UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO DEL SISTEMA AUTOMATIZADO DE ALMACÉN PALETIZADO

Trabajo Final de Grado

Autor

Alexander Rosales Martín

Tutor

Roberto Capilla Lladró

Cotutor

Carlos Sánchez Díaz

Universidad Politécnica de Valencia

Escuela Técnica Superior de Ingeniería del Diseño

Grado de Ingeniería Electrónica Industrial y Automática

Septiembre 2016. Curso 2015-2016.

AGRADECIMIENTOS

Quisiera empezar estas líneas recordando el comienzo de esta titulación, junto a los compañeros que me han acompañado durante todo este tiempo. Agradecer este periodo de mi vida a mis ya amigos: Pedro, Eric, Miriam, Rafa, Carmen, Sofía, Adrián, Ainoa y Mª José.

Por otro lado, agradecer a mi tutor del trabajo, Roberto Capilla; y al cotutor Carlos Sánchez, por hacer posible este proyecto. Desde la idea base de mi cabeza hasta esta creación. Ellos fueron los que me permitieron poder llegar a este trabajo final de grado.

Sin olvidarme de ti, esa persona que me ha estado aguantando tantas horas de trabajo. La que me ha apoyado en todo momento y la que me ha ayudado a poder tener este proyecto. Gracias Ana, te quiero.

Por último y no menos importante, me gustaría agradecer todo esto a mi Padre. Todo lo que sé hacer con estas manos es gracias a su experiencia y trabajo. Este proyecto tiene una parte que te pertenece. Gracias a toda mi familia por la ayuda prestada en este trabajo. Gracias a vosotros ahora soy lo que soy.

"Hoy puede ser un gran día, plantéatelo así,

aprovecharlo o que pase de largo, depende en parte de ti".

("*Hoy puede ser un gran día",* Joan Manuel Serrat)

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 MEMORIA

1.1 Resumen

Este proyecto tiene como finalidad el diseño y la programación de un sistema de almacenamiento mediante paletización automática.

Se ha realizado su puesta en marcha mediante la construcción de una maqueta con el objetivo de implementar y ejecutar dicho sistema de paletizado a través de una simulación real.

A continuación se desarrollan cada una de las etapas que se han seguido en su ejecución, así como las bases sobre las que se fundamentan.

1.2 Introducción

El proyecto objeto de este estudio consiste en el diseño, programación, construcción, implementación y ejecución de un sistema de almacenamiento automático mediante una maqueta a escala.

El funcionamiento de este sistema de almacenamiento está basado en una maqueta, donde un dispositivo móvil apila las unidades de carga, ya sean palets o bultos, en las posiciones que componen la estructura automáticamente, en función de la posición que le corresponda. Dicho sistema de almacenamiento se conoce como *paletizado inteligente o automatizado*.

Se ha construido un modelo a pequeña escala, resultado del proceso de diseño y programación del sistema. La solución final es una reproducción fiel a un sistema de almacenamiento automático real.

A lo largo de este documento se irán detallando cada una de las etapas y procesos realizados hasta la obtención del resultado final.

1.2.1 Alcance del proyecto

El alcance de este proyecto consiste en el diseño de los requerimientos mecánicos, eléctricos y automáticos con el objetivo de la construcción, implementación, ejecución y puesta en marcha final del sistema de almacenamiento automático en una reproducción real.

1.2.2 Antecedentes

Este proyecto es la continuación de un trabajo realizado anteriormente. Surgió de la idea de un grupo de alumnos de la Escuela Técnica Superior Industrial de Diseño, la cual se llevó a la práctica para la asignatura *Electrónica Digital* en 3º curso de la carrera de Grado en Ingeniería Electrónica Industrial y Automática. Esta idea original se llevó a cabo con éxito con una muy buena calificación de la asignatura.

En el mercado existen ya productos que ofrecen este servicio. Estos sistemas de almacenamiento son fruto de la evolución de empresas que se dedican principalmente

a diseñar almacenes de estanterías. En la *figura 1* se puede ver un ejemplo de cómo son estos almacenes automáticos.

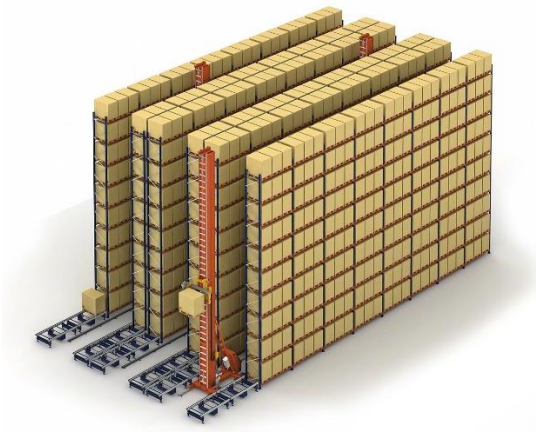


Figura 1: Almacén automático. Compañía Mecalux®

Estos almacenes se componen de filas de estanterías de tamaño acorde al emplazamiento a utilizar. Entre estas estanterías se colocan unas guías en el suelo por donde posteriormente serán instalados los mecanismos necesarios para el movimiento del carro, el cual coge y deja los objetos en dichas estanterías.

En la *figura 2* se puede observar el primer modelo de paletizadora automática junto al grupo que diseñó, programó y construyó la maqueta. Se donó a la Universidad Politécnica de Valencia para su exposición en la Escuela Técnica Superior de Ingeniería del Diseño. A día de hoy sigue expuesta.

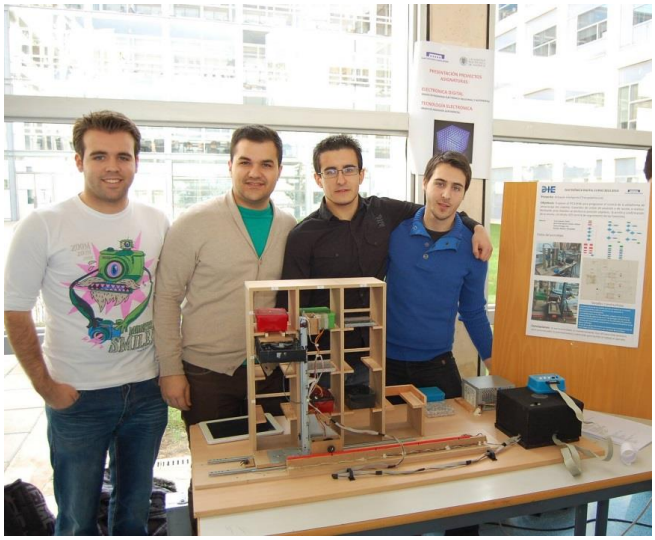


Figura 2: de izquierda a derecha. Eric García, Alexander Rosales, Rafa De la Concepción, Pedro Ante.

1.2.3 Búsqueda de información

En primer lugar, se han investigado las diferentes opciones que se pueden encontrar en el mercado, encontrando una amplia oferta de fabricantes y formas constructivas.

En la actualidad, existen diversos fabricantes que construyen almacenes con paletización automática. La compañía Mecalux® es uno de los mejores ejemplos de ello. Se dedica al montaje de estanterías para el almacenamiento de palets, y entre sus estructuras, destacan los transelevadores para palets, tal como se muestra en la *figura 3*.

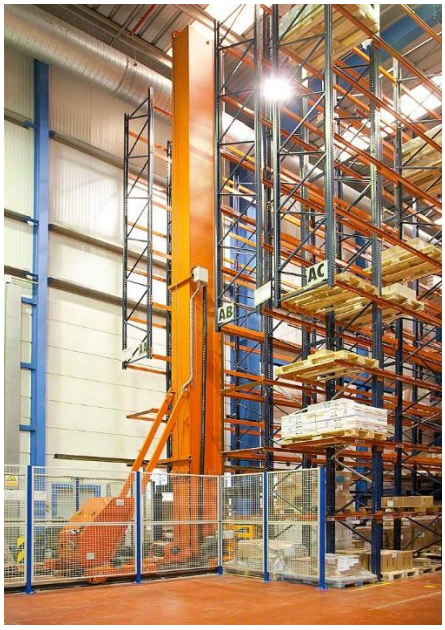


Figura 3: Transelevador trilateral automático, marca Mecalux®

El modelo objeto de este proyecto consiste en una reproducción a escala de este tipo de almacenamiento.

Por otro lado, existen otras máquinas similares como son los *robots cartesianos o robots de tres ejes.* La automatización del almacén se basa en los movimientos que éstos realizan en las tres dimensiones, es decir, eje X, eje Y y eje Z. En la *figura 4* se puede ver un ejemplo de lo que es un robot cartesiano.

También en la *figura 5* se puede observar las tres direcciones de movimiento en las que se basa su funcionamiento.

Figura 4: Robot Cartesiano

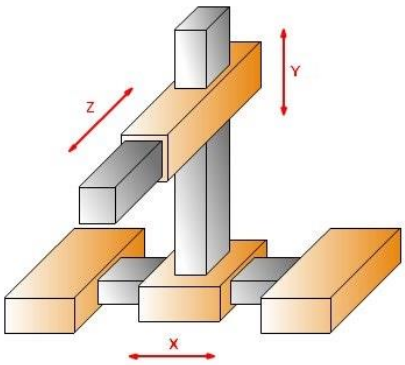


Figura 5: Sistema de 3 ejes cartesianos

Por todo ello, estas dos ideas son las bases del funcionamiento de esta maqueta.

1.3 Automatismo

En este apartado se va a detallar la elección del automatismo el cual procesará la información de la paletizadora.

En el mercado existen diversos dispositivos para la automatización industrial. Actualmente se pueden encontrar los controladores lógicos, más comúnmente llamados PLC (*Programmable Logic Controller)*, tarjetas de adquisición de datos, microprocesadores y demás placas controladoras.

Figura 6: PLC Siemens S7-1200

Figura 7: Tarjeta de adquisición de datos NI

Cada dispositivo ofrece ventajas e inconvenientes. El PLC es uno de los más utilizados en el ámbito industrial de la automatización, dejando en minoría el sistema de tarjeta de adquisición de datos por la necesidad de utilizar un ordenador obligatoriamente. Un ejemplo es el que se muestra en la *figura 6* de la marca Siemens®. El mundo del microcontrolador va en aumento para sistemas pequeños y automatismos sencillos.

Las tarjetas de adquisición de datos son un dispositivo en minoría, el cual va cada vez más en desuso, por lo que también se va a descartar esta posibilidad. Un ejemplo de ello es el que se muestra en la *figura 7*.

Dentro de los microprocesadores existen una amplia gama de dispositivos de diferentes marcas. En una primera selección se va a descartar el PLC por ser un dispositivo muy caro y poco accesible para nuestra maqueta.

En este caso, se ha optado por una solución con microprocesador, por ser suficiente para las funciones a implementar y por ser económicamente más viable que el resto de alternativas.

En definitiva, se ha optado por elegir la marca de microprocesadores Arduino®, por su sencillez de programación y por haber sido estudiada en la titulación, conociendo el lenguaje básico de programación y disponiendo de un gran soporte en Internet.

1.3.1 Automatización con Arduino®

Arduino® es una empresa que se fundó en 2005 por Massimo Banzi. Empezó como una placa para instituto económica y práctica, pero al poco tiempo se extendió y creció por todo el mundo como una empresa.

Esta empresa diseña y vende placas electrónicas, en las cuales se incluye un microprocesador ya implementado. En la actualidad es una marca muy utilizada.

Estas placas están disponibles para todo el público que quiera reproducirlas, ya que los esquemas se pueden descargar de la página web. Disponen de un código abierto de programación u "*Open Source*" (código abierto).

Este tipo de programación de software libre se está extendiendo por todo el mundo, incluso muchas marcas diseñan sus componentes para que puedan ser utilizados con este tipo de programación.

El software libre está en auge, ya que cualquier persona puede modificar su código o esquema eléctrico y darle diversas utilidades sin tener que pagar por dichas alteraciones. Gracias a esto, se puede hacer y deshacer de manera arbitraria, como se verá a continuación en la programación de este modelo.

1.3.2 Plataformas Arduino®

En el mundo de Arduino® existen diversas placas desarrolladas por ellos. Actualmente, debido a aspectos socioeconómicos la marca Arduino® se ha dividido en dos, Arduino® y Genuino®. Este nuevo nombre simplemente influye en la procedencia de la placa, siendo Arduino® procedente de Estados Unidos, y Genuino® de Europa.

Para nuestro caso, seguiremos llamando a esta plataforma Arduino®. Estas placas comerciales se encuentran disponibles para programar y utilizar en el entorno deseado.

Entre los modelos que existen en el mercado, podemos destacar algunos, tales como:

- **Arduino® Micro**

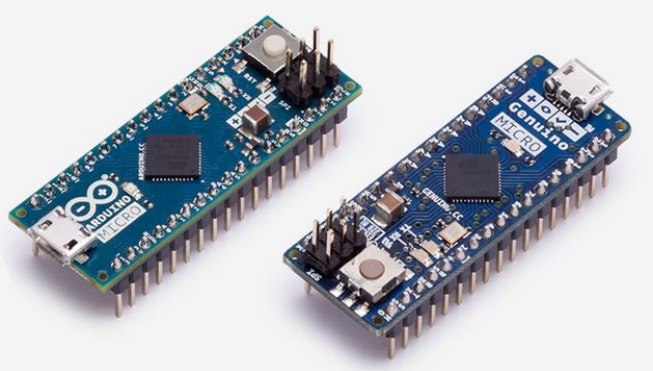


Figura 8: Arduino® Micro

Esta placa es una de las más pequeñas del mercado debido a su gran potencial y su buena versatilidad. Puede ser programada como cualquier otra placa y desempeña funciones similares a sus modelos superiores pero a un tamaño reducido, como se puede observar en la *figura 8*.

- **Arduino® LiliPad**

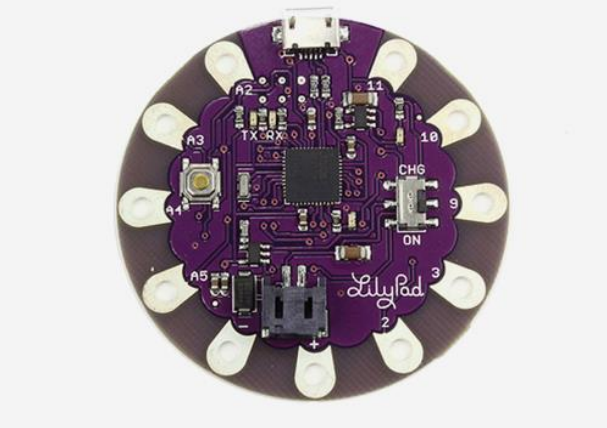


Figura 9: Arduino® LiliPad

Esta placa es un diseño especial de la marca Arduino®. Está pensada para incorporarla en las prendas de vestir o complementos textiles que se utilizan diariamente (ver *figura 9*). Sus utilidades son muy limitadas ya que no dispone de todas las prestaciones como las otras placas.

- **Arduino® Uno**

Figura 10: Arduino® Uno

Es la placa que se presenta como modelo estándar de la marca Arduino®, debido a su versatilidad y polivalencia (ver *figura 10*). Fue la primera placa en salir al mercado. En la actualidad se ha mejorado pero mantiene su forma original. Se trata de una de las placas más comercializadas y utilizadas en el mundo.

- **Arduino® Mega**

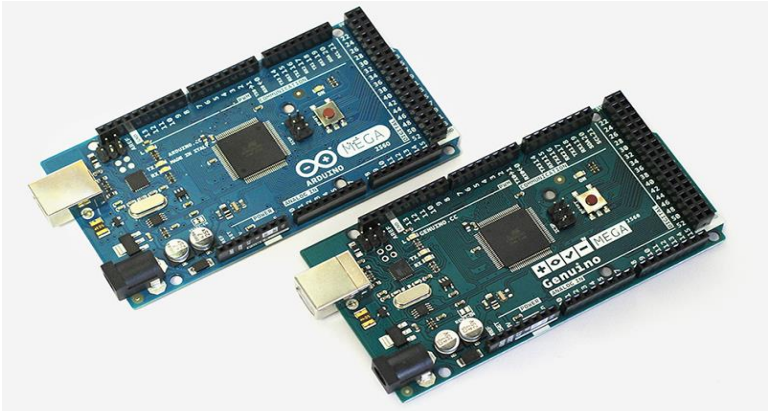


Figura 11: Arduino® Mega

Arduino® Mega es una de las placas más potentes de la marca. Es una placa muy polivalente con infinidad de usos y configuraciones (ver *figura 11*). Por eso, es una de las más vendidas y utilizadas. Es una placa con la que muchos sistemas se basan para diseñar sus proyectos tales como las impresoras 3D, cortadores laser, fresadoras y demás utilidades.

En el mercado existe una amplia gama de placas Arduino®, tales como: Arduino® 101, Arduino® Zero, Arduino® Yun, Arduino® DUE, Arduino® Robot, Arduino® Leonardo y demás. Las placas anteriormente descritas son solo un pequeño ejemplo de los diferentes modelos.

Aquí se puede ver una tabla comparativa con las características principales de cada una de las placas ya mencionadas (*tabla 1*):

	Arduino® Micro	Arduino® LilyPad	Arduino® UNO	Arduino® Mega
Microcontrolador	ATmega32U4	ATmega32U4	ATmega328P	ATmega2560
Velocidad del procesador	16MHz	8MHz	16MHz	16MHz
Memoria EEPROM	1Kb	1Kb	1Kb	4Kb
Memoria Flash	32Kb	32Kb	32Kb	256Kb
Pines E/S Digitales	20	9	14	54
Pines Entradas Analógicos	12	6	6	16
Pines con Interrupciones Externas	2	2	2	6
Salidas con PWM	7	6	6	15
Tensión de Trabajo	5V	3.3V	5V	5V
Corriente de salida	20mA	40mA	20mA	20mA

Tabla 1: Comparativa de las placas Arduino® nombradas

1.3.3 Entorno de programación Arduino®

Para poder programar este tipo de placas Arduino®, el mismo desarrollador suministra gratuitamente su entorno de programación y software necesario. Éste se conoce como Arduino® IDE (siglas de "*Integrated Development Environment*") o *Entorno de Desarrollo Integrado*. Es posible instalarlo en el ordenador tanto en entorno Windows, Linux o Mac.

El paquete de instalación lleva incluido todo lo necesario como software, drivers, librerías y complementos para poder utilizarlo. Este programa gestiona el lenguaje de programación con el que trabaja, así como las diferentes configuraciones a la hora de programar.

Es un entorno sencillo para programar sin demasiadas complicaciones, en la *figura 12* se puede ver el entorno gráfico. Se trata de un lenguaje de programación de alto nivel basado en una mezcla entre el *lenguaje C* y el *lenguaje C++,* siendo la mayoría de él compatible en el entorno IDE.

Muchas de las estructuras de programación son comunes al *lenguaje C++*; sin embargo, existen otras declaraciones específicas referentes a las entradas y salidas de la placa.

A continuación se muestra la barra de menús: Archivo, Editar, Programa, Herramientas y Ayuda. Es posible navegar por los distintos ficheros que se han creado para este proyecto.

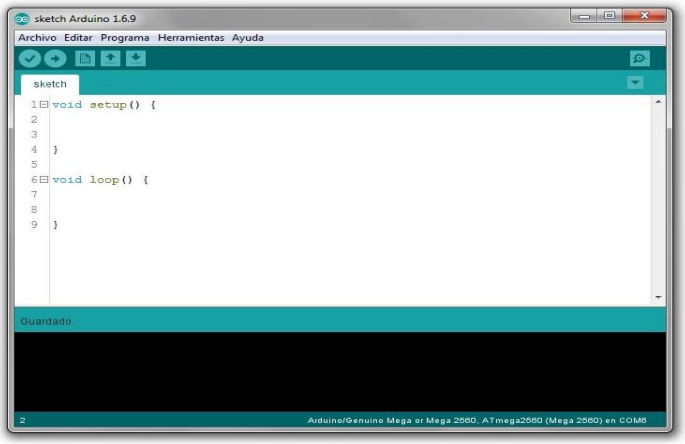


Figura 12: Arduino® IDE

En la parte superior también se encuentran los botones de: Verificar, Subir, Nuevo, Abrir y Guardar. Estos sirven para verificar la programación compilándola, subir el código al Arduino® volcándoselo en la memoria, crear un nuevo programa, abrir un nuevo programa ya guardado previamente o guardar el programa creado.

En la parte inferior se pueden apreciar los diferentes sketchs o módulos donde nos moveremos para programar las diferentes pestañas. El área negra es el espacio donde se informa sobre las notificaciones de error y de compilación.

En la esquina superior derecha se encuentra el botón del monitor serial. Este botón abre una ventana en la cual puedes ver la comunicación serie del Arduino® y a la vez enviar por el puerto al Arduino® desde el teclado del ordenador.

1.4 Desarrollo

1.4.1 Diseño mecánico

El diseño mecánico que se ha realizado para la construcción y correcto funcionamiento de la maqueta no es objeto principal de este proyecto. El desarrollo del proyecto se va a centrar en el diseño eléctrico, electrónico y automático.

Por ello, no se ha tenido en cuenta el cálculo estructural para su diseño. No obstante, se han empleado conceptos básicos aprendidos durante la titulación para la construcción del mecanismo, tales como movimientos lineales o transmisiones, los cuales se van a exponer a continuación.

1.4.1.1 EJE X

En este eje se han utilizado un sistema de guías lineales fijas con tres carros que deslizan sobre ellas. Éstas se han montado paralelas entre ellas sobre una base plana, la cual recrea el suelo de la instalación.

Para el movimiento de este eje se ha utilizado un sistema de piñón cremallera (véase *figura 13*). Para el correcto funcionamiento de deslizamiento del eje se han montado los finales de carrera correspondiente a Home X, y al final de carrera del tope del eje X en el otro extremo.

Figura 13: Eje x. Guías lineales y cremallera

Este sistema de transmisión está impulsado por un motor paso a paso modelo *Nema 17*.

El motor se encuentra unido a una caja de engranajes dentados que posteriormente trasmiten la fuerza a la cremallera (véase *figura 14*).

Este eje tiene un encoder óptico en el engranaje, el cual se une a la cremallera. Con este sistema conseguimos un bucle de realimentación que nos permite saber cuánto se ha movido el carro X, conociendo así la posición en todo momento para dicho eje.

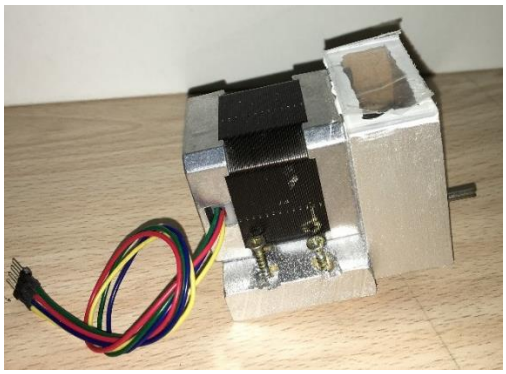


Figura 14: Eje X. Motor Nema 17 junto a caja de engranajes.

Todo ello en su conjunto forma el mecanismo del eje X. Este eje consigue posicionar la paletizadora en la posición correspondiente a las columnas.

1.4.1.2 EJE Y

Para el eje Y se ha diseñado un sistema lineal de movimiento con la ayuda de dos varillas cilíndricas fijas a la plataforma del eje X anteriormente nombrada, con dos rodamientos cilíndricos lineales. Es en estos dos rodamientos donde se ha montado el carro del eje Y, tal como puede verse en la *figura 15*.

Figura 15: Eje Y

Para impulsar este eje se ha empleado un motor paso a paso modelo *Nema 17*. Este motor mueve un sistema de poleas solidarias mediante una correa dentada hasta un tornillo sinfín. El tornillo sinfín es el encargado de crear un movimiento lineal en el carro del eje Y mediante una tuerca encastada en él.

En la *figura 16* se puede ver el motor paso a paso junto al sistema de polea y correa dentada para trasmitir la fuerza al tornillo sinfín.

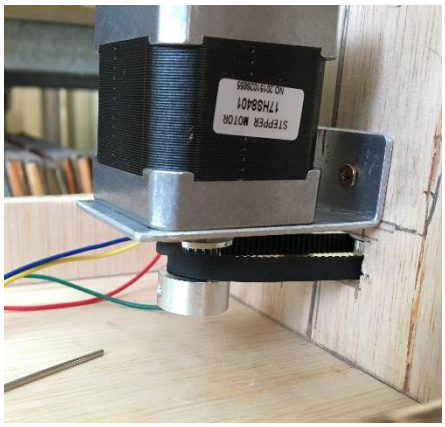


Figura 16: Eje Y. Transmisión de polea dentada y correa

Este sistema de transmisión tiene la función de elevar o descender el carro del eje Y montado con sendos rodamientos lineales cilíndricos y la tuerca del tornillo sinfín la cual genera el impulso. Dicho eje es capaz de posicionar la paletizadora en el lugar correspondiente a las filas de las estanterías.

1.4.1.3 EJE Z

Para esta sección de la maqueta, el carro móvil del eje Z se ha diseñado con dos guías telescópicas, como las utilizadas en los cajones extraíbles. Su función es semejante, pero con la mejora de que puede moverse en ambos sentidos.

Desde una posición centrada o posición básica, puede extenderse en un sentido u en otro. En la *figura 17* puede observarse la extensión del eje Z, así como su funcionamiento.

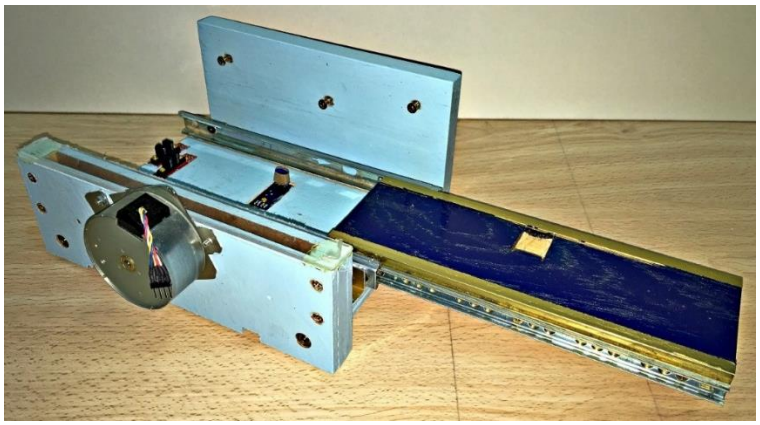


Figura 17: Eje Z

Para impulsar este eje se ha empleado un sistema de piñón cremallera como el del eje X.

Este sistema de movimiento lineal está inducido por un motor paso a paso modelo *Canon RG5-0764*.

El motor esta acoplado a un sistema diseñado especialmente para que en todo momento disponga de tracción al sistema de piñón cremallera.

En la *figura 18* se puede observar el mecanismo de doble piñón (color naranja) engranado con la cremallera, así como la caja de múltiples engranajes movidos por el motor.

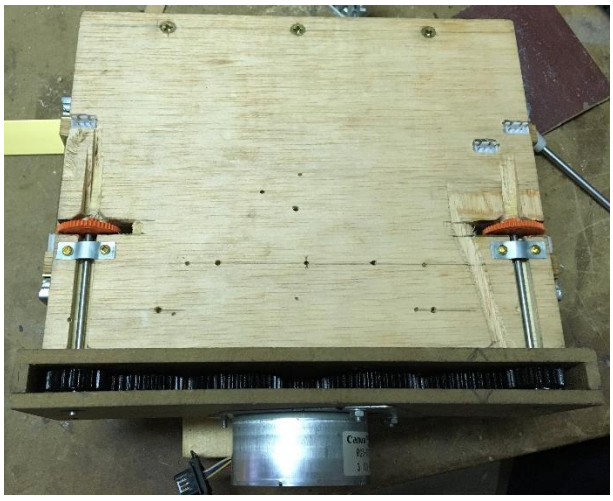


Figura 18: Eje Z. Motor junto a caja de engranajes y la transmisión piñón-cremallera

1.4.2 Diseño eléctrico

El diseño eléctrico se ha realizado con el programa *Autocad Electrical®*, mediante un esquema eléctrico de todos los componentes.

En el apartado de planos se puede ver todo el sistema de cableado empleado en la maqueta. Éste se ha diseñado teniendo en cuenta de que se trata de una maqueta a pequeña escala. No obstante, se ha realizado con la precisión requerida, de como si un cuadro eléctrico real se tratase.

A continuación, se va detallar los distintos sistemas eléctricos empleados en la automatización de la paletizadora.

1.4.2.1 Microprocesador (Arduino® Mega)

En este proyecto, como se ha justificado en el apartado 3, se va a emplear la placa de *Arduino® Mega*. A continuación se va a detallar las características principales.

Localización de las distintas partes del *Arduino® Mega 2560*:

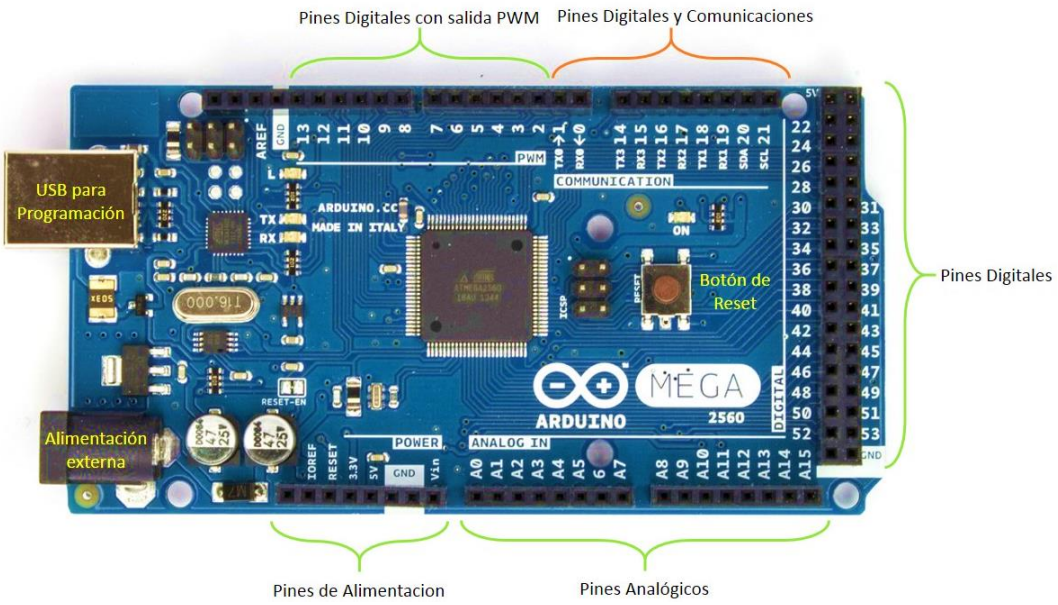


Figura 19: Estructura de un Arduino® Mega 2560

En la *figura 19* se pueden ver los distintos puertos que tiene la placa, así como las entradas y salidas, los pines dedicados a comunicaciones, alimentación, masa y demás.

1.4.2.2 Sensores

En esta maqueta se han empleado varios tipos de sensores. A continuación se va a detallar cada tipo de sensor así como su funcionamiento. Cada sensor se ha escogido acorde a su funcionamiento y las distintas acciones a procesar. Se ha tenido en cuenta la ubicación de cada uno de ellos, ya que no todos se pueden ajustar correctamente.

1.4.2.2.1 Sensores Ópticos

Esta clase de sensores corresponden a la familia de los sensores fotoeléctricos. Éstos trabajan mediante la luz y se componen de dos tipos de componentes electrónicos. Por un lado se encuentra un diodo, el cual emite luz infrarroja; y por otro lado se encuentra el fototransistor, el cual conduce la corriente cuando es excitado por el haz de luz procedente del diodo. En la *figura 20* se puede apreciar el funcionamiento de dichos componentes.

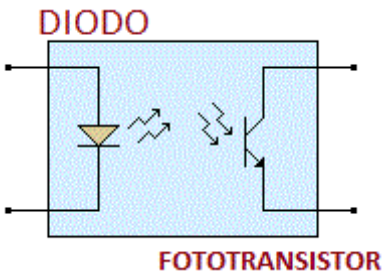


Figura 20: Esquema de un Sensor fotoeléctrico

Para esta maqueta se han empleado cinco sensores ópticos como el que se puede ver en la *figura 21.* Estos sensores ya llevan una pequeña electrónica implementada para una precisión y funcionamientos correctos. En este caso, estos sensores se comercializan ya montados y listos para su utilización.

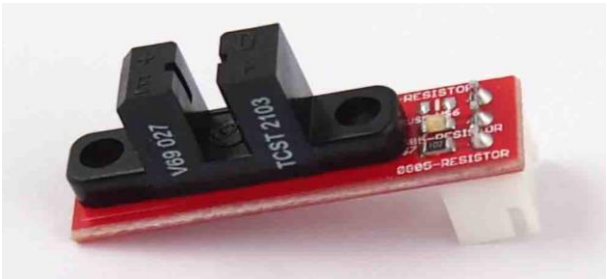


Figura 21: Sensor fotoeléctrico de barrera

Se han elegido este tipo de sensores debido a que no tienen partes mecánicas y por tanto, en su funcionamiento, no están sometidos a desgaste mecánico, ya que funcionan con un haz de luz.

Otro tipo de modelo empleado en el mundo de la industria son los sensores infrarrojos o de obstáculos. Esta clase de sensores detecta cuando un objeto cruza un haz de luz infrarroja, invisible al ojo humano.

Este funcionamiento es idóneo para todo tipo de objetos a detectar, ya que no intervienen otros factores externos. Los sensores infrarrojos se han empleado en la detección de objetos.

Figura 22: Sensor infrarrojo reflexion modelo 1

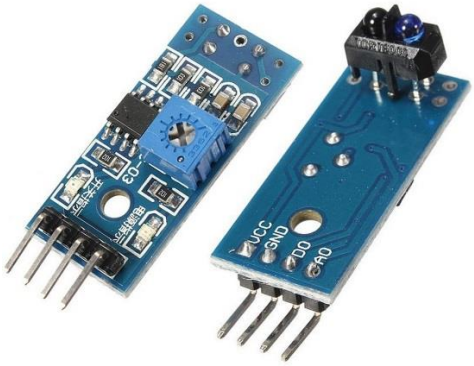


Figura 23: Sensor infrarrojo reflexion modelo 2

Para la detección del bulto o palet en la bahía de carga se ha empleado un sensor como el de la *figura 22*, que detecta si hay un palet cargado en la bahía.

Por otro lado, para el carro del eje Z, se ha utilizado un sensor como el de la *figura 23* más reducido, que detecta si se ha cargado un palet o bulto sobre dicho carro.

Por último, se ha utilizado un encoder óptico para el posicionamiento tanto del eje X como del eje Y.

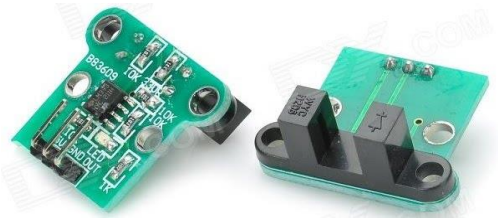


Figura 24: Encoder Óptico HC-020K

Esto proporciona un bucle cerrado de realimentación, con el cual podemos saber desde la posición de *Home* el desplazamiento que ha tenido lugar en cada uno de los dos ejes. Este mecanismo proporciona una alta precisión de posicionamiento. Los encoders utilizados para este proyecto son los que se pueden ver en la *figura 24*.

1.4.2.2.2 Finales de Carrera mecánicos

El funcionamiento de estos sensores corresponde a situaciones límite, por lo que su utilización va a ser esporádica o casi nula. Los finales de carrera que se han empleado en el diseño y construcción corresponden al tipo con roldana. En la *figura 25* se puede ver el modelo empleado en la maqueta. Este tipo de sensor se emplea, como su nombre indica, para detectar el final del recorrido de un movimiento. El objeto presiona la roldana y esta gira por contacto con el mismo.

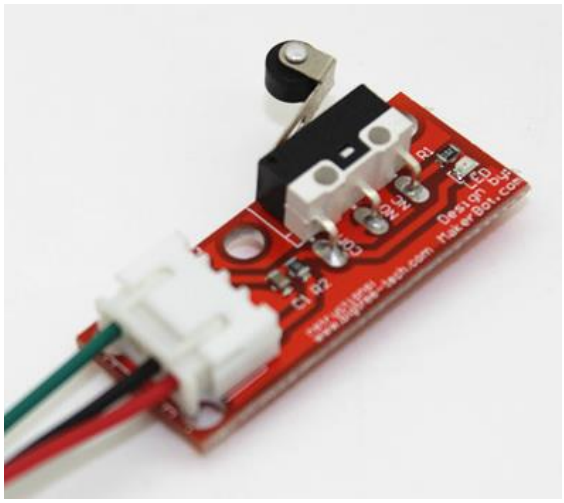


Figura 25: Final de carrera por roldana

Este tipo de sensor se ha empleado para detectar el final de carrera del eje X y del eje Y, activando una emergencia y parando los motores de los distintos ejes.

1.4.2.3 Actuadores

Para el diseño de la maqueta se han gastado diferentes modelos de actuadores, como motores paso a paso (PaP), placa de relés y pilotos luminosos.

1.4.2.3.1 Motores

Se han empleado varios modelos de motores PaP. Para el eje X e Y se han utilizado dos motores paso a paso idénticos (PaP) modelos *Nema 17*. Para el eje Z se ha empleado un modelo *Canon RG5-0764* también paso a paso.

Se han elegido esta clase de motores por su ajuste de posición y por su alto par a bajas revoluciones. El único inconveniente es que resulta necesario utilizar un driver o controlador específico para dichos motores.

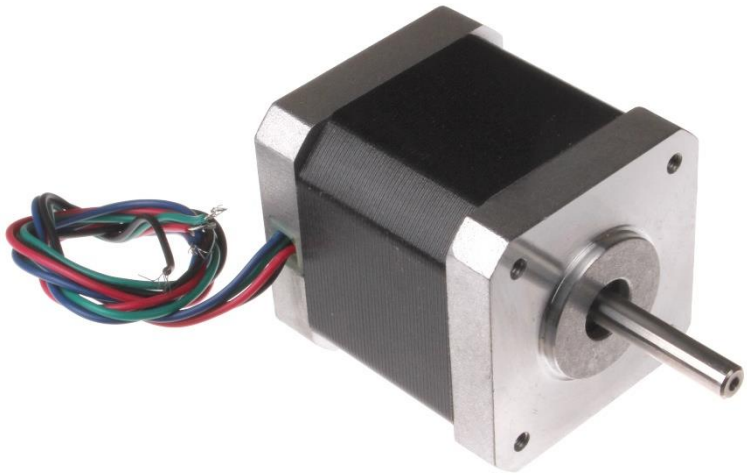


Figura 26: Motor Nema 17

Para los ejes X e Y se ha empleado un motor como el de la *figura 26* modelo *Nema 17.* En el eje Z se ha utilizado un motor como el de la *figura 27* modelo *Canon RG5-0764*.

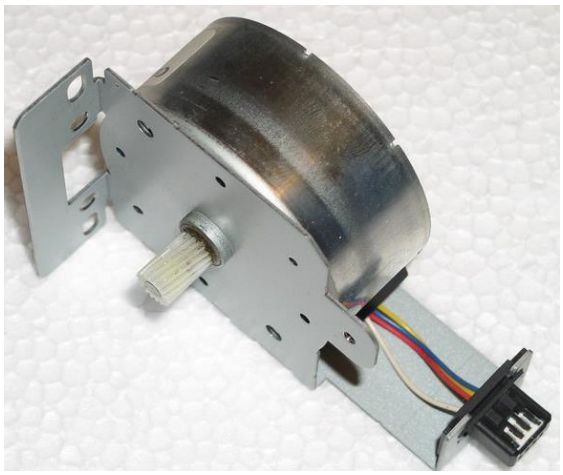


Figura 27: Motor Canon RG5-0764.

Para el correcto funcionamiento de los motores paso a paso se ha escogido un controlador específico. La placa Arduino® se encarga de controlar por medio de las salidas el funcionamiento de este driver. Los motores paso a paso funcionan con una secuencia específica de pulsos. Este controlador o driver ya implementa todo este sistema para generar dicha secuencia y controlar las distintas actuaciones o controles que tienen sobre el motor.

En este caso se han utilizado unos drivers *modelo TB6560* (véase *figura 28*). Estos drivers tienen un control sobre la intensidad consumida por el motor, la intensidad de frenado, el tipo de pasos y la deceleración.

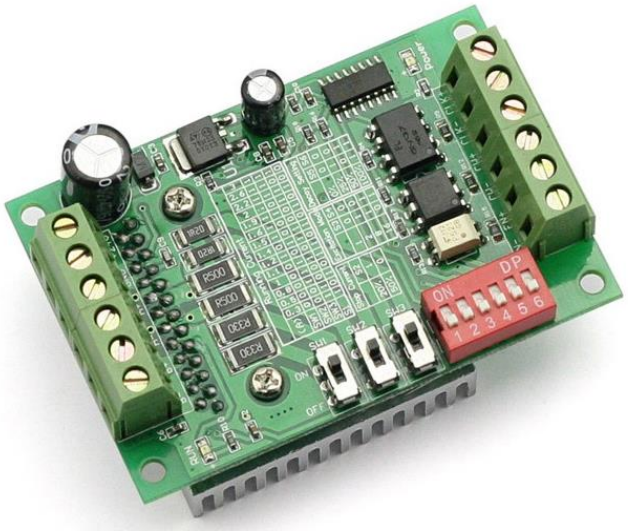


Figura 28: Driver de motor TB6560

1.4.2.3.2 Relés

La placa Arduino®, como ya se ha comentado anteriormente, no está capacitada para suministrar más de 50mA por pin de salida. Por ello, una salida no puede activar una carga de gran consumo de corriente. Por tanto, resulta necesario la utilización de relés para gobernar las cargas de mayor potencia con simples señales del Arduino® de baja potencia.

En el mercado se ha encontrado una placa comercial de 4 relés ya montada. En la *figura 29* se puede ver como es esta placa.

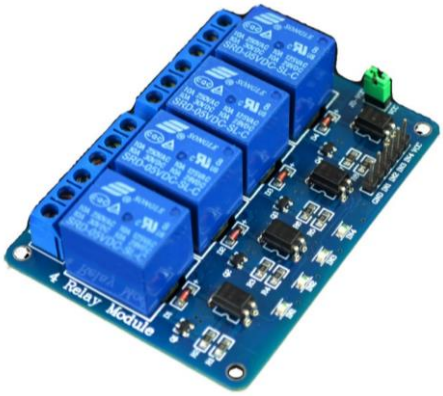


Figura 29: Placa de 4 relés

Dos de estos relés gobiernan sobre las cargas de la alimentación de los motores y de los pilotos luminosos. Por otro lado, los otros dos controlan la señal de giro de los drivers de los motores del eje X e Y. De esta forma, actúan como una emergencia para frenar dichos motores.

1.4.2.3.3 Pilotos luminosos

En esta maqueta se ha integrado un sistema de emergencia, el cual tiene varias actuaciones sobre la paletizadora. Para que estos estados de emergencia sean visibles, se han instalado unos pilotos luminosos en forma de baliza, como se puede observar en la *figura 30*. Ésta contiene dos pilotos luminosos; uno de color verde que se enciende cuando está todo correcto y otro de color rojo que se enciende cuando existe una situación de emergencia. Esta baliza está gobernada por uno de los relés de la placa comentada anteriormente.

Figura 30: Baliza Luminosa

1.4.2.4 Fuente de alimentación

En este proyecto se ha utilizado una fuente de alimentación conmutada modelo ATX, la cual pertenecía a un ordenador como se puede observar en la *figura 31*.

Figura 31: Fuente de Alimentación ATX

Se ha adaptado el cableado para utilizar las dos salidas de tensión, necesarias tanto para la electrónica (+5V) y la alimentación de los motores (+12V). Esta fuente de alimentación tiene una potencia suficiente para mantener en funcionamiento toda la maqueta y está ubicada en el interior de una caja de ordenador. Ésta ha sido empleada como estructura para el montaje del cuadro eléctrico, modificando su interior para la instalación de los componentes eléctricos y electrónicos.

1.4.3 Ajuste final

Para que todo el proyecto en su conjunto funcione correctamente es necesario realizar varios ajustes finales.

Uno de ellos son los sensores infrarrojos. Estos sensores tienen un ajuste fino para la detección de objetos delante de él. Se han adecuado para que detecten un objeto encima, así como cuando este objeto es retirado.

En el eje X de la maqueta es necesario ajustar correctamente el sensor de la posición de *Home* del eje X para que el carro móvil se pare en el mismo sitio en cada ciclo de funcionamiento. También es necesario ajustar el final de carrera de roldana colocado al final del recorrido del eje X, con el objetivo de que se active en caso de emergencia, parando el motor de dicho eje.

Para el eje Y se han montado dos sensores iguales que el eje X. El final de carrera de barrera se ha ajustado a la posición correcta para que se detenga siempre en la posición de *Home* del eje Y. Para el límite superior se ha adecuado la roldana para que detecte el tope superior y el carro móvil se detenga antes de llegar al final del recorrido.

A parte de estos sensores la maqueta dispone de unos topes mecánicos, de modo que en el caso de fallar los demás sistemas, éstos sean capaces de evitar mayores daños en la maqueta.

Con el objetivo de que los motores giren a una velocidad específica, resulta conveniente ajustar la velocidad por medio de la señal de reloj que recibe los drivers de cada motor. Esta señal de reloj es creada mediante la programación del Arduino®, y de esta forma se consigue variar la frecuencia de la señal de reloj. Como cada uno de los tres ejes utiliza distintos engranajes es necesario adaptar la velocidad independientemente en cada motor. Con todo ello, se logra modificar la velocidad de cada uno de los motores.

1.5 Programación

1.5.1 Protocolo de comunicación

Se ha establecido un sistema de comunicación conforme con el proyecto de mi compañero Pedro José Ante Espada (*Diseño Placa de Comunicación por Buses Industriales para Arduino®*), complementándose ambos dos.

Las órdenes que recibe el sistema de paletización para que éste ejecute la secuencia de programa, forman parte del proyecto de mi compañero. Por todo ello, es en esta parte donde se unen ambos proyectos.

Para llevar a cabo la comunicación se ha establecido un lenguaje con el objetivo de intercambiar información. Con este lenguaje se consigue que las dos partes puedan comunicarse de forma correcta, siendo participe ambos proyectos.

1.5.2 Sistema de programación

El algoritmo que se ha empleado en este proyecto ha sido diseñado desde cero y comprobado para su correcto funcionamiento.

Existen muchas formas de programar o de estructurar la programación. En este proyecto se ha optado por una segmentación y estructuración por funciones. Cada función se encuentra en un sketch o módulo separado.

Se va a dar paso al módulo principal que contiene el "*main*" o código principal el cual se ejecuta cíclicamente.

1.5.2.1 Main

En este módulo se inicia el código principal. En él se ha desarrollado el bucle principal y la declaración de todas las variables necesarias. Dentro de él se encuentran las dos declaraciones básicas para la programación del Arduino®. Estas declaraciones son el *setup* () y el *loop* ().

En la función *setup* se declara toda la configuración de pines, interrupciones y comunicación empleadas para la programación.

En la función *loop*, el código se ejecutará constantemente en bucle. Dentro de éste se encuentra la llamada a las diferentes tareas a realizar.

1.5.2.2 Comunicación

En este apartado se exponen todas las funciones empleadas para la recepción de las órdenes enviadas por parte del proyecto de mi compañero Pedro José Ante (*Diseño Placa de Comunicación por Buses Industriales para Arduino®*). Éstas incluyen todos los comandos necesarios para recibir y enviar las tramas de comunicación.

El lenguaje de comunicación entre ambos proyectos se compone de una trama de tres valores que recibe la paletizadora. Estos tres valores equivalen a:

- *Valor1:* este dato corresponde con el valor del comando a ejecutar: Dejar, Coger, Leer una posición de memoria, Borrar una posición de memoria, Borrar toda la memoria y Grabar una posición de memoria.
- *Valor2:* este dato corresponde con el valor de las columnas (eje X).
- *Valor3:* este dato corresponde con el valor de las filas (eje Y).

Para finalizar las órdenes de comunicación, el sistema de paletizado devuelve por el bus de comunicación una orden de tarea finalizada, quedando a la espera de recibir una nueva orden.

1.5.2.3 Cargar

En este módulo se ha realizado la programación de la secuencia de cargar bultos con el eje Z. En él se encuentran los diferentes comandos para poder mover dicho eje a derechas, a izquierdas o modo estático en la posición *Home*.

La implementación de la opción de cargar se ha hecho común para las distintas tareas involucradas. Es decir, se ha empleado la misma secuencia de programación para las acciones de carga desde la bahía de carga/descarga o análogamente, carga desde la estantería.

1.5.2.4 Descargar

En este sketch se ha estructurado la orden de descargar un palet o bulto. Como la paletizadora ofrece la posibilidad de descargar de dos formas diferentes, se ha

establecido una rutina común de descarga, pero realizando una preselección en función del lugar donde se quiera depositar el palet o bulto.

Tal como ya se ha comentado, una de las descargas que puede realizar es en la bahía de carga/descarga, o de forma análoga, cargarlo desde la estantería.

1.5.2.5 EEPROM

El sistema de automatización gestiona la memoria EEPROM (Electrically Erasable Programmable Read-Only Memory) de la que dispone Arduino®. La placa que se ha empleado para la automatización dispone de 4Kb de memoria. Se trata de un tipo de memoria no volátil, permaneciendo constante con la interrupción del suministro de corriente.

La EEPROM la gestiona el mismo Arduino® por medio de unos comandos ya implementados en el código de programación. De esta manera, evitamos que la paletizadora pierda el histórico de las posiciones libres y ocupadas.

En este módulo se han incluido todas las funciones necesarias para poder gestionar el espacio de la memoria interna. Entre ellas se encuentran:

- ***Checkeeprom ().*** Esta función comprueba la memoria EEPROM cuando recibe una orden, tanto para dejar como para extraer un bulto o palet en la estantería.
- ***Readeeprom1 ()***. Esta función lee el valor de la memoria EEPROM que se ha enviado por la comunicación. Guarda el valor leído para posteriormente enviarlo de nuevo.
- ***Deleprom1 ()***. Esta función permite eliminar el valor de la memoria EEPROM que se le ha enviado por la comunicación.
- ***Deleprom ()***. Esta función borra por completo las posiciones de la paletizadora almacenadas en la memoria EEPROM.
- ***Ripeeprom1 ()*** Esta función graba un nivel alto (1) en la posición de la memoria EEPROM correspondiente a la posición recibida en la orden.
- ***Ripeeprom ()***.Esta función graba un nivel alto o un nivel bajo según corresponda a la orden de coger (nivel bajo) o la orden de dejar (a nivel alto) un palet o bulto en la estantería.

1.5.2.6 Emergencia

En esta maqueta se han tenido en cuenta las medidas de seguridad necesarias, como si de un almacén real se tratase.

Se ha empleado una seta de emergencia así como los posibles casos de emergencia que puedan darse, los cuales se han incluido en este sketch. Para la programación de esta parte se ha seguido una lógica negada, tal como se utiliza en los sistemas de emergencia reales en automatismos.

- ***Emergencia ()***. Esta función es ejecutada en cualquier momento al pulsar la seta de emergencia.
- ***EmergenciaX ()***. Esta función se ejecuta en cualquier momento cuando se activa el final de carrera límite del eje X, cortando el funcionamiento del motor del mismo eje.
- ***EmergenciaY ()***. Esta función se ejecuta en cualquier momento cuando se activa el final de carrera límite del eje Y, cortando el funcionamiento del motor del mismo eje.

1.5.2.7 Entrar

En esta pestaña o sketch se encuentra declarada la función de entrar a la posición deseada, la cual viene dada por la orden recibida. El carro se posiciona, mediante el encoder del eje X y del eje Y, en el hueco de la estantería ordenada.

1.5.2.8 Funciones

En este apartado se han agrupado las diferentes funciones empleadas en función de las órdenes utilizadas por la paletizadora, tales como:

- ***Clock_X ()***. Genera el pulso de reloj a una frecuencia específica para el driver del motor X.
- ***Clock_Y ()***. Genera el pulso de reloj necesario para el driver del motor Y.
- ***Clock_Z ()***. Genera el pulso de reloj necesario para el driver del motor Z.
- ***Encoreset ()***. Resetea los contadores del encoder del eje X y del eje Y.
- ***Sysreset ()***. Resetea todas las variables necesarias para reiniciar el programa de la paletizadora y dejarla lista a la espera de una nueva orden.

- ***CuentaX ()***. Cuenta los impulsos del encoder del eje X.
- ***CuentaY ()***. Cuenta los impulsos del encoder del eje Y.

1.5.2.9 Home

En este sketch se ha implementado el algoritmo para que el sistema compruebe que todos los ejes de la paletizadora se encuentren en la posición *Home*.

Esto permite que al iniciar la paletizadora sea posible comprobar su posición en *Home*, estando así a la espera de una orden. Lee los sensores necesarios para saber dónde se encuentra, encendiendo así los motores oportunos para volver automáticamente a la posición *Home*.

1.5.2.10 Motores

Para simplificar la programación se han creado funciones específicas para cada uno de los tres motores que mueven los tres ejes, el eje X, el eje Y y el eje Z.

Las funciones de este sketch ayudan a programar con más sencillez y claridad dejando el código más estructurado y fácil de modificar en cualquier momento. Se compone de una serie de funciones sencillas para encender los motores.

Dentro se encuentran las funciones de:

- ***Xmotor_R*** (). Activa motor del eje X a la derecha.
- ***Xmotor_L*** (). Activa el motor del eje X a la izquierda.
- ***Xmotor_OFF*** (). Apaga el motor del eje X.
- ***Ymotor_Up*** (). Activa el motor del eje Y hacia arriba.
- ***Ymotor_Down*** (). Activa el motor del eje Y hacia abajo.
- ***Ymotor_OFF*** (). Apaga el motor del eje Y.
- ***Zmotor_R*** (). Activa el motor del eje Z a la derecha.
- ***Zmotor_R*** (). Activa el motor del eje Z a la izquierda.
- ***Zmotor_OFF*** (). Apaga el motor del eje Z.
- ***XYmotor_OFF*** (). Apaga los motores del eje X y del eje Y a la vez.
- ***Disable_motorX*** (). Desactiva el motor del eje X, dejándolo libre de movimiento.

- ***Disable_motorY*** (). Desactiva el motor del eje Y dejándolo libre de movimiento.

1.5.2.11 Salir

En este módulo se encuentra la función de salir de la estantería, la cual se ejecuta al finalizar la orden de coger o dejar un palet o bulto. Esta función envía directamente a la posición *Home* del eje X y del eje Y.

1.5.2.12 Tareas

En este sketch se encuentran las funciones principales para las tareas que puede realizar la paletizadora. Éstas se han separado, implementándolas en una función diferente. A continuación se explica en qué consiste cada tarea del programa.

- ***P01 ()***. Esta función realiza las tareas necesarias para introducir un palet o bulto en la estantería.

  La secuencia comienza cargando un bulto desde la bahía de carga/descarga, para posteriormente posicionarse en las coordenadas correspondientes que se le ha ordenado.

  Una vez alineado con la posición, el carro deslizador deposita el palet o bulto en dicha posición.

  Para finalizar, la paletizadora vuelve a la posición *Home* a la espera de una nueva orden.

- ***P02 ()***. Esta función realiza las tareas necesarias para extraer un palet o bulto de la estantería.

  Comienza situándose en la posición de la estantería recibida por la orden.

  Una vez en la posición, el carro extrae el palet o bulto y vuelve a la posición *Home*.

  Finalmente, el carro deposita el palet o bulto en la bahía de carga/descarga.

- ***P03 ()***. Esta función lee una posición específica de la memoria EEPROM. La posición es recibida por la orden de leer la EEPROM.

  Por último, la paletizadora envía el estado de la posición leída vacía u ocupada.

- ***P04 ()***. Esta función elimina una posición de la memoria EEPROM.
  Esta posición es recibida por la orden de borrar la memoria.
  Una vez borrada, la paletizadora manda una confirmación de que ha eliminado la posición correctamente.

- ***P05 ()***. En esta función la paletizadora borra todas las posiciones de la memoria EEPROM donde almacena el estado de las posiciones de las estanterías.

- ***P06 ()***. En esta función la paletizadora graba en la memoria EEPROM un estado ocupado (nivel alto) en una posición dada. Ésta es recibida por la orden de grabar una posición.

- ***P100 ()***. Se trata de una función extra. Si falla la comunicación debido al envío de una orden que no se corresponde con ninguna de las anteriores, envía un mensaje de error y espera una nueva orden.

1.6 Conclusiones

Este proyecto se ha llevado a cabo con las limitaciones pertinentes, referidos a espacio, tiempo y presupuesto. Este sistema se podría llevar a la práctica en una instalación de tamaño real.

Asimismo, con una financiación e investigación adecuadas, se podría diseñar una maquinaria capaz de competir en el actual sector de los almacenes automáticos.

La correcta ejecución del sistema, resultado del ensamblaje del diseño mecánico y eléctrico, ha supuesto uno de los principales inconvenientes en la ejecución de esta maqueta, ya que no se conocían de antemano todos los posibles problemas que podían surgir a posteriori.

Por otro lado, se ha tenido que adaptar la programación concebida inicialmente a la maqueta, debido a que siempre serán necesarios ajustes entre la programación lógica inicial y la ejecución final del programa. Las posteriores modificaciones en la programación han permitido el correcto funcionamiento de cada una de las tareas ejecutadas sobre la paletizadora.

No obstante, la gran dificultad a la que se ha tenido que hacer frente ha sido el correcto funcionamiento de la comunicación entre la paletizadora y el bus de comunicación, a través del cual se reciben las órdenes que ejecuta el sistema. Esto se ha solucionado mediante la adaptación de las partes implicadas, implementando las modificaciones necesarias para una correcta comunicación.

A pesar de todo ello, se ha podido finalizar satisfactoriamente esta propuesta, cumpliendo las expectativas fijadas al comienzo del proyecto.

1.7 Valoración Personal

En primer lugar, este proyecto ha sido el resultado de una idea ambiciosa, donde se ha realizado el diseño y el montaje desde cero. Se trata de una réplica a escala la cual puede servir como base para posibles diseños de mayor envergadura.

El ensamblaje de las partes eléctricas y mecánicas ha supuesto un inconveniente, ya que no se disponían de los conocimientos necesarios, por lo que ha sido un reto de superación personal.

En la realidad, este sistema podría haberse ejecutado desde un nivel superior, empleando un mayor grado de sofisticación y experiencia, elevando así el nivel de detalle en todos los sentidos.

En la realización de este proyecto se ha puesto mucho empeño y dedicación, partiendo de la idea principal hasta la maqueta final ya construida. Han surgido distintos problemas y obstáculos, los cuales se han sabido solventar satisfactoriamente.

En el ámbito de la comunicación, ha supuesto un reto poder conseguir que la paletizadora reconociese las órdenes recibidas a través del proyecto de mi compañero Pedro José Ante Espada en el (*Diseño Placa de Comunicación por Buses Industriales para Arduino®*).

Finalmente, con las limitaciones personales y materiales de los que se disponían, se ha conseguido construir y ejecutar una maqueta de una paletizadora totalmente funcional, alcanzando la meta personal marcada inicialmente.

1.8 Líneas de Investigación Futuras

Una vez finalizado este proyecto se presentan las siguientes mejoras, las cuales podrían desarrollarse en un futuro. Éstas podrían ser:

1. La creación de un **sistema *HMI*** *(Human Machine Interface)* o aplicación informática para la interacción hombre-máquina, mediante la cual un operario, desde cualquier dispositivo, puede enviar órdenes a la paletizadora. Ésta ejecutaría dichas disposiciones desde un entorno completamente visual.
2. La **ampliación del tamaño** del proyecto, con el consecuente estudio estructural por parte de un profesional del sector industrial. Esta ampliación no afectaría al algoritmo utilizado en la programación. No obstante, resultaría conveniente la sustitución de los sensores y actuadores por otros acorde a su tamaño y especificaciones.
3. A partir de la implementación del sistema *HMI*, comentado anteriormente, sería necesaria la **mejora** en el ámbito de las **comunicaciones** entre la paletizadora y la computadora, las cuales intercambian órdenes. Esto conllevaría la mejora de los buses de comunicación para el intercambio de información.

1.9 Bibliografía

- *Mecalux. Transelevador trilateral automático*.

http://www.mecalux.com.mx/almacenes-automatizados-para-tarimas/transelevador-trilateral-automatico

- *Arduino®*

https://www.arduino.cc

- *Driver TB6560*

http://hetpro-store.com/TUTORIALES/tb6560-controlador-motor-pasos/

- *Encoders HC-020K*

https://androminarobot.blogspot.com.es/2016/04/tutorial-sobre-el-encoder-fotoelectrico.html

- *Sensores Infrarojos*

http://forum.arduino.cc/index.php?topic=334000.0

- Placa 4 Reles

https://www.pccomponentes.com/modulo-rele-4-canales-compatible-con-arduino

- *Motor Nema 17*

http://reprap.org/wiki/NEMA_17_Stepper_motor

- *Simbología Eléctrica*

http://www.portaleso.com/usuarios/Toni/web_simbolos/unidad_simbolos_electricos_indice.html

- *Tarjeta de adquisición de datos National Instruments®.*

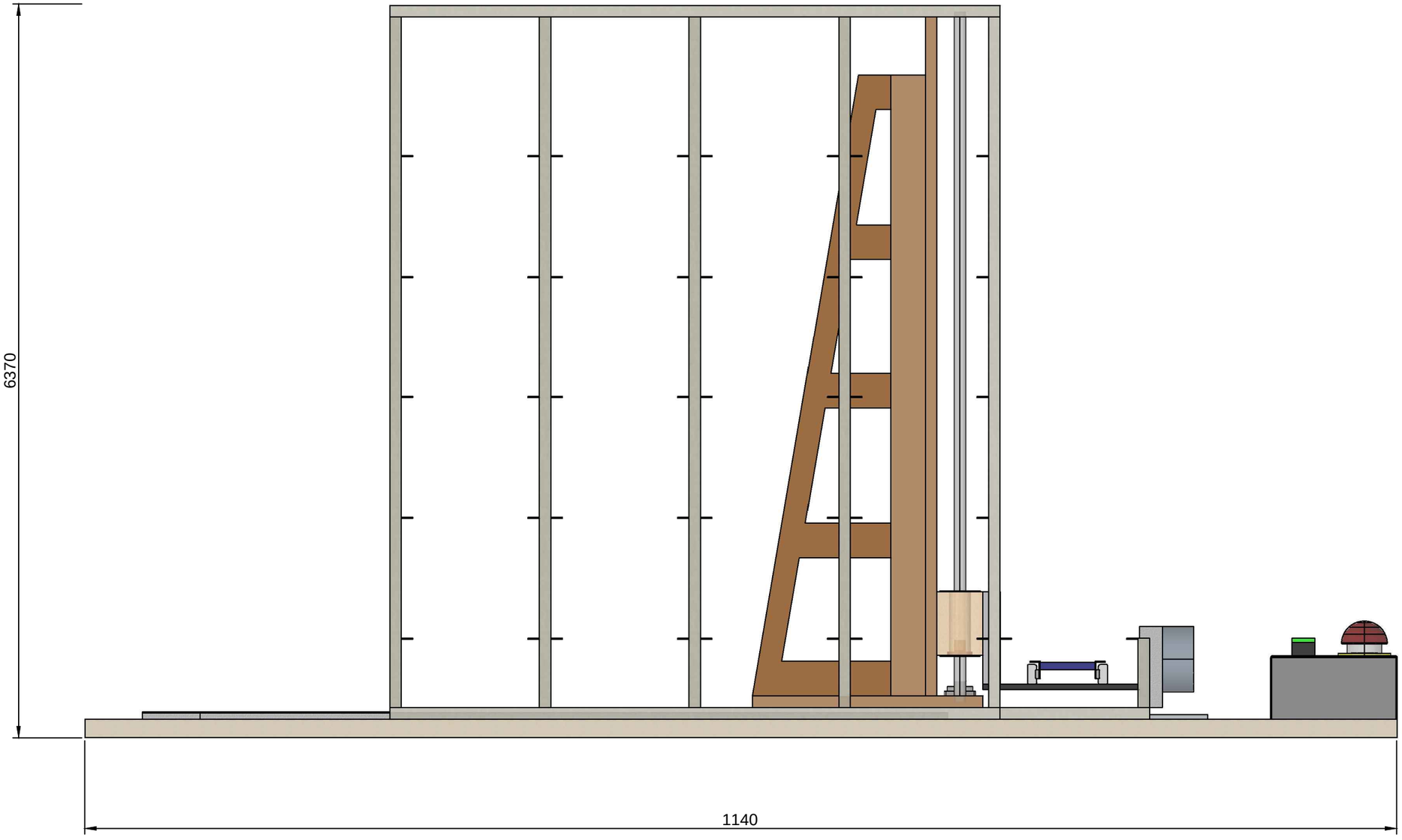
http://www.directindustry.es/prod/national-instruments/product-5074-981355.html

- *Siemens® Automatismos*

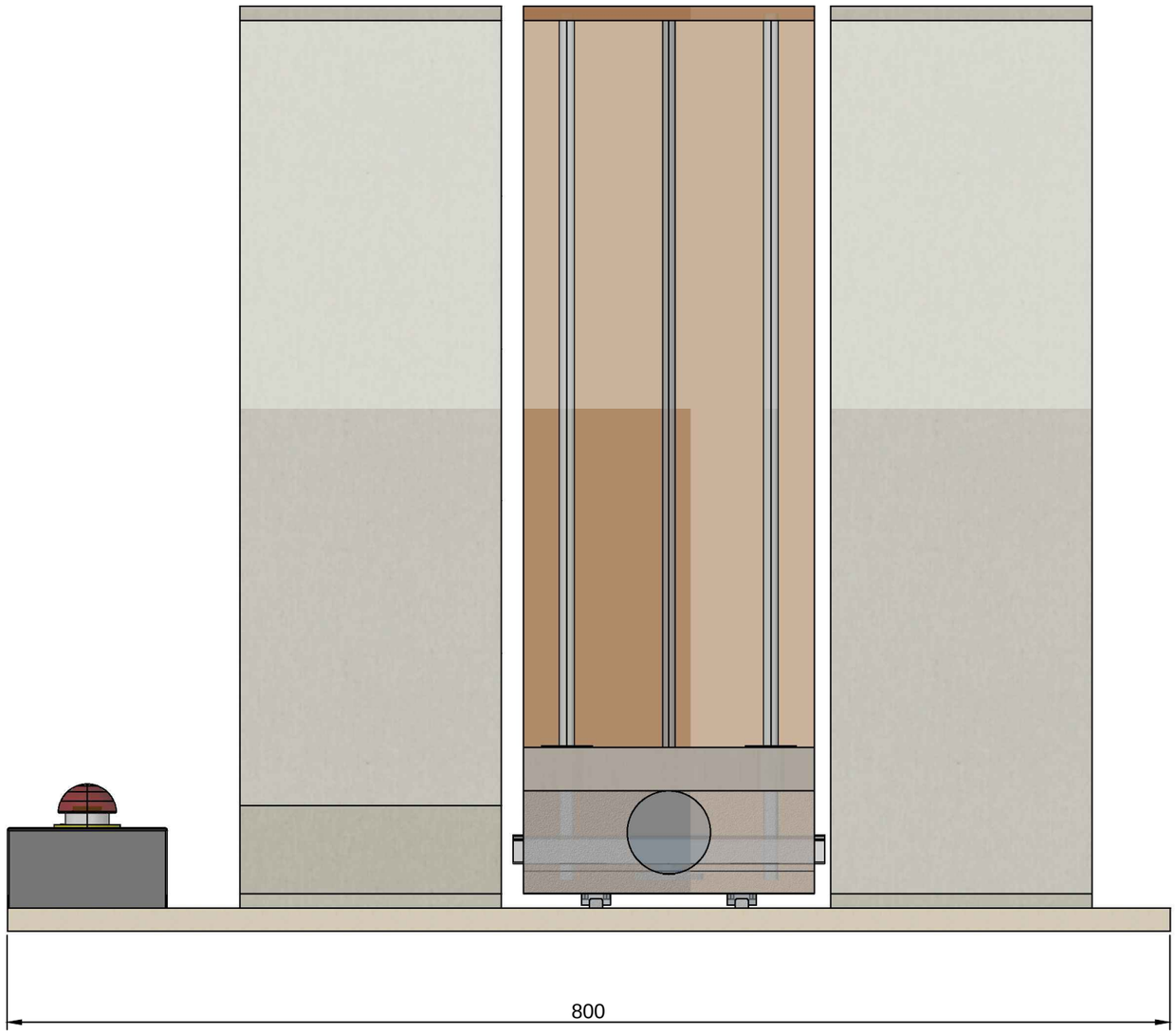
http://w3.siemens.com/mcms/programmable-logic-controller/en/pages/default.aspx

2 PLANOS

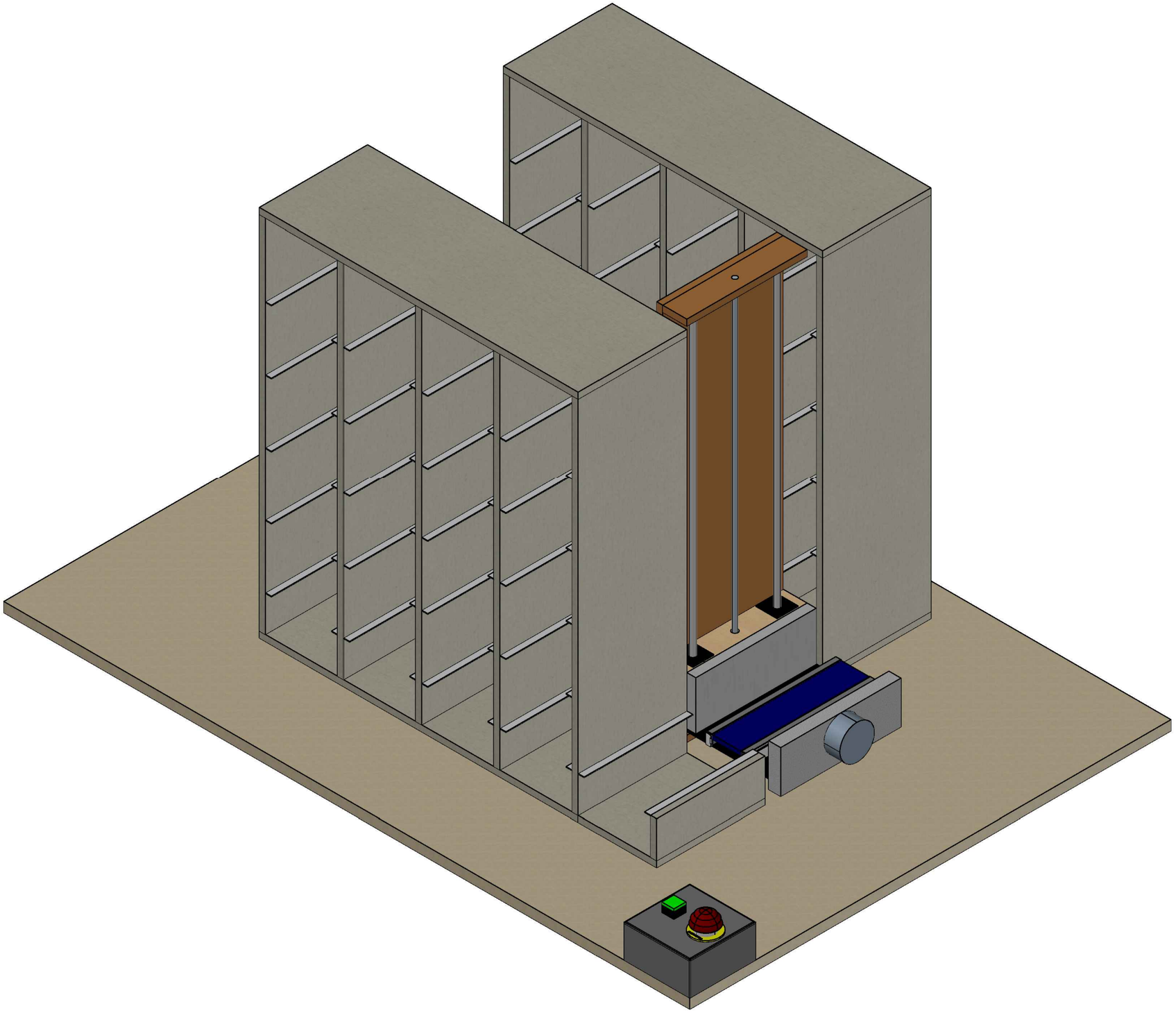
2.1 Planos Mecánicos



SIS.REP.	Escala	Unidades	PLANO	
	1:4	mm	PALETIZADORA VISTA LATERAL	
Plano Nº:	Fecha:	AUTOR	DISEÑO DEL SISTEMA AUTOMATIZADO DE UN ALMACÉN PALETIZADO	
1	07/08/2016	Alexander Rosales Martin		



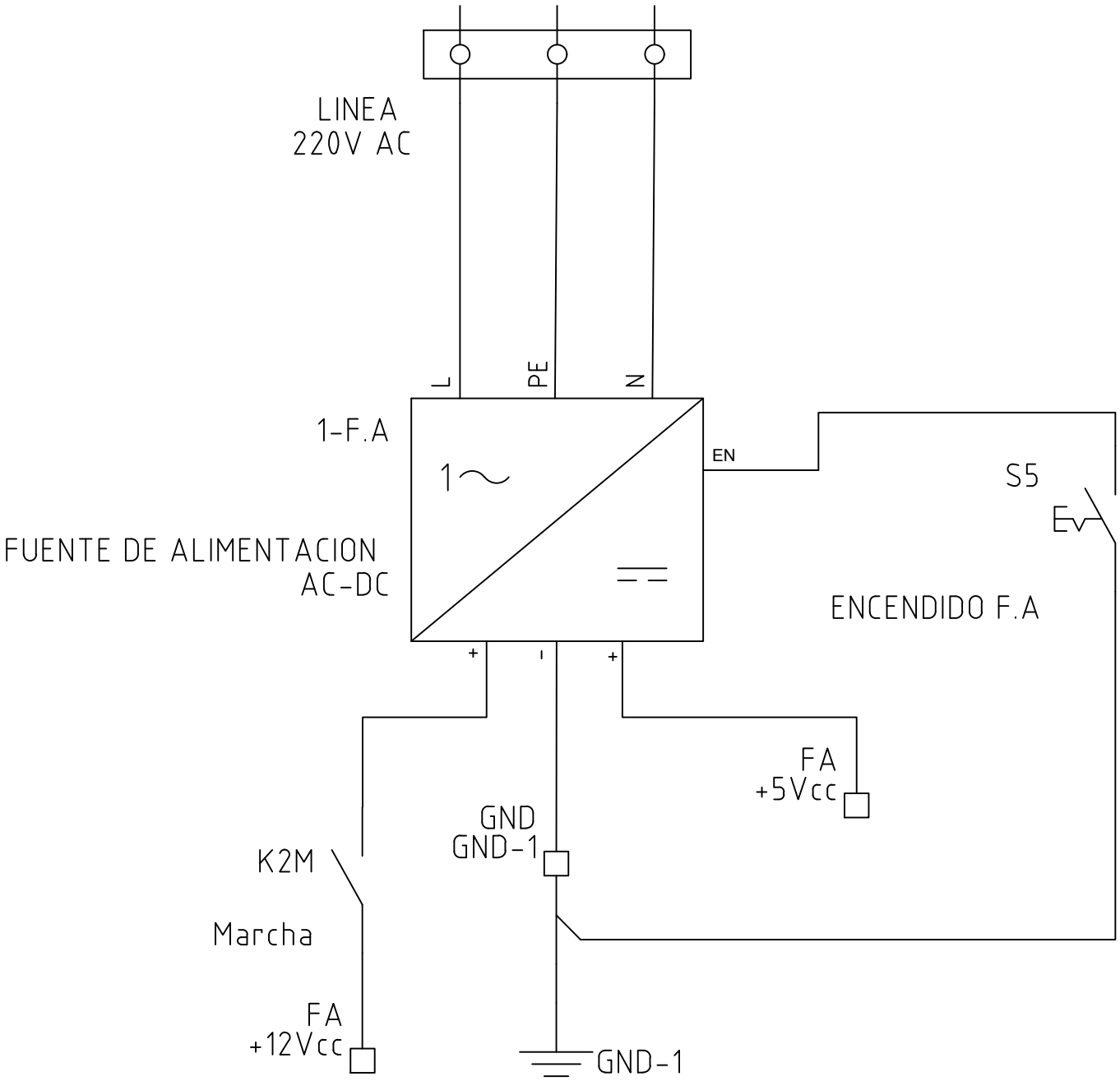
SIS.REP.	Escala	Unidades	PLANO	
	1:4	mm	PALETIZADORA VISTA FRONTAL	
Plano Nº:	Fecha:	AUTOR	DISEÑO DEL SISTEMA AUTOMATIZADO DE UN ALMACÉN PALETIZADO	
2	07/08/2016	Alexander Rosales Martin		



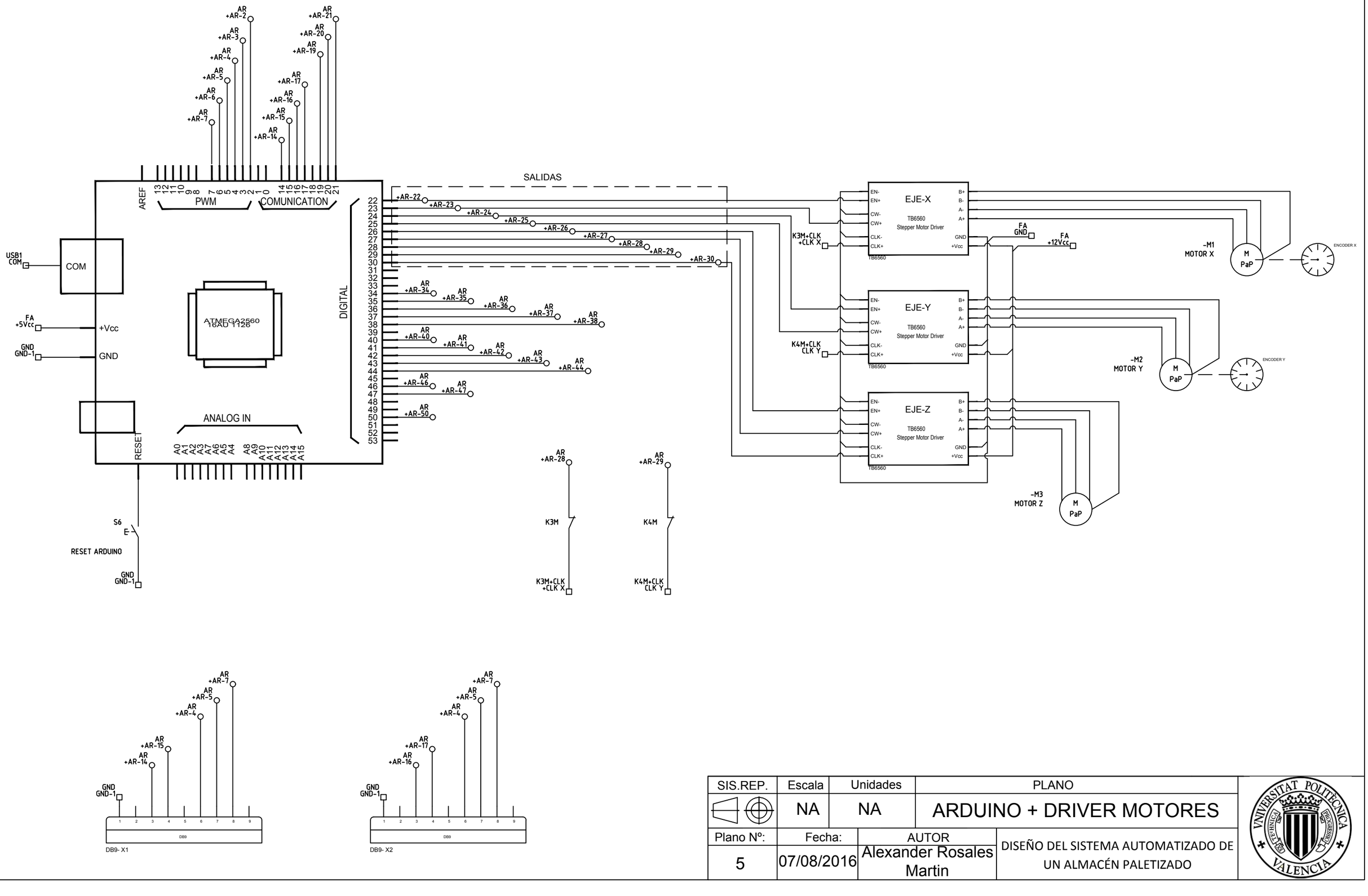
SIS.REP.	Escala	Unidades	PLANO	
	1:5	mm	PALETIZADORA VISTA PERSPECTIVA	
Plano Nº:	Fecha:	AUTOR	DISEÑO DEL SISTEMA AUTOMATIZADO DE UN ALMACÉN PALETIZADO	
3	07/08/2016	Alexander Rosales Martin		

VNIVERSITAT POLITÈCNICA VALENCIA

2.2 Planos Eléctricos

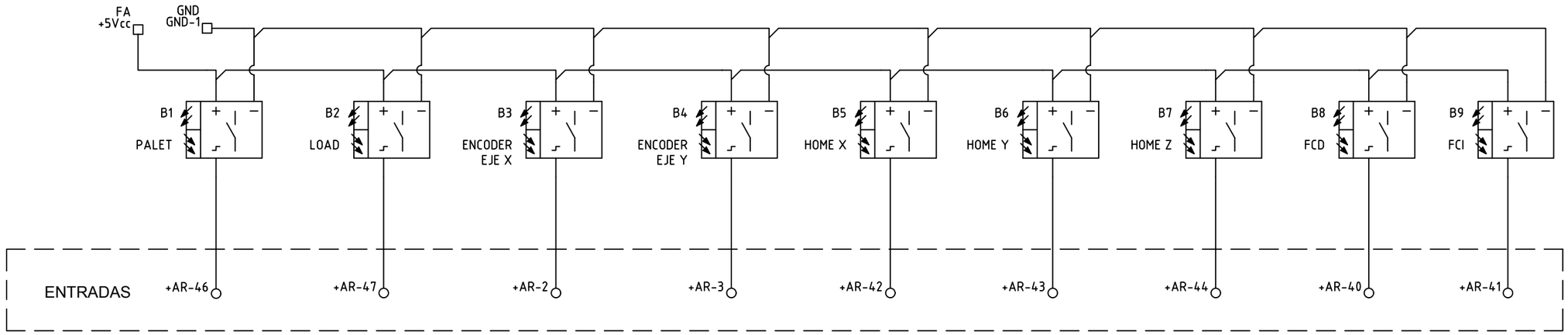


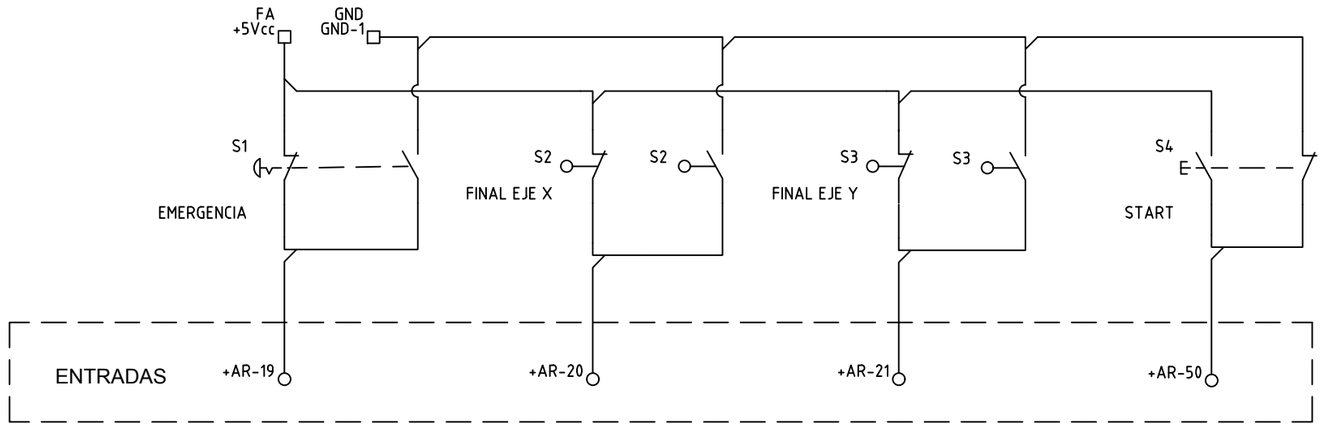
SIS.REP.	Escala	Unidades	PLANO	
	NA	NA	FUENTE DE ALIMENTACIÓN	
Plano Nº:	Fecha:	AUTOR	DISEÑO DEL SISTEMA AUTOMATIZADO DE UN ALMACÉN PALETIZADO	
4	07/08/2016	Alexander Rosales Martin		

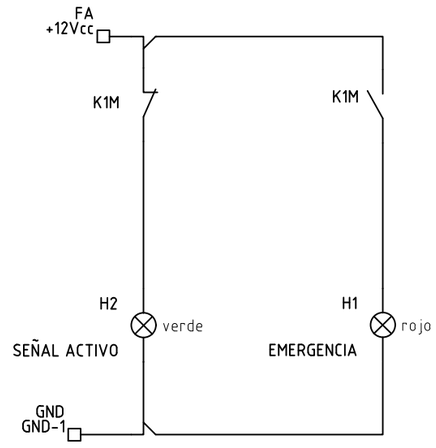


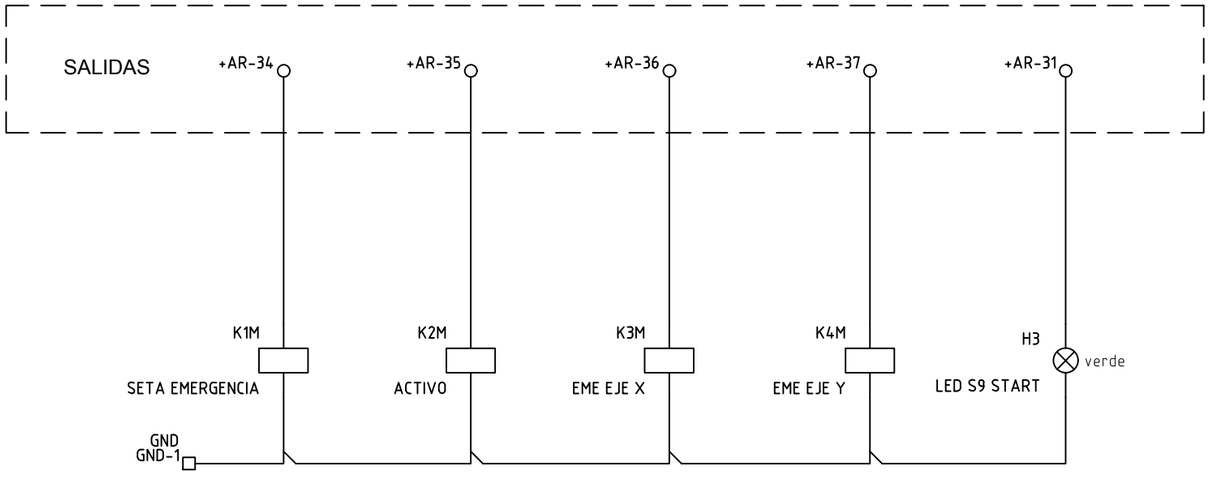
SIS.REP.	Escala	Unidades	PLANO
	NA	NA	ARDUINO + DRIVER MOTORES
Plano Nº:	Fecha:	AUTOR	DISEÑO DEL SISTEMA AUTOMATIZADO DE UN ALMACÉN PALETIZADO
5	07/08/2016	Alexander Rosales Martin	

VNIVERSITAT POLITÈCNICA VALENCIA

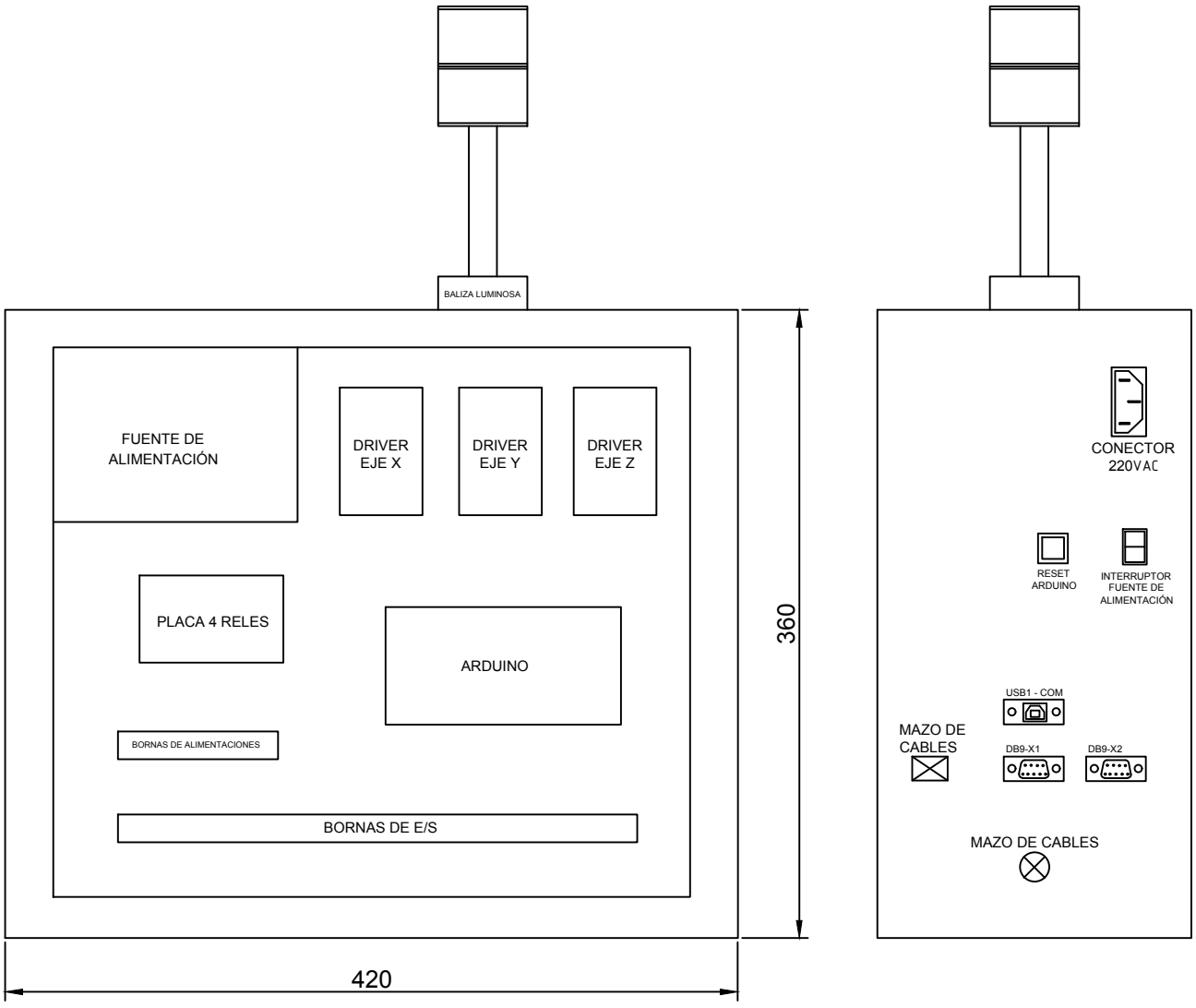


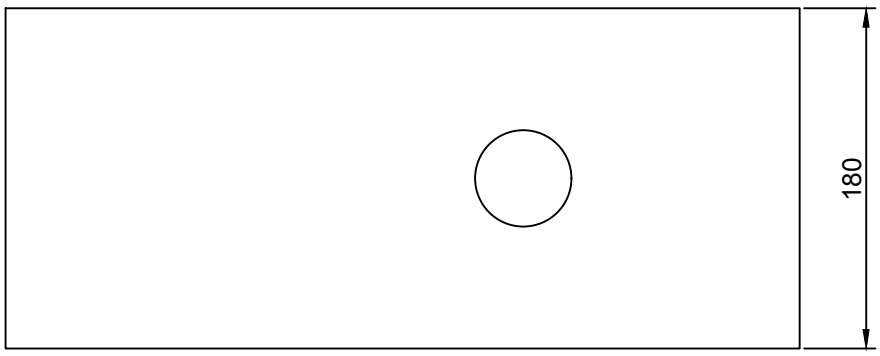






SIS.REP.	Escala	Unidades	PLANO
	NA	NA	ENTRADAS Y SALIDAS
Plano Nº:	Fecha:	AUTOR	DISEÑO DEL SISTEMA AUTOMATIZADO DE UN ALMACÉN PALETIZADO
6	07/08/2016	Alexander Rosales Martin	





SIS.REP.	Escala	Unidades	PLANO	
	NA	NA	CUADRO ELECTRICO	
Plano Nº:	Fecha:	AUTOR	DISEÑO DEL SISTEMA AUTOMATIZADO DE UN ALMACÉN PALETIZADO	
7	07/08/2016	Alexander Rosales Martin		

3 PRESUPUESTO

3.1 Partida de Materiales

MATERIALES	Unidades	Precio/Ud (€)	Coste (€)
Guía lineal eje X MGN9 350mm + I carro MG9H	2	14,79	29,58
Guía lineal eje X MGN9 600mm + 1 carros MG9H	1	24,91	24,91
Guía extraíble telescópica eje z 210mm	2	1,86	3,72
Varilla calibrada de acero Ø10mm eje z 600 mm	2	8,80	17,60
Rodamientos cilíndricos lineales LM10LUU 55mm	2	2,22	4,44
Bloque de rodamiento con amortiguación KFL08 Ø8mm	1	1,21	1,21
Tornillo sin fin trapezoidal THSL + tuerca 600mm Ø8mm	1	9,70	9,70
Rodamientos a bolas 3x10x4mm 10und	1	1,12	1,12
Polea dentada de aluminio 2GT Ø8mm 40 dientes	1	0,70	0,70
Polea dentada aluminio 2GT Ø5mm 30 dientes	1	0,63	0,63
Correa dentada 2GT 160mm	1	0,68	0,68
Cremallera dentada M1 eje X 5und	1	4,61	4,61
Cremallera dentada M0,5 eje Z 10und	1	2,36	2,36
Paquete con Juego de engranajes M1 Ø3mm	1	11,60	11,60
Cadeneta porta cabales articulada 10x20mm 1m	0,5	4,33	2,17
Cadeneta porta cabales articulada 10x7mm 1m	0,5	3,38	1,69
Placa de 4 relés 5V	1	4,56	4,56
Encoder óptico	2	2,74	5,48
Motor Nema 17 mod.17HS8401	2	12,67	25,34
Motor plano paso a paso RG5-764 Canon	1	14,76	14,76
Cables de 4 hilos para motores 20und	1	11,16	11,16
Caja eléctrica para botonera	1	2,76	2,76
Finales de carrera mecánicos de roldana	2	0,86	1,72
Finales de carrera ópticos de barrera	5	0,97	4,85
Escudo con tornillos para E/S para Arduino®	1	3,14	3,14
Pulsador verde	1	0,91	0,91
Regletas de conexión	5	0,84	4,20
Seta de emergencia	1	1,39	1,39
Conectores de alimentación para Arduino®	1	1,45	1,45
Sensor infrarrojos reflexivo	2	2,00	4,00
Interruptor para fuente de alimentación	1	1,45	1,45
Driver de motores paso a paso TB6560	3	6,00	18,00
Interruptor dos posiciones para comunicación	1	0,46	0,46
Maderas varios	1	58,00	58,00
Cables varios	1	19,00	19,00
Arduino® Mega 2560	1	35,00	35,00
Cable adaptador usb-usb	1	1,55	1,55
Conectores macho + hembra 4p 20und	1	5,46	5,46
Tornillería y perfilaría de aluminio	1	12,00	12,00
Fuente de alimentación	1	36,00	36,00
Caja de ordenador	1	29,00	29,00
		SUBTOTAL	418,36
		IVA 21%	87,85
		TOTAL	**506,21 €**

Tabla 2: Partida de materiales

3.2 Partida de Mano de Obra

PERSONAL	Horas	€/hora	Coste (€)
Mano de Obra de un Graduado en Ingeniería	300	11,50	3450,00
Mano de obra de un Técnico de Fabricación	200	9,50	1900,00
		TOTAL	**5.350,00 €**

Tabla 3: Partida de mano de obra

3.3 Presupuesto Total

PARTIDAS	Precio
Presupuesto de materiales	506,21
Presupuesto de mano de obra	5350,00
TOTAL	**5.856,21 €**

Tabla 4: Presupuesto Total

4 ANEXOS

CÓDIGO DEL PROGRAMA

MAIN

```
#include "EEPROM.h";


#define factor_x 20 //unidades para multiplicar cada posición eje X

#define factor_y 90 //unidades para multiplicar cada posición eje Y

#define factor_z 43 //pasos para subir/bajar cuando sube el carro o baja dejando o cogiendo caja


//------- VARIABLES --------

char accion = 'p' ; //acción del menú

char posx ; //posición X de la casilla a ir

char posy ; //posición Y de la casilla a ir

unsigned int prex = 0; //variable temporal de los pasos de x calculados

unsigned int prey = 0; //variable temporal de los pasos de y calculados


//Emergencia

volatile int eme = 1;


//checkeeprom()

int address = 0 ; //dirección donde lee o escribe la eeprom

bool readpos = 0;//variable donde guarda la información de la eeprom

int fallo = 0; //variable que activa el fallo en orden eeprom


//readeeprom(), deleprom()

int p = 0;

int e = 0;

int m = 0;
```

```
//cuentaX()
volatile unsigned int stepsx = 0; //variable donde cuenta los pasos de x
int q = 0; //variable para hacer el swich

//cuentaY()
volatile unsigned int stepsy = 0; //variable donde cuenta los pasos de y
int w = 0; //variable para hacer el swich

//clk X
bool clkEstadoX = LOW; //guarda el estado, inicializa a 0
unsigned long prevMillisX = 0; //inicia en 0 el primer tiempo de cuenta de x
unsigned long periodoX = 1;   //inversa de la frecuencia
unsigned long iniMillisX = 0; // guarda el tiempo inicial donde guarda milis

//clk Y
bool clkEstadoY = LOW; //guarda el estado, inicializa a 0
unsigned long prevMillisY = 0; //inicia en 0 el primer tiempo de cuenta de y
unsigned long periodoY = 1;   //inversa de la frecuencia
unsigned long iniMillisY = 0; // guarda el tiempo inicial donde guarda milis

//clk Z
bool  clkEstadoZ = LOW;  //guarda el estado, inicializa a 0
unsigned long prevMillisZ = 0; //inicia en 0 el primer tiempo de cuenta de z
unsigned long periodoZ = 4;   //inversa de la frecuencia
unsigned long iniMillisZ = 0; // guarda el tiempo inicial donde guarda milis

//------- COMUNICACION --------
//Variables del programa
int k = 0;
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}; // Pines 4 y 5 de arduino
int enable = 6; // pin 6 de arduino
int boton = 7; // pin 7 de arduino

//------- ENTRADAS -------- numero de pin
const int encox = 2;  //n pin encoder del eje x
const int encoy = 3;  //n pin encoder del eje y
const int emergency = 19; // seta de emergencia
const int emex = 20; //f.c de emergencia del eje x
const int emey = 21; //f.c de emergencia del eje y
const int fcD = 40;   //posicion eje z recogido
const int fcI = 41;   //posicion eje z extendido
const int home_x = 42; //home eje x fc
const int home_y = 43; //home eje y  fc
const int home_z = 44; //home eje z efecto hall
const int palet = 46; //sensor para detectar palet en la bahia de carga
const int load = 47;  //sensor para detectar palet cargado en el carro Z
const int start = 50;  //rearme
```

```
const int reset = 51; // Reset


//------- SALIDAS -------- numero de pin
const int puls = 31; //luz del pulsador verde
const int enx = 22;    //enable del motor x
const int girox = 23;  //sentido del motor x
const int eny = 24;    //enable del motor y
const int giroy = 25;  //sentido del motor y
const int enz = 26;    //enable del motor z
const int giroz = 27;  //sentido del motor z
const int clk_X =  28; //SeÃƒÂ±al de reloj para motor x
const int clk_Y =  29; //SeÃƒÂ±al de reloj para motor y
const int clk_Z =  30; //SeÃƒÂ±al de reloj para motor z
const int redL = 34; //luz de emergencia roja
const int greenL = 35; // salida de la luz verde
const int enable_x = 36; //enable driver X K3M
const int enable_y = 37; //enable driver Y K4M


void setup() {
  //------- ENTRADAS --------
  pinMode(fcD, INPUT);   //posicion eje z recogido
  pinMode(fcI, INPUT);   //posicion eje z extendido
  pinMode(home_x, INPUT); //home eje x
  pinMode(home_y, INPUT); //home eje y
  pinMode(home_z, INPUT); //home eje z
  pinMode(encox, INPUT); //lectura entrada de pasos del encoder
  pinMode(encoy, INPUT); //lectura entrada de pasos del encoder
  pinMode(palet, INPUT); //detectar palet en la bahia de carga
  pinMode(load, INPUT);  //detectar palet cargado en el carro Z
  pinMode(emex, INPUT);  //dectectar final de carrera del eje x
  pinMode(emey, INPUT);  //dectectar final de carrera del eje y
```

```
pinMode(emergency, INPUT);  //parada de emergencia

//------- SALIDAS --------
pinMode(puls, OUTPUT);    //salida de luz para el pulsador verde
pinMode(enx, OUTPUT);    //enable del motor x
pinMode(girox, OUTPUT);  //sentido del motor x
pinMode(eny, OUTPUT); //activar motor eje y
pinMode(giroy, OUTPUT);  //sentido del motor y
pinMode(enz, OUTPUT); //activar motor eje z
pinMode(giroz, OUTPUT);  //sentido del motor z
pinMode(redL, OUTPUT); //Luz de emergencia roja
pinMode(greenL, OUTPUT); //Luz de activacion verde
pinMode(clk_X, OUTPUT); // salida de reloj X
pinMode(clk_Y, OUTPUT); // salida de reloj Y
pinMode(clk_Z, OUTPUT); // salida de reloj Z
pinMode(enable_x, OUTPUT); // salida de rele reloj x
pinMode(enable_y, OUTPUT); // salida de rele reloj y

//------- INTERRUPCIONES --------
attachInterrupt(0, cuentaX, CHANGE);//pin 2 o rising
attachInterrupt(1, cuentaY, CHANGE); //pin 3
attachInterrupt(4, emergencia, LOW); //pin 19
attachInterrupt(3, emergenciaX, LOW); //pin 20
attachInterrupt(2, emergenciaY, LOW ); //pin 21

//------- PUERTOS --------
//comunicacion
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
  Serial2.begin(9600); //Inicia el puerto Serie 2//ok
  Serial3.begin(9600); //Inicia el puerto Serie 3 // ok
  Serial2.setTimeout(1000);
  Serial3.setTimeout(1000);
}

//---------------------------------- inicio programa -----------------------------------------------------
void loop() {
  inithome();//chequeo de la posicion de home lo primero
   Serial2.flush();
  Serial3.flush();
  Serial.flush();
  /*Espera a que se pulse el boton de inicio*/
  while (b == 0) {
    Inicio = digitalRead(boton);
    if (Inicio == LOW) {
      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();
    Serial2.flush();
  }
  while (Serial3.available()) {
    D = Serial3.read();
```

```
  Serial3.flush();
 }

 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
    posx =  Rx_232[0] - 48; //guarda la pos 0(primer numero) en la posx (restandole 48) para tener su valor original
    Serial.println(posx, DEC);
    posy = Rx_232[1] - 48;//guarda la pos 1(segundo numero) en la posy (restandole 48) para tener su valor original
    Serial.println(posy, DEC);
    accion = Rx_232[2];//guarda la pos 2(letra) en accion, sin restarle 48
    Serial.println(accion);
    Serial.println(accion, DEC);
   }
   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


      posx =  RxtT[0] - 48; //guarda la pos 0(primer numero) en la posx (restandole 48) para tener su valor original
      Serial.println(posx, DEC);
      posy = RxtT[1] - 48;//guarda la pos 1(segundo numero) en la posy (restandole 48) para tener su valor original
      Serial.println(posy, DEC);
      accion = RxtT[2];//guarda la pos 2(letra) en accion, sin restarle 48
      Serial.println(accion);
      Serial.println(accion, DEC);
    }
    break;


  default:
    break;
 }


 //orden(); //peticion de la orden
 switch (accion) { //Menu para las tareas
  case 65: //A 65
    P01(); //tarea 1 DEJAR pieza
    break;
  case 66: //B 66
    P02(); //tarea 2 COGER pieza
    break;
  case 67: //C 67
    P03(); //tarea 3 LEER 1 posicion de la EEPROM
    break;
  case 68: //D 68
    P04(); //tarea 4 BORRAR 1 posicion de la eeprom
    break;
```

```
   case 69: //E 69
     P05(); //tarea 5 BORRA toda la memoria eeprom
     break;
   case 70: //F 70
     P06(); //tarea 6 ESCRIBE 1 en una posicion la memoria eeprom
     break;
   default:
     P100(); //Tarea para errores, si falla algo
     break;
  }

  switch (Tipo_Out) { //Escoge entre salida por RS232 y RS485
   case 0:
     //Respuesta_232(); //Espera la respuesta
     Respuesta_trama_232(); //Devuelve la respuesta
     break;

   case 1:
     //Respuesta_485(); //Espera la respuesta
     Respuesta_trama_485(); //Devuelve la respuesta
     break;

   default:
     break;
  }
  Reset_var();
  sysreset();
}
```

CARGAR

```
//---------------------------------------- Cargar palet ----------------------------------------------------
void cargar() {
  if (accion == 65) { //(DEJAR)va a coger un palet de la bahía para dejarlo dentro
    if (digitalRead(palet) == HIGH) { //comprobar si hay palet en la bahia de carga(sensor)
      Serial.println("ERROR, No hay palet en la bahia"); // si no hay palet da error
      while (digitalRead(palet) == HIGH) {
      }
    }
    delay(3000); //espera un poco despues de poner el palet
  }
  if (address < 51 && accion == 66  ) { //quiere decir que esta cogiendo un palet de la otra
estanteria de la derecha (1 a la 4)
    while (digitalRead(home_z) == HIGH || digitalRead(home_z) == LOW) { //sacar el carro
      Zmotor_R ();
      if ((digitalRead(fcI) == LOW && digitalRead(fcD) == HIGH) && digitalRead(home_z) == LOW )
{
        Zmotor_OFF ();
        break;
      }
    }
    Zmotor_OFF ();//Apaga motores del carro
    delay(1000); //espera un poco para empezar a subir el carro
    encoreset();
    while (stepsy <= factor_z) { //subir el carro el numero del factor elegido
      Ymotor_Up ();//enciende motores para subir el carro un poco
      Serial.println(stepsy, DEC);
    }
    Ymotor_OFF ();//apaga motores el carro
    delay(800); //espera un poco para empezar a recoger el carro
    while (digitalRead(fcI) == LOW) { //recoge el carro
      Zmotor_L ();//enciende motores para recoger el carro
    }
```

```
   Zmotor_OFF ();
  }
  else {// hace el programa normal cogiendo de la estanteria (de la 5 a la 8)y la bahia de carga
   while (digitalRead(home_z) == HIGH) { //sacar el carro
    Zmotor_L();
    if ((digitalRead(fcI) == HIGH && digitalRead(fcD) == LOW) && digitalRead(home_z) == HIGH
) {
     Zmotor_OFF ();
     break;
    }
   }
   delay(500); //espera un poco para empezar a subir el carro
   encoreset();
   if (stepsy != factor_z) {
    while (stepsy <= factor_z) { //subir el carro el numero del factor elegido
     Ymotor_Up();
     Serial.println(stepsy, DEC);
    }
   }
   encoreset();
   Ymotor_OFF ();
   delay(500); //espera un poco para empezar a recoger el carro
   while (digitalRead(home_z) == HIGH) { //recoge el carro
    Zmotor_R();
    if ((digitalRead(fcI) == HIGH  &&  digitalRead(fcD) == HIGH) && digitalRead(home_z) ==
HIGH ) {
     Zmotor_OFF ();
     break;
    }
   }
   Zmotor_OFF ();
  }
```

```
switch (accion) {


 case 65: //va a coge un palet de la bahia para dejarlo dentro
  if (digitalRead(load) == HIGH) { //comprobar si hay palet en el carro cargado
   Serial.println("ERROR, NO SE HA CARGADO PALET");
   /////////buscar home Y y empezar de nuevo
   while (digitalRead(home_y) == LOW) {
    Serial.println("descargar 96");   // si el eje "y" esta desplazado
    while (digitalRead(home_y) == LOW) {
     Serial.println("home45");
     Ymotor_Down ();//enciende motores
    }
    //apagar motor
   }
   Ymotor_OFF ();
   encoreset();
   //while (1) {};
   delay(2000);
   cargar();
  }
  break;


 case 66: //coge un palet de dentro para dejarlo fuera en la bahia
  if (digitalRead(load) == HIGH) {
   Serial.println("ERROR, NO HAY PALET CARGADO");
   encoreset();
   if (stepsy != factor_z) {
    while (stepsy <= factor_z) { //bajar el carro el numero del factor elegido
     Ymotor_Down();
     cuentaY();
     Serial.println(stepsy, DEC);
```

```
          if (stepsy == factor_z) {
            Ymotor_OFF();
            break;
          }
        }
      }
      delay(2000);
    }
    cargar();
    break;
  }
  encoreset();
  return;
}
```

COMUNICACIÓN

```
////////////////////////////// RS 232 ////////////////////////////////////////////////////

//-------------------------Recepcion Rs232----------------------------
void Recepcion_Rs232(int a) {
  cont = 0;
  for (i = 0; i < 4; i++) {
    Rx_232[i] = ' ';
  }
  while (cont < a) {
    while (Serial2.available()) {
      if ((cont == 0) && (k == 1)) {
        Serial2.flush();
        Rx_232[cont] = Serial2.read();
        delayMicroseconds(1000);
```

```
      Serial.write(Rx_232[cont]);
    }
    Rx_232[cont] = Serial2.read();
    delayMicroseconds(1000);
    Serial.write(Rx_232[cont]);
    cont++;
   }
  }
  k = 0;
}

//-----------------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 1,2,3 para errores
      //Esperar_Respuesta();
      break;

    case 'B': /*Dejar*/
      //Mandar 0 cuando acaba, sino mandar 1,2,3 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.flush();
  Serial2.print(Res);
  Serial.println(Res);
  Serial2.flush();

}
////////////////////// RS485 ////////////////////////////////////////////////////////////

//-------------------------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)) {

   //Serial.print('D');
   D = Serial3.read();
   Serial3.flush();
   //Serial.println(D, BIN);
   //Serial.println(' ');
   z = z + 1;
  }

  if ((Serial3.available()) && (cont < 8) && (z >= 1) && (z < 5)) {

   if (z == 1) {
    Rx_485[cont] = Serial3.read();
    Serial3.flush();
    delayMicroseconds(1500);
    //Serial.print(cont);
    //Serial.println(Rx_485[cont], BIN);
    cont++;
    //delayMicroseconds(3000);
```

```
  if (cont == 8) {
    //Serial.println("z");
    z = z + 1;
    cont = 0;
  }
 } else if (z == 2) {
  Rx1_485[cont] = Serial3.read();
  Serial3.flush();
  //delayMicroseconds(1500);
  //Serial.print(cont);
  //Serial.println(Rx1_485[cont], BIN);
  cont++;
  //delayMicroseconds(3000);
  if (cont == 8) {
    //Serial.println("z2");
    z = z + 1;
    cont = 0;
  }
 } else if (z == 3) {
  Rx2_485[cont] = Serial3.read();
  Serial3.flush();
  //delayMicroseconds(1500);
  //Serial.print(cont);
  //Serial.println(Rx2_485[cont], BIN);
  cont++;
  //delayMicroseconds(3000);
  if (cont == 8) {
    //Serial.println("z3");
    z = z + 1;
    cont = 0;
  }
```

```
    } else if (z == 4) {
      Rx3_485[cont] = Serial3.read();
      Serial3.flush();
      //delayMicroseconds(1500);
      //Serial.print(cont);
      //Serial.println(Rx3_485[cont], BIN);
      cont++;
      //delayMicroseconds(3000);
      if (cont == 8) {
        //Serial.println("z4");
        z = 5;
        cont = 0;
      }
    }
   }
  }
  I++;
}

//-----------------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));
    //Serial.print(E0[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 1,2,3 para errores
      Esperar_Respuesta();
      break;

    case 'B': /*Dejar*/
      //Mandar 0 cuando acaba, sino mandar 1,2,3 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 de comunicacion ////////////////////////////////////////////////////////////

//----------------------------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();
  }
}
```

DESCARGAR

```
//------------------------------------------------ Descargar ------------------------------------------------

void descargar() {


  if (accion == 66) { //COGER //Deja un palet en la bahia de fuera

    if (digitalRead(palet) == LOW) {//comprobar si hay palet en la bahia de carga

      Serial.println("ERROR, Hay un palet en la bahia. \n Imposible Descargar ");

      while (digitalRead(palet) == LOW) {

        digitalWrite(redL, HIGH);

      }

      delay(1000);

      while (digitalRead(start) == LOW) {

        digitalWrite(puls, HIGH);

      }

      delay(3000);

    }

  }


  if (accion == 65 && address > 50) {

    if (stepsy != factor_z) {

      while (stepsy <= factor_z) { //subir el carro el numero del factor elegido

        Ymotor_Up();

        Serial.println(stepsy, DEC);

      }

    }

  }


  if (address < 51) { //quiere decir que esta dejando un palet de la otra estanteria de la derecha (1 a la 4)

    encoreset();

    if (stepsy != factor_z) {

      while (stepsy <= factor_z) { //bajar el carro el numero del factor elegido
```

```
      Ymotor_Up();
      Serial.println(stepsy, DEC);
      if (stepsy == factor_z) {
        Ymotor_OFF();
        break;
      }
     }
    }
    while (digitalRead(home_z) == HIGH || digitalRead(home_z) == LOW  ) { //sacar el carro
     Zmotor_R ();
     if ((digitalRead(fcI) == LOW && digitalRead(fcD) == HIGH) && digitalRead(home_z) == LOW )
{
      Zmotor_OFF ();
      break;
     }
    }
    Zmotor_OFF ();//Apaga motores del carro
    delay(1000); //espera un poco para empezar a subir el carro

    encoreset();
    while (stepsy <= factor_z) { //baja el carro el numero del factor elegido
     Ymotor_Down ();//enciende motores para bajar el carro un poco
     cuentaY();
     Serial.println(stepsy, DEC);
    }
    Ymotor_OFF ();//apaga motores el carro
    delay(800); //espera un poco para empezar a recoger el carro
    while (digitalRead(home_z) == HIGH || digitalRead(home_z) == LOW  ) { //recoge el carro
     Zmotor_L ();//enciende motores para recoger el carro
     if ((digitalRead(fcI) == HIGH && digitalRead(fcD) == HIGH) && digitalRead(home_z) == HIGH
) {
      Zmotor_OFF ();
```

```
      break;
     }
    }
    Zmotor_OFF ();
   }
   else {// hace el programa normal cogiendo de la estanteria (de la 5 a la 8)y la bahia de carga
    while (digitalRead(home_z) == HIGH) { //sacar el carro
     Zmotor_L();

     if ((digitalRead(fcI) == HIGH && digitalRead(fcD) == LOW) && digitalRead(home_z) == HIGH
) {
      Zmotor_OFF ();
      break;
     }
    }
    delay(500); //espera un poco para empezar a subir el carro

    encoreset();
    if (stepsy != factor_z) {
     while (stepsy <= factor_z) { //bajar el carro el numero del factor elegido
      Ymotor_Down();
      cuentaY();
      Serial.println(stepsy, DEC);
      if (stepsy == factor_z) {
       Ymotor_OFF();
       break;
      }
     }
    }
    Ymotor_OFF ();
    encoreset();
```

```
  delay(500); //espera un poco para empezar a recoger el carro


  while (digitalRead(home_z) == HIGH) { //recoge el carro
    Zmotor_R();
    if ((digitalRead(fcI) == HIGH  &&  digitalRead(fcD) == HIGH) && digitalRead(home_z) ==
HIGH ) {
      Zmotor_OFF ();
      break;
    }
  }
  Zmotor_OFF ();
 }


 switch (accion) {


  case 65: //DEJAR //Deja un palet dentro de la estanteria (dejar)
    if (digitalRead(load) == LOW) { //comprobar si hay palet en el carro
      Serial.println("ERROR, No se ha descargado el palet en la posicion");
      while (digitalRead(load) == LOW) {
        descargar();
      }
      delay(1000);
    }
    break;


  case 66: //COGER //Deja un palet en la bahia de fuera (coger)


    if (digitalRead(palet) == LOW) { //comprobar si hay palet en la bahia (sensor)
      Serial.println("ERROR, No se ha descargado palet en la bahia");
      while (digitalRead(palet) == LOW) {}
      delay(2000);
```

```
    }
    break;
  }
  return;
}
```

EEPROM

```
//------------------------------------ Chequeo de la memoria EEPROM -----------------------------------------
void checkeeprom() {
  //comprovacion de que en la posicion hay algo (para sacar) o no hay nada (para meter)
  address = ((posx) * 10) + (posy);//sumo "X" (decenas) e "y" (unidades) para crear un valor de direccion de la memoria eeprom
  Serial.print("Direccion de EEPROM: ");
  Serial.println(address, DEC);//muestro por pantalla el valor de la direccion

  readpos = EEPROM.read(address);
  switch (accion) {
   case 65://va a dejar pieza (por lo que tiene que estar vacia la posicion)
    if (readpos == 0) {
      Serial.println("La posicion para dejar es correcta");
    }
    else {// si no lo es, muestra un error y vuelve a llamar a orden() para volver a empezar
      Serial.println("Hubo un error inesperado");
      Serial.println("La posicion deseada ya esta ocupada");
      delay(3000);
      Serial.println("Seleccione una posicion vacia o cambie de opcion");
      fallo = 1;

    }
    break;
```

```
   case 66://va a coger pieza (por lo que tiene que haber algo en la posicion)
     if (readpos == 1 ) {
       Serial.println("La posicion para coger es correcta");
     }
     else {// si no lo es, muestra un error y vuelve a llamar a orden() para volver a empezar
       Serial.println("Hubo un error inesperado");
       Serial.println("La posicion para coger esta vacia");
       delay(3000);
       Serial.println("Seleccione una posicion ocupada o cambie de opcion");
       fallo = 1;
     }
     break;
  }
  return;
}

//---------------------------------- lee 1 posicione de la memoria EEPROM ---------------------------------
void readeeprom1() {
  address = ((posx) * 10) + (posy);
  Serial.print("POS.");
  Serial.print("\t");
  Serial.println("ESTADO");
  readpos = EEPROM.read(address);//lee la eeprom
  Serial.print(address, DEC);
  Serial.print("\t");
  Serial.println(readpos, DEC);
  delay(1000);
  if (readpos == 1) {
    Res = '1';
  }
  if (readpos == 0) {
```

```
    Res = '0';
  }
  return;
}

//------------------------------ borra 1 posicion de la memoria EEPROM -----------------------------------
void deleeprom1() {
  //vaciar la posicion deseada de la eeprom (0)
  address = ((posx) * 10) + (posy);
  EEPROM.write(address, 0);
  Serial.print(address, DEC);//esto no es necesario
  Serial.println("\t Memoria EEPROM borrada satisfactoriamente");
  delay(2000);
  return;
}

//------------------------- borra todas las posiciones de la memoria EEPROM ------------------------------
void deleeprom() {
  //vaciar todas las casillas de la eeprom
  p = 0;
  e = 0;
  address = 85; //la direccion mas alta es la columna 8 fila 5
  for (e = 0; e <= 7; e++) {
    for (p = 0; p <= 4; p++) {
      EEPROM.write(address, 0);
      Serial.println(address, DEC);
      address = address - 1; // empieza restando uno hasta 5 veces para saltar filas
    }
    address = address - 5;//hace una resta de 5 para saltar de columna
  }
  Serial.println("\t Memoria EEPROM borrada satisfactoriamente");
```

```
  delay(5000);
  return;
}

//---------------------------------- escribe 1 posicion de la memoria EEPROM -------------------------------
void ripeeprom1() {
  //escribir una posicion deseada de la eeprom (1)
  address = ((posx) * 10) + (posy);
  EEPROM.write(address, 1);
  Serial.print(address, DEC);//esto no es necesario
  Serial.println("\t Memoria EEPROM escrita satisfactoriamente");
  delay(2000);
  return;
}



//---------------------------------- Grabar 1 - 0 en la memoria EEPROM ---------------------------------------
void ripeeprom() {
  switch (accion) { //segun si coge o deja pieza grabara 1 o 0
    case 65://va a dejar pieza (tiene que grabar 1) es la letra D (dejar)en ascii
      Serial.println("Grabando EEPROM...");
      delay(1000);
      EEPROM.write(address, 1);
      Serial.println("Finalizado con exito");
      break;

    case 66://coger pieza (tiene que grabar 0) es la letra C (coger) en ascii
      Serial.println("Grabando EEPROM...");
      delay(1000);
      EEPROM.write(address, 0);
      Serial.println("Finalizado con exito");
```

```
    break;
  }
  return;

}
```

EMERGENCIA

```
//-------------------------------------- Parada de Emergencia ---------------------------------------------
void emergencia() {
  eme = 0;
  while ( digitalRead(emergency) == LOW && eme == 0) {
    digitalWrite(redL, HIGH);
    if (digitalRead(emergency) == LOW) {
      Serial.println("Parada de emergencia activa");
      while (digitalRead(emergency) == LOW && digitalRead(start) != HIGH  ) {
        disable_motorX();
        disable_motorY();
        digitalWrite(redL, HIGH);
      }
    }
  }
  while (digitalRead(start) == LOW) {
    digitalWrite(redL, HIGH);
    digitalWrite(puls, HIGH);
  }
  eme = 1;
  digitalWrite(redL, LOW);
  digitalWrite(puls, LOW);
}

void emergenciaX() {
  eme = 0;
```

```
  while ( digitalRead(emex) == LOW && eme == 0) {
    digitalWrite(redL, HIGH);
    digitalWrite(enable_x, HIGH);
    if (digitalRead(emex) == LOW) {
      while (digitalRead(emex) == LOW && digitalRead(start) != HIGH  ) {
        disable_motorX();
        digitalWrite(redL, HIGH);
      }
    }
  }
  while (digitalRead(start) == LOW) {
    digitalWrite(redL, HIGH);
    digitalWrite(puls, HIGH);
  }
  eme = 1;
  digitalWrite(redL, LOW);
  digitalWrite(puls, LOW);
  digitalWrite(enable_x, LOW);
}

void emergenciaY() {
  eme = 0;
  while ( digitalRead(emey) == LOW && eme == 0) {
    digitalWrite(redL, HIGH);
    digitalWrite(enable_y, HIGH);
    if (digitalRead(emey) == LOW) {
      while (digitalRead(emey) == LOW && digitalRead(start) != HIGH  ) {
        disable_motorY();
        digitalWrite(redL, HIGH);
      }
    }
```

```
  }
  while (digitalRead(start) == LOW) {
    digitalWrite(redL, HIGH);
    digitalWrite(puls, HIGH);
  }
  eme = 1;
  digitalWrite(redL, LOW);
  digitalWrite(puls, LOW);
  digitalWrite(enable_y, LOW);
}
```

ENTRAR

```
//--------------------------------------- Entrar a coger/dejar ---------------------------------------------
void entrar() {


  if (posx == 5) {
    posx = 4;
  }
  if (posx == 6) {
    posx = 3;
  }
  if (posx == 7) {
    posx = 2;
  }
  if (posx == 8) {
    posx = 1;
  }


  prex = factor_x * posx; //multiplicar el factor de proporción(1 unidad) por la posicion de x e y introducida en la orden
  prey = (factor_y * posy) - factor_y; //estos valores seran los numeros de pasos que tiene que contar el encoder
```

```
 if (accion == 66) {
  //(COGER)coge un palet de dentro para dejarlo fuera
  if (digitalRead(load) == LOW) { //comprobar si hay palet en el carro
   Serial.println("ERROR. Ya lleva un palet, imposible cargar otro"); // si hay palet da error
   descargar();
   while (digitalRead(load) == LOW) {//espera a no tener un palet en  cargarlo
   }
  }
 }
 //este while hace que el carro se mueva en diagonal encendiendo los dos motores a la vez
 encoreset();
 while (stepsx <= prex && stepsy <= prey) {
  Xmotor_L (); //enciende motor x
  Ymotor_Up (); //enciende motor y
  Serial.print("pasos X:");
  Serial.println(stepsx, DEC);
  Serial.println(prex, DEC);
  Serial.print("pasos Y:");
  Serial.println(stepsy, DEC);
  Serial.println(prey , DEC);
 }

 /* Como podria ser, alguno de los dos ejes puede llegar antes que el otro, por lo que saldria
  del while. Con estas dos condiciones, comprueba los dos contadores para que lleguen a su
posicion.
  Siempre por debajo de la posicion de dejar */
 if (stepsx <= prex) {
  Ymotor_OFF ();      //apaga a su vez el otro eje
  while (stepsx <= prex) {
   Xmotor_L ();
```

```
    Serial.print("pasos X:");
    Serial.println(stepsx, DEC);
    Serial.println(prex, DEC);
   }
  }
  if (stepsy <= prey ) {
   Xmotor_OFF ();      //apaga a su vez el otro eje
   while (stepsy <= prey ) {
    Ymotor_Up ();

    Serial.print("pasos Y:");
    Serial.println(stepsy, DEC);
    Serial.println(prey , DEC);
    if (stepsy == prey ) {
     Ymotor_OFF ();
     break;
    }
   }
  }
  //apaga todos los motores, siempre por debajo de la posicion
  XYmotor_OFF();
  Ymotor_OFF ();
  Xmotor_OFF ();
  return;
}
```

FUNCIONES

```
void clock_X() {
  iniMillisX = millis(); //guarda el tiempo que va contando para hacer la comparacion
```

```
  Serial.println(iniMillisX, DEC);
  Serial.println(prevMillisX, DEC);
  Serial.println(periodoX, DEC);

  if (iniMillisX - prevMillisX > (periodoX) ) { //resta los millis iniciales - los millis previos(en el inicio=0)
    // si la condicion se cumple inicia la cuenta con los previosmillis guardandolos en el principio de inicial
    prevMillisX = iniMillisX;

    if (clkEstadoX == LOW)
      clkEstadoX = HIGH;
    else
      clkEstadoX = LOW;

    // set the LED with the ledState of the variable:
    digitalWrite(clk_X, clkEstadoX);
  }
}

void clock_Y() {

  iniMillisY = micros(); //guarda el tiempo que va contando para hacer la comparacion
  Serial.println(iniMillisY, DEC);
  Serial.println(prevMillisY, DEC);
  Serial.println((iniMillisY-prevMillisY), DEC);
  if (iniMillisY - prevMillisY > (1600 )) { //resta los millis iniciales - los millis previos(en el inicio=0)
    // si la condicion se cumple inicia la cuenta con los previosmillis guardandolos en el principio de inicial

    prevMillisY = iniMillisY;
```

```
Serial.println("me cago en la puta y yauu");
   if (clkEstadoY == LOW)
     clkEstadoY = HIGH;
   else
     clkEstadoY = LOW;

   // set the LED with the ledState of the variable:
   digitalWrite(clk_Y, clkEstadoY);
  }
}

void clock_Z() {

  iniMillisZ = millis(); //guarda el tiempo que va contando para hacer la comparacion

  if (iniMillisZ - prevMillisZ > (periodoZ  )) { //resta los millis iniciales - los millis previos(en el inicio=0)
   // si la condicion se cumple inicia la cuenta con los previosmillis guardandolos en el principio de inicial
   prevMillisZ = iniMillisZ;

   if (clkEstadoZ == LOW)
     clkEstadoZ = HIGH;
   else
     clkEstadoZ = LOW;

   // set the LED with the ledState of the variable:
   digitalWrite(clk_Z, clkEstadoZ);

  }
}
```

```
//------------------------------------ Resetea contador de pasos -------------------------------------------
void encoreset() {
  stepsx = 0; //inicializa el contador de pasos
  stepsy = 0;
  return;
}

//------------------------------------ Resetea todas las variables iniciales ---------------------------------
void sysreset() {
  posx = 0; //inicializa las variables de la posicion
  posy = 0; //inicializa las variables de la posicion
  accion = 'p';
  address = 0;
  readpos = 0;
  stepsx = 0; //inicializa el contador de pasos
  stepsy = 0;
  Serial.flush();
  fallo = 0;
  return;
}

//------------------------------------ cuenta los pasos de encoder X -----------------------------------------
void cuentaX() {//contador de impulsos del encoder
  //evita que cuente por duplicado los pasos, o se salte impulsos del encoder
  Serial.println("cuenta X 81");/////////
  if (q == 0) {
    if (digitalRead(encox) == 1) {
      stepsx ++;
      q = 1;

      Serial.println("cuenta X 88");/////////
```

```
    }
  }
  if (q == 1) {
    if (digitalRead(encox) == 0) {
      q = 0;

      Serial.println("cuenta X 94");/////////
    }

  }

  return;
}

//---------------------------------- cuenta los pasos de encoder Y ----------------------------------------
void cuentaY() {//contador de impulsos del encoder
  //evita que cuente por duplicado los pasos, o se salte impulsos del encoder
  if (w == 0) {
    if (digitalRead(encoy) == 1) {
      stepsy ++;
      w = 1;
    }
  }
  if (w == 1) {
    if (digitalRead(encoy) == 0) {
      w = 0;
    }
  }
}
```

HOME

```
//------------------------------ chequeo de inicio del programa home xyz ------------------------------------
void inithome() {
  if (digitalRead(home_x) == LOW || digitalRead(home_y) == LOW ||
     (digitalRead(fcD) == LOW && digitalRead(fcI) == LOW ) ) {
    Serial.println(" No estoy en Home!");//mensaje de error
    Serial.println("Inicializando sistema...");




    if (digitalRead(fcD) == LOW && digitalRead(fcI) == LOW ) {  //si el carro esta fuera...
      Serial.println("Inicializando eje Z");
      delay(2000);
      while (digitalRead(fcD) == LOW && digitalRead(fcI) == LOW ) {
        Serial.println("home 15");
        // elcarro esta movido a derechas, retrocede a izquierdas hasta home
        if (digitalRead(fcI) == LOW && digitalRead(fcD) == LOW &&  digitalRead(home_z) == LOW
||
           digitalRead(fcI) == LOW && digitalRead(fcD) == HIGH &&  digitalRead(home_z) == LOW )
{


          while ( digitalRead(fcI) == LOW ) {
            Zmotor_L ();//enciende motores
            Serial.println("home 22");
          }
        }
        Zmotor_OFF ();//apagar motores
        Serial.println("home 26");
        //el carro esta salido a izquierdas, retrocede a derechas hasta home
        if (digitalRead(fcI) == LOW && digitalRead(fcD) == LOW &&  digitalRead(home_z) == HIGH
||
```

```
        digitalRead(fcI) == HIGH && digitalRead(fcD) == LOW &&  digitalRead(home_z) == HIGH
) {
       while ( digitalRead(fcD) == LOW) {
         Zmotor_R ();//enciende motores
       }
      }
      Zmotor_OFF ();//apagar motores
    }
   }
   while (digitalRead(home_y) == LOW) {

    while (digitalRead(home_y) == LOW) {
     Serial.println("home45");
     Ymotor_Down ();//enciende motores
    }
    //apagar motor
   }
   Ymotor_OFF ();

   while (digitalRead(home_x) == LOW) {
    // si el eje x esta desplazado
    Serial.println("Inicializando eje X");
    //recoger motor X
    while (digitalRead(home_x) == LOW) {
     Xmotor_R ();//enciende motores
     if (digitalRead(home_x) == HIGH) {
      break;
     }
    }
   }
   Xmotor_OFF (); //apagar motor
```

```
   Serial.flush(); //limpia el buffer
  }
}
```

MOTORES

```
void Xmotor_R () { //enciende motor X R
  digitalWrite(enx, LOW);
  digitalWrite(girox, HIGH);
  digitalWrite(clk_X, HIGH);
  clock_X();
}

void Xmotor_L () { //enciende motor X L
  digitalWrite(enx, LOW);
  digitalWrite(girox, LOW);
  digitalWrite(clk_X, HIGH);
  clock_X();
  cuentaX();
}

void Xmotor_OFF () { //apaga motor X
  digitalWrite(enx, LOW);
  digitalWrite(girox, LOW);
  digitalWrite(clk_X, LOW);
}

void Ymotor_Up () { //enciende motor Y R
  digitalWrite(eny, LOW);
  digitalWrite(giroy, HIGH);
  digitalWrite(clk_Y, HIGH);
```

```
  clock_Y();
  cuentaY();
}

void Ymotor_Down () { //enciende motor Y L
  digitalWrite(eny, LOW);
  digitalWrite(giroy, LOW);
  clock_Y();
  digitalWrite(clk_Y, HIGH);
}

void Ymotor_OFF () { //apaga motor Y
  digitalWrite(eny, LOW);
  digitalWrite(giroy, HIGH);
  digitalWrite(clk_Y, LOW);
}

void Zmotor_R () { //enciende motor Z R
  digitalWrite(enz, LOW);
  digitalWrite(giroz, HIGH);
  clock_Z();
}

void Zmotor_L () { //enciende motor Z L
  digitalWrite(enz, LOW);
  digitalWrite(giroz, LOW);
  clock_Z();
}

void Zmotor_OFF () { //apaga motor Z
  digitalWrite(enz, LOW);
```

```
  digitalWrite(giroz, HIGH);
  digitalWrite(clk_Y, LOW);
}

void XYmotor_OFF() { //apaga motor XY
  digitalWrite(enx, LOW);
  digitalWrite(eny, LOW);
  digitalWrite(clk_X, LOW);
  digitalWrite(clk_Y, LOW);
  encoreset();
}

void disable_motorX() {
  digitalWrite(enx, HIGH);
  digitalWrite(clk_X, LOW);
}

void disable_motorY() {
  digitalWrite(eny, LOW);
  digitalWrite(clk_Y, LOW);;
}
```

SALIR

```
//------------------------------------------- Salir de coger/dejar -----------------------------------------
void salir() {
  if (digitalRead(fcD) == LOW && digitalRead(fcI) == LOW) {
    Serial.println("ERROR, El carro no esta en posicion");
    while (digitalRead(fcD) == LOW && digitalRead(fcI) == LOW) {
      digitalWrite(redL, HIGH);
      //mensaje de error
```

```
  }
  delay(2000);
 }

 while (digitalRead(home_x) == LOW && digitalRead(home_y) == LOW) {
  Xmotor_R(); //enciende motor x
  Ymotor_Down (); //enciende motor y
 }

 while (digitalRead(home_x) == LOW ) {
  Xmotor_R();
  //apaga a su vez el otro eje
  Ymotor_OFF ();
 }

 while (digitalRead(home_y) == LOW) {
  Ymotor_Down ();
  //apaga a su vez el otro eje
  Xmotor_OFF ();
 }

 //apaga todos los motores, siempre por debajo de la posicion
 XYmotor_OFF();
 return;
}
```

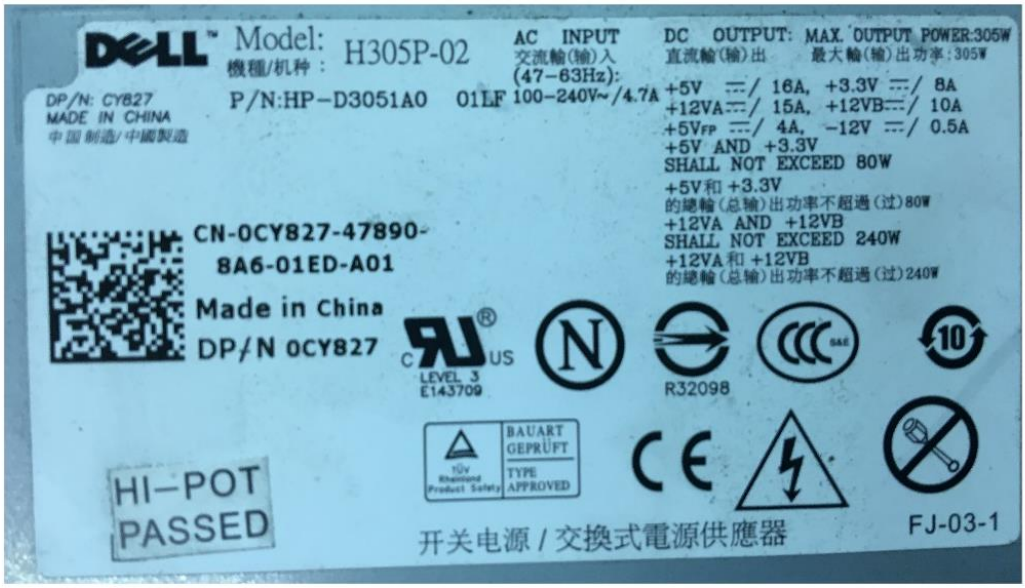
TAREAS

```
//------- TAREA 1 --------//DEJAR pieza A
void P01() {
  checkeeprom();
  if (fallo == 0) {
    cargar();
    entrar();
    descargar();
    ripeeprom();
    salir();
    Res = '0';
  }

  if (fallo == 1) {
    Res = '1';
  }
  return;
}

//------- TAREA 2 --------//COGER pieza B
void P02() {
  checkeeprom();
  if (fallo == 0) {
    entrar();
    cargar();
    ripeeprom();
    salir();
    descargar();
    Res = '0';
  }
```

```
  if (fallo == 1) {
    Res = '1';
  }
  return;
}

//------- TAREA 3 --------//LEER 1 posicion de la EEPROM C
void P03() {
  readeeprom1();
  return;
}

//------- TAREA 4 --------//BORRAR 1 posicion de la eeprom D
void P04() {
  deleeprom1();
  Res = '0';
  return;
}

//------- TAREA 5 --------//BORRA toda la memoria eeprom E
void P05() {
  deleeprom();
  Res = '0';
  return;
}

//------- TAREA 6 --------//ESCRIBE 1 en una posicion la memoria eeprom F
void P06() {
  ripeeprom1();
  Res = '0';
  return;
```

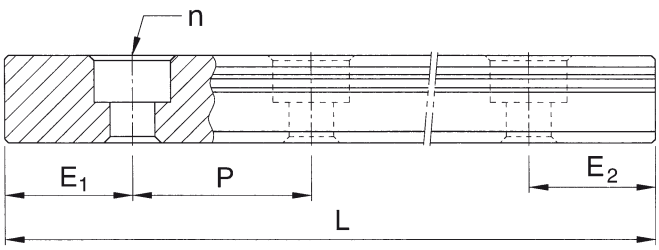
```
}



//------- TAREA 100 --------//Tarea para errores, si falla algo
void P100() {
  delay(3000);
  Serial.println("Fallo en la orden, algun dato es incorrecto.");
  Res = '1';
}
```

FUENTE DE ALIMENTACIÓN



1.3.11 Maximumlengtes van profielrailgeleidingen

Om bij niet-standaardlengtes uit te sluiten dat het einde van de profiel instabiel wordt, mag de waarde E de halve afstand tussen de montageboringen (P) niet overschrijden. Tegelijk mag de waarde $E_{1/2}$ niet kleiner dan $E_{1/2}$ min en niet groter dan $E_{1/2}$ max zijn, zodat de montageboring niet uitbreekt.



Formule 1.3 $$L = (n-1) \cdot P + E_1 + E_2$$

L: totale lengte van de rail [mm]
n: aantal montageboringen
P: afstand tussen twee montageboringen [mm]
$E_{1/2}$: afstand van het midden van de laatste montageboring tot het einde van de profielrail [mm]

Tabel 1.29:

Rail/grootte	MGNR 7	MGNR 9	MGNR 12	MGNR 15	MGWR 7	MGWR 9	MGWR 12	MGWR 15
Boringsafstand (P)	15	20	25	40	30	30	40	40
E1/2 min	5	5	5	6	6	6	8	8
E1/2 max	10	15	20	34	24	24	32	32
Max. lengte (1 stuk)	600	1000	1000	1000	600	1200	1200	1000
Max. lengte voor E1=E2=P/2*	585	980	975	960	570	960	960	960

Eenheid: [mm]

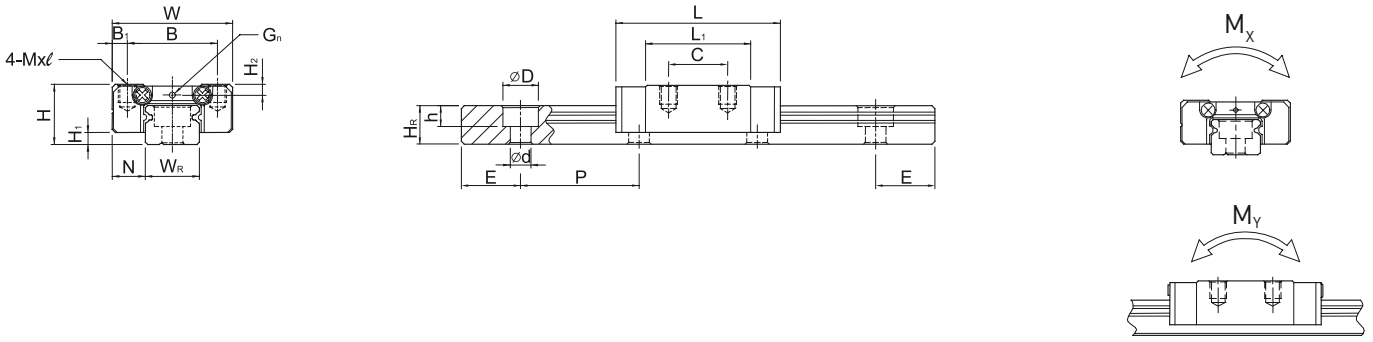
Opmerking:
1. De tolerantie voor E bedraagt voor standaardrails 0,5 tot –0,5 mm, bij kopstaartverbindingen 0 tot –0,3 mm
2. Type "M" is van roestbestendig staal
3. Zonder opgave van de $E_{1/2}$-maten wordt rekening houdende met $E_{1/2}$min het maximaal mogelijke aantal montageboringen bepaald

Profielrailgeleiding
MG serie

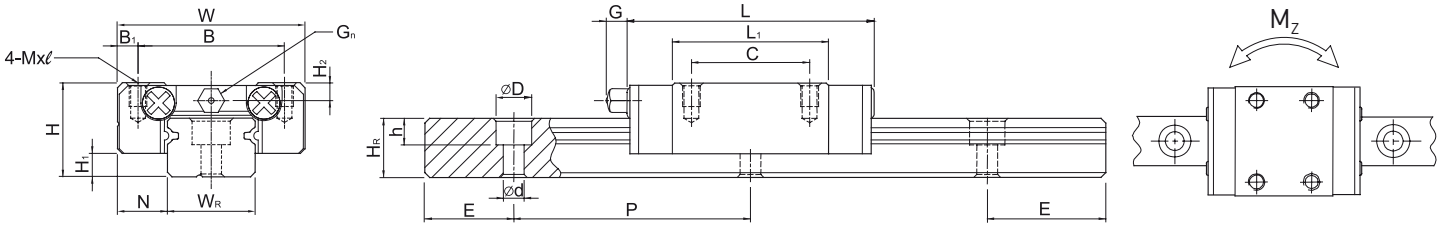
1.3.12 Afmetingen voor HIWIN MGN/MGW serie

1. MGN-C / MGN-H

- MGN7, MGN9, MGN12



- MGN15

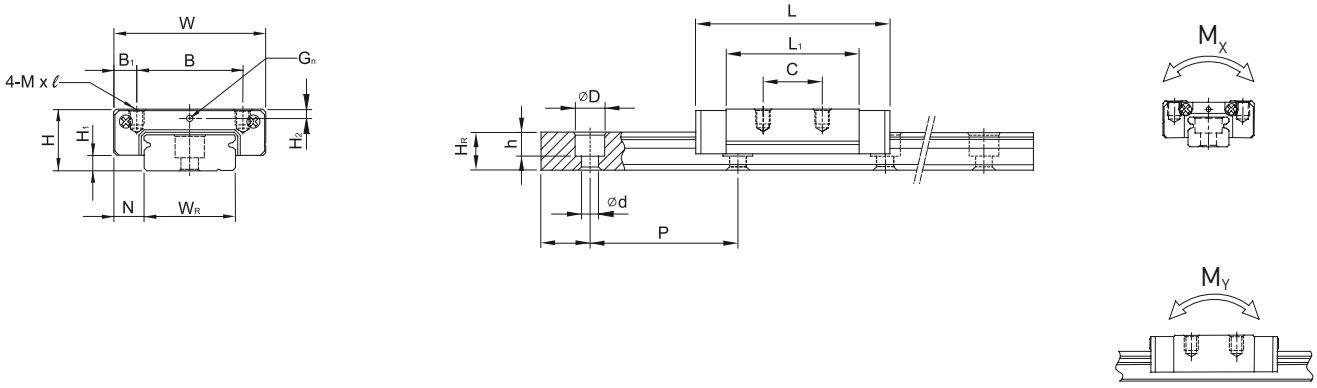


Model	Montagematen [mm]			Afmetingen van de loopwagen [mm]									Afmetingen van de profielrail [mm]							Schroef voor rail [mm]	Dynamisch draaggetal C_{dyn} [N]	Statisch draaggetal C0 [N]	Statisch moment			Gewicht	
	H	H1	N	W	B	B1	C	L1	L	G	Gn	M x l	H2	WR	HR	D	h	d	P	E				M_X [Nm]	M_Y [Nm]	M_Z [Nm]	Wagen [g]	Rail [kg/m]
MGN7C	8	1,5	5	17	12	2,5	8	13,5	22,5	-	Ø 0,8	M2 x 2,5	1,5	7	4,8	4,2	2,3	2,4	15	*	M2x6	1000	1270	4,8	2,9	2,9	10	0,22
MGN7H							13	21,8	30,8													1400	2000	7,8	4,9	4,9	15	
MGN9C	10	2	5,5	20	15	2,5	10	18,9	28,9	-	Ø 0,8	M3 x 3	1,8	9	6,5	6	3,5	3,5	20	*	M3x8	1900	2600	12	7,5	7,5	16	0,38
MGN9H							16	29,9	39,9													2600	4100	20	19	19	26	
MGN12C	13	3	7,5	27	20	3,5	15	21,7	34,7	-	Ø 0,8	M3 x 3,5	2,5	12	8	6	4,5	3,5	25	*	M3x8	2900	4000	26	14	14	34	0,65
MGN12H							20	32,4	45,4													3800	6000	39	37	37	54	
MGN15C	16	4	8,5	32	25	3,5	20	26,7	42,1	4,5	GN3S	M3 x 4	3	15	10	6	4,5	3,5	40	*	M3x10	4700	5700	46	22	22	59	1,06
MGN15H							25	43,4	58,8													6500	9300	75	59	59	92	

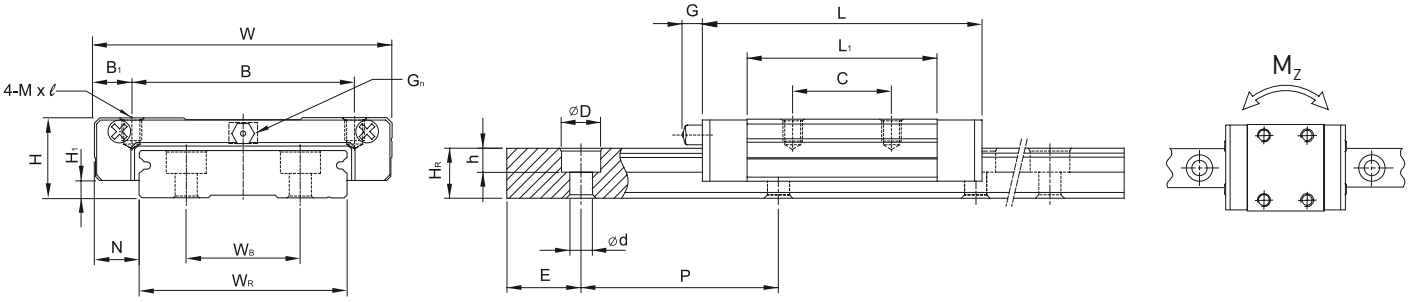
*zie p. 37, tab 1.29

2. MGW-C / MGW-H

- MGW7, MGW9, MGW12



- MGW15



Model	Montagema-ten [mm]			Afmetingen van de loopwagen [mm]										Afmetingen van de profielrail [mm]								Schroef voor rail [mm]	Dynamisch draag-getal C_{dyn} [N]	Statisch draag-getal C_0 [N]	Statisch moment			Gewicht	
	H	H_1	N	W	B	B_1	C	L_1	L	G	G_n	M x l	H_2	W_R	W_B	H_R	D	h	d	P	E				M_X [Nm]	M_Y [Nm]	M_Z [Nm]	Wagen [g]	Rail [kg/m]
MGW7C	9	1.9	5.5	25	19	3	10	21	31.2	-	ø 0,9	M3 x 3	1.85	14	-	5.2	6	3.2	3.5	30	*	M3x6	1400	2100	16	7.3	7.3	20	0.51
MGW7H							19	30.8	41														1800	3200	23.9	15.8	15.8	29	
MGW9C	12	2.9	6	30	21	4,5	12	27.5	39.9	-	ø 1,0	M3 x 3	2.4	18	-	7	6	4.5	3.5	30	*	M3x8	2800	4200	40.9	19.3	19.3	40	0.91
MGW9H					23	3,5	24	38.5	50.7														3500	6000	55.6	34.7	34.7	57	
MGW12C	14	3.4	8	40	28	6	15	31.3	46.1	-	ø 1,8	M3 x 3,6	2.8	24	-	8.5	8	4.5	4.5	40	*	M4x8	4000	5700	71.7	28.3	28.3	71	1.49
MGW12H							28	45.6	60.4														5200	8400	104.7	58.5	58.5	103	
MGW15C	16	3.4	9	60	45	7.5	20	38	54.8	5.2	GN3S	M4 x 4,2	3.2	42	23	9.5	8	4.5	4.5	40	*	M4x10	6900	9400	203.2	57.8	57.8	143	2.86
MGW15H							35	57	73.8														9100	14100	304.8	125	125	215	

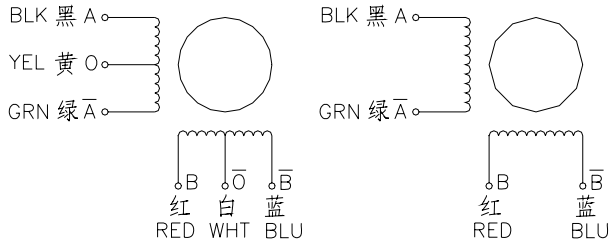
*zie p. 37, tab 1.29

MATRIZ DE POSICIONES

FILAS										
5	8,5	7,5	6,5	5,5		4,5	3,5	2,5	1,5	**5**
4	8,4	7,4	6,4	5,4		4,4	3,4	2,4	1,4	**4**
3	8,3	7,3	6,3	5,3		4,3	3,3	2,3	1,3	**3**
2	8,2	7,2	6,2	5,2		4,2	3,2	2,2	1,2	**2**
1	8,1	7,1	6,1	5,1		4,1	3,1	2,1	1,1	**1**
	8	**7**	**6**	**5**		**4**	**3**	**2**	**1**	
	COLUMNAS				EJE SIMETRIA	COLUMNAS				FILAS

2 Phase Hybrid Stepper Motor
17HS series-Size 42mm(1.8 degree)

Wiring Diagram:



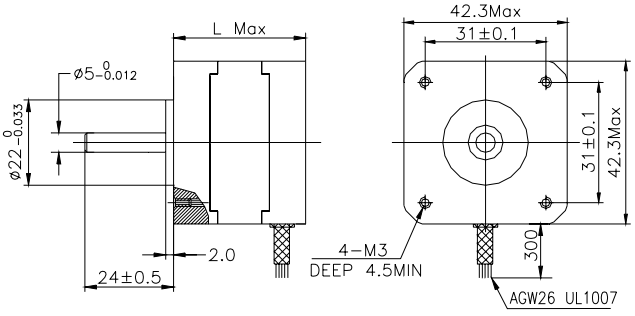
UNI-POLAR(6 LEADS) BI-POLAR(4LEADS)

Electrical Specifications:

Series Model	Step Angle (deg)	Motor Length (mm)	Rated Current (A)	Phase Resistance (ohm)	Phase Inductance (mH)	Holding Torque (N.cm Min)	Detent Torque (N.cm Max)	Rotor Inertia (g.cm²)	Lead Wire (No.)	Motor Weight (g)
17HS2408	1.8	28	0.6	8	10	12	1.6	34	4	150
17HS3401	1.8	34	1.3	2.4	2.8	28	1.6	34	4	220
17HS3410	1.8	34	1.7	1.2	1.8	28	1.6	34	4	220
17HS3430	1.8	34	0.4	30	35	28	1.6	34	4	220
17HS3630	1.8	34	0.4	30	18	21	1.6	34	6	220
17HS3616	1.8	34	0.16	75	40	14	1.6	34	6	220
17HS4401	1.8	40	1.7	1.5	2.8	40	2.2	54	4	280
17HS4402	1.8	40	1.3	2.5	5.0	40	2.2	54	4	280
17HS4602	1.8	40	1.2	3.2	2.8	28	2.2	54	6	280
17HS4630	1.8	40	0.4	30	28	28	2.2	54	6	280
17HS8401	1.8	48	1.7	1.8	3.2	52	2.6	68	4	350
17HS8402	1.8	48	1.3	3.2	5.5	52	2.6	68	4	350
17HS8403	1.8	48	2.3	1.2	1.6	46	2.6	68	4	350
17HS8630	1.8	48	0.4	30	38	34	2.6	68	6	350

***Note:** We can manufacture products according to customer's requirements.

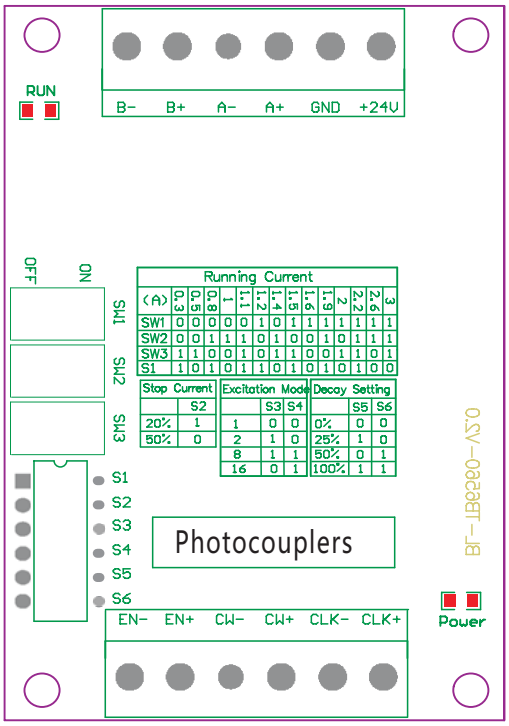
Dimensions: unit=mm



Motor Length:

Model	Length
17HS2XXX	28 mm
17HS3XXX	34 mm
16HS4XXX	40 mm
16HS8XXX	48 mm

Tb6560 stepping motor driver V20



Warning:

1、Check the connection twice! The Tb6560 chipset can be damaged if the motor or the power supply are not connected properly.

2、Dont apply a motor that its rated current is more than 3A to this driver.

3、Do not set the current more than the motor rated current!

Wiring Terminal symbol	Description
+24V, GND	Power positive and negative
A+, A-	Motor phase A
B+, B-	Motor phase B
CLK+, CLK-	Pulse positive and negative
CW+, CW-	Direction positive and negative
EN+, EN-	Enable positive and negative

Note:

1、6 input terminals, can be connected as common anode or cathode。

2、The normal input voltage is **5V**, if it is more than 5V, than a series resistor is needed. this resistance is 1K case 12V and 2.4K case 24V.

3、when pulse is applied to **CLK**, the stepping motor will rotate, and stop when there is none, and the motor driver will change its current to the half current mode as setting to hold the motor still.

4、Motor rotate clockwise when CW is low level and counterclockwise when **CW** is high level.

5、Motor is enable when **EN** is low level and disable when EN is high leve.

Running Current														
(A)	0.3	0.5	0.8	1	1.1	1.2	1.4	1.5	1.6	1.9	2	2.2	2.6	3
SW1	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	ON	ON	ON	ON	ON	ON
SW2	OFF	OFF	ON	ON	ON	OFF	ON	OFF	OFF	ON	OFF	ON	ON	ON
SW3	ON	ON	OFF	OFF	ON	OFF	ON	ON	OFF	OFF	ON	ON	OFF	ON
S1	ON	OFF	ON	OFF	ON	ON	OFF	ON	OFF	ON	OFF	ON	OFF	OFF

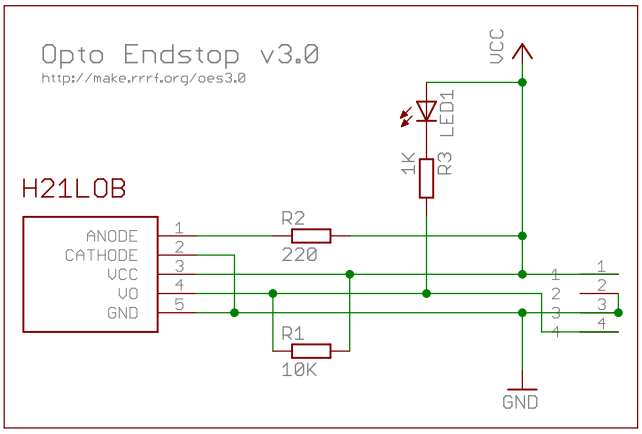
Stop Current	
	S2
20%	ON
50%	OFF

Excitation Mode		
Step	S3	S4
whole	OFF	OFF
half	ON	OFF
1/8	ON	ON
1/16	OFF	ON

Decay Setting		
	S5	S6
0%	OFF	OFF
25%	ON	OFF
50%	OFF	ON
100%	ON	ON

Asignacion de Entradas y Salidas

	Pin	Tipo	Declaración	Función
RX0	0			
TX0	1			
PWM/interrup0	2	E	encox	n pin encoder del eje x
PWM/interrup1	3	E	encoy	n pin encoder del eje y
PWM	4	E	232	Selección de comunicación 232
PWM	5	E	485	Selección de comunicación 485
PWM	6	E	enable485	Enable 485
PWM	7	E	pulscom	Señal para comunicación de orden de comunicación
PWM	8			
PWM	9			
PWM	10			
PWM	11			
PWM	12			
PWM	13			
TX3	14	COM		serial 3 Tx
RX3	15	COM		serial 3 Rx
TX2	16	COM		serial 2 Tx
RX2	17	COM		serial 2 Rx
TX1/interrup 5	18			
RX1/interrup 4	19	E	emergency	Seta de emergencia (NC)
SDA/interrup 3	20	E	emex	f.c de emergencia del eje x (NC)
SCL/interrup 2	21	E	emey	f.c de emergencia del eje y(NC)
	22	S	enx	enable del motor x
	23	S	girox	sentido del motor x
	24	S	eny	enable del motor y
	25	S	giroy	sentido del motor y
	26	S	enz	enable del motor z
	27	S	giroz	sentido del motor z
	28	S	clk_X	Señal de CLK motor X
	29	S	clk_Y	Señal de CLK motor Y
	30	S	clk_Z	Señal de CLK motor Z
	31	S	puls	luz del pulsador verde start PWM
	32			
	33			
	34	S	redL	luz de emergencia roja K1M
	35	S	greenL	salida de la luz verde, ARDUINO ACTIVO K2M
	36	S	enable_x	corta la señal de reloj del driver X (K3M)
	37	S	enable_y	corta la señal de reloj del driver Y (K4M)
	38			
	39			
	40	E	fcD	posicion eje z recogido (carro) derechas (NO)
	41	E	fcI	posicion eje z extendido (carro) izquierdas (NO)
	42	E	homex	home eje x fc (NO)
	43	E	homey	home eje y fc (NO)
	44	E	homez	home eje z fc (NO)
	45			
	46	E	palet	sensor para detectar palet en la bahia de carga (NO)
	47	E	load	sensor para detectar palet cargado en el carro Z (NO)
	48			
	49			
	50	E	start	rearme despues de emergencia boton verde
	51			
	52			
	53			

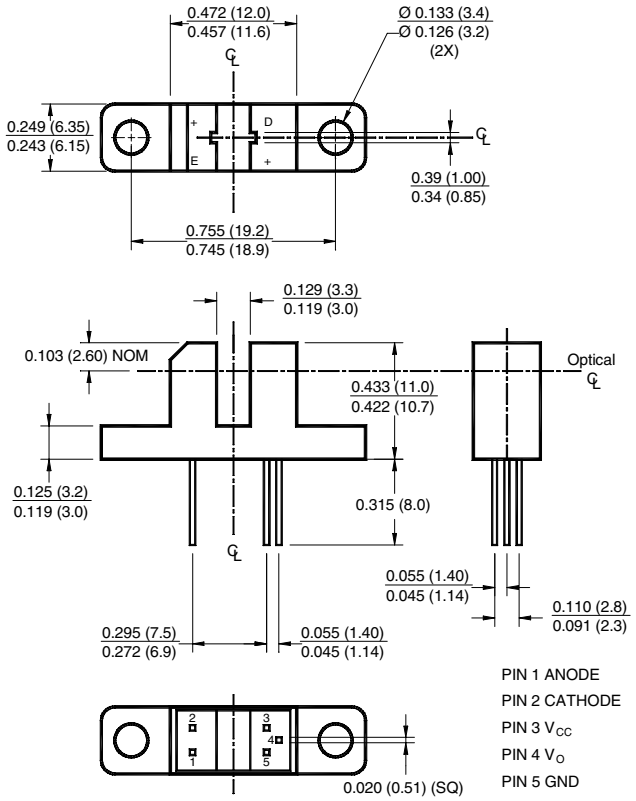

Opto Endstop v3.0
http://make.rrrf.org/oes3.0
VCC
LED1
1K
R3
H21L0B
ANODE
CATHODE
VCC
VO
GND
1
2
3
4
5
R2
220
R1
10K
GND

FAIRCHILD
SEMICONDUCTOR®

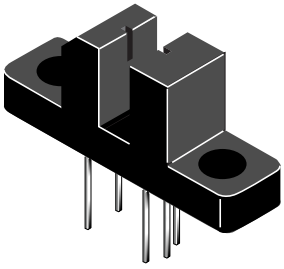
OPTOLOGIC® OPTICAL INTERRUPTER SWITCH

H21LTB H21LTI H21LOB H21LOI

PACKAGE DIMENSIONS



NOTES:
1. Dimensions for all drawings are in inches (mm).
2. Tolerance of ± .010 (.25) on all non-nominal dimensions unless otherwise specified.



PART NUMBER DEFINITIONS	
H21LTB	Totem-pole, buffer output
H21LTI	Totem-pole, inverter output
H21LOB	Open-collector, buffer output
H21LOI	Open-collector, inverter output

INPUT/OUTPUT TABLE		
Part Number	**LED**	**Output**
H21LTB	On	High
H21LTB	Off	Low
H21LTI	On	Low
H21LTI	Off	High
H21LOB	On	High
H21LOB	Off	Low
H21LOI	On	Low
H21LOI	Off	High

DESCRIPTION

The H21L series are slotted optical switches designed for multipurpose non contact sensing. They consist of a GaAs LED and a silicon OPTOLOGIC®. sensor packaged in an injection molded housing and facing each other across a .124" (3.15 mm) gap. The output is either inverting or non-inverting, with a choice of totem-pole or open-collector configuration for TTL/CMOS compatibility

FEATURES

- Low cost
- 0.035" apertures
- Black plastic opaque housing
- Mounting tabs on housing
- Choice of inverter or buffer output functions
- Choice of open-collector or totem-pole output configuration
- TTL/CMOS compatible output functions

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise specified)

Parameter	Symbol	Rating	Units
Operating Temperature	T_{OPR}	-40 to +85	°C
Storage Temperature	T_{STG}	-40 to +85	°C
Soldering Temperature (Iron)(3,4,5,6)	T_{SOL-I}	240 for 5 sec	°C
Soldering Temperature (Flow)(3,4,6)	T_{SOL-F}	260 for 10 sec	°C
INPUT (EMITTER)			
Continuous Forward Current	I_F	50	mA
Reverse Voltage	V_R	6	V
Power Dissipation(1)	P_D	100	mW
OUTPUT (SENSOR)			
Output Current	I_O	50	mA
Supply Voltage	V_{CC}	4.0 to 16	V
Output Voltage	V_O	30	V
Power Dissipation(2)	P_D	150	mW

NOTES (Applies to Max Ratings and Characteristics Tables.)

1. Derate power dissipation linearly 1.67 mW/°C above 25°C.
2. Derate power dissipation linearly 2.50 mW/°C above 25°C.
3. RMA flux is recommended.
4. Methanol or isopropyl alcohols are recommended as cleaning agents.
5. Soldering iron 1/16" (1.6mm) from housing.
6. As long as leads are not under any stress or spring tension.

ELECTRICAL / OPTICAL CHARACTERISTICS (T_A =25°C)

PARAMETER	TEST CONDITIONS	SYMBOL	MIN.	TYP.	MAX.	UNITS
INPUT (EMITTER)						
Forward Voltage	I_F = 20 mA	V_F	—		1.5	V
Reverse Leakage Current	V_R = 5 V	I_R	—		10	µA
OUTPUT (SENSOR)						
Supply Current	V_{CC} = 5 V	I_{CC}	—		5	mA
COUPLED						
Low Level Output Voltage H21LTB, H21LOB	I_F = 0 mA, V_{CC} = 5 V, R_L = 100 Ω	V_{OL}	—		0.4	V
Low Level Output Voltage H21LTI, H21LOI	I_F = 15 mA, V_{CC} = 5 V, R_L = 360 Ω	V_{OL}	—		0.4	V
High Level Output Voltage H21LTB	I_F = 15 mA, V_{CC} = 5 V, I_{OH} = -800 µA	V_{OH}	2.4		—	V
High Level Output Voltage H21LTI	I_F = 0 mA, V_{CC} = 5 V, I_{OH} = -800 µA	V_{OH}	2.4		—	V
High Level Output Current H21LOB	I_F = 0 mA, V_{CC} = 5 V, I_{OH} = -800 µA	I_{OH}			100	µA
High Level Output Current H21LOI	I_F = 0 mA, V_{CC} = 5 V, V_{OH} = 30 V	I_{OH}	—		100	µA
Turn on Threshold Current	V_{CC} = 5 V, R_L = 360 Ω	$I_F(+)$	—		15	mA
Turn off Threshold Current	V_{CC} = 5 V, R_L = 360 Ω	$I_F(-)$	0.50		—	mA
Hysteresis Ratio		$I_F(+) / I_F(-)$		1.2		
Propagation Delay	V_{CC} = 5 V, R_L = 360 Ω (See Fig, 9)	t_{PLH}, t_{PHL}		5		µs
Output Rise and Fall Time	V_{CC} = 5 V, R_L = 360 Ω (See Fig, 9)	t_r, t_f		70		ns