a == 2) {
```

```
    Tx_485[i] = E2[i];
   } else if (a == 3) {
    Tx_485[i] = E3[i];
   }
   break;

 case 1:
   if (a == 0) {
    Tx_485[i] = E0[i];
   } else if (a == 1) {
    Tx_485[i] = E1[i];
   } else if (a == 2) {
    Tx_485[i] = E2[i];
   } else if (a == 3) {
    Tx_485[i] = E3[i];
   }
   break;

 case 2:
   if (a == 0) {
    Tx_485[i] = E0[i];
   } else if (a == 1) {
    Tx_485[i] = E1[i];
   } else if (a == 2) {
    Tx_485[i] = E2[i];
   } else if (a == 3) {
    Tx_485[i] = E3[i];
   }
   break;

 case 3:
   if (a == 0) {
    Tx_485[i] = E0[i];
   } else if (a == 1) {
    Tx_485[i] = E1[i];
   } else if (a == 2) {
    Tx_485[i] = E2[i];
   } else if (a == 3) {
    Tx_485[i] = E3[i];
   }
   break;

 case 4:
   if (a == 0) {
    Tx_485[i] = E0[i];
   } else if (a == 1) {
    Tx_485[i] = E1[i];
   } else if (a == 2) {
    Tx_485[i] = E2[i];
   } else if (a == 3) {
```

```
        Tx_485[i] = E3[i];
       }
       break;

     case 5:
       if (a == 0) {
        Tx_485[i] = E0[i];
       } else if (a == 1) {
        Tx_485[i] = E1[i];
       } else if (a == 2) {
        Tx_485[i] = E2[i];
       } else if (a == 3) {
        Tx_485[i] = E3[i];
       }
       break;

     case 6:
       if (a == 0) {
        Tx_485[i] = E0[i];
       } else if (a == 1) {
        Tx_485[i] = E1[i];
       } else if (a == 2) {
        Tx_485[i] = E2[i];
       } else if (a == 3) {
        Tx_485[i] = E3[i];
       }
       break;

     case 7:
       if (a == 0) {
        Tx_485[i] = E0[i];
       } else if (a == 1) {
        Tx_485[i] = E1[i];
       } else if (a == 2) {
        Tx_485[i] = E2[i];
       } else if (a == 3) {
        Tx_485[i] = E3[i];
       }
       break;

     default:
       break;
    }
  }
}

//-----------------Asigna Respuesta----------------------
void Respuesta_485() {
  switch (RxtT[2]) {
    case 'A': /*Coger*/
```

```
    //Mandar 0 cuando acaba, sino mandar 1 para errores
    Esperar_Respuesta();
    break;

  case 'B': /*Dejar*/
    //Mandar 0 cuando acaba, sino mandar 1 para errores
    Esperar_Respuesta();
    break;

  case 'C': /*Leer*/
    //Mandar 0 para libre y 1 para ocupado
    Esperar_Respuesta();
    break;

  case 'D': /*Borrar*/
    //Mandar 0 cuando acaba de borrar
    Esperar_Respuesta();
    break;

  case 'E': /*Borrar todo*/
    //Mandar 0 cuando acaba de borrar
    Esperar_Respuesta();
    break;

  case 'F': /*Escribir*/
    //Mandar 0 cuando lea y espera el valor de escritura
    Esperar_Respuesta();
    break;
 }
}

//--------------------------Recepcion Respuesta-----------------------------
void Recepcion_Escritura() {
 cont = 0;
 z = 0;
 while (z < 2) {
  /*Lee el inicio de la trama*/
  if ((Serial3.available()) && (z == 0)) {
   D = Serial3.read();
   z = z + 1;
  }
  /*Lee el primer valor de la trama*/
  if ((Serial3.available()) && (cont < 8) && (z == 1)) {
   Rx_485[cont] = Serial3.read();
   cont++;
   if (cont == 8) {
    z = z + 1;
    cont = 0;
   }
  }
```

```
  }
}

//-----------------Conversión valor de respuestaa--------------------
void Conversion_recep_485() { /*Agrupa el primer valor de la trama en un byte para su uso*/
  Rxt[0] = 255;
  for (i = 0; i < 8; i++) {
    Rx_485[i] = Rx_485[i] & 1;
    Rx_485[i] = Rx_485[i] << (7 - i);
    Rxt[0] = Rxt[0] ^ Rx_485[i];
  }
  RxtT[0] = Rxt[0];
  RxtT[0] = RxtT[0] ^ 255;
  Serial.write(RxtT[0]); //Transformar a ASCII
}


////////////////////////////////////////////// funciones ////////////////////////////////////////////
//----------------------------Tipo de salida----------------------------------------------
int salida(){

  for(i=0;i<2;i++){
    Out=digitalRead(Out_Switch[i]);
    if(Out==1){
      return i;
    }
  }
}

//----------------------------Resetea Variables-------------------------------------------
void Reset_var(){
  Res=' ';
  b=0;
  x=0;
  z=0;
  a=0;
  Tipo_Out=0;
  Out=0;

  Inicio=0;
  cont=0;
  Chk_Out=0;
  Ok=0;

  for(i=0;i<4;i++){
     Rxt[i]= 255;
     RxtT[i]= 255;
     Rx_232[i]=' ';
  }
```

```
 for(i=0;i<8;i++){

   E0[i] = 0;
   E1[i] = 0;
   E2[i] = 0;
   E3[i] = 0;

   Rx_485[i]=' ';
   Rx1_485[i]=' ';
   Rx2_485[i]=' ';
   Rx3_485[i]=' ';


   Tx_485[i]=' ';
 }
 i=0;
 while (Serial2.available()) {
   D = Serial2.read();
 }
 while (Serial3.available()) {
   D = Serial3.read();
 }
}
```

12.1.4. SCADA

```
////////////////////////////////////////////// MAIN ////////////////////////////////////////////
#include <Keypad.h>
#include <UTFT.h>

// Standard Arduino Mega shield: <display model>,38,39,40,41
UTFT    myGLCD(ILI9481, 38, 39, 40, 41);//tipo de display de la pantalla
extern uint8_t BigFont[]; // declaraciones de libreria
extern uint8_t SmallFont[];

//------------- Variables auxiliares-----------
int x; // auxiliar de for();(menu)
int menu = 0; //valor donde guarda la opccion menu
char columna = 0;// donde guarda columna
char fila = 0;// donde guarda fila
int ini = 0;
int posicion = 2;//para que sea disitnto de 0 o 1 // valor donde guarda la recepcion del dato = 0/1 de la
memoria
int rst = 0; // variable auxiliar para cuando apretas volver al menu

//------- COMUNICACION -------- // todo tuyo baby¡¡
char Rx[3] = {' ', ' ', ' '};
```

```
//int Rx_chk[2] = {0, 0};
int cont = 0;
char posx ; //posicion x COLUMNAS
char posy ; //posicion y FILAS

//------------- teclados numericos ----------------
////////////////////// TECLADO 1 ////////////////////////////
const byte filas = 4;         //teclado  de 4 filas
const byte columnas = 3;          //y 4 columnas
byte pins_filas[] = {A0, A1, A2, A3};     //Pines Arduino para las filas.
byte pins_columnas[] = { A4, A5, A6};     // Pines Arduino para las columnas.
char teclas [ filas ][ columnas ] = {
 {'1', '2', '3'},
 {'4', '5', '6'},
 {'7', '8', '9'},
 {'*', '0', '#'}
};
Keypad teclado1 = Keypad(makeKeymap(teclas), pins_filas, pins_columnas, filas, columnas);

////////////////// TECLADO 2 ////////////////////////////////
const byte filas2 = 1;         //teclado  de 1 filas
const byte columnas2 = 4;          //y 4 columnas
byte pins_filas2[] = { 8};     //Pines Arduino para las filas.
byte pins_columnas2[] = {11, 12, 9, 10};    // Pines Arduino para las columnas.
char teclas2 [ filas ][ columnas ] = {
 'A', 'B', 'C', 'D'
};
Keypad teclado2 = Keypad(makeKeymap(teclas2), pins_filas2, pins_columnas2, filas2, columnas2);

////////////////////////// TECLADO 3 ////////////////////////////
const byte filas3 = 1;         //teclado  de 1 filas
const byte columnas3 = 4;          //y 4 columnas
byte pins_filas3[] = { A12} ;  //Pines Arduino para las filas.
byte pins_columnas3[] = {A10, A11, A8, A9};      // Pines Arduino para las columnas.
char teclas3 [ filas ][ columnas ] = {
 'A', 'T', 'M', 'F'
};
Keypad teclado3 = Keypad(makeKeymap(teclas3), pins_filas3, pins_columnas3, filas3, columnas3);

void setup() {
 myGLCD.InitLCD(); // Inicializa la pantalla
 myGLCD.clrScr(); //limpia la pantalla

 // -----  E/S -------
 Serial.begin(9600);
 Serial.flush();

}
```

```
void loop() {

  Serial.println("INICIO");// para saber por donde va el cursor del programa

 if (ini == 0) {//para que la pantalla de inicio solo aparezca una vez
  inicio();
  //es el valor ascii de la letra B que corresponde a la tecla 4
  while (teclado2.getKey() != 66) {} //espera a apretar la tecla de inicio
  ini = 1;
  Serial.println("uno"); // para saber por donde va el cursor del programa
 }
 Serial.println("dos"); // para saber por donde va el cursor del programa
 MENU();//pantalla de menu
 while (menu < 65 || menu > 70 ) {
  if (menu == 0) {
   menu = teclado2.getKey(); // esto guarda un char/pero como esta declarado como int guarda un
numero
  }
  if (menu == 0) {
   menu = teclado3.getKey();
  }
 }

 switch  (menu) {
  case 65://P01 - 65 - A //
   Rx[0] = 'A'; //guarda la letra de la accion en la matriz que supongoque enviaremos, es igual en todos

   P01();
   colum();//numero de columna

   if (rst == 0) {// esto es para saltarselo si apretamos menu, esto no ejecutaria
    myGLCD.printNumI(columna - 48, 310, 120);//escribe en la pantalla el valor de columna //adapta el
valor de columna ascii a un DEC
    P11();
    fil(); //numero de fila

    if (rst == 0) {//igual si apretas menu esto no se hace
     myGLCD.printNumI(fila - 48, 280, 160);//escribe elvalor de fila en la pantalla
     P12();
     while (teclado3.getKey() != 70) {// espera a que apretes la tecla aceptar (boton8) //es el valor ascii
de la letra F que corresponde a la tecla 8
      if (teclado2.getKey() == 66) {// si apretas la tecla de menu condicion para resetear y salir del bucle
       rst = 1;
       break;
      }
     }
     if (rst == 0) {
      //si apretas reset esto se lo salta
      P13();
     }
```

```
   }
  }
  Serial.println(Rx[0]);
  Serial.println(Rx[1]);
  Serial.println(Rx[2]);
  //comprobacion de valores correctos de fila y columna
  Serial.println("FIN");
  break;

 case 66://P02 - 66 - B //
  Rx[0] = 'B';
  P02();
  colum();//numero de columna

  if (rst == 0) {
   myGLCD.printNumI(columna - 48, 310, 120);// escribe valor de columna en pantalla
   P21();
   fil();

   if (rst == 0) {
    myGLCD.printNumI(fila - 48, 280, 160);
    P22();
    while (teclado3.getKey() != 70) { //es el valor ascii de la letra F que corresponde a la tecla 8
     if (teclado2.getKey() == 66) {
      rst = 1;
      break;
     }
    }
    if (rst == 0) {
     P23();
    }
   }
  }
  Serial.println(Rx[0]);
  Serial.println(Rx[1]);
  Serial.println(Rx[2]);
  //comprobacion de valores correctos de fila y columna
  Serial.println("FIN");
  break;

 case 67: //P03 - 67 - C //
  Rx[0] = 'C';
  P03();
  colum();//numero de columna

  if (rst == 0) {
   myGLCD.printNumI(columna - 48, 310, 120);
   P31();
   fil();
```

```
  if (rst == 0) {
    myGLCD.printNumI(fila - 48, 280, 160);
    P32();
    while (teclado3.getKey() != 70) { //es el valor ascii de la letra F que corresponde a la tecla 8
      if (teclado2.getKey() == 66) {
        rst = 1;
        break;
      }
    }
    if (rst == 0) {
      Serial.flush();
      while (posicion == 2) {// espera a leer el valor de la posicion enviada llega 1 o 0
        if (Serial.available() > 0) {
          posicion = Serial.read() - 48;
        }
      }
      Serial.println(posicion);//sin ajuste de lineaAA¡¡¡ OJO
      if (posicion == 0) {
        P33();// bacia
      }
      if (posicion == 1) {
        P34();//ocupada
      }
      while (teclado2.getKey() != 66) {} //espera a apretar la tecla de menu
    }
  }
 }
 Serial.println(Rx[0]);
 Serial.println(Rx[1]);
 Serial.println(Rx[2]);
 //comprobacion de valores correctos de fila y columna
 Serial.println("FIN");
 break;

case 68: //P04 - 68 - D //
 Rx[0] = 'D';
 P04();
 colum();//numero de columna

 if (rst == 0) {
   myGLCD.printNumI(columna - 48, 310, 120);
   P41();
   fil();

   if (rst == 0) {
     myGLCD.printNumI(fila - 48, 280, 160);
     P42();
     while (teclado3.getKey() != 70) {//es el valor ascii de la letra F que corresponde a la tecla 8
       if (teclado2.getKey() == 66) {
         rst = 1;
```

```
        break;
       }
      }
      if (rst == 0) {
       P43();
       while (teclado2.getKey() != 66) {} //espera a apretar la tecla de menu
      }
     }
    }
    Serial.println(Rx[0]);
    Serial.println(Rx[1]);
    Serial.println(Rx[2]);
    // comprobacion de valores correctos de fila y columna
    Serial.println("FIN");
    break;

  case 69: //P05 - 69 - E //
    Rx[0] = 'E';
    P05();
    do {
     if (teclado2.getKey() == 66) {// menu
      rst = 1;
      break;
     }
    }
    while (teclado3.getKey() != 70);//aceptar

    if (rst == 0) {
     P51();
     while (teclado2.getKey() != 66) {} //espera a apretar la tecla de menu
    }
    Serial.println(Rx[0]);
    Serial.println(Rx[1]);
    Serial.println(Rx[2]);
    // comprobacion de valores correctos de fila y columna
    Serial.println("FIN");
    break;

  case 70: //P06 - 70 - F //
    Rx[0] = 'F';
    P06();
    colum();//numero de columna

    if (rst == 0) {
     myGLCD.printNumI(columna - 48, 310, 120);
     P61();
     fil();

     if (rst == 0) {
      myGLCD.printNumI(fila - 48, 280, 160);
```

```
        P62();
        while (teclado3.getKey() != 70) {//es el valor ascii de la letra F que corresponde a la tecla 8
          if (teclado2.getKey() == 66) {
            rst = 1;
            break;
          }
        }
        if (rst == 0) {
          P63();
        }
        if (rst == 0) {
          while (teclado2.getKey() != 66) {} //espera a apretar la tecla de menu
        }
       }
     }
     Serial.println(Rx[0]);
     Serial.println(Rx[1]);
     Serial.println(Rx[2]);
     // comprobacion de valores correctos de fila y column
     Serial.println("FIN");
     break;
  }
  reset();
}

//////////////////////////////////////////////// Inicio  ////////////////////////////////////////////
void inicio() {
  myGLCD.setColor(153, 247, 210); //tinta el fondo de la pantalla
  myGLCD.fillRoundRect (-1, -1, 500, 500 );//dibuja cuadro
  //TITULO
  myGLCD.setFont(BigFont);//elige la fuente
  myGLCD.setBackColor(153, 247, 210);//elige el color de atrás de la fuente
  myGLCD.setColor(0, 0, 0);//elige el color de lo que va debajo
  myGLCD.print("ANTE & ROSALES AUTOMATION", CENTER, 33);
  myGLCD.print("ALMACEN AUTOMATICO", CENTER, 250);
  //FOTO
  myGLCD.setColor(113, 255, 61);//vuelve a cambiar el color
  myGLCD.fillRoundRect (100, 80, 380, 230 );//dibuja cuadro
  //INICIO
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (10, 283, 108 , 305 );
  myGLCD.setColor(255, 255, 255);
  myGLCD.drawRoundRect (10, 283, 108 , 305 );//dibuja el recuadro (borde)
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setFont(SmallFont);
  myGLCD.print("INICIO", 34, 288);
  return;
}

//////////////////////////////////////////////// Menu ////////////////////////////////////////////
```

```
void MENU() {
  myGLCD.clrScr();
  //TECLAS MENU
  for (x = 0; x < 4; x++) {
    myGLCD.setColor(0, 0, 255);
    myGLCD.fillRoundRect (10, 15 + (x * 65), 210, 65 + (x * 65));
    myGLCD.setColor(255, 255, 255);
    myGLCD.drawRoundRect (10, 15 + (x * 65), 210, 65 + (x * 65));
  }

  for (x = 0; x < 4; x++) {
    myGLCD.setColor(0, 0, 255);
    myGLCD.fillRoundRect (270, 15 + (x * 65), 470, 65 + (x * 65));
    myGLCD.setColor(255, 255, 255);
    myGLCD.drawRoundRect (270, 15 + (x * 65), 470, 65 + (x * 65));
  }
 //TEXTO
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("P01 DEJAR", 38, 33);
  myGLCD.print("P02 COGER", 38, 97);
  myGLCD.print("P03 LEE.1", 40, 162);
  myGLCD.print("P04 BORRA.1", 23, 227);
  //
  myGLCD.print("P05 LEE.T", 300, 33);
  myGLCD.print("P06 BORRA.T", 284, 97);
  myGLCD.print("P07 GRABA.1", 284, 162);
  myGLCD.print("---------", 300, 227);
  //ELIGE OPCION
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("ELIGE UNA ACCION", CENTER, 290);
  return;
}

////////////////////////////////////////////// P01 //////////////////////////////////////////////
//OPCION DEJAR
void P01() {
  myGLCD.clrScr();
  myGLCD.setColor(153, 247, 210); //tinta el fondo de la pantalla
  //OPCION
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (100, 20, 380 , 60 );
  myGLCD.setColor(255, 0, 0);
  myGLCD.drawRoundRect (100, 20, 380 , 60 );
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setColor(255, 255, 255);
```

```
  myGLCD.print("OPCION DE DEJAR", CENTER, 33);
  //ELEGIR
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("ELIGIR POSICION DONDE DEJAR", CENTER, 80);
  //COLUMNA
  myGLCD.setFont(BigFont);
  myGLCD.print("COLUMNA:", CENTER, 120);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 310, 120);
  //FILA
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_WHITE);
  myGLCD.print("FILA: ", CENTER, 160);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 280, 160);
  //MENU
  myGLCD.setColor(0, 85, 200);
  myGLCD.fillRoundRect (10, 260, 170, 310);
  myGLCD.setColor(0, 85, 200);
  myGLCD.drawRoundRect (10, 260, 170, 310);
  myGLCD.setBackColor(0, 85, 200);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("MENU", 60, 278);
  //CURSOR EN COLUMNA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 120);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.8", CENTER, 210);
  return;
}

void P11() {
  //TAPA 1 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 120);
  //CURSOR EN FILA
  myGLCD.setBackColor(0, 0, 0);
```

```
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 160);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.5", CENTER, 210);
  return;
}

void P12() {
  //TAPA 2 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 160);
  //ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("ESTAS SEGURO ?", 130, 210);
  //CURSOR EN ESTAS SEGURO?
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 110, 210);
  //ACEPTAR EN VERDE
  myGLCD.setColor(0, 180, 30);
  myGLCD.fillRoundRect (320, 260, 470 , 310 );
  myGLCD.setColor(0, 255, 0);
  myGLCD.drawRoundRect (320, 260, 470 , 310 );
  myGLCD.setBackColor(0, 180, 30);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("ACEPTAR", 340, 278);
  return;
}

void P13() {
  //BORRA ELEGIR
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.print("                          ", CENTER, 80);
  //BORRA COLUMNA
  myGLCD.print("          ", CENTER, 120);
  myGLCD.print("     ", 310, 120);
  //BORRA FILA
  myGLCD.print("              ", CENTER, 160);
  myGLCD.print("       ", 280, 160);
```

```
 //BORRA CURSOR
 myGLCD.print(" ", 110, 210);
 //BORRA MENU
 myGLCD.fillRoundRect (9, 260, 172, 310);
 //BORRA ACEPTAR
 myGLCD.fillRoundRect (320, 260, 472 , 310 );
 //PROCESANDO
 myGLCD.setColor(0, 18, 182);
 myGLCD.fillRoundRect (100, 260, 375 , 310 );
 myGLCD.setColor(0, 18, 182);
 myGLCD.drawRoundRect (100, 260, 375 , 310 );
 myGLCD.setBackColor(0, 18, 182);
 myGLCD.setColor(255, 255, 255);
 myGLCD.setFont(BigFont);
 myGLCD.print("PROCESANDO...", CENTER, 278);
 //BORRA ESTAS SEGURO?
 myGLCD.setFont(BigFont);
 myGLCD.setColor(VGA_WHITE);
 myGLCD.setBackColor(VGA_BLACK);
 myGLCD.print("              ", 120, 210);
 //ANIMACION
 myGLCD.setFont(BigFont);
 myGLCD.setBackColor(0, 0, 0);
 myGLCD.setColor(255, 255, 255);
 ////////while comunicacion ==0
 for (int q = 0; q < 5; q++) {
   myGLCD.print(">             ", CENTER, 100);
   myGLCD.print(" >            ", CENTER, 100);
   myGLCD.print("  >           ", CENTER, 100);
   myGLCD.print("   >          ", CENTER, 100);
   myGLCD.print("    >         ", CENTER, 100);
   myGLCD.print("     >        ", CENTER, 100);
   myGLCD.print("      >       ", CENTER, 100);
   myGLCD.print("       >      ", CENTER, 100);
   myGLCD.print("        >     ", CENTER, 100);
   myGLCD.print("         >    ", CENTER, 100);
   myGLCD.print("          >   ", CENTER, 100);
   myGLCD.print("           >  ", CENTER, 100);
   myGLCD.print("            > ", CENTER, 100);
   myGLCD.print("             >", CENTER, 100);
 }
 return;
}

//////////////////////////////////////////// P02 ////////////////////////////////////////////
//OPCION COGER
void P02() {
```

```
myGLCD.clrScr();
myGLCD.setColor(153, 247, 210); //tinta el fondo de la pantalla
//OPCION
myGLCD.setColor(0, 0, 255);
myGLCD.fillRoundRect (100, 20, 380 , 60 );
myGLCD.setColor(255, 0, 0);
myGLCD.drawRoundRect (100, 20, 380 , 60 );
myGLCD.setBackColor(0, 0, 255);
myGLCD.setFont(BigFont);
myGLCD.setBackColor(0, 0, 255);
myGLCD.setColor(255, 255, 255);
myGLCD.print("OPCION DE COGER", CENTER, 33);
//ELEGIR
myGLCD.setFont(BigFont);
myGLCD.setBackColor(0, 0, 0);
myGLCD.setColor(255, 255, 255);
myGLCD.print("ELIGIR POSICION DONDE COGER", CENTER, 80);
//COLUMNA
myGLCD.setFont(BigFont);
myGLCD.print("COLUMNA:", CENTER, 120);
myGLCD.setColor(VGA_BLACK);
myGLCD.setBackColor(VGA_WHITE);
myGLCD.print(" ", 310, 120);
//FILA
myGLCD.setFont(BigFont);
myGLCD.setBackColor(VGA_BLACK);
myGLCD.setColor(VGA_WHITE);
myGLCD.print("FILA: ", CENTER, 160);
myGLCD.setColor(VGA_BLACK);
myGLCD.setBackColor(VGA_WHITE);
myGLCD.print(" ", 280, 160);
//MENU
myGLCD.setColor(0, 85, 200);
myGLCD.fillRoundRect (10, 260, 170, 310);
myGLCD.setColor(0, 85, 200);
myGLCD.drawRoundRect (10, 260, 170, 310);
myGLCD.setBackColor(0, 85, 200);
myGLCD.setColor(255, 255, 255);
myGLCD.setFont(BigFont);
myGLCD.print("MENU", 60, 278);
//CURSOR EN COLUMNA
myGLCD.setBackColor(0, 0, 0);
myGLCD.setColor(0, 255, 0);
myGLCD.setFont(BigFont);
myGLCD.print(">", 150, 120);
//LIMITES
myGLCD.setFont(BigFont);
myGLCD.setColor(VGA_BLACK);
myGLCD.setBackColor(VGA_WHITE);
myGLCD.print("Min.1 <> Max.8", CENTER, 210);
```

```
  return;
}

void P21() {
  //TAPA 1 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 120);
  //CURSOR EN FILA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 160);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.5", CENTER, 210);
  return;
}

void P22() {
  //TAPA 2 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 160);
  //ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("ESTAS SEGURO ?", 130, 210);
  //CURSOR EN ESTAS SEGURO?
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 110, 210);
  //ACEPTAR EN VERDE
  myGLCD.setColor(0, 180, 30);
  myGLCD.fillRoundRect (320, 260, 470 , 310 );
  myGLCD.setColor(0, 255, 0);
  myGLCD.drawRoundRect (320, 260, 470 , 310 );
  myGLCD.setBackColor(0, 180, 30);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("ACEPTAR", 340, 278);
  return;
}
```

```
void P23() {
  //BORRA ELEGIR
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.print("                          ", CENTER, 80);
  //BORRA COLUMNA
  myGLCD.print("          ", CENTER, 120);
  myGLCD.print("     ", 310, 120);
  //BORRA FILA
  myGLCD.print("              ", CENTER, 160);
  myGLCD.print("       ", 280, 160);
  //BORRA CURSOR
  myGLCD.print(" ", 110, 210);
  //BORRA MENU
  myGLCD.fillRoundRect (9, 260, 172, 310);
  //BORRA ACEPTAR
  myGLCD.fillRoundRect (320, 260, 472 , 310 );
  //PROCESANDO
  myGLCD.setColor(0, 18, 182);
  myGLCD.fillRoundRect (100, 260, 375 , 310 );
  myGLCD.setColor(0, 18, 182);
  myGLCD.drawRoundRect (100, 260, 375 , 310 );
  myGLCD.setBackColor(0, 18, 182);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("PROCESANDO...", CENTER, 278);
  //BORRA ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_WHITE);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.print("              ", 120, 210);
  //ANIMACION
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(255, 255, 255);
  ///////while comunicacion ==0
  for (int q = 0; q < 5; q++) {
    myGLCD.print(">             ", CENTER, 100);
    myGLCD.print(" >            ", CENTER, 100);
    myGLCD.print("  >           ", CENTER, 100);
    myGLCD.print("   >          ", CENTER, 100);
    myGLCD.print("    >         ", CENTER, 100);
    myGLCD.print("     >        ", CENTER, 100);
    myGLCD.print("      >       ", CENTER, 100);
    myGLCD.print("       >      ", CENTER, 100);
    myGLCD.print("       >      ", CENTER, 100);
    myGLCD.print("        >     ", CENTER, 100);
    myGLCD.print("         >    ", CENTER, 100);
    myGLCD.print("          >   ", CENTER, 100);
    myGLCD.print("          >   ", CENTER, 100);
```

```
   myGLCD.print("           >   ", CENTER, 100);
   myGLCD.print("            >  ", CENTER, 100);
   myGLCD.print("             > ", CENTER, 100);
   myGLCD.print("              >", CENTER, 100);
  }
  return;
}

//////////////////////////////////////////// P03 ////////////////////////////////////////////
//OPCION LEER MEMORIA 1
void P03() {
  myGLCD.clrScr();
  myGLCD.setColor(153, 247, 210); //tinta el fondo de la pantalla
  //OPCION
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (50, 20, 430 , 60 );
  myGLCD.setColor(255, 0, 0);
  myGLCD.drawRoundRect (50, 20, 430 , 60 );
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("OPCION DE LEER MEMORIA", CENTER, 33);
  //ELEGIR
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("ELIGIR POSICION PARA LEER", CENTER, 80);
  //COLUMNA
  myGLCD.setFont(BigFont);
  myGLCD.print("COLUMNA:", CENTER, 120);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 310, 120);
  //FILA
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_WHITE);
  myGLCD.print("FILA: ", CENTER, 160);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 280, 160);
  //MENU
  myGLCD.setColor(0, 85, 200);
  myGLCD.fillRoundRect (10, 260, 170, 310);
  myGLCD.setColor(0, 85, 200);
  myGLCD.drawRoundRect (10, 260, 170, 310);
  myGLCD.setBackColor(0, 85, 200);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
```

```
  myGLCD.print("MENU", 60, 278);
  //CURSOR EN COLUMNA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 120);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.8", CENTER, 210);
  return;
}

void P31() {
  //TAPA 1 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 120);
  //CURSOR EN FILA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 160);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.5", CENTER, 210);
  return;
}

void P32() {
  //TAPA 2 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 160);
  //ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("ESTAS SEGURO ?", 130, 210);
  //CURSOR EN ESTAS SEGURO?
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 110, 210);
  //ACEPTAR EN VERDE
```

```
  myGLCD.setColor(0, 180, 30);
  myGLCD.fillRoundRect (320, 260, 470 , 310 );
  myGLCD.setColor(0, 255, 0);
  myGLCD.drawRoundRect (320, 260, 470 , 310 );
  myGLCD.setBackColor(0, 180, 30);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("ACEPTAR", 340, 278);
  return;
}

void P33() {
  //BORRA ACEPTAR
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.fillRoundRect (320, 260, 472 , 310 );
  //BORRA ESTAS SEGURO?
  myGLCD.setColor(VGA_WHITE);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.print("              ", 110, 210);
  //VACIA
  myGLCD.setColor(0, 140, 0);
  myGLCD.fillRoundRect (150, 195, 325 , 240 );
  myGLCD.setBackColor(0, 140, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("VACIA", CENTER, 210);
  return;
}

void P34() {
  //BORRA ACEPTAR
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.fillRoundRect (320, 260, 472 , 310 );
  //BORRA ESTAS SEGURO?
  myGLCD.setColor(VGA_WHITE);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.print("              ", 110, 210);
  //OCUPADA
  myGLCD.setColor(255, 0, 0);
  myGLCD.fillRoundRect (150, 195, 325 , 240 );
  myGLCD.setBackColor(255, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("OCUPADA", CENTER, 210);
  return;
}
```

```
////////////////////////////////////////// P04 //////////////////////////////////////////
//OPCION BORRAR 1
void P04() {
  myGLCD.clrScr();
  myGLCD.setColor(153, 247, 210); //tinta el fondo de la pantalla
  //OPCION
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (40, 20, 440 , 60 );
  myGLCD.setColor(255, 0, 0);
  myGLCD.drawRoundRect (40, 20, 440 , 60 );
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("OPCION DE BORRAR MEMORIA", CENTER, 33);
  //ELEGIR
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("ELIGIR POSICION PARA BORRAR", CENTER, 80);
  //COLUMNA
  myGLCD.setFont(BigFont);
  myGLCD.print("COLUMNA:", CENTER, 120);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 310, 120);
  //FILA
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_WHITE);
  myGLCD.print("FILA: ", CENTER, 160);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 280, 160);
  //MENU
  myGLCD.setColor(0, 85, 200);
  myGLCD.fillRoundRect (10, 260, 170, 310);
  myGLCD.setColor(0, 85, 200);
  myGLCD.drawRoundRect (10, 260, 170, 310);
  myGLCD.setBackColor(0, 85, 200);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("MENU", 60, 278);
  //CURSOR EN COLUMNA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 120);
  //LIMITES
  myGLCD.setFont(BigFont);
```

```
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.8", CENTER, 210);
  return;
}

void P41() {
  //TAPA 1 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 120);
  //CURSOR EN FILA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 160);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.5", CENTER, 210);
  return;
}

void P42() {
  //TAPA 2 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 160);
  //ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("ESTAS SEGURO DE BORRAR?", CENTER, 210);
  //CURSOR EN ESTAS SEGURO?
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 35, 210);
  //ACEPTAR EN VERDE
  myGLCD.setColor(0, 180, 30);
  myGLCD.fillRoundRect (320, 260, 470 , 310 );
  myGLCD.setColor(0, 255, 0);
  myGLCD.drawRoundRect (320, 260, 470 , 310 );
  myGLCD.setBackColor(0, 180, 30);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("ACEPTAR", 340, 278);
```

```
  return;
}

void P43() {
  //BORRA ACEPTAR
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.fillRoundRect (320, 260, 472 , 310 );
  //BORRA ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_WHITE);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.print("                          ", 15, 210);
  //BORRADA
  myGLCD.setColor(255, 0, 0);
  myGLCD.fillRoundRect (150, 195, 325 , 240 );
  myGLCD.setBackColor(255, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("BORRADA", CENTER, 210);
  return;
}

//////////////////////////////////////////// P05 ////////////////////////////////////////////
//OPCION BORRAR TODA EEPROM
void P05() {
  myGLCD.clrScr();
  myGLCD.setColor(153, 247, 210); //tinta el fondo de la pantalla
  //OPCION
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (20, 20, 460 , 60 );
  myGLCD.setColor(255, 0, 0);
  myGLCD.drawRoundRect (20, 20, 460 , 60 );
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("OPCION BORRAR TODA MEMORIA", CENTER, 33);
 //CURSOR EN ESTAS SEGURO?
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 35, 210);
  //ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("ESTAS SEGURO DE BORRAR?", CENTER, 210);
  //ACEPTAR EN VERDE
  myGLCD.setColor(0, 180, 30);
```

```
  myGLCD.fillRoundRect (320, 260, 470 , 310 );
  myGLCD.setColor(0, 255, 0);
  myGLCD.drawRoundRect (320, 260, 470 , 310 );
  myGLCD.setBackColor(0, 180, 30);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("ACEPTAR", 340, 278);
  //MENU
  myGLCD.setColor(0, 85, 200);
  myGLCD.fillRoundRect (10, 260, 170, 310);
  myGLCD.setColor(0, 85, 200);
  myGLCD.drawRoundRect (10, 260, 170, 310);
  myGLCD.setBackColor(0, 85, 200);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("MENU", 60, 278);
  return;
}

void P51() {
 //BORRA ACEPTAR
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.fillRoundRect (320, 260, 472 , 310 );
  //BORRA ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_WHITE);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.print("                          ", 15, 210);
  //BORRADA
  myGLCD.setColor(255, 0, 0);
  myGLCD.fillRoundRect (150, 195, 325 , 240 );
  myGLCD.setBackColor(255, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("BORRADA", CENTER, 210);
  return;
}

//////////////////////////////////////////// P06 ////////////////////////////////////////////
//OPCION DE MODIFICAR EEPROM
void P06() {
  myGLCD.clrScr();
  myGLCD.setColor(153, 247, 210); //tinta el fondo de la pantalla
  //OPCION
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (40, 20, 440 , 60 );
  myGLCD.setColor(255, 0, 0);
  myGLCD.drawRoundRect (40, 20, 440 , 60 );
  myGLCD.setBackColor(0, 0, 255);
```

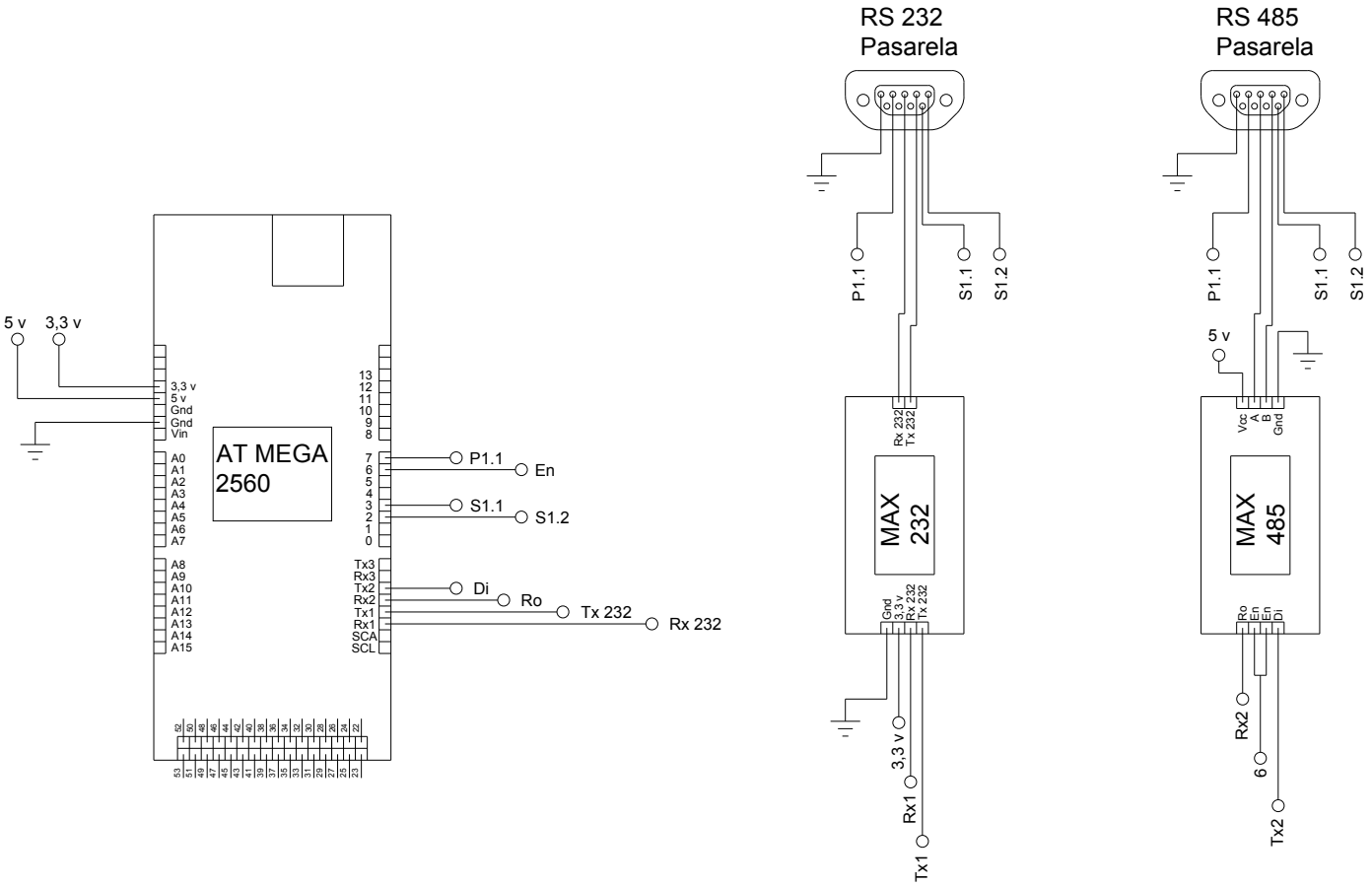
```
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 255);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("OPCION DE GRABAR MEMORIA", CENTER, 33);
  //ELEGIR
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.print("ELIGIR POSICION PARA GRABAR", CENTER, 80);
  //COLUMNA
  myGLCD.setFont(BigFont);
  myGLCD.print("COLUMNA:", CENTER, 120);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 310, 120);
  //FILA
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_WHITE);
  myGLCD.print("FILA: ", CENTER, 160);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print(" ", 280, 160);
  //MENU
  myGLCD.setColor(0, 85, 200);
  myGLCD.fillRoundRect (10, 260, 170, 310);
  myGLCD.setColor(0, 85, 200);
  myGLCD.drawRoundRect (10, 260, 170, 310);
  myGLCD.setBackColor(0, 85, 200);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("MENU", 60, 278);
  //CURSOR EN COLUMNA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 120);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.8", CENTER, 210);
  return;
}

void P61() {
  //TAPA 1 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
```

```
  myGLCD.print(" ", 150, 120);
  //CURSOR EN FILA
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 150, 160);
  //LIMITES
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("Min.1 <> Max.5", CENTER, 210);
  return;
}

void P62() {
  //TAPA 2 CURSOR
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setFont(BigFont);
  myGLCD.print(" ", 150, 160);
  //ESTAS SEGURO?
  myGLCD.setFont(BigFont);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("ESTAS SEGURO ?", 130, 210);
  //CURSOR EN ESTAS SEGURO?
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.setFont(BigFont);
  myGLCD.print(">", 110, 210);
  //ACEPTAR EN VERDE
  myGLCD.setColor(0, 180, 30);
  myGLCD.fillRoundRect (320, 260, 470 , 310 );
  myGLCD.setColor(0, 255, 0);
  myGLCD.drawRoundRect (320, 260, 470 , 310 );
  myGLCD.setBackColor(0, 180, 30);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("ACEPTAR", 340, 278);
  return;
}

void P63() {
  //BORRA ACEPTAR
  myGLCD.setBackColor(VGA_BLACK);
  myGLCD.setColor(VGA_BLACK);
  myGLCD.fillRoundRect (320, 260, 472 , 310 );
  //BORRA ESTAS SEGURO?
  myGLCD.setColor(VGA_WHITE);
  myGLCD.setBackColor(VGA_BLACK);
```

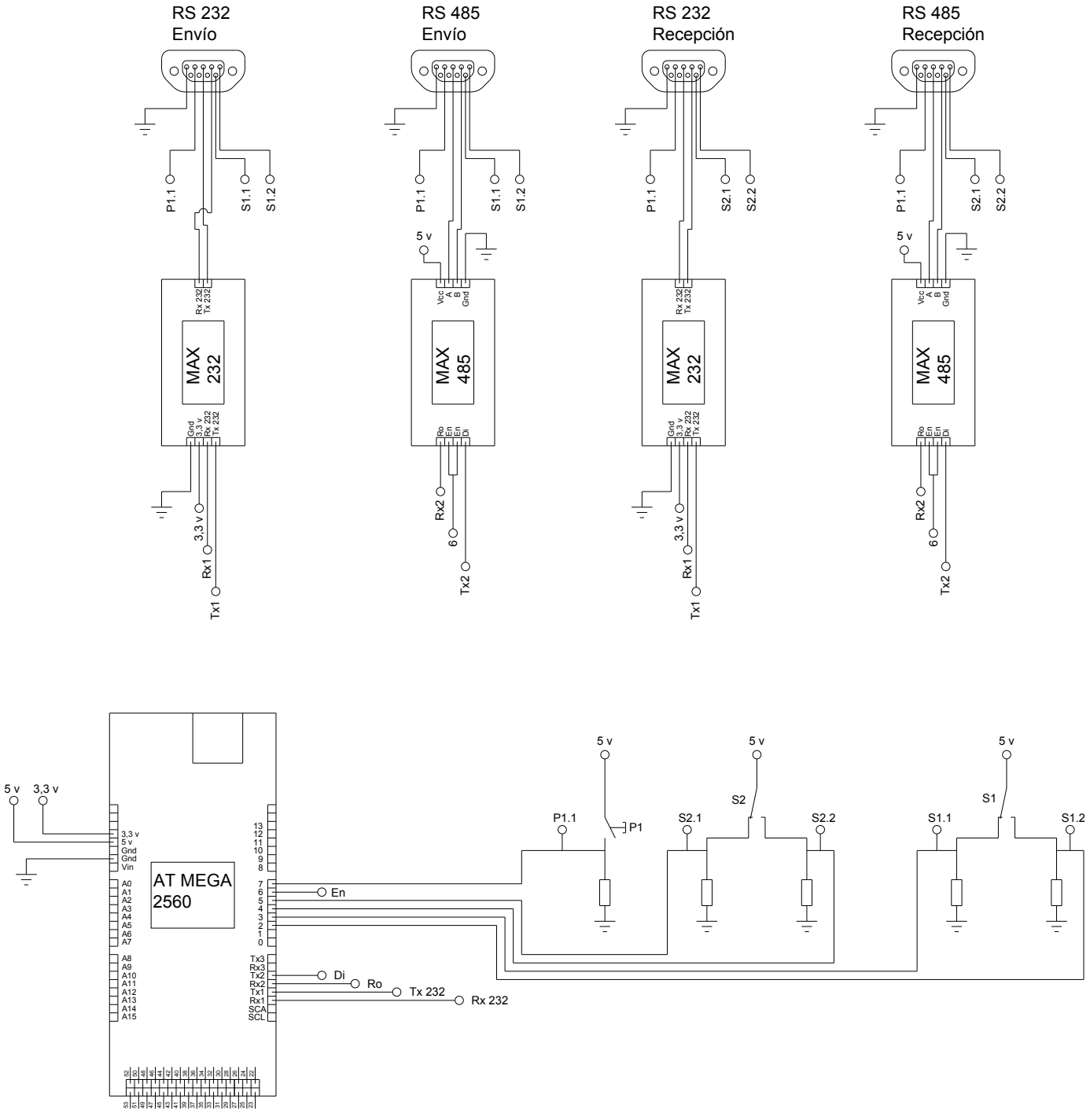
```
  myGLCD.print("               ", 110, 210);
  //OCUPADA
  myGLCD.setColor(255, 0, 0);
  myGLCD.fillRoundRect (150, 195, 325 , 240 );
  myGLCD.setBackColor(255, 0, 0);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setFont(BigFont);
  myGLCD.print("OCUPADA", CENTER, 210);
  return;
}

//////////////////////////////////////////// funciones ////////////////////////////////////////
void reset() { //resetea todas las variables que intervienen en la programacion
  columna = 0;
  fila = 0;
  menu = 0;
  Rx[3] = '0', '0', '0';
  posicion = 2;
  rst = 0;
  Serial.flush();
}

void colum() {
  do {// esto es para que espere a recibir un valor de columna
    if (columna == 0) {
      columna = teclado1.getKey(); //esto guarda el numero
    }
    if (teclado2.getKey() == 66) { //si apretas menu esto activa la variable reset para luego salir del bucle
do while
      rst = 1;
    }
    myGLCD.setColor(VGA_BLACK);
    myGLCD.setBackColor(VGA_WHITE);
    Rx[1] = columna;
    if (rst == 1) {//boton de menu(atras) si se apreta tiene que saltar todas las ordenes i finalizar programa
      break;
    }
    if (columna > 56) {//valor maximo de columnas 8=56 ascii
      columna = 0;
    }
  }
  while (columna == 0);
  return;
}

void fil() {
  do { // esto es para que espere a recivir un valor de fila
    if (fila == 0) {
      fila = teclado1.getKey();
    }
```

```
    if (teclado2.getKey() == 66) {
      rst = 1;
    }
    myGLCD.setColor(VGA_BLACK);
    myGLCD.setBackColor(VGA_WHITE);
    Rx[2] = fila;
    if (rst == 1) {
      break;
    }
    if (fila > 53) {//valor maximo de filas 5=53 ascii
      fila = 0;
    }
  }
  while (fila == 0);
  return;
}
```

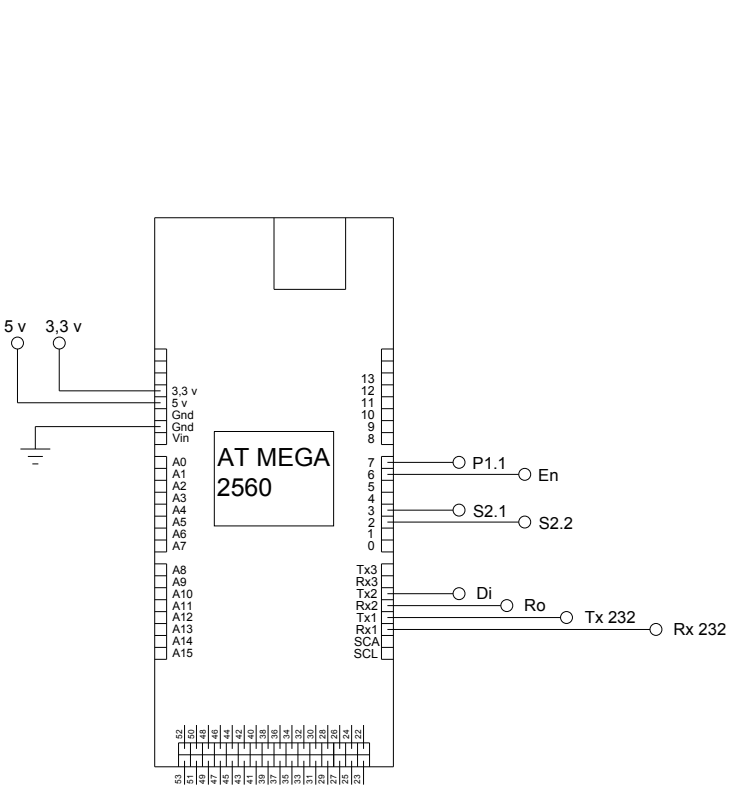
12.2. Planos.

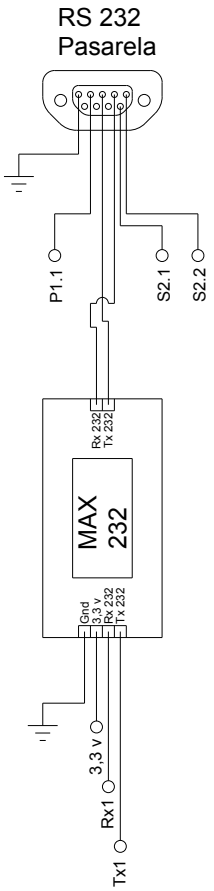


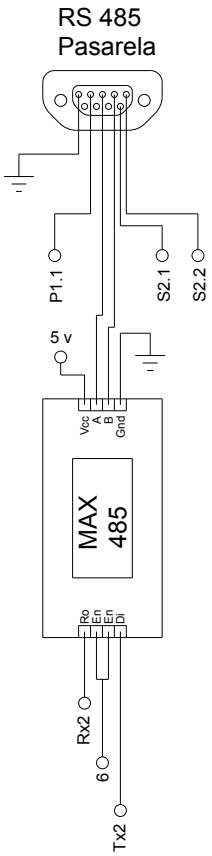
SIS.REP.	Escala	Unidades	PLANO	
	NA	NA	Circuito Transmisión	
Plano Nº:	Fecha:	AUTOR		
1	01/09/2016	Pedro Jose Ante Espada	DISEÑO PLACA DE COMUNICACIÓN POR BUSES INDUSTRIALES PARA ARDUINO	



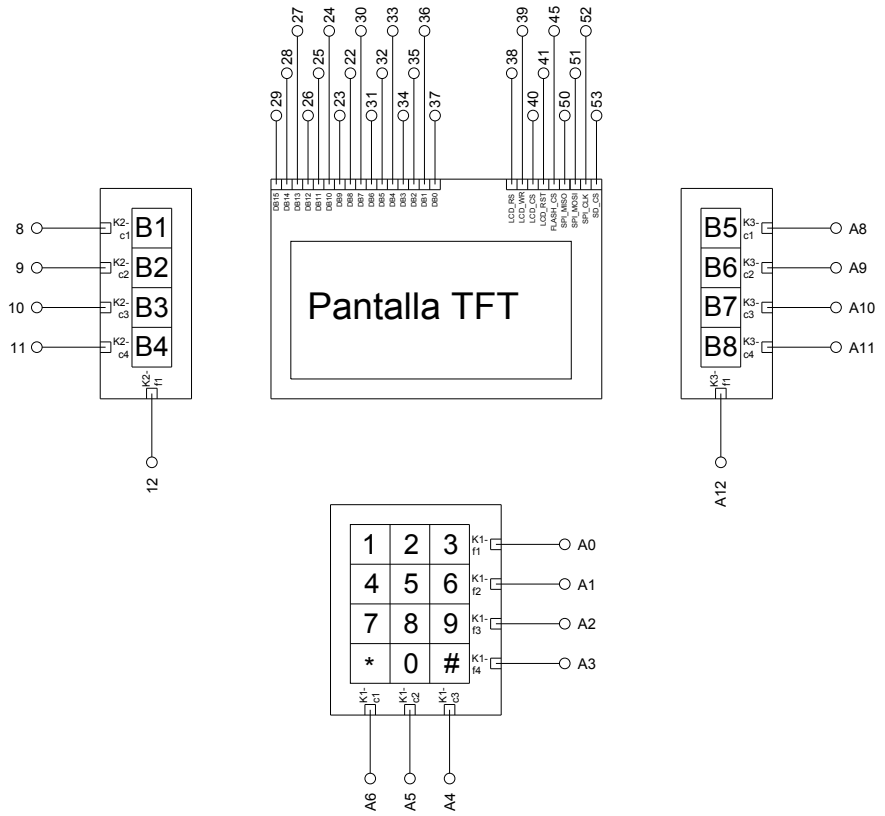
SIS.REP.	Escala	Unidades	PLANO	
	NA	NA	Circuito Pasarela	
Plano Nº:	Fecha:	AUTOR	DISEÑO PLACA DE COMUNICACIÓN POR BUSES INDUSTRIALES PARA ARDUINO	
2	01/09/2016	Pedro Jose Ante Espada		

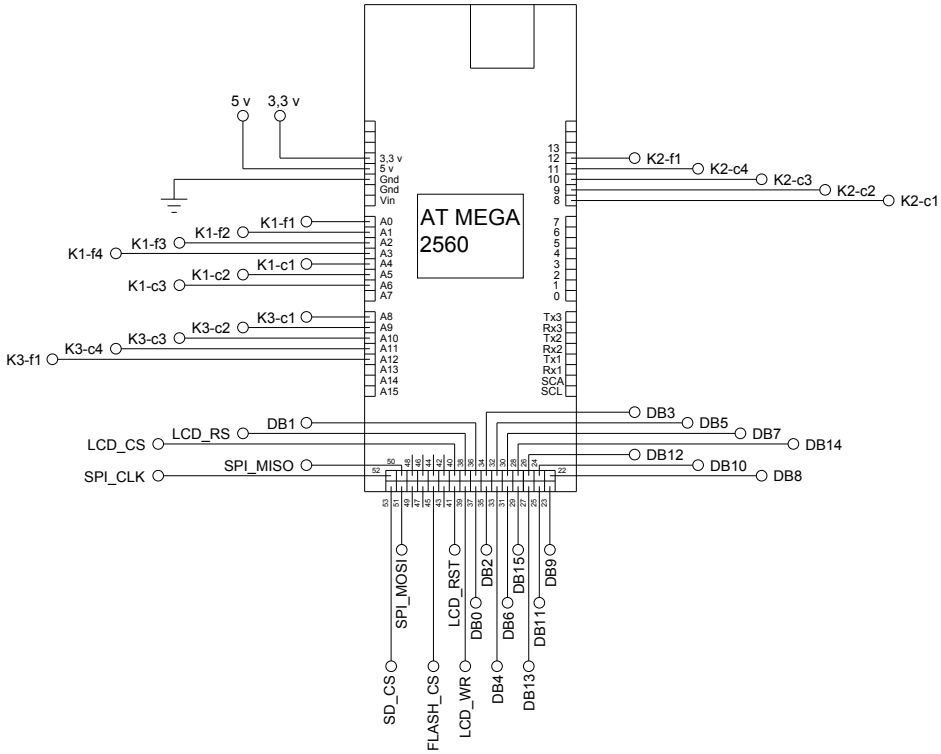






SIS.REP.	Escala	Unidades	PLANO	
	NA	NA	Circuito Recepción	
Plano Nº:	Fecha:	AUTOR	DISEÑO PLACA DE COMUNICACIÓN POR BUSES INDUSTRIALES PARA ARDUINO	
3	01/09/2016	Pedro Jose Ante Espada		





SIS.REP.	Escala	Unidades	PLANO	
	NA	NA	Circuito Pantalla TFT	
Plano Nº:	Fecha:	AUTOR	DISEÑO PLACA DE COMUNICACIÓN POR BUSES INDUSTRIALES PARA ARDUINO	
4	01/09/2016	Pedro Jose Ante Espada		