Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Diseño de un sistema automatizado de aparcamiento de vehículos

Autor: Alejandro Villalba Hernando

Tutor: José Manuel Villadangos Carrizo

2022

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

“Diseño de un sistema automatizado de aparcamiento de vehículos”

Autor: Alejandro Villalba Hernando

Tutor: José Manuel Villadangos Carrizo

TRIBUNAL:

Presidente: César Mataix Gómez

Vocal 1º: Álvaro de la Llana Calvo

Vocal 2º: José Manuel Villadangos Carrizo

FECHA: SEPTIEMBRE 2022

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGO FUENTE

RESUMEN

Este trabajo de fin de grado tiene como objetivo el diseño de un sistema automatizado de aparcamiento para vehículos, centrándose especialmente en el diseño de su sistema de control.

Dicho automatismo se implementará con un microcontrolador ARM Cortex-M3 LPC1768, utilizando la tarjeta Mini-DK2. A ella se conectarán distintos sensores, periféricos y subsistemas para poder realizar automáticamente la tarea de aparcar un vehículo.

Se estudia un caso concreto: un aparcamiento de catorce alturas, siendo diez de ellas para el almacenamiento de los vehículos, de ocho coches por planta, y cuatro plantas para alojar los diferentes componentes del sistema. Será diseñado para un bloque de viviendas ficticio y además de este ejemplo, se estudiará cómo modificar el automatismo para poder aumentar su capacidad, modificar ciertos parámetros y poder aplicarlo a otros ámbitos, como un parking privado de un solo dueño o un parking público.

Palabras clave: Aparcamiento, ARM, Cortex M3, LPC1768, Arduino, Microcontrolador, Motor AC, PLC.

ABSTRACT

This final project aims to design an automated parking system for vehicles, focusing especially on the design of its control system.

This automation will be implemented with an ARM Cortex-M3 LPC1768 microcontroller, using the Mini-DK2 card. Different sensors, peripherals and subsystems shall be connected to it to enable the automatic parking of a vehicle

We study a concrete case: a parking lot of fourteen heights, ten of them for the storage of vehicles, eight cars per floor, and four floors to accommodate the different components of the system. It will be designed for a fictitious housing block and in addition to this example, it will study how to modify the automation to increase its capacity, modify certain parameters and apply it to other areas, such as a private parking of a single owner or a public parking

Keywords: Parking, ARM, Cortex M3, LPC1768, Arduino, Microcontroller, Motor AC, PLC

RESUMEN EXTENDIDO

Este trabajo de fin de grado desarrolla el diseño de un aparcamiento automatizado para vehículos, centrándose especialmente en su sistema de control. Para ello, se estructura en seis etapas: arquitectura del parking, funcionamiento del automatismo, periféricos de la tarjeta Mini-DK2, hardware y software del automatismo y conclusiones y trabajos futuros. Conocer la forma o arquitectura del aparcamiento y el funcionamiento es necesario para poder diseñar correctamente el automatismo, pues se deben tener en cuenta la arquitectura y todos los posibles modos de funcionamiento y sus procedimientos para diseñar el software y hardware.

ARQUITECTURA

Se estudia un caso concreto: un aparcamiento de catorce alturas, siendo diez de ellas para el almacenamiento de los vehículos, y ocho coches por planta, diseñado para un bloque de viviendas ficticio, por tanto será capaz de almacenar 80 vehículos.

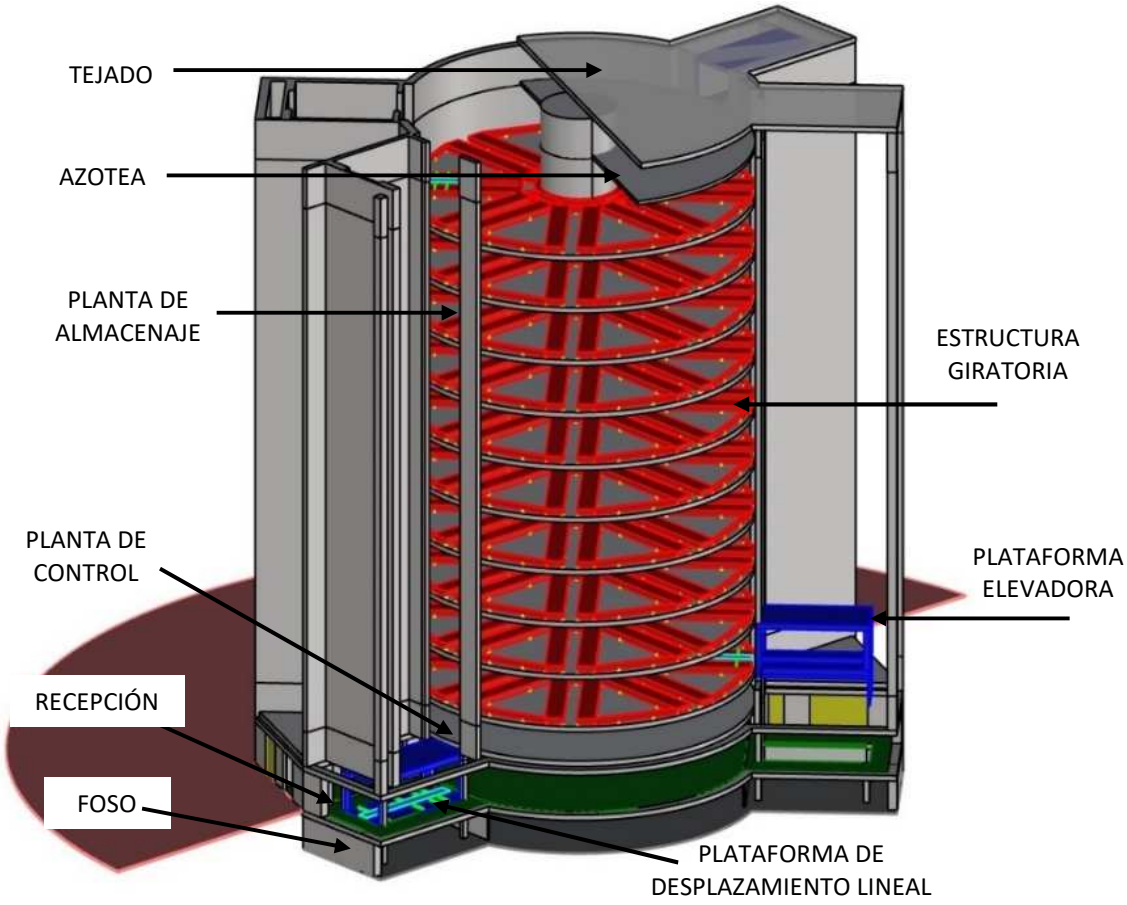


Figura 1. Diseño 3D del aparcamiento automatizado propuesto

El edificio se divide en tres bloques, dos de ellos iguales los cuales albergan las cuatro plataformas elevadoras, y un núcleo circular central que será el encargado de recoger el almacenaje de los vehículos.

La superficie de este aparcamiento es de 530 m^2 y su altura es de 35 m, por lo que la distancia será crucial en el envío de datos.

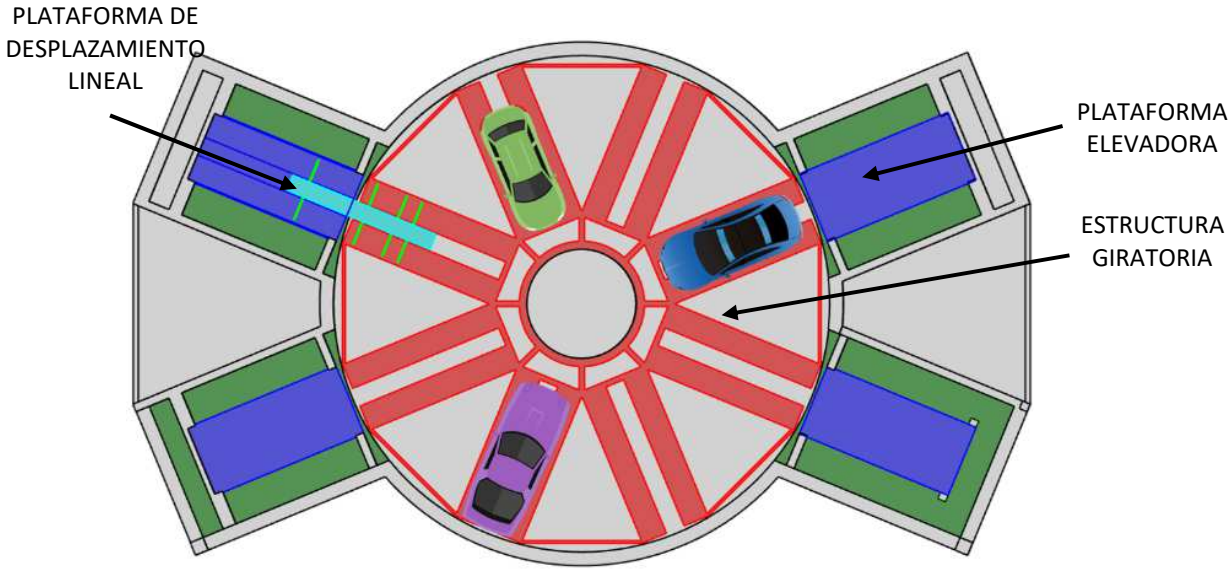


Figura 2. Vista en planta del aparcamiento.

FUNCIONAMIENTO

Una vez se haya estacionado el vehículo en la plataforma elevadora (con el freno de mano) se pasará una tarjeta RFID por un lector, al ser detectada el automatismo buscará si la plaza asociada a dicha tarjeta está libre, activará las medidas de seguridad y el sistema móvil se pondrá en marcha. Este sistema consta de una plataforma elevadora que situará en altura al vehículo, una planta rotativa que centrará la plaza libre frente al elevador y un robot que introducirá el vehículo al aparcamiento. Una vez el coche ha sido aparcado el elevador descenderá y las medidas de seguridad se desactivarán.

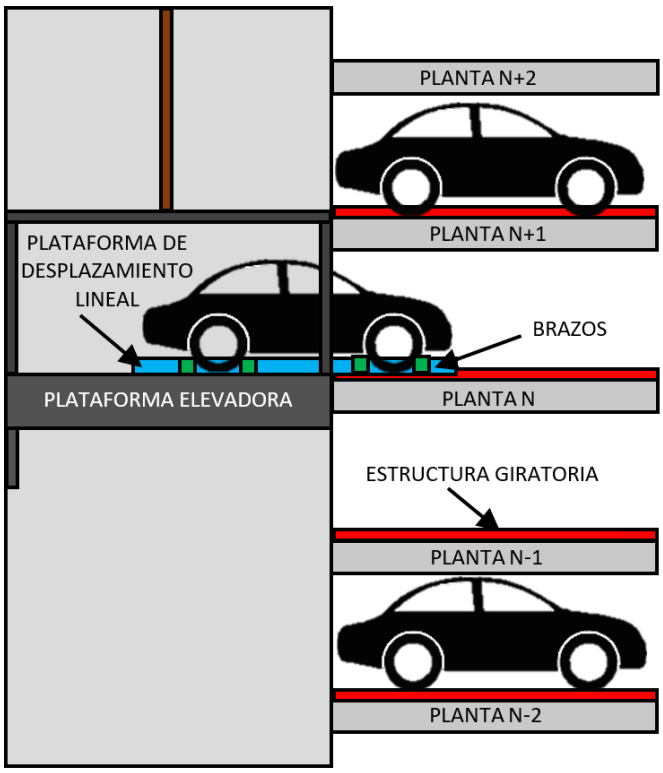


Figura 3. Representación del sistema de aparcamiento o retirada de vehículos.

En el caso de querer retirar el vehículo solo será necesario pasar la tarjeta RFID por un lector de un elevador que no esté en uso y al igual que anteriormente el vehículo descenderá.

Cuando la plataforma elevadora esté disponible se iluminará de color verde y aparecerá por un display situado encima del lector un mensaje de aviso. Al igual, si la plataforma está en ejecución la luz será amarilla parpadeante y se emitirá otro mensaje, en el caso de haberse producido una avería la luz amarilla quedará fija y el mensaje se modificará para notificarlo. Por último, si el aparcamiento está lleno la luz será roja.

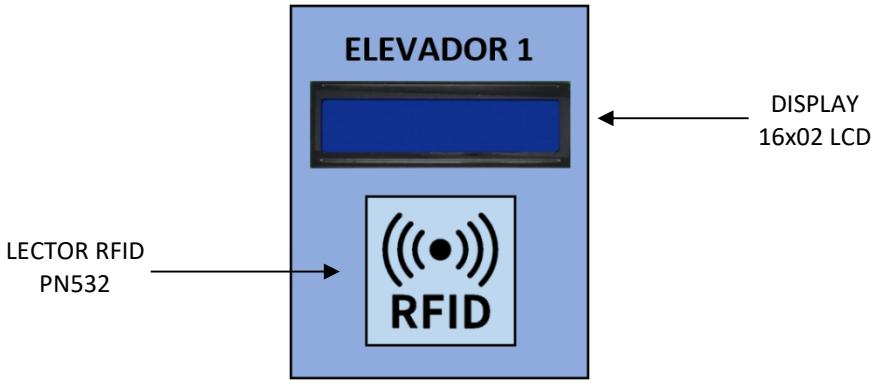


Figura 4. Representación del lector de tarjetas RFID más el display informativo situado en la sala del lector

Por otro lado, se pueden dar los casos de pasar la tarjeta por el lector no habiendo coches ni en el elevador ni en el aparcamiento, o el caso contrario, que los haya ambos. En estos casos se emitirá un mensaje de aviso en el display situada encima del lector.

HARDWARE

El sistema requiere de multitud de subsistemas basados en autómatas programables, sensores y módulos externos. Estos componentes se deben conectar al microprocesador central teniendo en cuenta la distancia a la que se sitúan y el ruido que se puede producir.

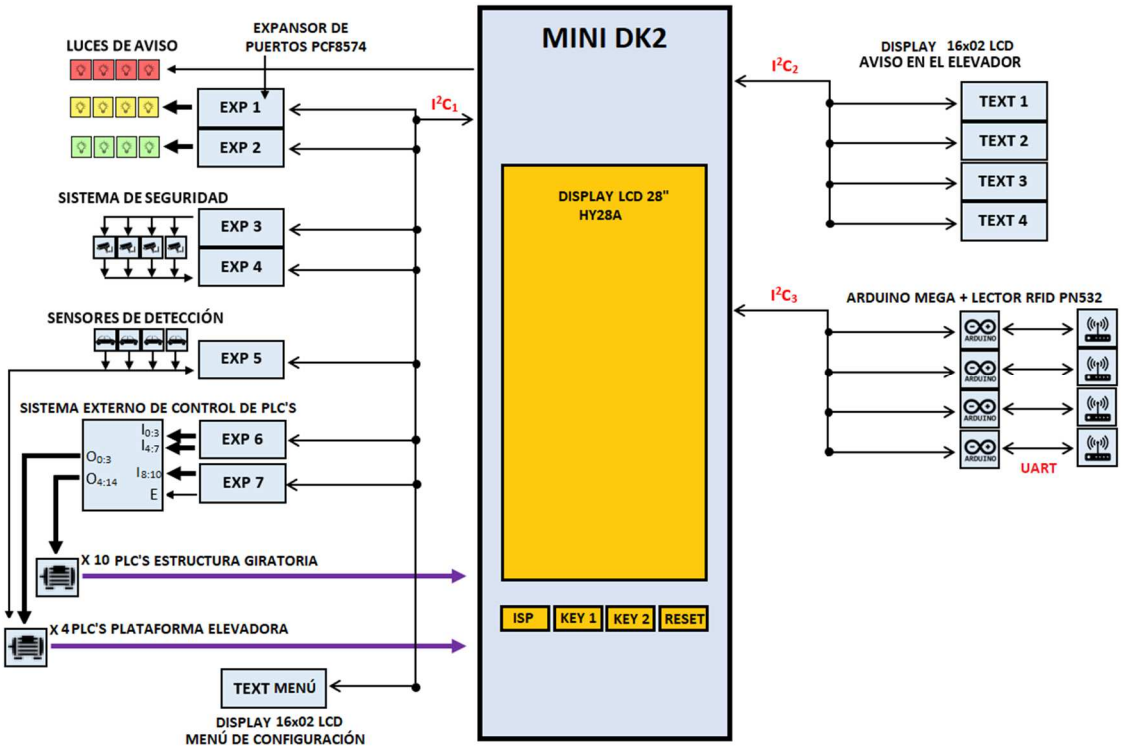


Figura 5. Resumen del hardware del automatismo.

SOFTWARE

Dicho automatismo se implementará con un microcontrolador ARM Cortex-M3 LPC1768, el cual está acoplado en la tarjeta Mini-DK2. La programación se hará mediante el programa KEIL v4 y cargada a la tarjeta directamente mediante un debugger.

El programa se diseña teniendo en cuenta los componentes necesarios descritos en Hardware y los requisitos descritos en el funcionamiento e intentando que sea lo más sencillo, compacto y correcto posible.

ÁMBITOS DE APLICACIÓN

El automatismo está preparado para tener una capacidad máxima de 192 plazas (repartidas en 16 plantas y 12 vehículos por planta) con 8 plataformas elevadoras, además, se pueden enterrar plantas para disminuir la altura a nivel de suelo.

Actualmente cada tarjeta está asociada a una plaza, cosa poco práctica que se puede mejorar enlazando la tarjeta leída con la plaza más accesible en ese instante.

La arquitectura de este aparcamiento deja multitud de huecos vacíos (como se observa sombreado de verde en la imagen inferior) que se podrían rellenar con vehículos de menor tamaño añadiendo otro sistema y conectándolo con el descrito.

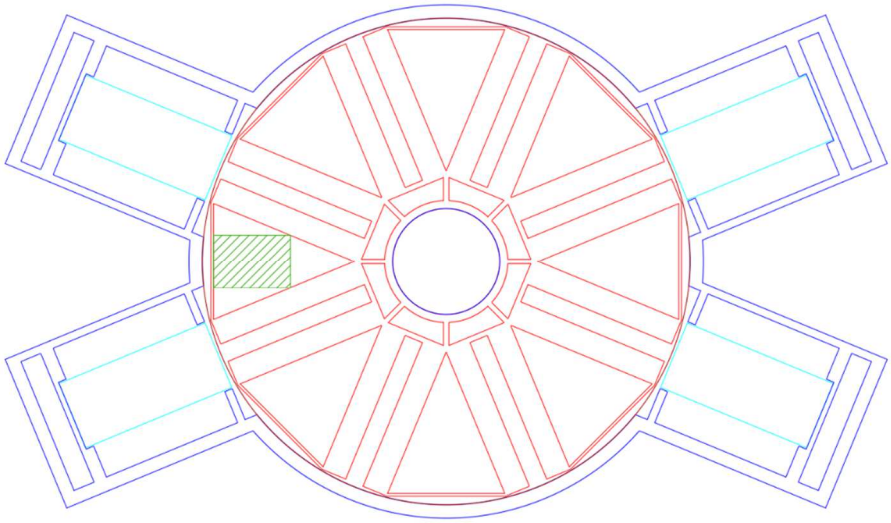


Figura 6. Representación de los espacios muertos del aparcamiento.

CAPÍTULO I

INTRODUCCIÓN

1. INTRODUCCIÓN

En este trabajo se diseña un sistema automatizado basado en un aparcamiento de vehículos. Para diseñar dicho automatismo, se necesita conocer la arquitectura que va a tener, las funciones que va a realizar y los sistemas externos al microcontrolador principal que se van a utilizar. Con estas premisas, se diseña el software y hardware del sistema.

1.1. MOTIVACIÓN Y FASES DE ESTUDIO

En estos días es muy difícil encontrar aparcamiento por el centro de las ciudades, o se tiene que dejar el coche en la tercera planta subterránea del centro comercial, pues los aparcamientos comúnmente son extensas explanadas o liosos entramados de columnas donde nunca se encuentra la salida, por ello, el autor se ve motivado a elaborar este trabajo de fin de grado llamado "Diseño de un sistema automatizado de aparcamiento de vehículos", donde sea sencillo despreocuparse del estacionamiento, evitando roces y golpes innecesarios y concentrando la mayor cantidad de vehículos en el menor espacio de la manera más cómoda posible.

Diseñar un sistema automatizado de aparcamiento de vehículos es un proyecto complicado, por lo que se ha estructurado en varias fases de estudio.

La **primera fase** responde a la pregunta de qué forma tendrá el aparcamiento, por lo que se hablará de la arquitectura del edificio.

La **segunda fase** responde a la pregunta de cómo funcionará el aparcamiento, pues ya se sabe qué se quiere hacer y qué forma tendrá. Por tanto, en esta segunda fase se hablará del funcionamiento del automatismo.

La **tercera fase** introduce los sistemas externos al sistema principal, la tarjeta MiniDK2, que se usarán para llevar a cabo el proyecto.

La **cuarta fase** entra directamente en el interior del estudio, pues se ve cómo se conectan físicamente los sistemas externos y componentes al sistema principal, es decir, se estudia el hardware.

La **quinta fase** está muy relacionada con la fase anterior, pues una vez se conocen las conexiones de los subsistemas se programa el microprocesador para controlarlos, es decir, se elabora el software.

En la **sexta fase** se sacan las conclusiones y se tratan los trabajos futuros, pues es interesante saber qué cambios hay que realizar interna y externamente para enfocar el automatismo en diferentes situaciones y ampliaciones y no contemplar una única opción.

1.2. PRESENTACIÓN

Las dos figuras siguientes (Figura 7 y 8) muestran una vista 3D y una vista en planta del aparcamiento, en ellas se observa las distintas plantas y sistemas móviles que lo conforman, el cual tiene un aforo de 80 vehículos cuyas medidas máximas son de 5,5 metros de largo por 2,4 metros de ancho y 2,2 metros de alto, dimensiones escogidas para albergar el máximo número de modelos de turismos y furgonetas de viaje. Las 80 plazas se recogen en 10 plantas de 8 plazas o divisiones cada una.

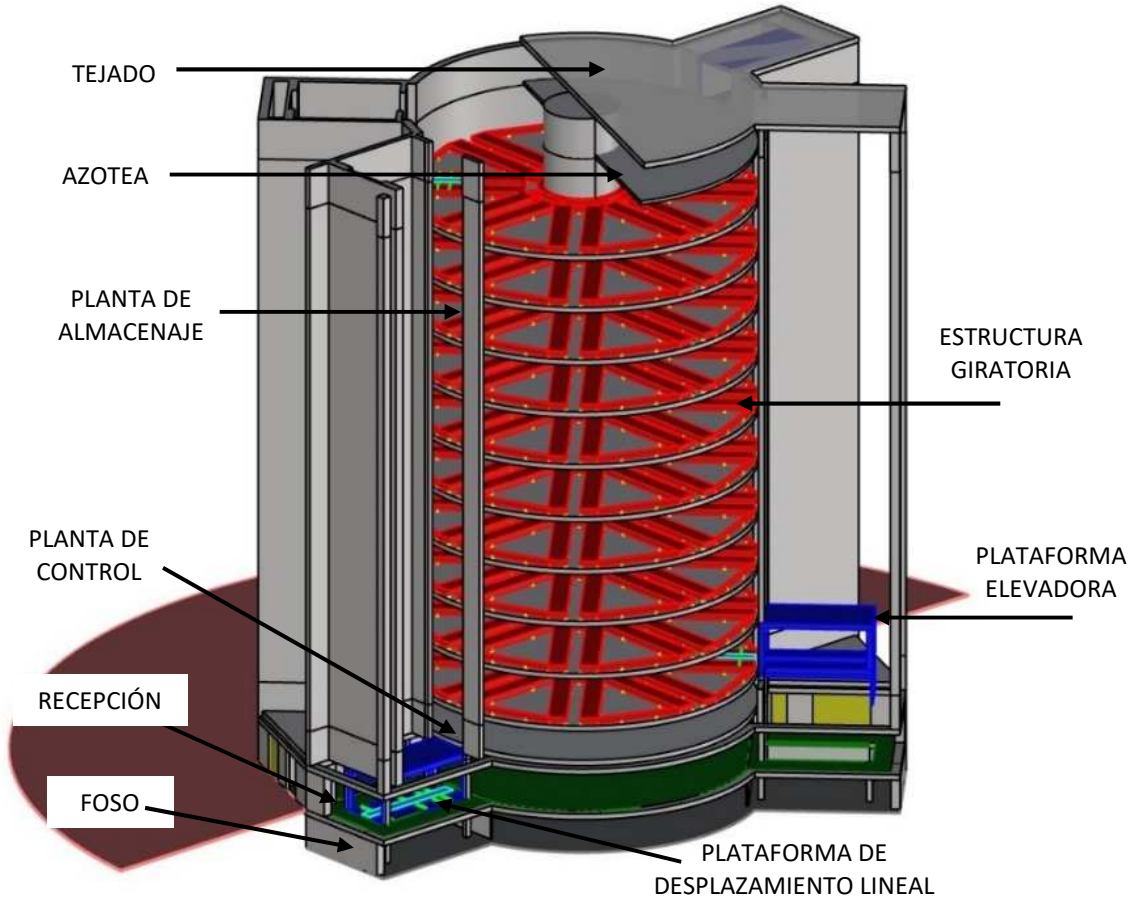


Figura 7. Diseño 3D del aparcamiento automatizado propuesto.

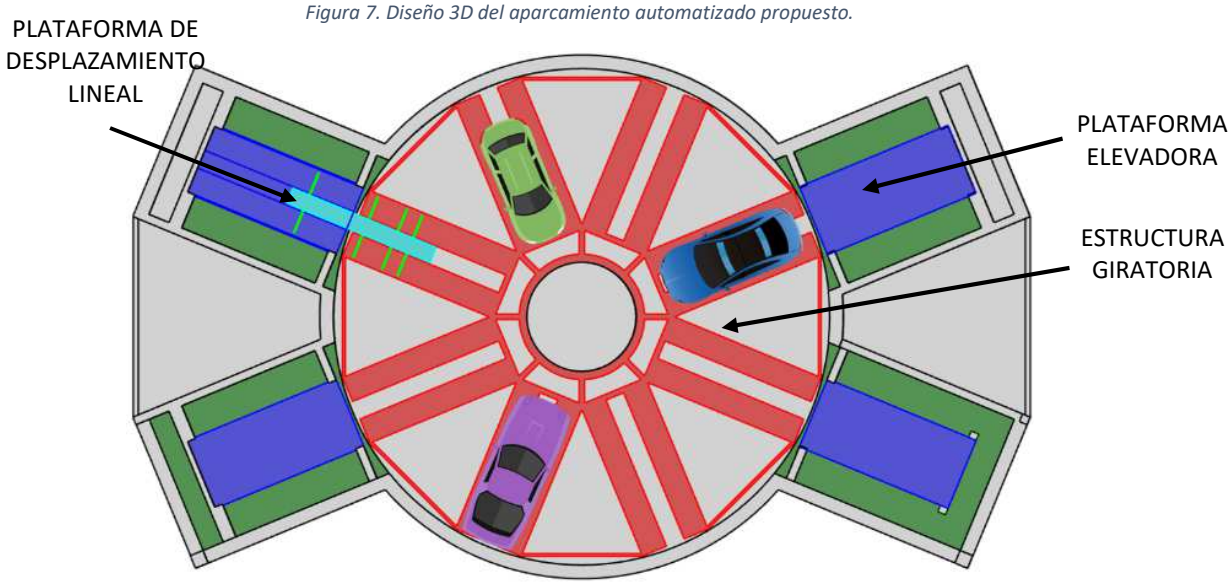


Figura 8. Vista en planta del aparcamiento.

Para poder aparcar o retirar el vehículo es necesario pasar una tarjeta RFID previamente configurada por un lector, en este caso cada tarjeta tiene asociada una plaza, es decir, si se lee la tarjeta 23 se accede a la plaza número 23. Una vez leído el número de tarjeta, el automatismo determina mediante una memoria de plazas libres y un sensor de presencia en la plataforma elevadora si se debe aparcar, retirar o si es imposible las dos acciones anteriores por la inexistencia de un vehículo en el interior del automatismo y en la plataforma, o, por el contrario, la existencia de dos vehículos.

Una vez se obtenga la acción a realizar el sistema muestra por un display situado encima del lector RFID un mensaje informativo y la plataforma elevadora se ilumina de verde, amarillo o rojo dependiendo de su estado, es decir, se ilumina de verde si está disponible y listo para usar, de amarillo parpadeante si se está utilizando, amarillo fijo si está averiado o rojo si se han ocupado las 80 plazas.

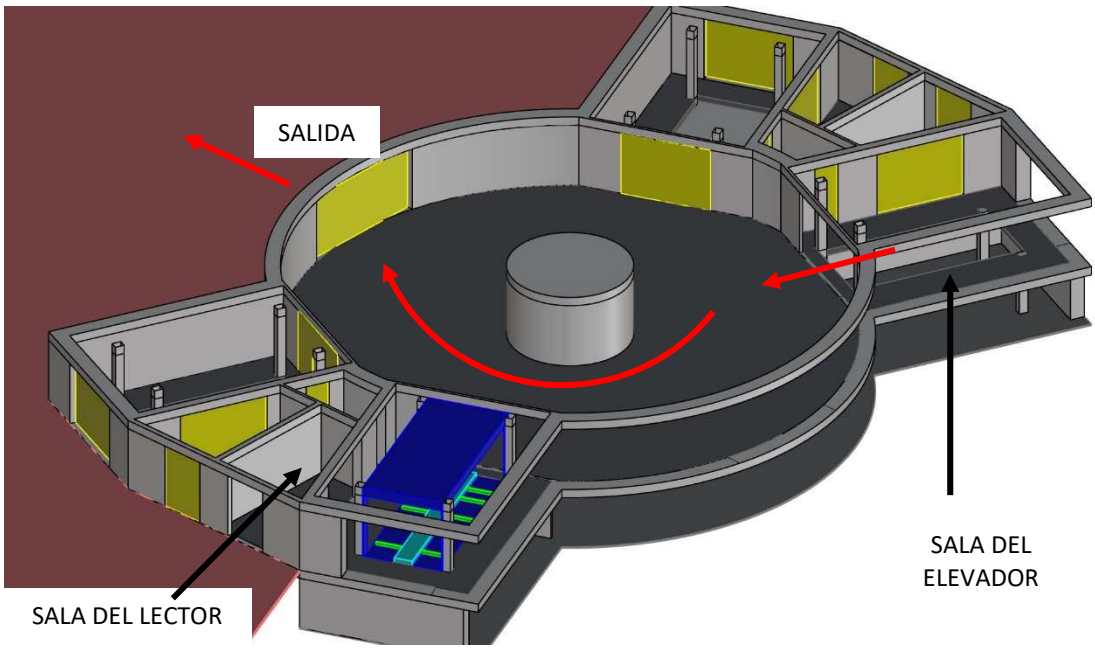


Figura 9. Vista 3D de la planta de recepción del aparcamiento.

La figura 9 es una vista de la planta de recepción donde se detalla la sala del elevador y del lector y el recorrido que hará el vehículo al retirarlo, en la figura 10 se muestra el dispositivo que recoge el lector RFID PN532 y el display informativo LCD 16x02.

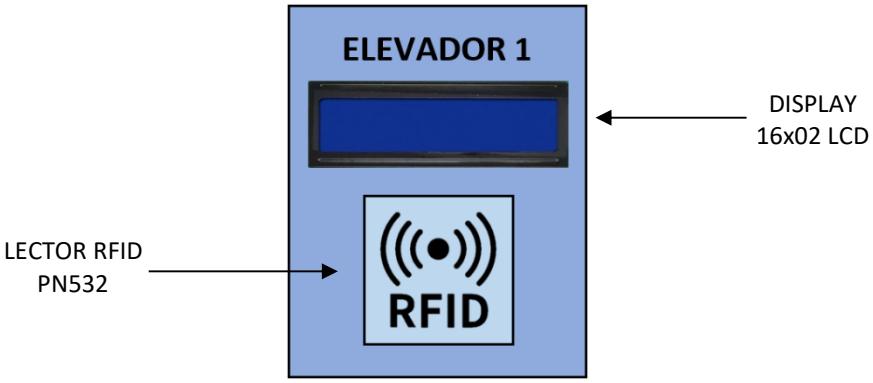


Figura 10. Representación del lector de tarjetas RFID más el display informativo situado en la sala del lector

Para poder guardar o retirar el vehículo es necesaria la actuación de tres sistemas móviles: la plataforma elevadora, la estructura giratoria y la plataforma de desplazamiento lineal. La plataforma elevadora situará el vehículo en la altura correspondiente, al mismo tiempo la estructura giratoria rotará y situará la división o hueco enfrente del elevador y por último la plataforma de desplazamiento lineal, integrada en el propio elevador, introducirá o retirará el vehículo. En la figura 11 se ve una representación del aparcamiento o retirada de un vehículo.

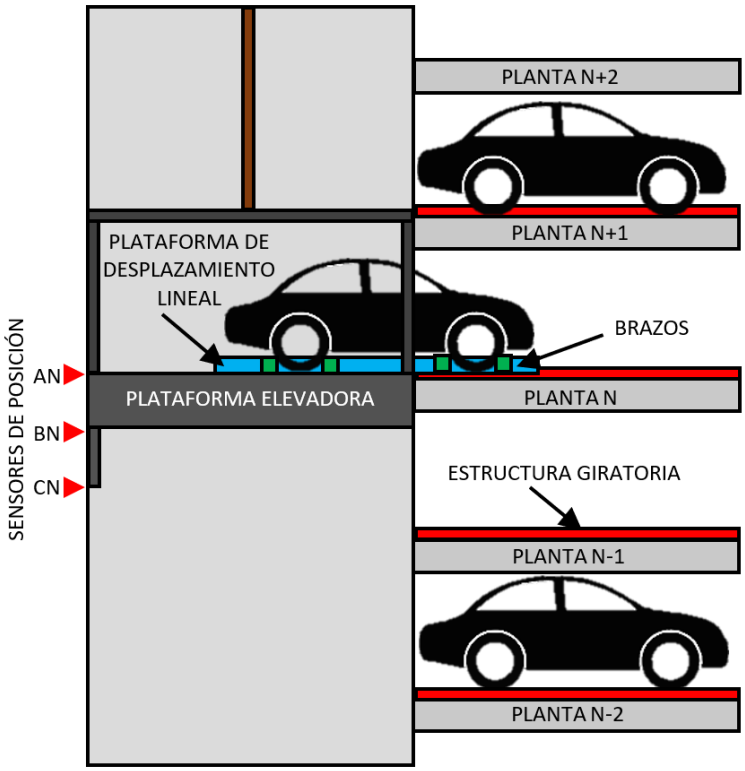


Figura 11. Representación del sistema de aparcamiento o retirada de vehículos.

Estos sistemas móviles serán controlados individualmente por autómatas. El PLC de la plataforma elevadora requerirá del número de planta a la que debe situarse, la estructura giratoria necesitará la plaza y el elevador que debe enfrentar (se recuerda que habrá un total de ocho divisiones por planta y cuatro elevadores), y la plataforma de desplazamiento lineal requerirá la orden de introducir o retirar el vehículo, todos estos datos se transmitirán desde el sistema principal, la Mini-DK2.

En esta figura (figura 12) se representa la conexión entre el control externo de PLC's y el motor de la plataforma elevadora, donde se usa un variador de frecuencia controlado por el PLC para regular la velocidad y sentido de giro del motor.

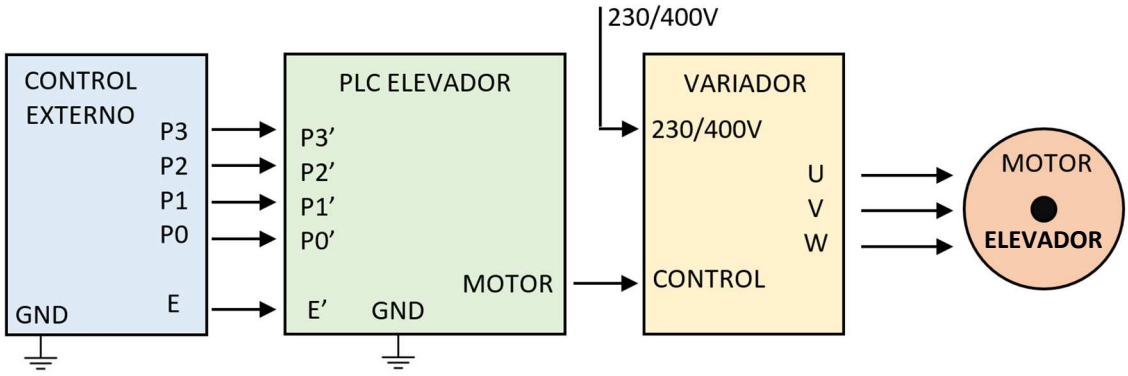


Figura 12. Representación de la conexión entre el motor del elevador y la Mini-DK2.

Dicho control externo de elevadores será controlado por el sistema principal, la tarjeta Mini-DK2. A continuación se muestra en la siguiente figura (figura 13) un resumen del hardware de la tarjeta con los diferentes módulos y PLC's.

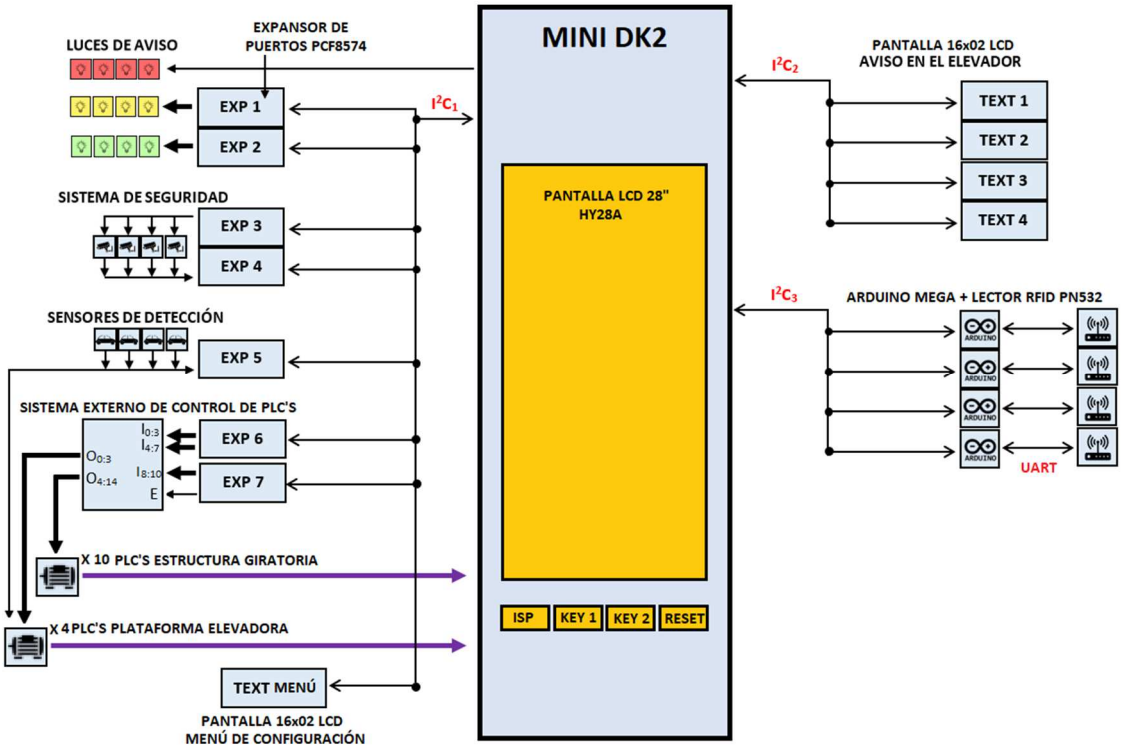


Figura 13. Resumen del hardware del automatismo relacionado con los distintos módulos y periféricos usados.

Para poder enviar la información necesaria a los autómatas programables de las plataformas elevadoras o de las estructuras giratorias es necesario de un sistema de control externo basado en multiplexación que envíe en el momento preciso los datos a un único PLC, pues sino sería necesario el uso de multitud de expansores de puertos, pues no existen suficientes puertos en la tarjeta Mini-DK2.

Al no disponer de suficientes pines en el sistema principal (Mini-DK2) se usarán expansores de puertos basados en el sistema de comunicación I^2C que añaden ocho pines adicionales. Estos expansores se representan en la figura superior con el nombre de EXP N, siendo N el número de expansor. En el caso de los displays LCD 1602A es necesario otro expansor de puertos para conectarlas al bus I^2C, estos displays se representan en la figura como TEXT N, siendo N el número de display o MENÚ.

CAPÍTULO II

ARQUITECTURA FÍSICA DEL PARKING

2. ARQUITECTURA FÍSICA DEL PARKING

Conocer la arquitectura del automatismo antes de adentrarse en el hardware y software del proyecto es esencial, pues para escoger correctamente los componentes y ubicarlos en él se debe de conocer su ubicación y características.

Este aparcamiento tendrá forma de caramelo, una forma muy visual e innovadora que se aleja del clásico aparcamiento en forma de estantería, donde se guardan los vehículos a un lado u otro del robot que los almacena. En las siguientes imágenes se puede observar los dos tipos de aparcamiento descritos, a la izquierda el aparcamiento con forma de estantería (figura 14) y a la derecha el aparcamiento estudiado (figura 15).

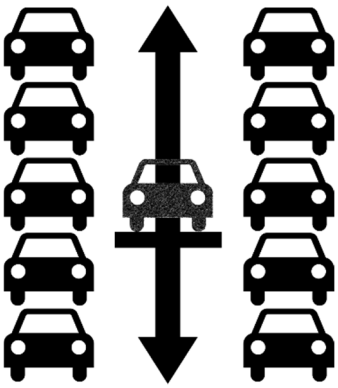


Figura 14. Vista frontal de un ejemplo de aparcamiento en forma de estantería.

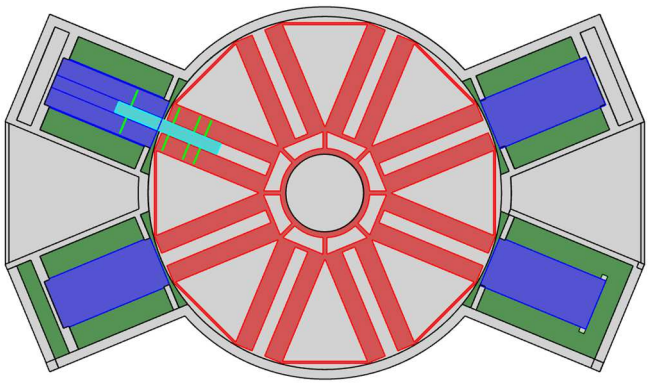


Figura 15. Planta del boceto 3D del aparcamiento estudiado.

Respecto al aparcamiento estudiado se puede dividir en dos edificios, el central circular y los cuatro edificios anexos poliédricos. Ambas construcciones tienen 14 alturas, donde en la planta principal o planta cero interactuarán los clientes, pudiendo dejar y recoger sus vehículos.

Una planta más arriba, planta primera, se encuentra la sala de control y eléctrica, que contendrá los sistemas de control del automatismo y los cuadros eléctricos principales, entre otros.

Las siguientes alturas están reservadas para el aparcamiento de los coches, en este caso serán 10 plantas, todas ellas encima de la sala de control, en el apartado 7 de “Conclusiones y futuros trabajos” se estudiará la posibilidad de ampliar este número de plantas y la capacidad de colocarlas debajo de la planta cero o recepción.

En lo más alto de la torre se encuentra la azotea, encargada de contener los sistemas de elevación de la plataforma elevadora y el sistema de ventilación, y el tejado, donde habrá un parque de placas fotovoltaicas para intentar autoabastecer el aparcamiento, además de los diferentes aparatos de ventilación y comunicación.

Respecto a la última planta, el foso, se situará en la parte más profunda del aparcamiento y alojará parte de la construcción de la plataforma elevadora para su correcto posicionamiento en la penúltima altura, también contendrá los cimientos del edificio y las acometidas de los diferentes servicios.

En la figura 16 se muestra una vista 3D del boceto del aparcamiento.

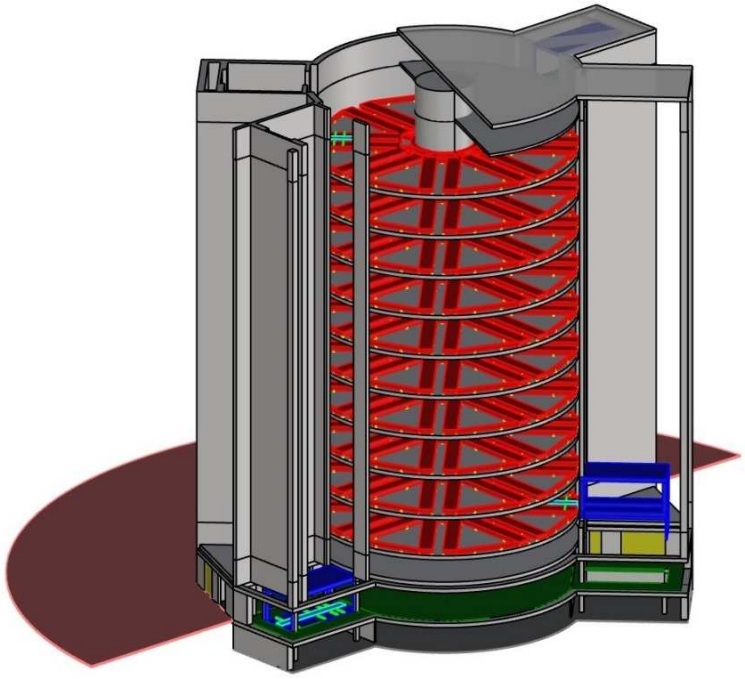


Figura 16. Vista de lado del boceto 3D del aparcamiento.

2.1. PLATAFORMA ELEVADORA Y PLATAFORMA DE DESPLAZAMIENTO LINEAL

La plataforma elevadora es la encargada de recibir, entregar, posicionar e introducir o extraer los vehículos, por lo que su diseño debe estar adaptado a ello. Se compone de seis elementos: plataforma, base, columnas, techo, extensión para control y plataforma de desplazamiento lineal. La podemos observar en la figura 17.

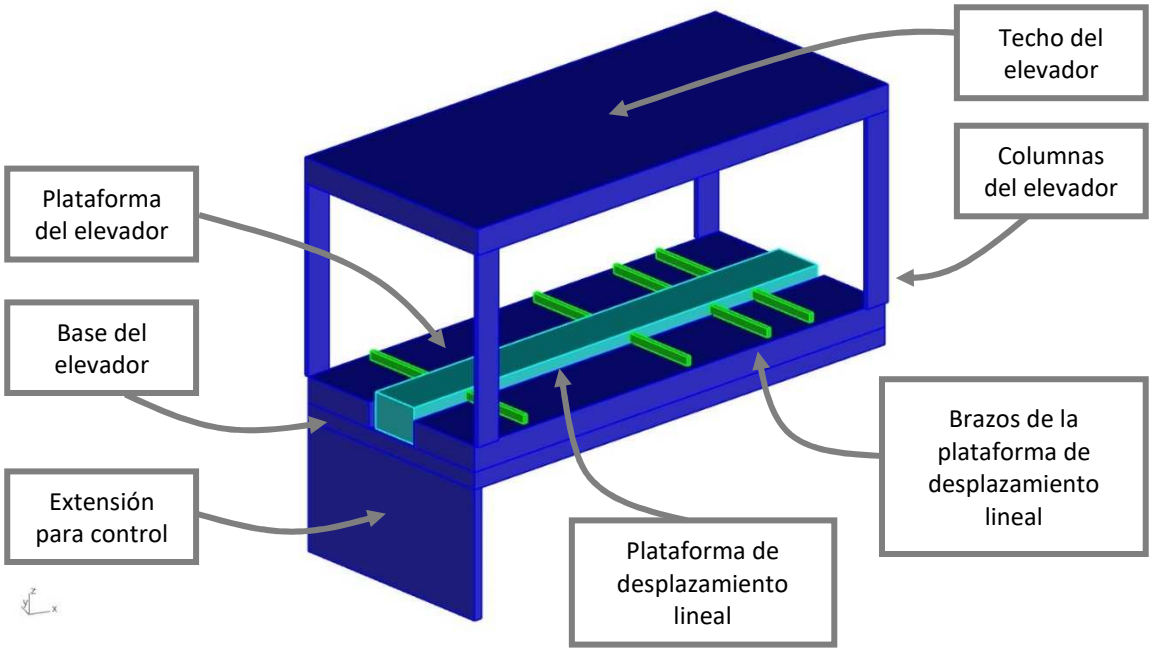


Figura 17. Representación 3D de la plataforma elevadora y de la plataforma de desplazamiento lineal.

En primer lugar, se diseñará la plataforma, es decir, donde se situará el vehículo, la cual debe estar preparada para albergar la máxima cantidad de modelos de coches, pues se busca la funcionalidad del proyecto. Las dimensiones del perímetro exterior se basarán en la búsqueda de los coches más largos y anchos [1], y las dimensiones del hueco central (donde se situará la plataforma de desplazamiento lineal encargada de introducir o extraer los vehículos) se hallarán con las dimensiones del coche más estrecho.

El ancho de la plataforma será de 2,4 metros, pues actualmente el BMW X6M cuenta con un ancho de 2,019 metros de anchura (sin tener en cuenta los retrovisores), siendo el modelo de coche más ancho en el mercado actual; si tenemos en cuenta los retrovisores, esta medición asciende a 2,212 metros. Los casi 200 milímetros restantes se proporcionan para extender en el tiempo la vida útil del aparcamiento, pues cada año el ancho de los vehículos va aumentando para aumentar la tracción, también se aportan para poder alojar a coches de competición que hayan sido modificados en anchura, pues se busca la máxima polivalencia posible.

El largo de la plataforma será de 5,7 metros, pues actualmente el coche más largo del mercado mide 5,48 metros de longitud siendo el Renault Trafic. Como se explicará más adelante, la plataforma de desplazamiento lineal dispone de ocho brazos (estando abiertos los dos brazos delanteros, ya que sirven como tope máximo para los coches), se han situado en la plataforma a 0,89 metros desde donde se situaría el morro del vehículo.

El voladizo más largo también pertenece al Renault Trafic, con un valor de 1,014 metros de largo. Si tenemos en cuenta que la altura del brazo abierto es de 100 mm (realmente es de 80mm) y el diámetro de la rueda (50,6 mm el neumático más pequeño) obtenemos que la distancia desde el eje de la rueda hasta el final de la plataforma es de 1,091 metros (superior al voladizo máximo), dejando aproximadamente 77 milímetros de margen en el caso más restrictivo, en el caso más permisivo, con el neumático de diámetro mayor (787,1 mm) la distancia de margen sería de 138 milímetros. Dicho vehículo se quedará en un rango de 82 a 143 milímetros del borde de la plataforma.

Respecto a la altura de la plataforma, esta será de 0,3 metros. La plataforma de desplazamiento lineal no debe sobresalir más de ocho centímetros por encima de la superficie de la plataforma, siendo su altura de 38 centímetros, por lo que la superficie de la plataforma será de ese valor.

El hueco central donde se depositará la plataforma de desplazamiento lineal será de 0,7 metros de ancho centrado en la plataforma, pues si tenemos en cuenta que el vehículo más estrecho (el Invicta Electric de 1,499 metros de anchura), está centrado con la plataforma y que usa los neumáticos más anchos (335 mm) obtenemos una anchura máxima de 0,83 metros, a esto le restamos los 130 milímetros de holguras y descuadres.

En resumen, el largo de la plataforma será de 5,7 metros, el ancho de 2,4 metros, el hueco central será de 0,7 metros de ancho, el brazo fijo de la plataforma de desplazamiento lineal se situará a 0,89 metros del extremo delantero y su altura será de 8 centímetros. La figura 18 representa la plataforma elevadora con cotas respecto al BMW X3 de 2005.

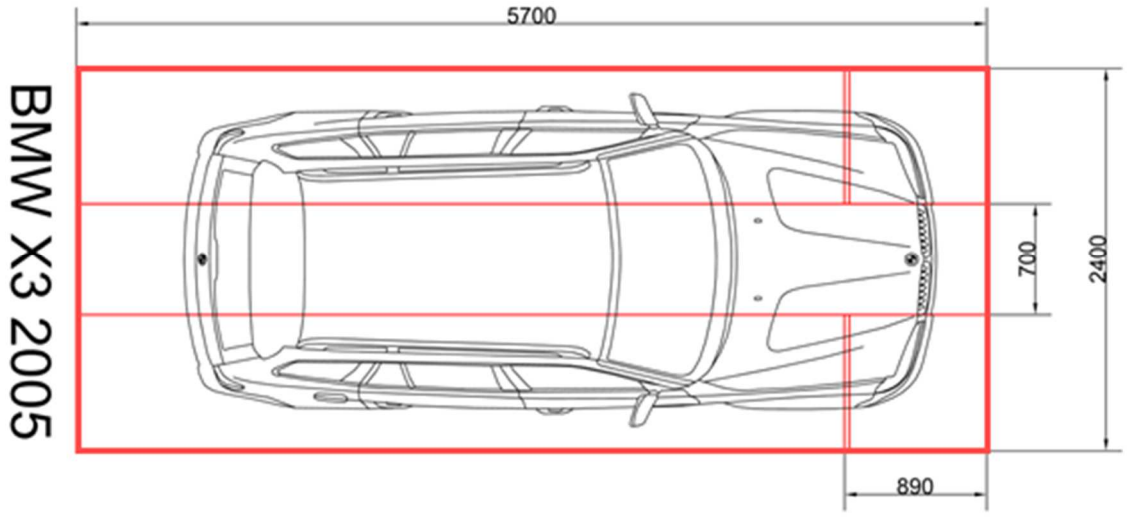


Figura 18. Representación de la plataforma elevadora en comparación con el BMW X3 de 2005.

La base del elevador tendrá las mismas medidas de ancho y largo que la plataforma debido a que es la zona que sostiene a la plataforma de desplazamiento lineal, es decir, será de 5,7 metros de largo, 2,4 metros de ancho y una altura de 0,2 metros.

Las columnas tendrán 2,5 metros de altura e irán situadas al nivel de la base, por lo que el habitáculo tendrá una altura de 2,2 metros. La altura del habitáculo se obtiene del nivel del Ineos Grenadier (2,036 m), el coche más alto del mercado actual. Los 164 milímetros de holgura servirán para albergar cofres de techo, pues para coches más normales como el BMW X3 (1,676 m de alto) la holgura sería de 524 milímetros, altitud suficiente para recoger los cofres u otras extensiones de almacenamiento. El ancho y largo de las columnas será arbitrario al igual que la altura de la base del elevador. Para el diseño se han usado unas medidas de 300 milímetros de longitud por 50 milímetros de anchura.

El techo debe albergar las columnas, pues estas van unidas a este por la parte inferior (no por el lateral), por lo que las dimensiones de su superficie tendrán parte de arbitrariedad. La altura también estará basada en la percepción del autor, sin basarse en cálculos. Sus dimensiones son de 5,9 metros de largo, 2,7 metros de ancho y 0,3 metros de alto.

La extensión de control será una estructura hueca que irá unida a la base y servirá para situar los elementos de control de posicionamiento del elevador, sus dimensiones serán de 2,4 x 0,1 x 1,5 metros (largo, ancho y alto). La altura depende de la altura del foso, mientras que el ancho y la longitud no se basan en medidas especiales, simplemente constituirán una estructura sólida y fina.

La plataforma de desplazamiento lineal dispone de unos pequeños brazos que se ajustan a las ruedas del vehículo para su almacenamiento o retirada, su posición en la plataforma se basa en el rango de posibles posiciones de las ruedas, calculado mediante la batalla del vehículo. Se situará dentro del hueco central, será de 5,6 metros de largo, 10 centímetros menos para poder acoplarlo adecuadamente, 70 centímetros de grosor y de 38 centímetros de altura. En la figura 19 se puede observar una representación de la plataforma de desplazamiento lineal levantando al vehículo, los ocho centímetros que sobresalen de la plataforma se representan con el color verde y los brazos en color rojo. En la referencia [2] se puede encontrar un video de un ejemplo real.

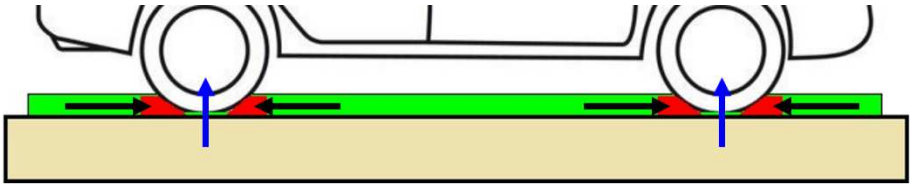


Figura 19. Plataforma de desplazamiento lineal levantando el coche para poder desplazarse.

2.2. PLANTA PRINCIPAL DEL PARKING

Esta planta tiene la función de recepcionar y entregar los vehículos. Posee también la sala eléctrica de mandos y la sala del lector de tarjetas. Se situará a nivel del suelo de la calle para poder acceder con más facilidad.

Esta planta se puede dividir en dos zonas: la zona interior circular y las dos zonas exteriores poligonales. Las medidas del primer sector se basan en las medidas de la estructura que albergará los vehículos (apartado 2.4.), la segunda zona basa sus medidas en la comodidad y accesibilidad, pues posee pasillos y puertas accesibles para minusválidos. La altura de esta planta será de 2,8 metros, 0,3 metros de anchura del suelo y 2,5 metros desde este al techo.

En la figura 20 se observa una representación 3D, en los anexos se adjuntan planos más detallados con las dimensiones.

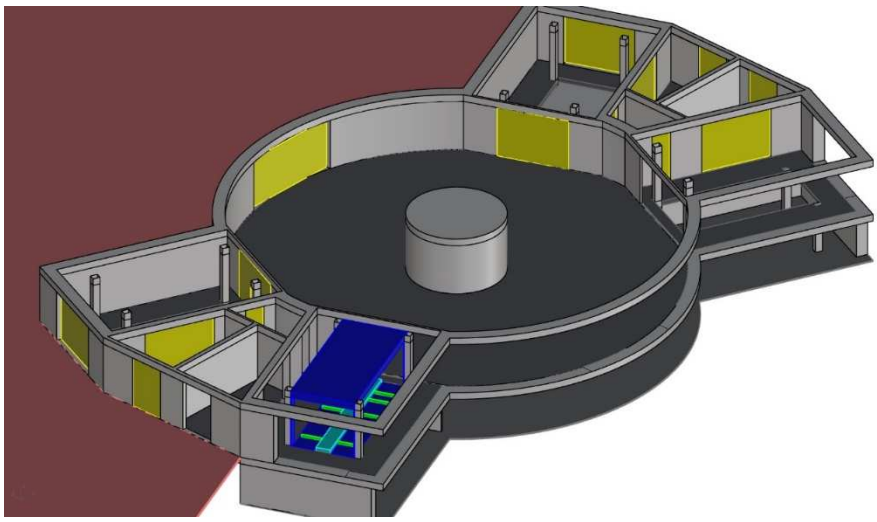


Figura 20. Representación 3D de la planta principal o recepción del aparcamiento.

2.3. PLANTA INFERIOR DEL PARKING O FOSO

El foso se situará en lo más profundo del aparcamiento, su función principal es albergar la plataforma elevadora cuando esta se encuentre en la penúltima altura (en este caso la planta de recepción), ya que debe de estar a nivel y, como se ha descrito anteriormente, tiene una extensión en un extremo para el control de posición de esta.

Sus dimensiones se basan en la planta principal, siendo de 1,8 metros de altura, suficiente para realizar trabajos sencillos en la base de la plataforma.

En la figura 21 se observa una representación 3D, en los anexos se adjuntan planos más detallados con las dimensiones.

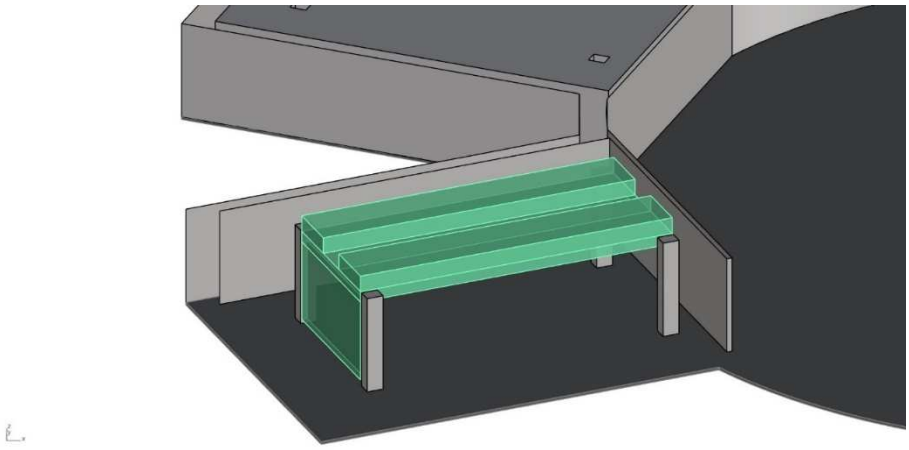


Figura 21. Representación 3D del foso del aparcamiento.

2.4. PLANTA DE CONTROL DEL PARKING

La construcción del aparcamiento y la estructura de control de la plataforma elevadora obligan a añadir una planta justo encima de la de recepción. Aprovechando dicha estancia, se instalará en ella el sistema y cuadros eléctricos y el sistema de control del automatismo, es decir, el microcontrolador **LPC1768** y sus componentes.

Esta planta tendrá una altura de 2,5 metros, 0,3 metros de anchura del suelo y el resto desde este al techo para que sea cómodo trabajar en los momentos de mantenimiento. El resto de las dimensiones se basan en la planta de almacén de vehículos.

En la planta de recepción (anexo al círculo central) se hallan las salas del elevador del lector y una sala para cuadros eléctricos, que no aparecerá en el resto de las plantas del automatismo.

En la figura 22 se observa una representación 3D, en los anexos se adjuntan planos con las dimensiones.

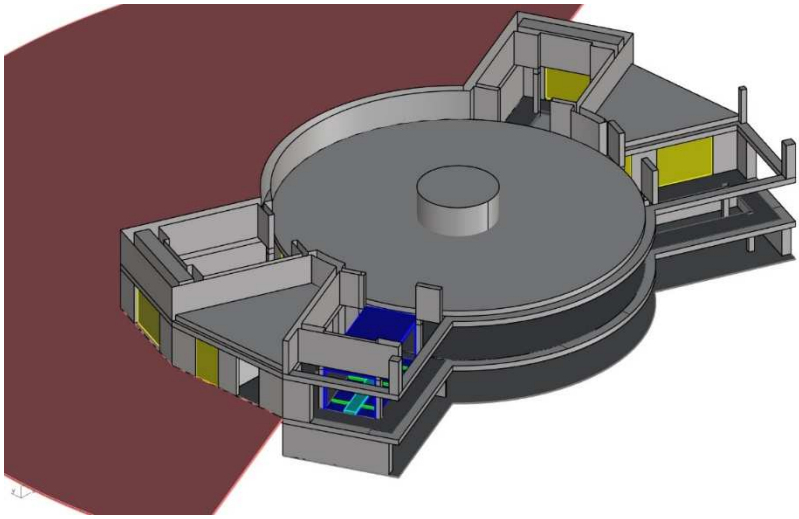


Figura 22. Representación 3D de la planta de control del aparcamiento.

2.5. PLANTA DE ALMACENAMIENTO DE VEHÍCULOS DEL PARKING

Esta planta se utiliza para dimensionar todas las demás alturas, pues es necesario conocer el número de vehículos que se van a alojar en ella y de qué manera. Añadir o eliminar un vehículo supone expandir o disminuir su superficie.

En este proyecto se dimensiona el aparcamiento para alojar a ocho vehículos por planta por lo que dichas medidas se basarán en las medidas de la plataforma elevadora (apartado 2.1.). La zona de almacenaje de vehículos debe ser circular, pues esta tiene una estructura que rotará para situar los aparcamientos enfrente del elevador indicado. Para ello, se descarta el disponer las plazas en paralelo y se opta por juntar las esquinas interiores de las ocho plazas, formando la estructura de la imagen superior. En el centro de dicha planta (al igual que en todas las demás), se encontrará un cilindro que servirá tanto de acceso peatonal, en caso de mantenimiento, como de barrera estructural para aguantar el peso y como canaleta para los cables. La altura de esta planta también será de 2,5 metros, 0,3 metros de anchura del suelo y el resto desde este al techo.

En la figura 23 se observa una representación 3D, en los anexos se adjuntan planos con las dimensiones.

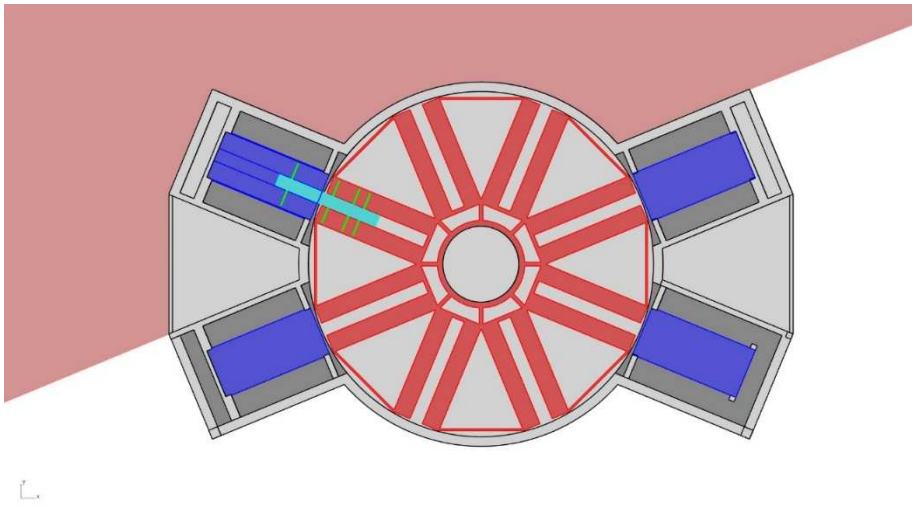


Figura 23. Representación 3D de la planta de almacenamiento de coches del aparcamiento.

2.6. PLANTA SUPERIOR DEL PARKING O AZOTEA

Para poder albergar el sistema de elevación y el control de las placas solares, se requiere de la azotea. Sus dimensiones dependen de la estructura de almacenamiento de vehículos, al igual que el resto de las plantas. La altura de esta planta será de 2,5 metros, 0,3 metros de anchura del suelo y el resto desde este al techo.

En la figura 24 se observa una representación 3D, en los anexos se adjuntan planos más detallados con las dimensiones.

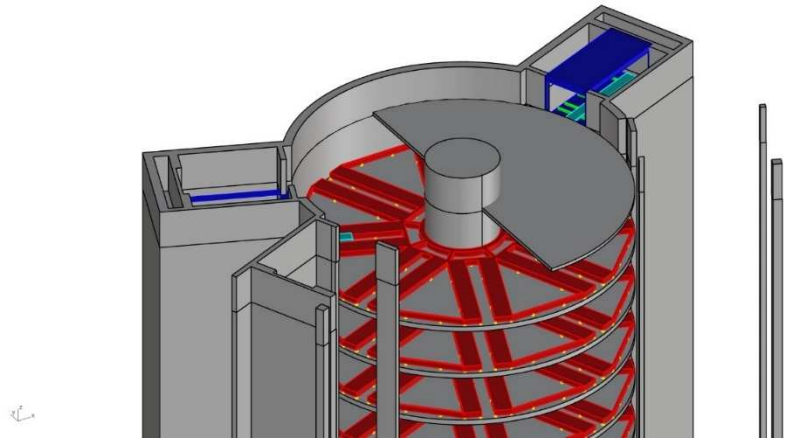


Figura 24. Representación 3D de la azotea del aparcamiento.

2.7. TEJADO DEL PARKING

Coronando la torre se encuentra el tejado, el cual está pensado para ser cubierto en su totalidad de placas solares, que abastecerán al automatismo. El área útil del tejado es aproximadamente de 470 metros cuadrados, pues dispone de cuatro rectángulos de aproximadamente 7,5 metros de largo por 6 metros de ancho y un círculo interno de aproximadamente 9,6 metros de diámetro. Se debe respetar el área del círculo interno, ya que se corresponderá con una escalera para poder acceder al tejado.

En la figura 25 se observa una representación 3D, en los anexos se adjuntan planos más detallados con las dimensiones.

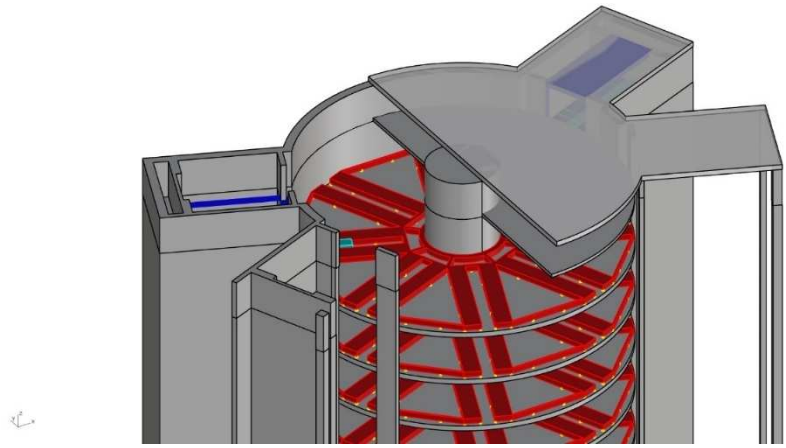


Figura 25. Representación 3D del tejado del aparcamiento.

En la figura 26 se encuentran las cotas del tejado del parking para obtener la superficie en la que se pueden poner placas solares.

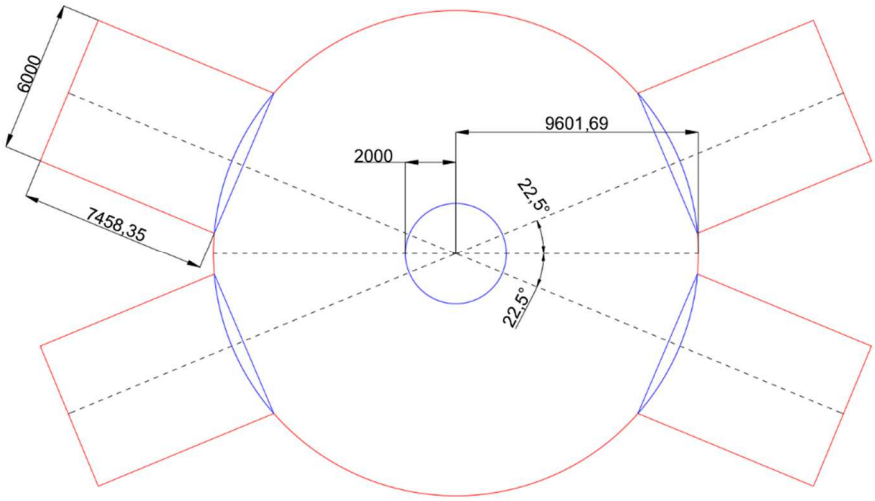


Figura 26. Dimensiones de la superficie del tejado en milímetros

$$Área = 4 \cdot Área_{RECTÁNGULO} + Área_{CÍRCULO_{EXTERNO}} - Área_{CÍRCULO_{INTERNO}}$$

$$Área = 4 \cdot (a \cdot b) + \pi \cdot r_E^2 - \pi \cdot r_I^2 = 4 \cdot (7{,}46 \cdot 6) + \pi(9{,}6^2 - 2^2) = 4 \cdot 45 + 277 = 457m^2$$

Las placas solares escogidas poseen una potencia de 200W [3] y tienen unas dimensiones de 1332 x 992 x 35mm. Normalmente, las placas solares deben tener una inclinación de 30 grados [4] para aprovechar al máximo la luz en invierno, por lo que respecto al suelo supone una superficie de 1153 x 992 mm =1,14 m^2, si se aproxima esta área a 1,2 m^2 por placa se obtiene un total de 548 placas solares, lo que es equivalente a aproximadamente 110kW.

2.8. ANÁLISIS DE LA SUPERFICIE OCUPADA DEL PARKING

La superficie total ocupada por el aparcamiento propuesto (80 plazas divididas en 8 divisiones o huecos por planta, en 10 plantas y 4 elevadores) es de 530 m^2 aproximadamente.

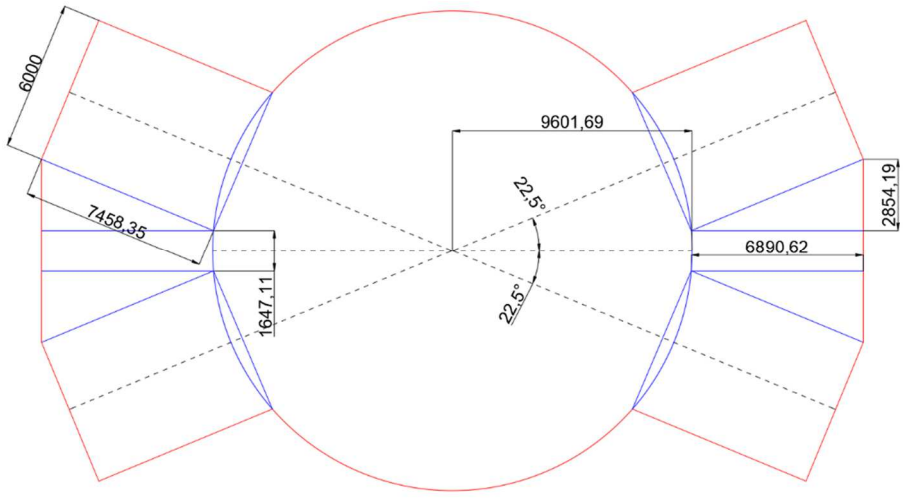


Figura 27. Medidas de la base del aparcamiento propuesto, medidas en milímetros.

En la imagen 27 se muestran las cotas para obtener la superficie ocupada por el aparcamiento.

$$Área_{TOTAL} = Área_{INTERNO} + Área_{ANEXO} = 290 + 240 = 530m^2$$

$$Área_{INTERNO} = Área_{CÍRCULO_{EXTERNO}} = \pi \cdot r^2 = \pi \cdot 9{,}6^2 = 290m^2$$

$$Área_{ANEXO} = 4 \cdot Área_{RECTÁNGULO_{ELEVADOR}} + 2 \cdot Área_{RECTÁNGULO_{LECTOR}} + 4 \cdot Área_{TRIÁNGULO}$$

$$Área_{ANEXO} = 4 \cdot (a \cdot b) + 2 \cdot (c \cdot d) + 4 \cdot \frac{c \cdot e}{2} = 4 \cdot (7{,}46 \cdot 6) + 2 \cdot (6{,}89 \cdot 1{,}65) + 2 \cdot (6{,}89 \cdot 2{,}85)$$
$$= 179 + 22{,}7 + 39{,}3 = 241\ m^2$$

La altura total del aparcamiento propuesto será de 35 metros, teniendo en cuenta las diez plantas de almacenamiento, la planta de recepción, control, foso, azotea y tejado.

Para saber si el aparcamiento diseñado es rentable respecto al precio del suelo, es conveniente conocer la superficie de un parking convencional, esta se aproximará mediante las dimensiones de las plazas de este proyecto (5,7 metros de largo por 2,4 metros de ancho), donde el pasillo central sea de, al menos, un 50% más amplio que el ancho de la plaza, es decir, de 3,6 metros.

Se comparan el aparcamiento con tres parkings convencionales: el primero (figura 28) ordena las plazas en una explanada de una única altura con cuatro filas, el segundo (figura 29) los situará en un edificio de dos plantas con un pasillo de 6 metros de ancho para albergar la rampa de subida y bajada de dos carriles, y el tercero (figura 30) también estará situado en un edificio de cuatro plantas, suponiendo que la rampa de subida cabe en dicho espacio.

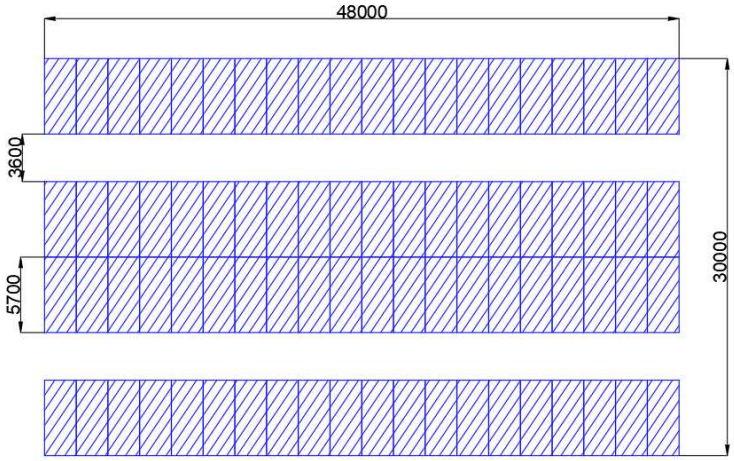


Figura 28. Primer ejemplo de aparcamiento convencional de 80 plazas, medidas en milímetros.

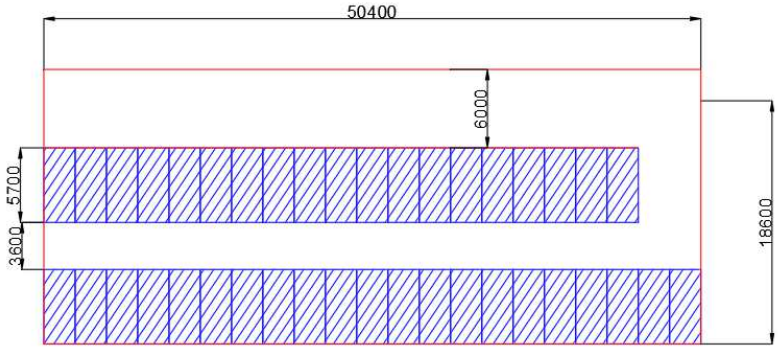


Figura 29. Segundo ejemplo de aparcamiento convencional de 80 plazas, medidas en milímetros.

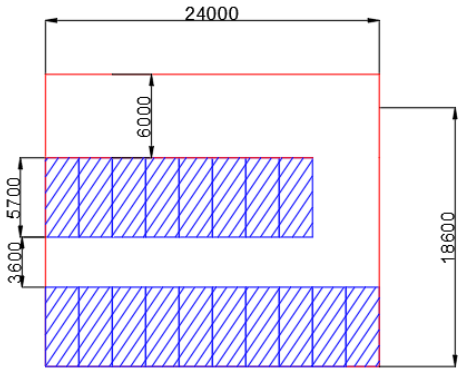


Figura 30. Tercer ejemplo de aparcamiento convencional de 80 plazas, medidas en milímetros.

El artículo 109 del Plan General de Ordenación Urbana de *Barcelona [5]* dice lo siguiente:

"Los locales tendrán una altura libre mínima en todos sus puntos de dos metros veinte centímetros (2,20 m), que no se podrá reducir con canalizaciones o instalaciones análogas en las zonas de circulación.

En el exterior se indicará la altura máxima de los vehículos que puedan penetrar, inferior en cincuenta centímetros (0,50 m) a la altura libre de paso del local y acceso al mismo."

Si se tiene en cuenta que todo el sistema de ventilación y demás instalaciones se sitúan en un falso techo, la distancia entre el suelo y este deberá ser 2,20 metros al menos. Dicho falso techo aproximadamente medirá 0,8 metros de altura, por lo que cada altura de aparcamiento ocupará aproximadamente 3 metros.

Como conclusión, la superficie del primer ejemplo de aparcamiento es de 1440 m^2, un 272% más grande que la superficie usada en este automatismo, el segundo ejemplo posee una superficie de 2246m^2, un 177% más grande. Sin embargo, el tercer ejemplo posee una superficie de 450m^2, un 15% más pequeño que el estudiado.

La altura del parking propuesto es de 35 metros, mientras que las alturas de los ejemplos son de 4, 7 y 13 metros respectivamente, considerando 3 metros por planta y 1 metro de tejado.

CAPÍTULO III

FUNCIONAMIENTO DEL PARKING

3. FUNCIONAMIENTO DEL AUTOMATISMO

Conocer el funcionamiento del automatismo antes de introducirse de lleno en el estudio es importante, ya que para diseñar cualquier parte del proyecto se tiene que conocer el funcionamiento del sistema. Dicho funcionamiento se puede clasificar en cuatro operaciones: estacionar, retirar y dos casos anómalos, los cuales contemplan el aparcamiento de un vehículo habiendo ya otro en el interior o retirar un coche sin haber uno dentro.

3.1. PROCEDIMIENTO DE APARCAMIENTO DE UN VEHÍCULO

Una persona llegará montada en su vehículo hasta la entrada del aparcamiento, si hay plazas libres estará abierto. Entonces se aproximará hacia una plataforma en el interior del edificio y una vez haya estacionado el vehículo con el freno de mano (como lo haría normalmente) se bajará de él y acudirá hacia la sala de recepción donde se sitúa el lector de tarjetas, acercará su tarjeta y seguidamente se activará el automatismo.

El automatismo detecta que el vehículo se encuentra correctamente situado en la plataforma elevadora y comprueba si la plaza referenciada a la tarjeta está libre, ya que cada tarjeta tiene su plaza (por ejemplo: Planta 3, división 7). Si todo es correcto, el automatismo se activa.

Lo primero que el sistema hará será activar las medidas de seguridad. Una vez recibida una respuesta por parte de los sistemas de protección, el automatismo enviará una señal al subsistema encargado de elevar el vehículo, situándolo en la altura o planta correspondiente (en el ejemplo anterior, planta 3). Dicha planta tendrá una estructura que girará tantos grados como se necesiten para posicionar la entrada de la plataforma elevadora frente a la plaza libre (división 7). A continuación, una plataforma de desplazamiento lineal automatizada introducirá el vehículo en la plaza libre. La plataforma descenderá a su posición de reposo, los sistemas de seguridad se desactivarán y habrá finalizado el aparcamiento.

3.2. PROCEDIMIENTO DE RETIRADA DE UN VEHÍCULO

Una persona llegará andando hasta la sala de recepción y activará el automatismo acercando la tarjeta al lector. Seguidamente, el sistema detecta que no hay vehículo en la plataforma elevadora y comprueba que la plaza asociada a la tarjeta está ocupada (Planta 3, división 7). Si todo esto es correcto, el automatismo se activa procediendo a retirar el vehículo.

Lo primero que se hace es activar las medidas de seguridad. Una vez recibida una respuesta por parte de los sistemas de seguridad, el sistema enviará una señal al subsistema encargado de bajar el vehículo, situando la plataforma elevadora en la altura o planta correspondiente (planta 3). La estructura giratoria de dicha planta girará tantos grados como se necesiten para posicionar la entrada del elevador enfrente del vehículo (división 7). A continuación, una plataforma de desplazamiento lineal automatizada extraerá el vehículo hasta la plataforma elevadora, la cual descenderá a la posición de reposo o salida. Por último, se desactivarán las medidas de seguridad y el propietario podrá retirar el vehículo.

3.4. ESQUEMA DEL FUNCIONAMIENTO

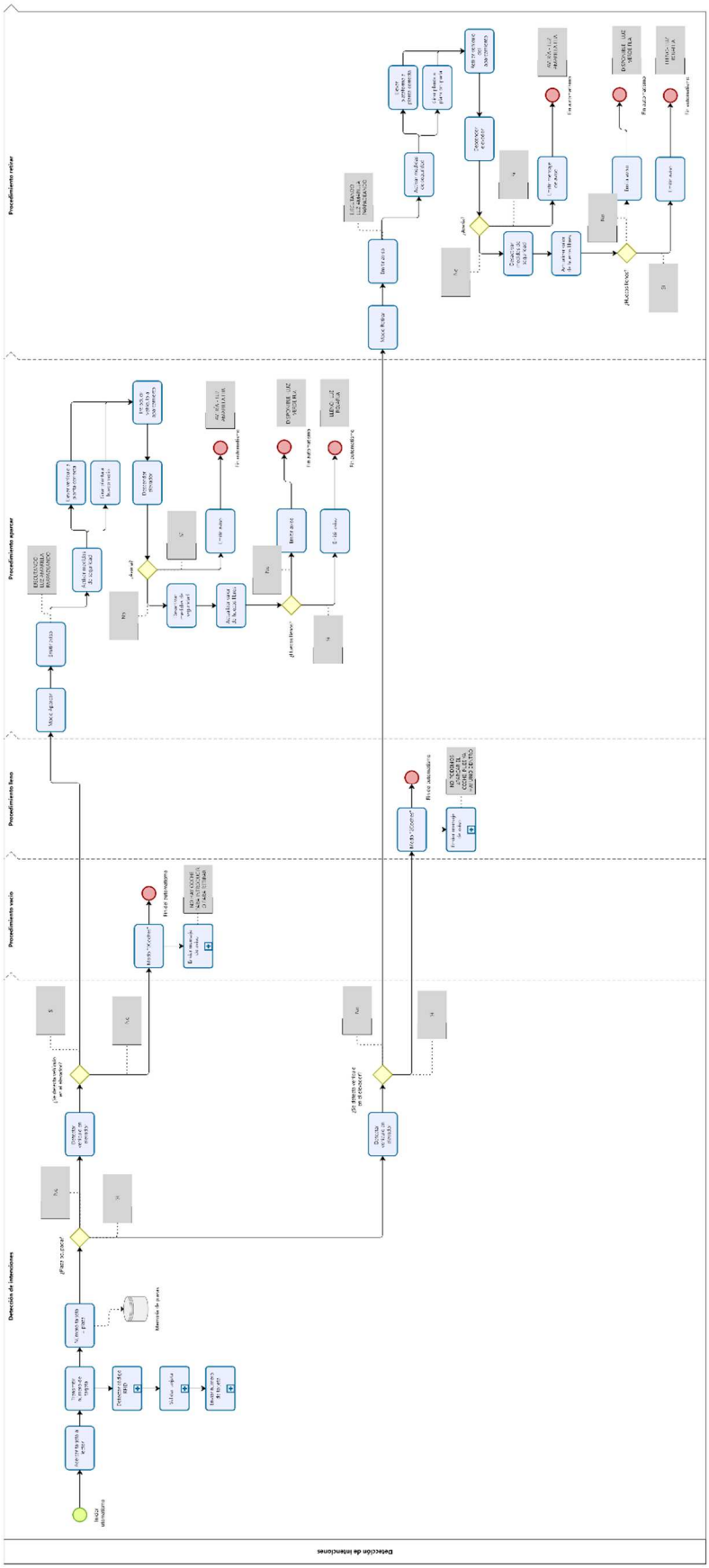


Figura 31. Esquema completo del automatismo realizado con Bizagi.

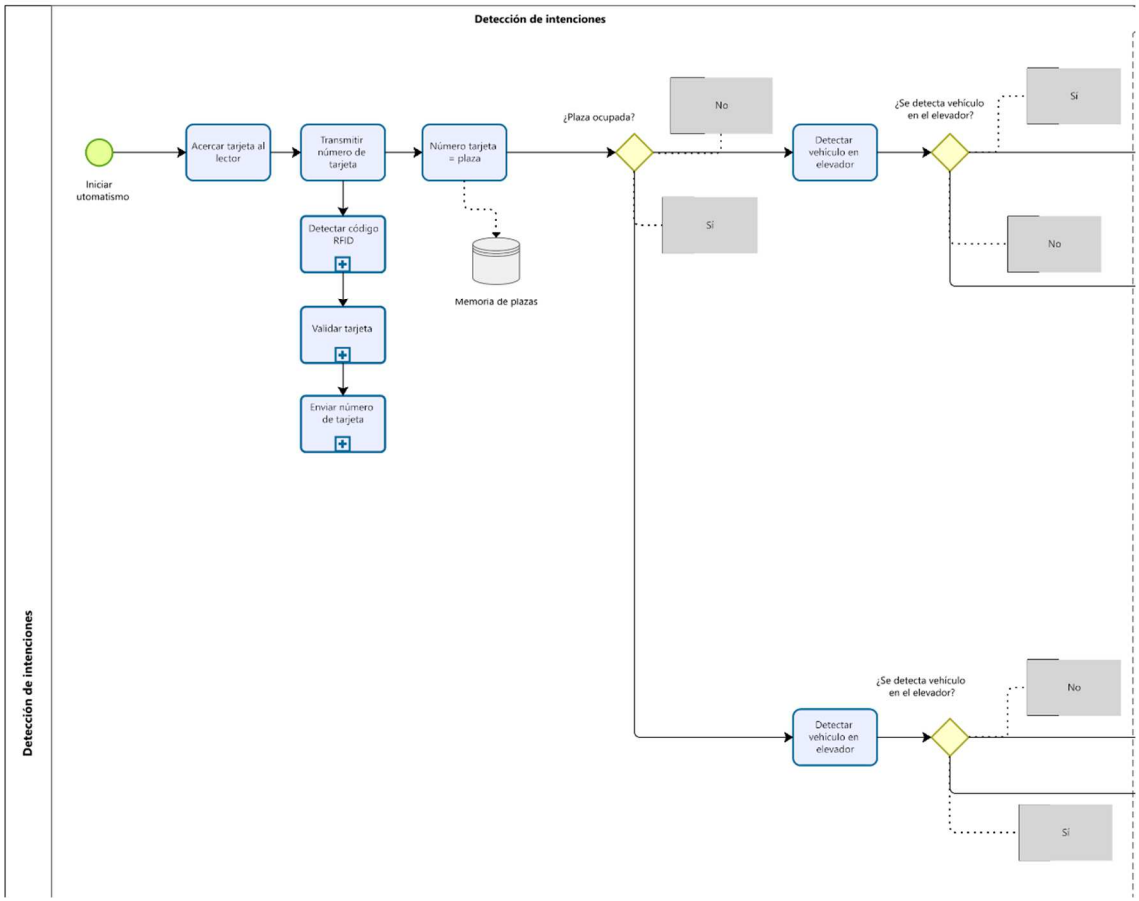


Figura 32. Primera parte del esquema del automatismo realizado con Bizagi.

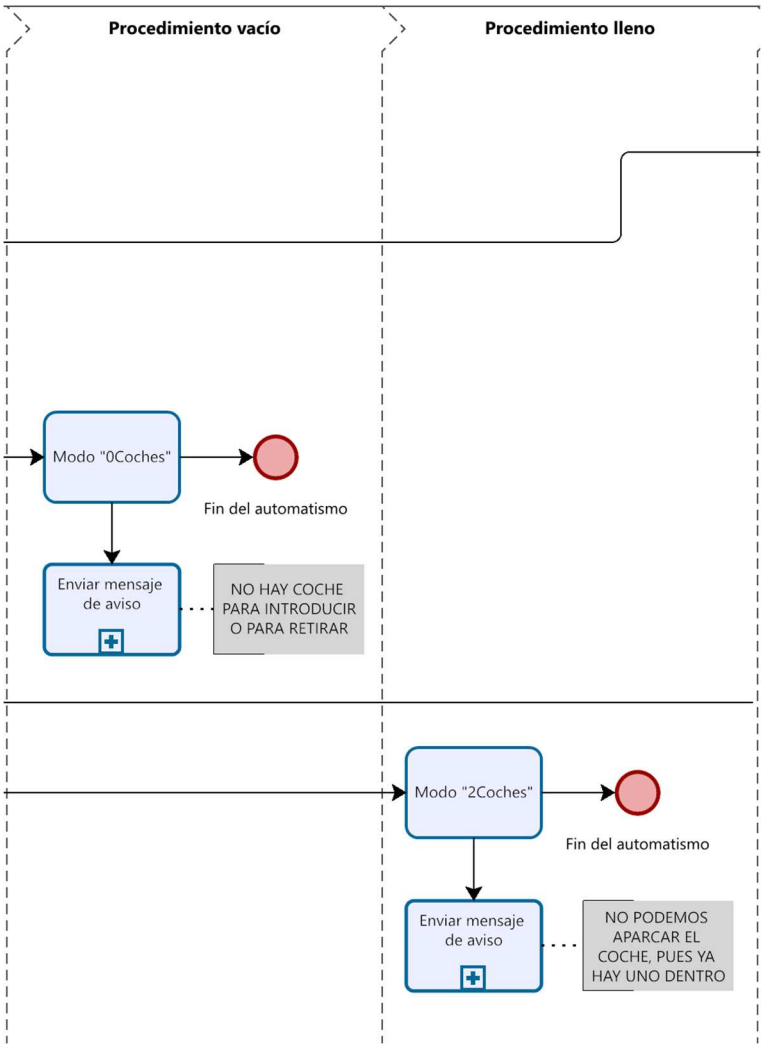


Figura 33. Segunda parte del esquema del automatismo realizado con Bizagi.

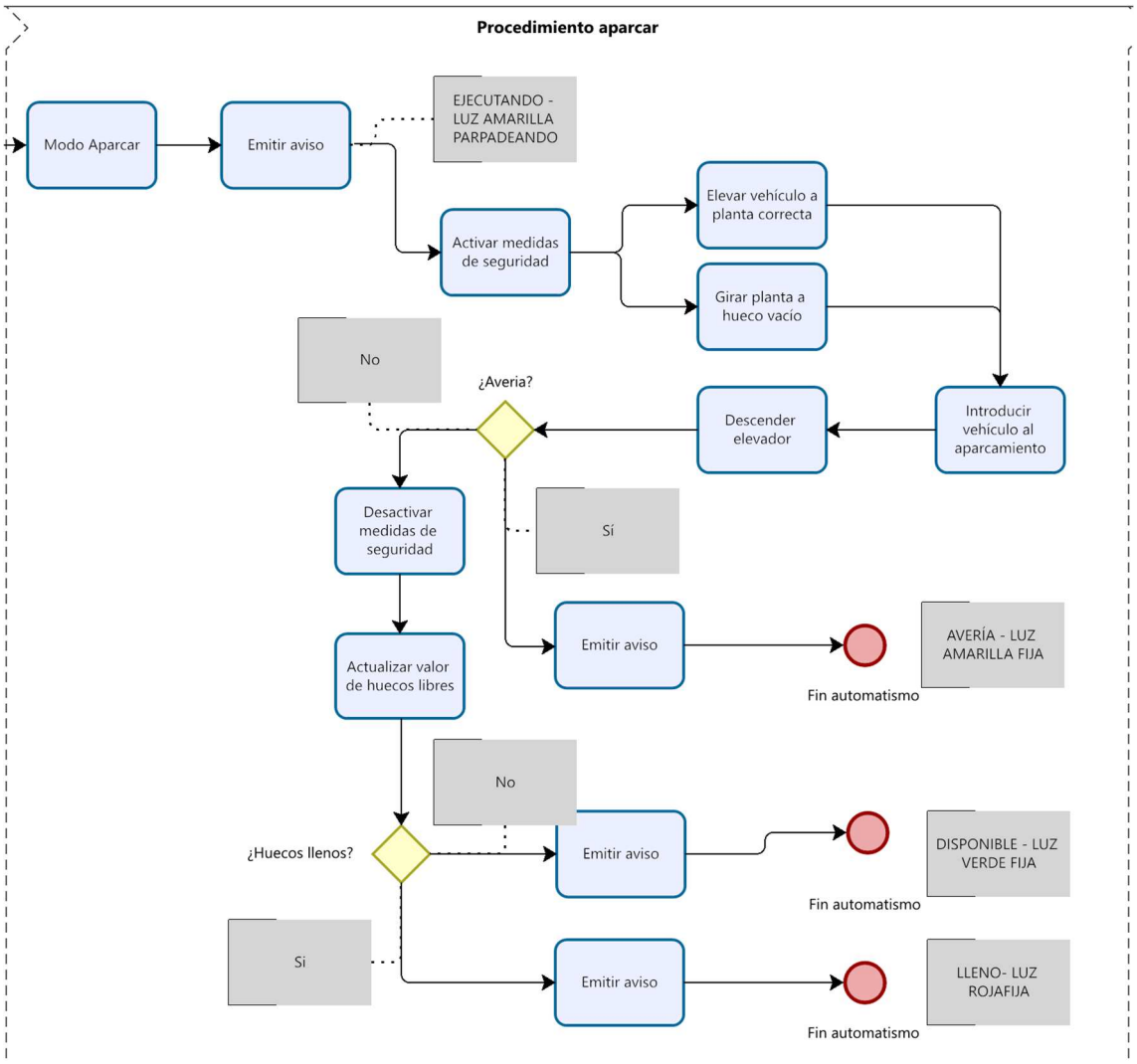


Figura 34. Tercera parte del esquema del automatismo realizado con Bizagi.

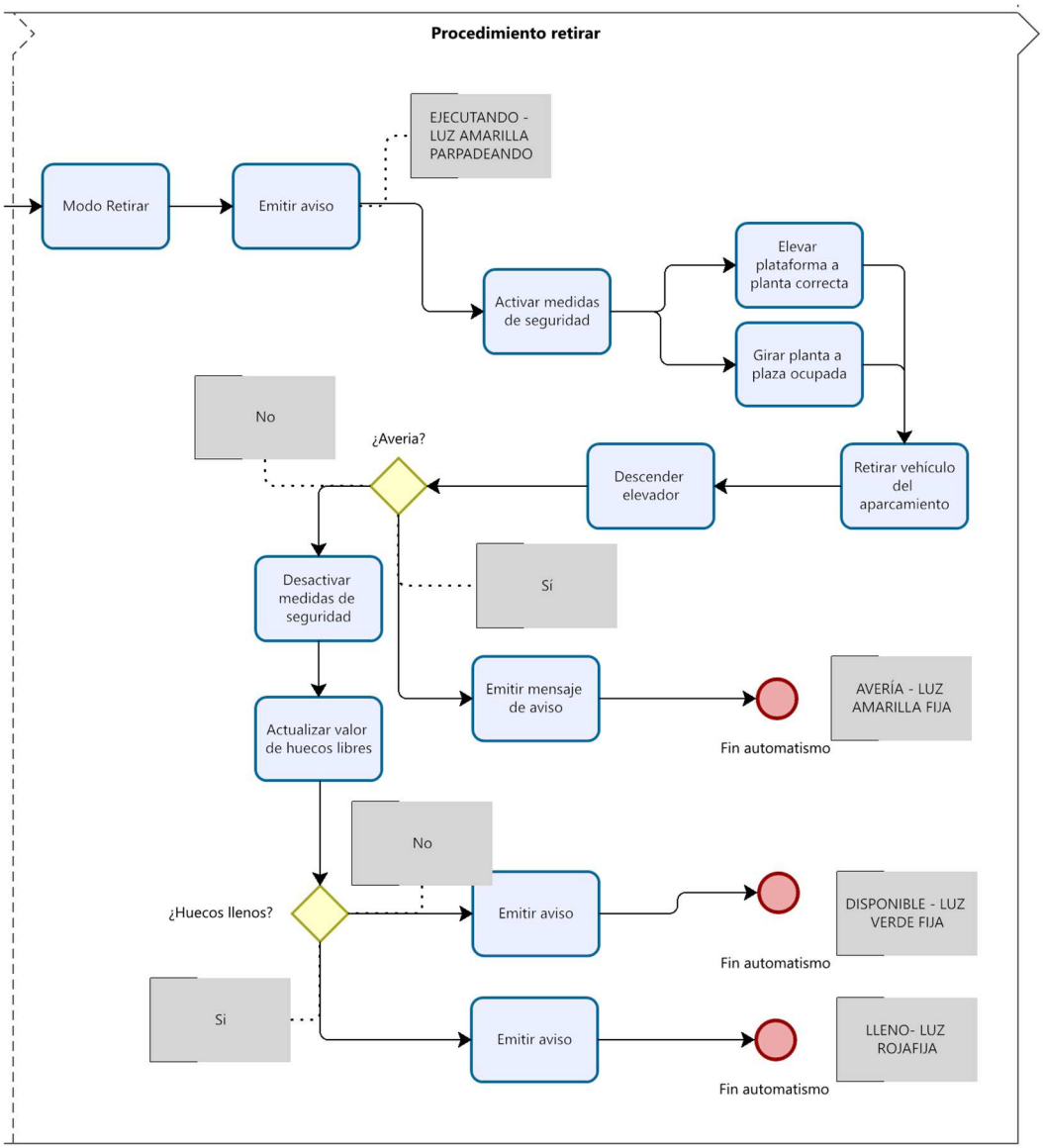


Figura 35. Cuarta parte del esquema del automatismo realizado con Bizagi.

CAPÍTULO IV

PERIFÉRICOS DE LA TARJETA MINI-DK2

4. PERIFÉRICOS DE LA TARJETA MINI-DK2

En este apartado se estudian tres tipos de subsistemas usados en el aparcamiento: sistema basado en PLC's (para el control de la plataforma elevadora, estructura rotativa y plataforma de desplazamiento lineal), sistema de detección de vehículos y sistema de seguridad.

Estos subsistemas son difíciles de integrar en el software principal debido a su complejidad, por lo que se opta por controlarlos de otro modo y conectarlos a este.

4.1. PLATAFORMA ELEVADORA

Este subsistema es el encargado de desplaza la plataforma elevadora, la cual sitúa el vehículo a la altura correspondiente. Será controlado por un autómata programable o PLC que recibirá la orden del microcontrolador principal, ayudándose de varios sensores dispuestos a lo largo del aparcamiento.

El estudio consta de cuatro partes: la parte de control, la parte de conexionado eléctrico, la estructura mecánica y las características eléctricas y mecánicas del motor elevador.

4.1.1. SISTEMA DE CONTROL DE LA PLATAFORMA ELEVADORA

El sistema de control del elevador se llevará a cabo mediante un PLC, el cual será el encargado de situar a la plataforma elevadora en la planta correspondiente con gran precisión.

El microcontrolador **LPC1768** enviará al PLC de esta plataforma elevadora el número de planta a la que debe situarse, lo hará mediante cuatro bits, el autómata recibirá la petición y el motor de la plataforma elevadora iniciará el ascenso.

Los cuatro bits combinados en lenguaje binario representarán mediante la siguiente tabla de verdad (tabla 1) el funcionamiento del elevador.

Tabla 1. Función del PLC según las entradas que recibe.

ENTRADAS					FUNCIÓN
E	**PLANTA [3]**	**PLANTA [2]**	**PLANTA [1]**	**PLANTA [0]**	
0	X	X	X	X	No tener en cuenta
1	0	0	0	0	Mover a planta 1
1	0	0	0	1	Mover a planta 2
1	0	0	1	0	Mover a planta 3
1	0	0	1	1	Mover a planta 4
1	0	1	0	0	Mover a planta 5
1	0	1	0	1	Mover a planta 6
1	0	1	1	0	Mover a planta 7
1	0	1	1	1	Mover a planta 8
1	1	0	0	0	Mover a planta 9
1	1	0	0	1	Mover a planta 10
1	1	0	1	0	No tener en cuenta
1	1	0	1	1	No tener en cuenta
1	1	1	0	0	No tener en cuenta
1	1	1	0	1	No tener en cuenta
1	1	1	1	0	No tener en cuenta
1	1	1	1	1	No tener en cuenta

Las cuatro entradas **PLANTA[x]** indica la planta a la que la plataforma elevadora se tendrá que mover. La entrada **E** se corresponde con la señal de validación o *enable*, pues el sistema no debe leer las entradas hasta que reciba la señal de validación.

Una vez se ha recibido la altura a la que se debe de situar, el elevador iniciará el ascenso. Este dispondrá de tres velocidades que dotarán al aparcamiento de rapidez, precisión y bajo coste, pues desplazarse en tres velocidades permite poder ascender a gran velocidad y detenerse lentamente para lograr dicha precisión, y arrancar a velocidad baja evita generar picos de intensidad que requieran de conductores y controladores que soporten dichas corrientes.

Para poder situar la planta y controlar las velocidades se instalarán tres sensores inductivos para cada planta ("**A_N**", "**B_N**" y "**C_N**"), donde N significa el número de planta siendo 0 la planta de recepción y de 1 a 10 las plantas de almacenaje de vehículos. Al aparcar o retirar el vehículo la plataforma partirá desde la planta de recepción, la plataforma será detectada por los tres sensores de dicha planta por lo que arrancará en velocidad baja, inmediatamente dejará de detectar "**C_0**", al ir ascendiendo "**B_0**" no detectará y el controlador pondrá la velocidad media, por último, al no detectar el sensor "**A_0**" el elevador llegará a velocidad máxima, de igual modo, al llegar a la planta en concreto la velocidad irá disminuyendo siendo "**C_N**" el primer sensor en detectar, al llegar a "**A_N**" el automatismo se detendrá y se iniciará el proceso de aparcamiento o retirada. En la figura 36 se puede observar la distribución de dichos sensores, al igual que las diferentes partes del aparcamiento.

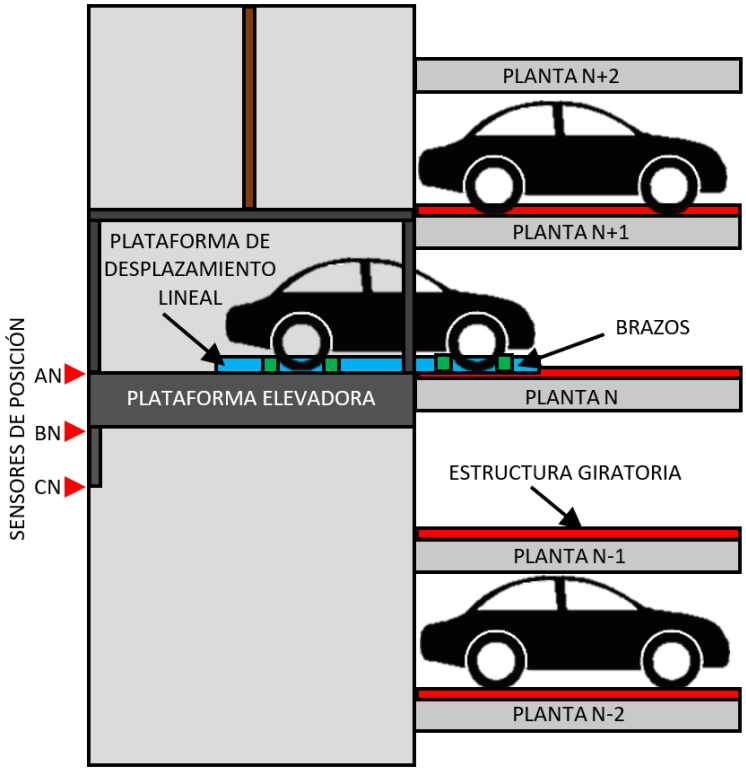


Figura 36. Disposición de los sensores inductivos en la plataforma elevadora.

Con esta configuración la planta de recepción requiere de una planta inferior (foso) para poder albergar los sensores que detengan la plataforma en la recepción, al igual, la planta superior a esta no tendrá acceso, pues en la recepción no se instalarán sensores. Esta planta superior será la encargada de albergar los sistemas eléctricos y de control, lo que no requiere acceso.

Una vez la plataforma se sitúe a la altura requerida se leerá el estado del sensor de detección de vehículo para determinar si la plataforma de desplazamiento lineal debe extraer o aparcar el vehículo y se le enviará dicha orden. Una vez dicha plataforma de desplazamiento lineal ha concluido su tarea el elevador descenderá hasta la planta baja o recepción y enviará al microcontrolador principal (**LPC1768**) una respuesta sobre si la ejecución ha sido exitosa o si se ha producido una avería.

Dicha respuesta se observa en la figura 37 y debe seguir las siguientes premisas:

- Debe tener dos modos, uno para representar la ejecución correcta y otra para representar avería en el elevador.
- La frecuencia de la señal correcta debe ser el doble a la frecuencia de lectura del microcontrolador.
- La frecuencia de la señal avería debe ser cuatro veces mayor a la frecuencia de lectura del microcontrolador.
- El periodo de lectura de la respuesta del microcontrolador será: 0,1s, 0,2s, 0.5s, 1s o 2s.

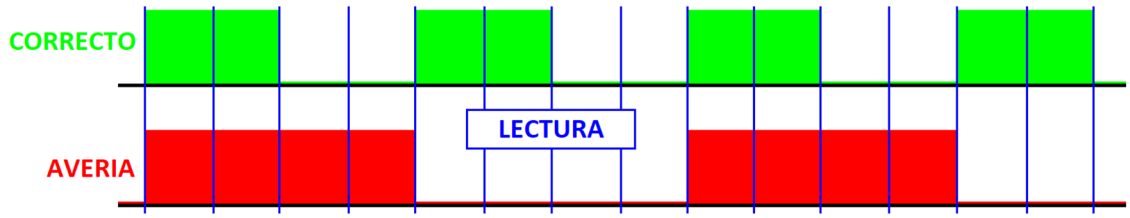


Figura 37. Representación de la señal de respuesta del PLC.

4.1.2. CONEXIONADO ELÉCTRICO

El PLC de la plataforma elevadora deberá recoger la señales de activación proveniente del microcontrolador, la señal respuesta de la estructura rotativa y plataforma de desplazamiento lineal, la lectura del sensor de detección de vehículo y las señales de control de los sensores inductivos. Estas últimas ascienden a 33 en el caso de haber 10 plantas, además debe enviar una señal de respuesta al microcontrolador, las señales de activación y dirección de dicho motor y dos señales para activar la plataforma de desplazamiento lineal. Ningún autómata dispone de 38 entradas y 5 salidas por lo que se opta por recudir el número de las entradas de los sensores mediante un control externo. En la siguiente imagen (figura 38) se representa todo el conexionado que debe de tener el sistema de control del motor elevador.

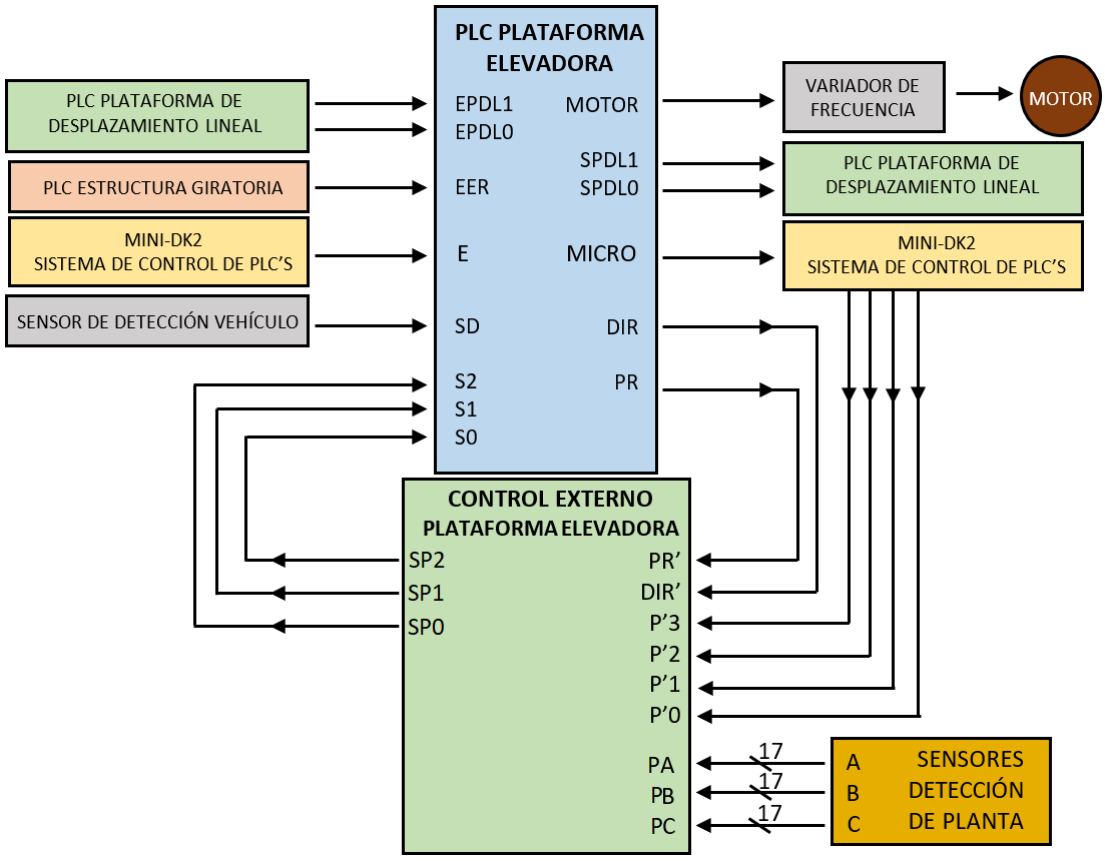


Figura 38. Esquema del conexionado del sistema de control del motor elevador.

El control externo funciona como un multiplexor de 17 entradas y una salida, ambas de 3 canales, con una selección de cinco bits, siendo cuatro de ellos la planta (**P'x**) en la que se debe situar y el quinto si se debe situar o no en la planta de reposo (**Pr'**).

Con esto se consigue reducir a tres entradas y dos salida para poder controlar los sensores, recibiendo por **S0**, **S1** y **S2** el estado de los sensores de la planta seleccionada por **P3**, **P2**, **P1** y **P0**. Al activarse **Pr** se recibirá el valor de los sensores de la planta de reposo y con el bit **DIR** se podrá preparar al PLC para leer sensores de plantas subterráneas.

En las siguientes tablas (tabla 2 y 3) de verdad se muestra el funcionamiento de dicho sistema de control:

Tabla 2. Obtención de la lectura del sensor inductivo dependiendo de la planta.

Pr'	P'3	P'2	P'1	P'0	SP2	SP1	SP0	FUNCIÓN
0	0	0	0	0	A1	B1	C1	PRIMERA PLANTA
0	0	0	0	1	A2	B2	C2	SEGUNDA PLANTA
0	0	0	1	0	A3	B3	C3	TERCERA PLANTA
0	0	0	1	1	A4	B4	C4	CUARTA PLANTA
0	0	1	0	0	A5	B5	C5	QUINTA PLANTA
0	0	1	0	1	A6	B6	C6	SEXTA PLANTA
0	0	1	1	0	A7	B7	C7	SÉPTIMA PLANTA
0	0	1	1	1	A8	B8	C8	OCTAVA PLANTA
0	1	0	0	0	A9	B9	C9	NOVENA PLANTA
0	1	0	0	1	A10	B10	C10	DÉCIMA PLANTA
0	1	0	1	0	-	-	-	NO SE TIENE EN CUENTA
0	1	0	1	1	-	-	-	NO SE TIENE EN CUENTA
0	1	1	0	0	-	-	-	NO SE TIENE EN CUENTA
0	1	1	0	1	-	-	-	NO SE TIENE EN CUENTA
0	1	1	1	0	-	-	-	NO SE TIENE EN CUENTA
0	1	1	1	1	-	-	-	NO SE TIENE EN CUENTA
1	X	X	X	X	Ar	Br	Cr	PLANTA DE RECEPCIÓN

Tabla 3. Velocidad del elevador según la lectura del sensor inductor y la dirección de movimiento.

	DIR	S2 - PA	S1 - PB	S0 - PC	FUNCIÓN
DIR= 0 → ASCENDER	0	0	0	0	ALTA VELOCIDAD
	0	0	0	1	MEDIA VELOCIDAD
	0	0	1	0	PARO POR FALLO → IMPOSIBLE
	0	0	1	1	BAJA VELOCIDAD
	0	1	0	0	PARO POR FALLO → IMPOSIBLE
	0	1	0	1	PARO POR FALLO → IMPOSIBLE
	0	1	1	0	PARO POR FALLO → IMPOSIBLE
	0	1	1	1	PARO POR LLEGADA
DIR = 1 → DESCENDER	1	0	0	0	ALTA VELOCIDAD
	1	0	0	1	PARO POR FALLO → IMPOSIBLE
	1	0	1	0	PARO POR FALLO → IMPOSIBLE
	1	0	1	1	PARO POR FALLO → IMPOSIBLE
	1	1	0	0	MEDIA VELOCIDAD
	1	1	0	1	PARO POR FALLO → IMPOSIBLE
	1	1	1	0	BAJA VELOCIDAD
	1	1	1	1	PARO POR LLEGADA

Por tanto, el PLC requerirá de ocho entradas y seis salidas. La entrada **E** entrega la validación o *enable* desde el microcontrolador, y a este se le enviará la señal de respuesta mediante la salida **MICRO**. Al control externo se enviará la dirección de movimiento mediante la salida **DIR**, la petición de leer los sensores de la planta de recepción mediante **PR** y se obtendrá la lectura de los sensores de la planta a la que se dirige mediante las entradas **S2**, **S1** y **S0**. El motor elevador se activará con la salida **MOTOR**.

Se enviará mediante las salidas **SPDL1 y SPDL0** información a la plataforma de desplazamiento lineal sobre si debe retirar (**SPDL0**=1) o aparcar (**SPDL1**=1) un vehículo, y se recibirá por **EPDL1** y **EPDL0** la respuesta de la plataforma de desplazamiento lineal, donde si **EPDL0**=1 signigica que la plataforma de desplazamiento lineal ha terminado y si **EPDL1**=1 signigica que ha ocurrido una avería. El PLC de la estructura rotativa también responderá, pues hasta que la planta no haya concluido el giro este no debe activar la plataforma de desplazamiento lineal, dicha señal se recogerá mediante **EER**.

El sensor de detección de vehículo será leído por la entrada **SD** para poder determianar si la función de la plataforma de desplazamiento lineal es introducir (**SD**=1, se detecta) o retirar el vehículo (**SD**=0, no se detecta).

En la figura 39 se observa el diagrama de bloques simplificado de dicho autómata.

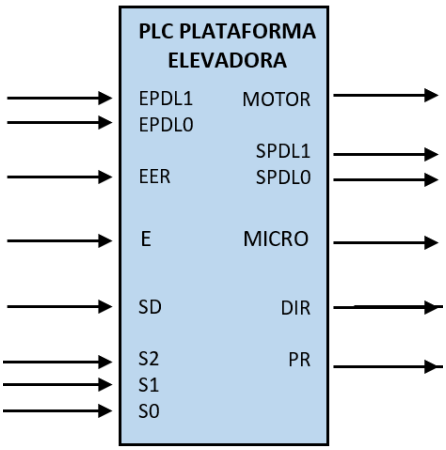


Figura 39. Representación de las entradas y salidas del PLC elevador.

4.1.3. ESTRUCTURA MECÁNICA

Para efectuar el movimiento de la plataforma elevadora se necesita un sistema mecánico que convierta el par de rotación del motor en movimiento lineal. Diseñar dicho sistema mecánico requiere conocer los requerimientos y limitaciones.

El subsistema debe aguantar el peso de la plataforma y del vehículo, debe moverse a la máxima velocidad posible y debe de ser fiable. Para ello se aportan una posible solución que se basa en el desplazamiento por poleas o polipastos. Según el Real Decreto 818/2009 la masa máxima de un turismo será de 3500 kg, si además tomamos el peso de la plataforma como 1000 kg obtenemos un peso de 4500 kg, es decir, la plataforma ejerce aproximadamente 45 kN de fuerza.

$$\boldsymbol{P_{max} = M_{max} \cdot \omega_{max}}$$

$$\boldsymbol{M_{max} = F_{max} \cdot r}$$

$$\boldsymbol{\omega_{max} = \frac{v_{max}}{r}}$$

Donde P_{max} es la potencia máxima que ejerce la plataforna, M_{max} el par de giro máximo que ejerce, ω_{max} la velocidad angular máxima, F_{max} la fuerza máxima ejercida por la plataforma, v_{max} la velocidad máxima alcanzada y r la distancia entre el eje de rotación y el pundo de aplicación de la fuerza.

$$\boldsymbol{P_{max} = F_{max} \cdot V_{max}}$$

Si suponemos que la velocidad máxima de ascensión de la plataforma es de 0,4 m/s, es decir, recorrerá 50 metros en 125 segundos, aproximadamente dos minutos, obtenemos una potencia de 18 kW. Para poder reducir más la potencia del motor se doratá al sistema de elevación de un polipasto reductor (figura 40) 4 a 1, es decir, el polipasto será capaz de recudir el peso de la plataforma elevadora a un cuatro. En condiciones reales esta relación disminuirá obteniendo una aproximación de un poco menos de un tercio de dicho peso.

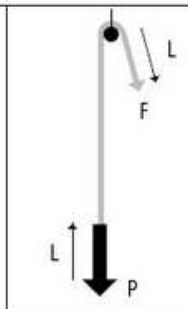
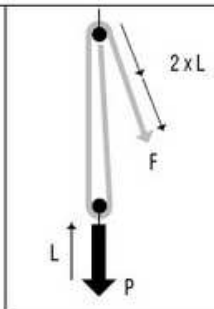
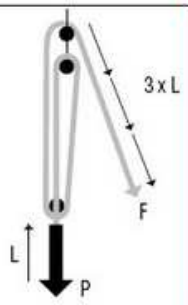
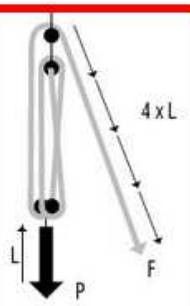
Eficiencias				
Fuerza teórica	F = P	F = 0,5 x P	F = 0,33 x P	F = 0,25 x P
Fuerza real	F = 1,05 x P	F = 0,54 x P	F = 0,37 x P	F = 0,3 x P
Longitud de cuerda estirada	1 vez la longitud recorridad por el peso	2 veces la longitud recorridad por el peso	3 veces la longitud recorridad por el peso	4 veces la longitud recorridad por el peso

Figura 40. Ejemplos de polipastos respecto a su fuerza teórica, real y longitud de cuerda estirada. [6]

Escogiendo este último polipasto obtendremos un peso de 1350kg, es decir, la fuerza máxima será de 13,5kN aproximadamente, y una potencia de 5,4kW.

El radio de las poleas no será un problema, pues dicho sistema de elevación estará dispuesto en la planta denominada como la azotea del edificio, es decir, entre el techo y la última planta de aparcamiento.

La figura 41 muestra un boceto representativo del sistema de elevación.

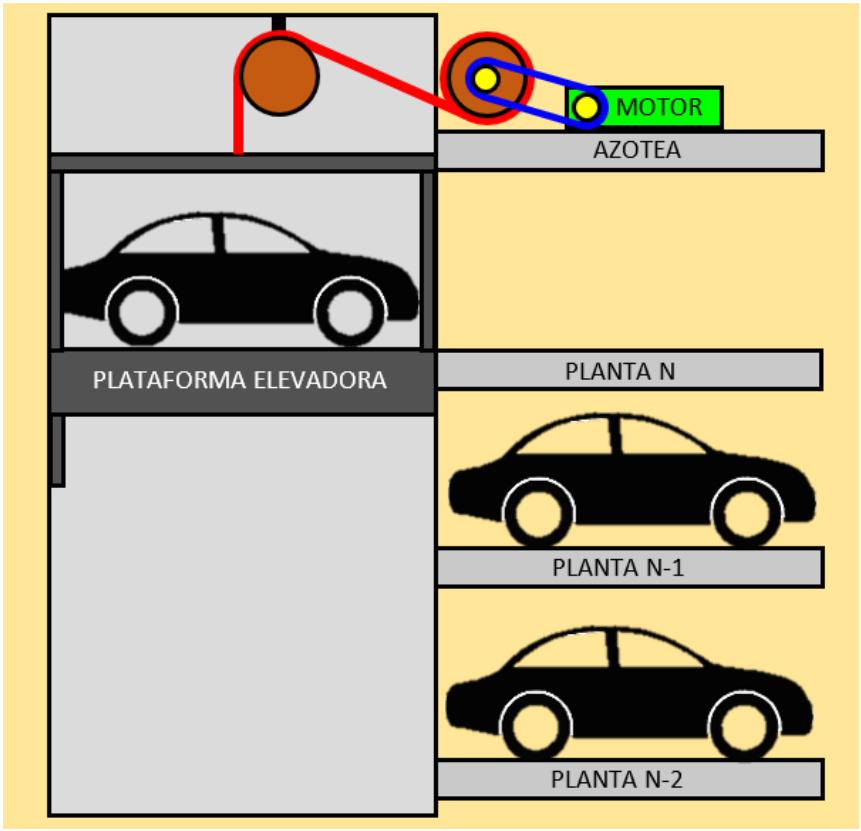


Figura 41. Representación del sistema de elevación.

4.1.4. CARACTERÍSTICAS ELÉCTRICAS Y MECÁNICAS DEL MOTOR ELEVADOR

La siguiente etapa para diseñar el subsistema es conocer las características de potencia del motor, es decir, se elige si será una máquina de corriente continua o alterna, trabajará en monofásica o en trifásica, y sus tipos de motor y arranque

Los motores usados en este subsistemas requieren de gran potencia por lo que motores trifásicos es la mejor opción, ya que los motores monofásicos o de corriente continua no son suficientemente potentes y económicos.

Los motores trifásicos se clasifican, entre otros, en máquinas síncronas o asíncronas (de inducción), las diferencias más notables son que los motores síncronos trabajarán a la velocidad de sincronismo, independientemente de la carga aplicada, en cambio, el motor asíncrono trabajará por debajo de esa velocidad de sincronismo, además, los motores síncronos necesitan una etapa de sincronización de frecuencia en el arranque. En cuanto al precio también notamos una diferencia y es que los motores asíncronos son más baratos. Normalmente se utilizan las máquinas asíncronos en aplicaciones industriales como motores y las máquinas síncronas en generadores como alternadores.

Por tanto, se usan motores trifásicos asíncronos por su gran potencia y versatilidad y reducido precio.

Respecto al arranque del motor, si un motor se alimenta directamente de la instalación eléctrica, este genera un pico de corriente de hasta siete veces la intensidad nominal que puede dañar al sistema, la instalación eléctrica, incluso al propio motor. Por ello, la ley española en el REBT limita en motores de corriente alterna la relación de la intensidad de arranque y plena carga según la tabla siguiente (tabla 4):

Tabla 4. REBT ITC 47.

POTENCIA NOMINAL DEL MOTOR DE CORRIENTE ALTERNA	CONSTANTE MÁXIMA DE PROPORCIONALIDAD ENTRE LA INTENSIDAD DE LA CORRIENTE DE ARRANQUE Y DE LA DE PLENA CARGA
De 0,75 kW a 1,5 kW	4,5
De 1,5 kW a 5,0 kW	3,0
De 5,0 kW a 15,0 kW	2,0
De más de 15,0 kW	1,5

Se buscan métodos de arranque alternativos al arranque directo para cumplir con el reglamento y para proteger la instalación:

El arranque **estrella-triángulo** consiste en conectar mediante relés y contactores el motor en modo estrella y antes de llegar a la velocidad de sincronismo cambiar a modo triángulo, con el que se continuará hasta el paro.

Este método consigue decrementar el pico de intensidad en el arranque, con la desventaja de una pérdida de potencia inicialmente, pues el modo estrella conectaremos el motor con la tensión de fase a 230V y no con la tensión de línea a 400V.

Respecto a la complejidad de diseño e instalación, es una instalación barata y un poco compleja, pues necesitaremos cablear diferentes relés y contactores, siendo casi nulo el control que se puede hacer sobre el motor.

El arranque **variando la frecuencia** consiste en utilizar un variador de frecuencia para conseguir que el motor incremente la velocidad poco a poco, hasta llegar a la velocidad esperada.

Este método consigue reducir el pico de intensidad completamente ya que aumentará la velocidad suavemente, mediante el incremento de frecuencia, generando una rampa de intensidad que termina con la corriente nominal.

Su instalación es bastante más sencilla que en el método anterior, pues solo necesitaremos el variador y un sistema de control si queremos controlarlo externamente, pues el variador permite un control detallado y preciso del motor. La desventaja es la dificultad a la hora de programar el variador.

El concepto de arranque con **arrancador suave** es parecido al del variador de frecuencia, estando más limitado, pues, aunque la instalación sea bastante más barata y sencilla que la del variador, el control que aporta este es muy reducido, pudiendo configurar el tiempo de duración de la rampa principalmente, la corriente en el arranque seguirá siendo superior a lo obtenido con el variador de frecuencia.

El arranque por **resistencias estatóricas o rotóricas** solo se puede aplicar en motores de rotor bobinado, pues consiste en variar la resistencia del rotor para conseguir aumentar el par de arranque. Realmente este sistema no garantiza que la intensidad de arranque se reduzca lo suficiente por lo que no lo usaremos.

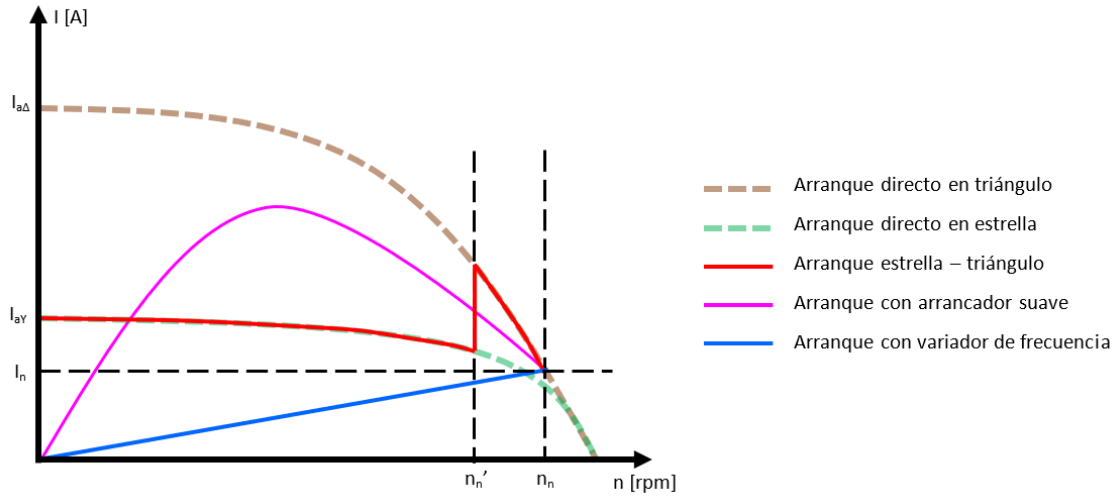


Figura 42. Curva aproximada de corriente frente a velocidad para los distintos tipos de arranque.

En el gráfico (figura 42) se puede observar de manera aproximada la relación entre la intensidad y la velocidad de los diferentes tipos de arranque de motores de corriente alterna trifásicos.

ELECCIÓN DEL MOTOR

Las conclusiones obtenidas son que el motor debe ser capaz de entregar una potencia máxima de 5,4 kW, debe ser trifásico y asíncrono, y dispondrá de variador de frecuencia.

La velocidad a la que debe ascender el elevador será de 0,4 metros por segundo, los motores asíncronos de cuatro polos suelen rotar a 1450 revoluciones por minuto, por lo que se debe escoger el radio de las poleas para poder obtener la relación de transmisión y poder escoger una caja reductora correctamente.

Se dispone de tres poleas (A, B y C) (figura 43), donde “A” y “B” serán del mismo tamaño. Por lo que la relación de transmisión será la siguiente.

$$\boldsymbol{P_{max} = F_{max} \cdot V_{max}}$$

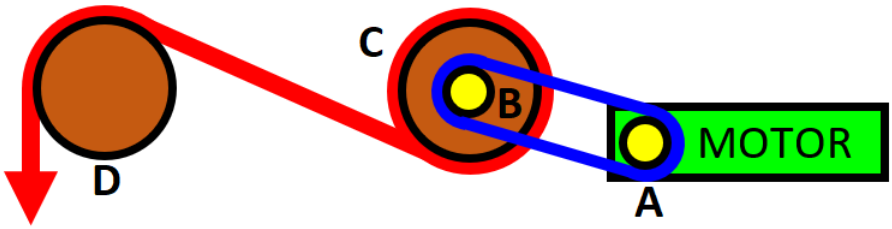


Figura 43. Sistema de poleas en el sistema elevador.

Pues la polea motriz será “A” y moverá a “B” con la misma velocidad, pues los radios son iguales. La polea “B” estará fija al mismo eje que “C”, por lo que la velocidad angular de “A” se transmitirá a “C”.

$$v = \omega \cdot r$$

$$\omega_A = \omega_B = \omega_C = \omega_{REDUCTORA}$$

$$v_{RECOGIDA} = \omega_{REDUCTORA} \cdot r_C$$

La polea “C” recogerá la cuerda que elevará la plataforma y “D” representará el sistema de polipasto escogido anteriormente. Dicho polipasto recude el peso de la carga hasta cuatro veces a cambio de tener que recoger cuatro veces más de cuerda que de ascensión, por lo que la velocidad de recogida será cuatro veces menor a la velocidad de ascenso.

$$v_{RECOGIDA} = 4 \cdot v_{ASCENSO}$$

$$v_{ASCENSO} = 0.25 \cdot \omega_{REDUCTORA} \cdot r_C$$

La velocidad angular de la reductora debe ser lo máximo posible para disminuir la relación de reducción de dicho sistema, además se requiere una velocidad de ascenso de 0,4 metros por segundo se requerirá una polea demasiado pequeña, por lo que se fija su radio a 15 centímetros, obteniendo una velocidad angular de la reductora de 100 revoluciones por minuto.

$$\omega_{REDUCTORA} = \frac{4 \cdot v_{ASCENSO}}{r_C} = 4 \cdot \frac{0,4\ [m/s]}{0,15} = 10,66\ rad/s = 102\ rpm$$

La relación de reducción debe de ser de 14,5, pues la velocidad del motor asíncrono de cuatro polos suele ser de 1450 rpm.

En el catálogo de motorreductores de JALMAC (figura 44) se encuentra un motor con caja reductora de modelo **MSF130** que puede acoplarse con bastante precisión a los requerimientos, pues cuenta con una potencia de 5,5kW y una velocidad reducida de 94 rpm, lo que supondría una velocidad de ascenso de 0,37 metros por segundo, es decir, asciende 50 metros en 135 segundos o 2,25 minutos.

Motor Kw		n2 rpm	i	M2 Nm	f.s	Tipo Type
5.50	4P n1=1400	186	7.5	253	1.9	MSF 110
		140	10	334	1.6	
		94	15	484	1.2	
		70	20	638	0.9	
		186	7.5	256	3.0	MSF 130
		140	10	334	2.5	
		94	15	490	1.9	
		70	20	645	1.4	
		56	25	788	1.2	
		47	30	900	1.2	
		35	40	1171	0.9	

Figura 44. Extracto del catálogo de motorreductores del grupo JALMAC. [8]

4.2. PLATAFORMA DE DESPLAZAMIENTO LINEAL

Este subsistema es el encargado de introducir el vehículo en la plaza o de retirarlo hacia la plataforma elevadora. Será controlado por un autómata programable o PLC que recibirá la orden del PLC de la plataforma elevadora.

El estudio consta de dos partes: sistema de control y conexionado eléctrico y estructura mecánica.

4.2.1. SISTEMA DE CONTROL Y CONEXIONADO DE LA PLATAFORMA DE DESPLAZAMIENTO LINEAL

El sistema de control de la plataforma de desplazamiento lineal se llevará a cabo mediante un PLC al igual que en la plataforma elevadora. Este autómata será el encargado de controlar el motor que desplace la plataforma linealmente e introduzca o retire los vehículos.

El PLC de la plataforma elevadora enviará al sistema la orden de aparcar mediante la entrada **APARCAR** o la orden de retirar con **RETIRAR**. Una vez la plataforma de desplazamiento lineal haya terminado la ejecución enviará al PLC del elevador una respuesta basada en una salida a nivel alto mediante la **EXITO** si ha sido exitoso o **ERROR** si se ha producido una avería.

Los dos bits combinados en lenguaje binario representarán mediante la siguiente tabla de verdad (tabla 5) el funcionamiento de la plataforma de desplazamiento lineal.

Tabla 5. Función del PLC según las entradas que recibe.

ENTRADAS		FUNCIÓN
RETIRAR	APARCAR	
0	X	No tener en cuenta
1	0	Retirar del aparcamiento
1	1	Introducir en el aparcamiento

La plataforma de desplazamiento lineal necesitará de dos finales de carrera para poder detectar cuando concluye con el desplazamiento, dichos finales de carrera se conectarán a las entradas **FC0** y **FC1**. De igual modo el sistema enviará por **SB** la señal para activar los brazos y leerá por **EB** si los brazos se han ajustado a las ruedas. Mediante la salida **MOTOR** y **DIR** se controlará el avance o retroceso de la plataforma. El conexionado de este PLC se puede observar en la siguiente figura (figura 45).

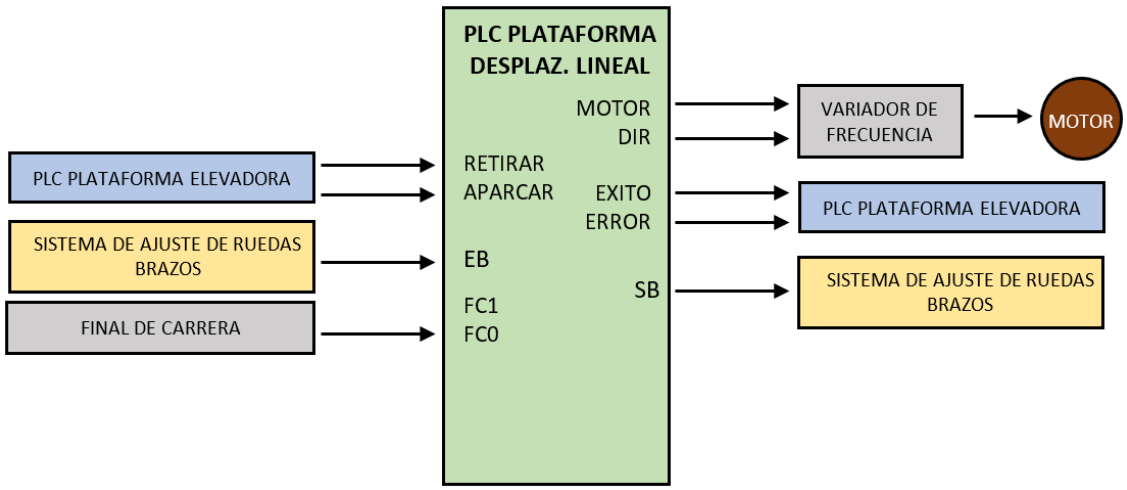


Figura 45. Representación del conexionado del PLC de la plataforma de desplazamiento lineal.

4.2.2. ESTRUCTURA MECÁNICA

La plataforma de desplazamiento lineal (figura 46) deberá mover el vehículo que estará estacionado con el freno de mano, para ello cuenta con ocho brazos móviles que se replegarán y ajustarán a la posición de las ruedas, una vez estén correctamente situados se aproximarán para conseguir elevar el vehículo un par de centímetros, lo suficiente para poder desplazar el vehículo sin necesidad de hacer rodar las ruedas (figura 47).

Una vez se ha levantado el vehículo, la plataforma de desplazamiento lineal se desplazará, soportando dicho peso, hasta la plaza libre donde separará los brazos con cuidado para que las ruedas toquen el suelo y acto seguido se plegarán los brazos y la plataforma de desplazamiento lineal volverá al interior de la plataforma elevadora.

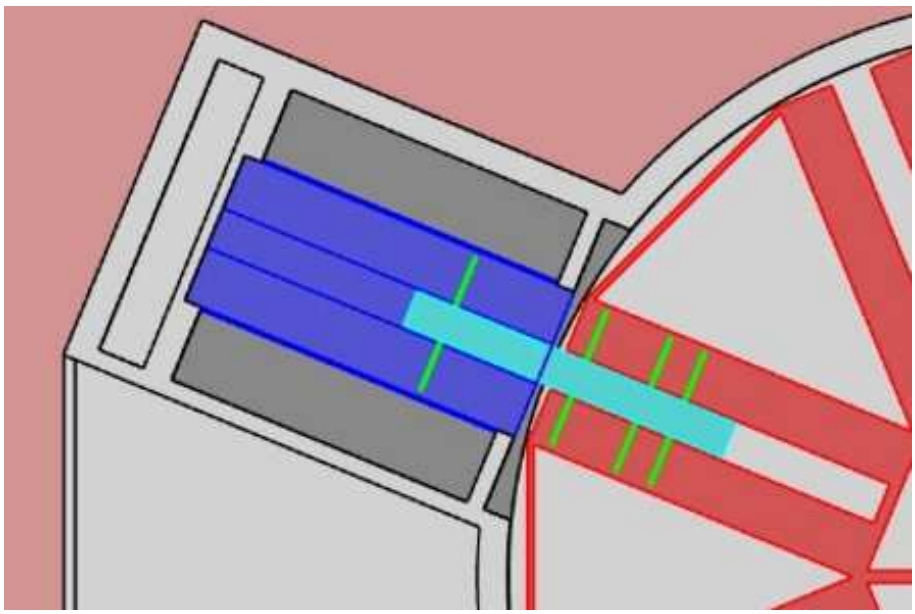


Figura 46. Representación del desplazamiento de la plataforma de desplazamiento lineal.

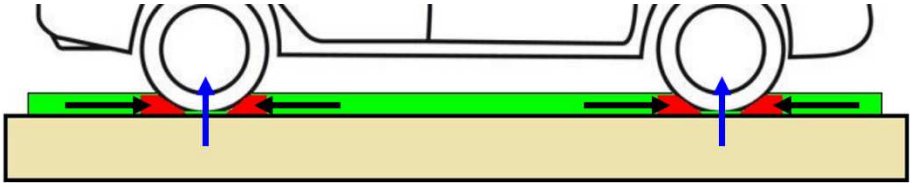


Figura 47. Representación de la elevación del vehículo con la plataforma de desplazamiento lineal.

4.3. ESTRUCTURA ROTATIVA

Este subsistema es el encargado de rotar la planta para situar la plaza libre enfrente al elevador que contiene el vehículo. Será controlado por un autómata programable o PLC que recibirá la orden del microcontrolador principal (**LCP1768**).

El estudio consta de tres partes: el sistema de control, el conexionado eléctrico, y la estructura mecánica.

4.3.1. SISTEMA DE CONTROL DE LA ESTRUCTURA ROTATIVA

El sistema de control de la estructura rotativa se llevará a cabo mediante un PLC, el cual será el encargado de situar la plaza o división de la planta enfrente a la plataforma elevadora con gran precisión.

El microcontrolador **LPC1768** enviará al PLC el número de elevador y la plaza o la división que debe enfrentar, lo hará mediante ocho bits, tres para escoger el elevador, cuatro para escoger la división y la habilitación o *enable*.

Los tres primeros bits (**DIVISION**) y el último (**E**) combinados en lenguaje binario representarán mediante la siguiente tabla de verdad (tabla 6)la plaza que debe enfrentar.

Tabla 6. Elección de la división en la planta según las entradas que el PLC recibe.

ENTRADAS					FUNCIÓN
E	**DIVISION [3]**	**DIVISION [2]**	**DIVISION [1]**	**DIVISION [0]**	
0	X	X	X	X	No tener en cuenta
1	0	0	0	0	División 1
1	0	0	0	1	División 2
1	0	0	1	0	División 3

1	0	0	1	1	División 4
1	0	1	0	0	División 5
1	0	1	0	1	División 6
1	0	1	1	0	División 7
1	0	1	1	1	División 8
1	1	0	0	0	No tener en cuenta
1	1	0	0	1	No tener en cuenta
1	1	0	1	0	No tener en cuenta
1	1	0	1	1	No tener en cuenta

Los tres siguientes bits y el último combinados en lenguaje binario representarán mediante la siguiente tabla de verdad (tabla 7) el elevador que debe enfrentar.

Tabla 7. Elección del elevador según las entradas que el PLC recibe.

ENTRADAS				**FUNCIÓN**
E	**ELEVADOR [2]**	**ELEVADOR [1]**	**ELEVADOR [0]**	
0	X	X	X	No tener en cuenta
1	0	0	0	Elevador 1
1	0	0	1	Elevador 2
1	0	1	0	Elevador 3
1	0	1	1	Elevador 4
1	1	0	0	No tener en cuenta
1	1	0	1	No tener en cuenta
1	1	1	0	No tener en cuenta
1	1	1	1	No tener en cuenta

Las cuatro entradas **DIVISION[x]** indica la plaza y las tres entradas **ELEVADOR[x]** indican el elevador que tendrá que enfrentar. La entrada **E** se corresponde con la señal de validación o *enable*, pues el sistema no debe leer las entradas hasta que reciba la señal de validación.

Una vez se ha recibido la plaza que debe enfrentar con el elevador el sistema iniciará la rotación. La plata dispondrá de tres velocidades al igual que la plataforma elevadora.

Para poder situar la planta y controlar las velocidades se instalarán tres sensores inductivos para cada elevador ("**A_N**", "**B_N**" y "**C_N**") y otros tres por plaza ("**A'_M**", "**B'_M**" y "**C'_M**"), dependiendo del sentido de giro el automatismo irá aumentando o disminuyendo la velocidad al igual que ocurría en la plataforma elevadora.

Una vez la planta se ha enfrentado al elevador el autómata enviará al PLC del elevador un aviso de la finalización del proceso, para poder accionar la plataforma de desplazamiento lineal, y al microcontrolador principal (**LPC1768**) una respuesta sobre si la ejecución ha sido exitosa o si se ha producido una avería.

Dicha respuesta debe seguir las siguientes premisas:

- Debe tener dos modos, uno para representar la ejecución correcta y otra para representar avería en el elevador.
- La frecuencia de la señal correcta debe ser el doble a la frecuencia de lectura del microcontrolador.
- La frecuencia de la señal avería debe ser cuatro veces mayor a la frecuencia de lectura del microcontrolador.
- El periodo de lectura de la respuesta del microcontrolador será: 0.1s, 0.2s, 0.5s, 1s o 2s.

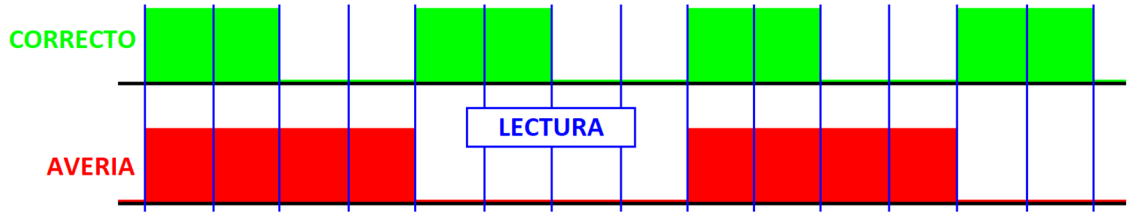


Figura 48. Representación de la señal de respuesta del PLC.

4.3.2. CONEXIONADO ELÉCTRICO

El autómata deberá recoger la señal de activación o *enable* proveniente del microcontrolador y las señales de control de los sensores inductivos. Estas últimas ascienden a 36 en el caso de haber 8 divisiones o plazas por planta y 4 elevadores, además debe controlar la marcha y sentido del motor y enviar una señal de aviso al PLC elevador. Ningún autómata dispone de 37 entradas y 6 salidas por lo que se opta por recudir el número de las entradas de los sensores mediante un control externo.

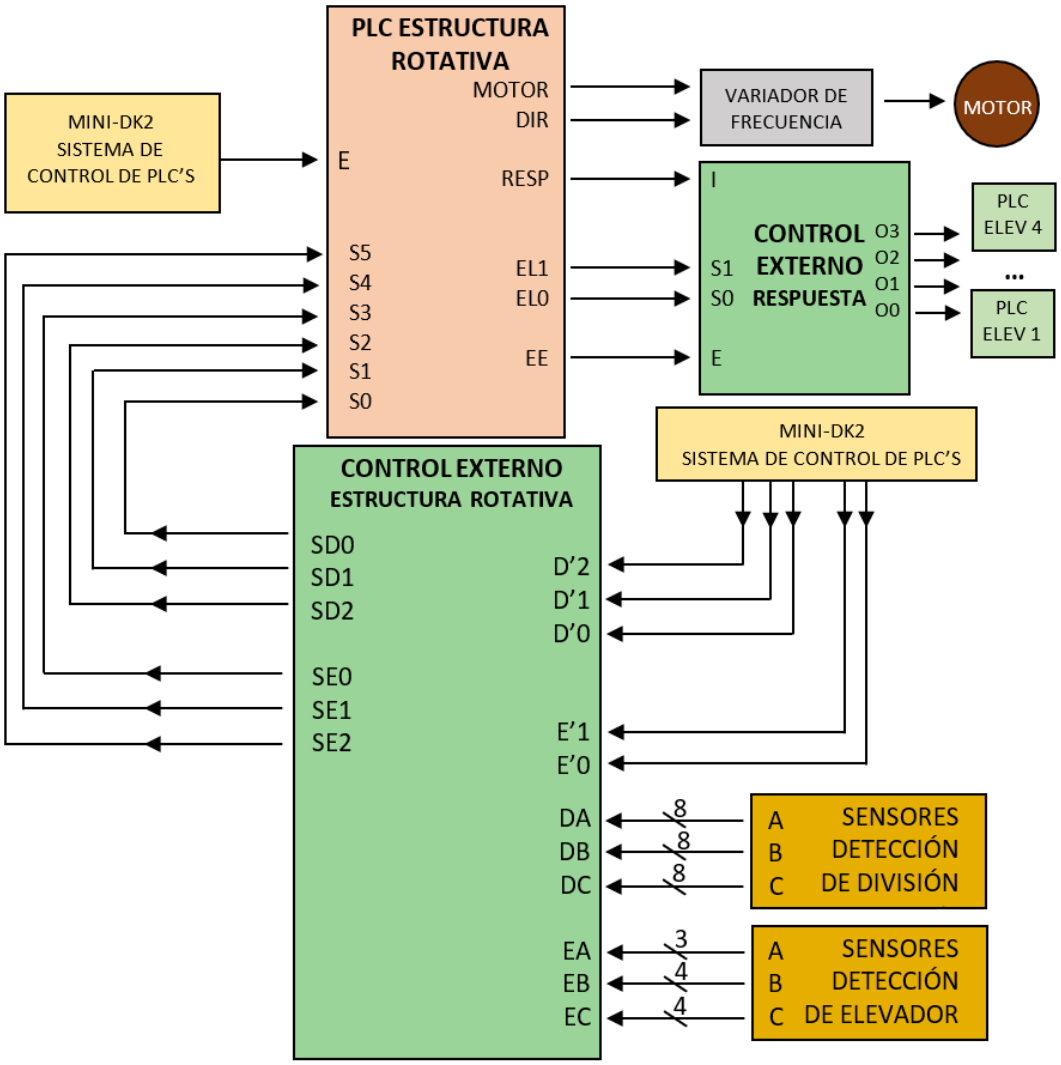


Figura 49. Representación del conexionado del PLC de la estructura rotativa.

Este control externo funcionará como un dos multiplexores, el primero controla los sensores de las divisiones o plazas de la estructura rotativa y el segundo controla los sensores de los cuatro elevadores. El primero de ellos es un multiplexor complejo de 8 entradas y una salida, ambas de 3 canales, con una selección de cuatro bits, siendo la división o plaza que se debe enfrentar.

El segundo es también un multiplexor complejo pero de 4 entradas y una salida, ambas de 3 canales, con una selección de dos bits, siendo el elevador que se debe enfrentar.

Con esto se consigue reducir a seis entradas para poder controlar los sensores, recibiendo por **S0**, **S1** y **S2** el estado de los sensores de las divisiones indicado por **D'2**, **D'1** y **D'0**, y por **S3**, **S4** y **S5** el estado de los sensores del elevador indicado por **E'1** y **E'0**.

En las siguientes tablas de verdad (tabla 8, 9 y 10) se muestra el funcionamiento de dicho sistema de control:

Tabla 8. Obtención de la lectura del sensor inductivo dependiendo de la división.

D'2	D'1	D'0	SD2	SD1	SD0	FUNCIÓN
0	0	0	DA1	DB1	DC1	DIVISIÓN 1
0	0	1	DA2	DB2	DC2	DIVISIÓN 2
0	1	0	DA3	DB3	DC3	DIVISIÓN 3
0	1	1	DA4	DB'4	DC4	DIVISIÓN 4
1	0	0	DA5	DB5	DC5	DIVISIÓN 5
1	0	1	DA6	DB6	DC6	DIVISIÓN 6
1	1	0	DA7	DB7	DC7	DIVISIÓN 7
1	1	1	DA8	DB8	DC8	DIVISIÓN 8

Tabla 9. Obtención de la lectura del sensor inductivo dependiendo del elevador.

E1	E0	R2	R1	R0	FUNCIÓN
0	0	A1	B1	C1	ELEVADOR 1
0	1	A2	B2	C2	ELEVADOR 2
1	0	A3	B3	C3	ELEVADOR 3
1	1	A4	B4	C4	ELEVADOR 4

Tabla 10. Velocidad del elevador según la lectura del sensor inductor y el sentido de giro.

	S5 - AC	S4 - EB	S3 - EA	S2 – DA	S1 – DB	S0 – DC	FUNCIÓN
D = 0 → ANTI-HORARIO	0	0	0	0	0	0	ALTA VELOCIDAD
	0	0	1	0	0	1	MEDIA VELOCIDAD
	0	1	0	0	1	0	PARO POR FALLO → IMPOSIBLE
	0	1	1	0	1	1	BAJA VELOCIDAD
	1	0	0	1	0	0	PARO POR FALLO → IMPOSIBLE
	1	0	1	1	0	1	PARO POR FALLO → IMPOSIBLE
	1	1	0	1	1	0	PARO POR FALLO → IMPOSIBLE
	1	1	1	1	1	1	PARO POR LLEGADA
D = 1 → HORARIO	0	0	0	0	0	0	ALTA VELOCIDAD
	0	0	1	0	0	1	PARO POR FALLO → IMPOSIBLE
	0	1	0	0	1	0	PARO POR FALLO → IMPOSIBLE
	0	1	1	0	1	1	PARO POR FALLO → IMPOSIBLE
	1	0	0	1	0	0	MEDIA VELOCIDAD
	1	0	1	1	0	1	PARO POR FALLO → IMPOSIBLE
	1	1	0	1	1	0	BAJA VELOCIDAD
	1	1	1	1	1	1	PARO POR LLEGADA

Se enviará mediante la salida **RESP** el aviso al PLC del elevador, para ello se requiere de otro control externo para poder escoger entre el elevador apropiado.

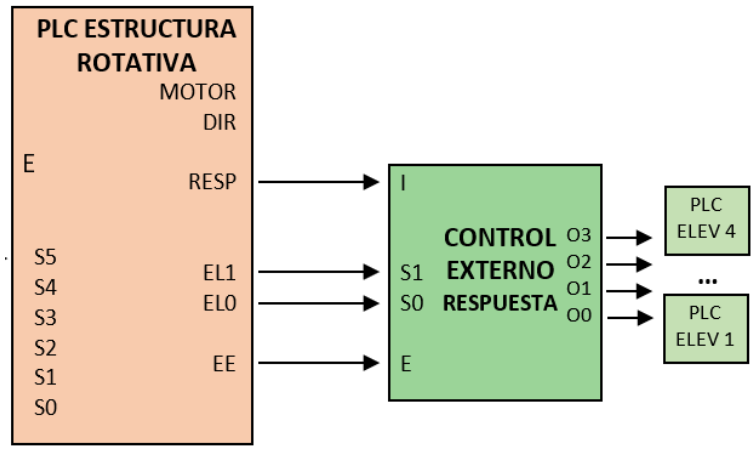


Figura 50. Control externo para responder al PLC del elevador indicado.

Este control externo funcionará como un demultiplexor de una entrada y cuatro salidas, ambas de 1 bit, con una selección de dos bits, siendo el elevador en la que se quiere enviar el aviso.

En la siguiente tabla de verdad (tabla 11) se muestra el funcionamiento de dicho sistema de control:

Tabla 11. Sistema de control externo para enviar al PLC del elevador apropiado el aviso de proceso concluido.

S1	S0	FUNCIÓN
0	0	ELEVADOR 1
0	1	ELEVADOR 2
1	0	ELEVADOR 3
1	1	ELEVADOR 4

4.3.3. ESTRUCTURA MECÁNICA

Para efectuar la rotación de la planta se necesita un sistema mecánico que traslade el par de rotación del motor a la estructura sólida de la planta. Diseñar dicho sistema mecánico requiere conocer los requerimientos y limitaciones.

La estructura debe aguantar el peso de los vehículos y el suyo propio, debe rotar a la máxima velocidad posible y debe de ser fiable. Para ello se aportan una posible solución que se basa en la rotación mediante un motor y varios apoyos móviles.

En la imagen (figura 51) se aprecia en color naranja la estructura que albergará ocho vehículos, en este caso, que debe rotar para enfrentar los huecos libres y las plataformas elevadoras. Para que dicha estructura no se vea sobrepasada por el peso se situarán ruedas o soportes móviles en varios puntos, representado de color azul.

El motor encargado de la rotación de dicha planta se situará en la caja roja, es decir, pegado al cilindro central, pues dicho cilindro tendrá una cremallera anclada a él y por la cual se desplazará el motor a modo de planetario.

No será necesario un gran motor para rotar dicha planta, pues la velocidad angular será de aproximadamente 0,66 rpm, es decir, hará una rotación completa en 40 segundos. Realmente su máximo recorrido será media circunferencia, pues dispone de cambio de sentido de giro.

Su velocidad lineal en el extremo de la estructura será de 0,62 metros por segundo, pues el radio máximo es de 9 metros.

$$v_{max} = \omega_{max} \cdot r$$

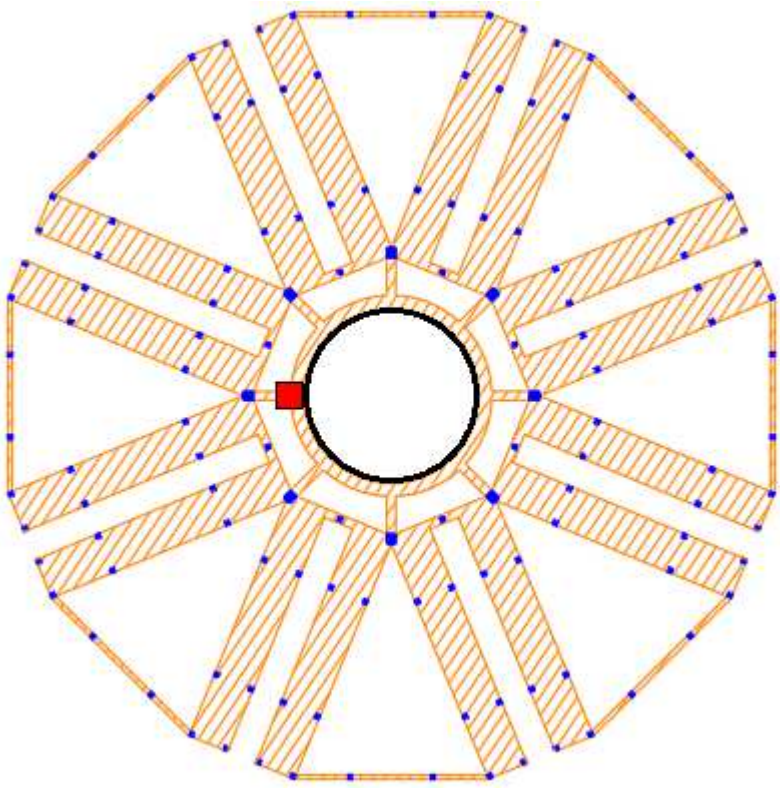


Figura 51. Estructura que conforma la planta de almacenaje de vehículos.

4.4. SISTEMA DE DETECCIÓN DE VEHÍCULO EN EL ELEVADOR

Cada plataforma elevadora contará con un subsistema que detectará la presencia del vehículo. Dicho subsistema estará formado únicamente por un sensor de detección basado en la fotocélula con espejo **E3FA-RN11** de la casa OMRON (figura 52).

Sensores (Carcasa de plástico E3FA)

Luz roja Luz infrarroja

Tipo de sensor	Distancia de detección	Método de conexión	Modelo	
			Salida NPN	Salida PNP
Reflexión sobre espejo con función MSR*2.	0,1 a 4 m con E39-R1S	con cable	**E3FA-RN11 2M**	**E3FA-RP11 2M**
		conector M12	**E3FA-RN21**	**E3FA-RP21**

Figura 52. Extracto de las hojas de características de la fotocélula E3FA-RN11 de OMRON. [9]

El sensor **E3FA-RN11** (figura 53) posee las siguientes características: su carcasa es de plástico, cilíndrica y recta, posee reflexión sobre espejo con función MSR, salida NPN, conexión por cable mediante una sescuncia, es de luz roja y dispone de una distancia efectiva de 0,1 a 4 metros. Estas características son ideales para el aparcamiento, pues de esta manera se reduce su mantenimiento. Además, dispone de un potenciómetro para variar la sensibilidad y poder asegurar la detección.

Figura 53. Sensor detector de presencia E3FA-RN11. [9]

Este sensor se encontrará en el techo de la plataforma elevadora y se colocará el espejo en la base (entre las dos ruedas delanteras del vehículo), de este modo cualquier coche que se encuentre estacionado en la plataforma el sensor lo detectará y enviará una señal.

Su esquema de conexión se puede observar en la siguiente fotografía (figura 54). Se conectará el cable marrón al positivo de la fuente de alimentación, que será de 24V en corriente continua. El cable azul se conectará al negativo de dicha fuente de alimentación. Por su parte, el cable rosa especifica si se quiere trabajar en la oscuridad o en un entorno iluminado y, por último, la señal de control se obtiene con el cable negro.

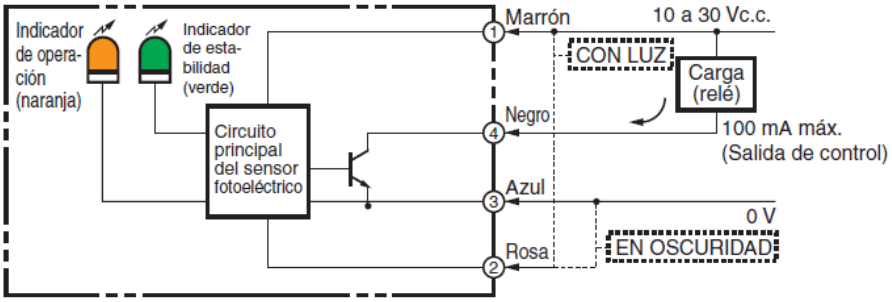


Figura 54. Circuito de salida del sensor E3FA-RN11. [9]

La mencionada señal de control se conectará a la alimentación (A1 o A2) de un relé, cuando dicho sensor detecte presencia el circuito que alimente al relé se activará y este cerrará los contactos generando una señal de control que llegará al microcontrolador. Esta conexión se realiza para evitar conectar directamente el sensor a la placa de evaluación y evitar posibles sobretensiones.

En la siguiente fotografía se puede observar la conexión del relé para obtener la señal de control, la cual irá al expansor de puertos quinto (**EXP 5)**. Es muy importante que las tierras, negativos o GND de la fuente de alimentación y de la tarjeta Mini-DK2 estén conectados, así la transmisión de voltaje será la correcta.

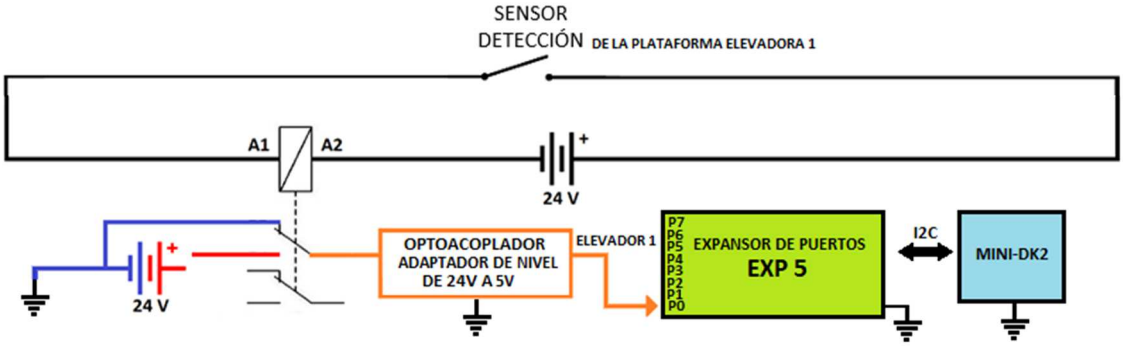


Figura 55. Diagrama de conexión del sensor de presencia.

4.5. SISTEMA DE SEGURIDAD DE SALA DEL ELEVADOR

Al igual que el subsistema de detección de vehículo, cada plataforma elevadora contará con un subsistema de seguridad el cual asegurará, en la medida de lo posible, que la zona móvil se encuentra vacía para evitar accidentes. Dicho sistema de seguridad contará con cuatro sensores de presencia y tres sensores magnéticos que garanticen que las puertas o barreras estén cerradas.

Este sistema requiere una señal que lo active o desactive. Una vez recibida y ejecutada, enviará una respuesta para indicar que ha sido correctamente o no ejecutado.

4.5.1. COMPONENTES

Se recuerda que el sistema de seguridad estará compuesto por cuatro sensores de presencia y tres barreras físicas.

Figura 56. Sensor de movimiento HUBER Motion 3LV. [20]

HUBER Motion 3LV (figura 56) es un detector de movimiento mediante luz infrarroja. Posee un ángulo de 180° y una protección IP44. Su tensión de alimentación es de 12-24V DC y su esquema de conexión se muestra a continuación (figura 57):

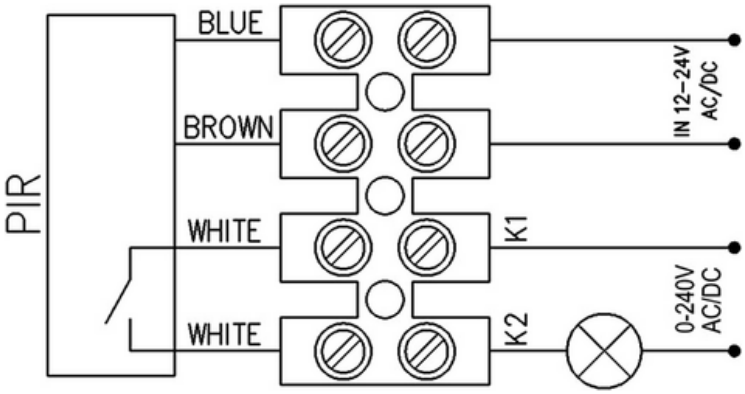


Figura 57. Diagrama de conexión del sensor de movimiento HUBER Motion 3LV. [20]

Se alimentará con 24V DC entre los cables marrón (positivo) y azul (negativo), y se obtendrá continuidad en el cable blanco si detecta movimiento.

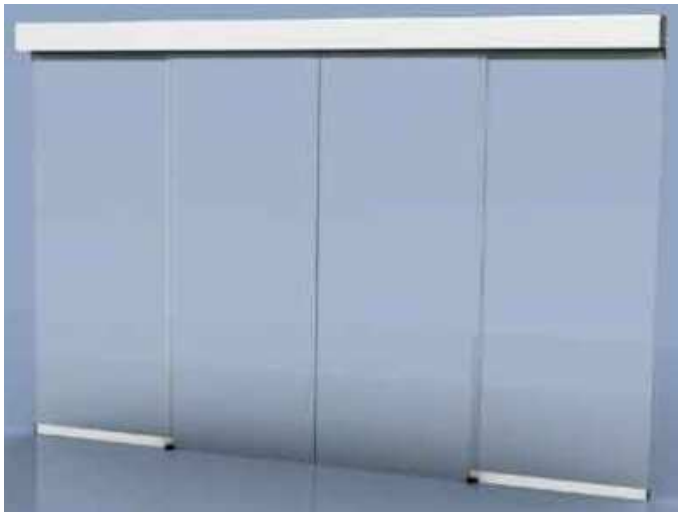


Figura 58. Puerta corredera automática. [21]

La puerta que se encuentra entre la sala del elevador y la sala del lector será una puerta corredera que se cerrará y bloqueará al recibir la señal de activación (figura 59). El grupo Aprimatic cuenta con todo lo necesario para llevar a cabo este automatismo.

Se utilizará un subsistema formado por un motor para puerta corredera peatonal (**NS 120 BS**) de dos hojas y un programador digital (**NS5-DIG)** que controlará en función de la señal de entrada, la señal de activación, el movimiento y bloqueo de la puerta.

Además, dispondrá de dos sensores de presencia que accionarán la apertura de la puerta cuando el bloqueo no esté activo para evitar tener las puertas abiertas y ahorrar energía.

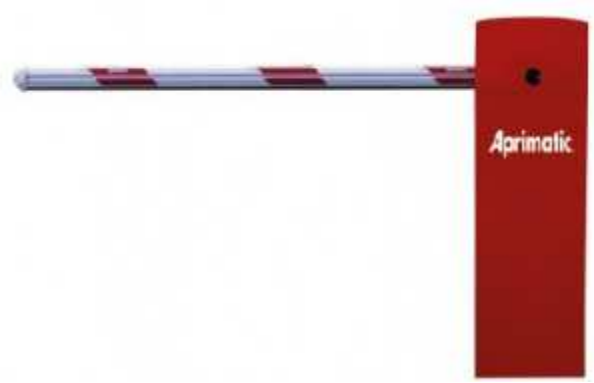


Figura 59. Barrera automática Park 30 Plus. [11]

Tanto a la entrada como a la salida de la zona de elevación, es decir, por donde llegan y salen los vehículos, se instalarán dos barreras automáticas (**Park 30 Plus)** (figura 59) del grupo Aprimatic, que formarán junto a un programador digital (**NS5-DIG)** un subsistema que garantizará el movimiento de la barrera. Dicho programador digital recibirá la señal de activación mandada por el sistema principal y enviará a la barrera la orden de apertura o cierre. Además, dispondrá de dos sensores que garanticen la inexistencia de obstáculos debajo de la barra.

Las mencionadas barreras y puerta se encontrarán en las entradas y salidas del recinto de la plataforma elevadora (exactamente donde se indica en la figura 60). Los sensores de presencia se ubicarán en cada una de las cuatro esquinas de la sala.

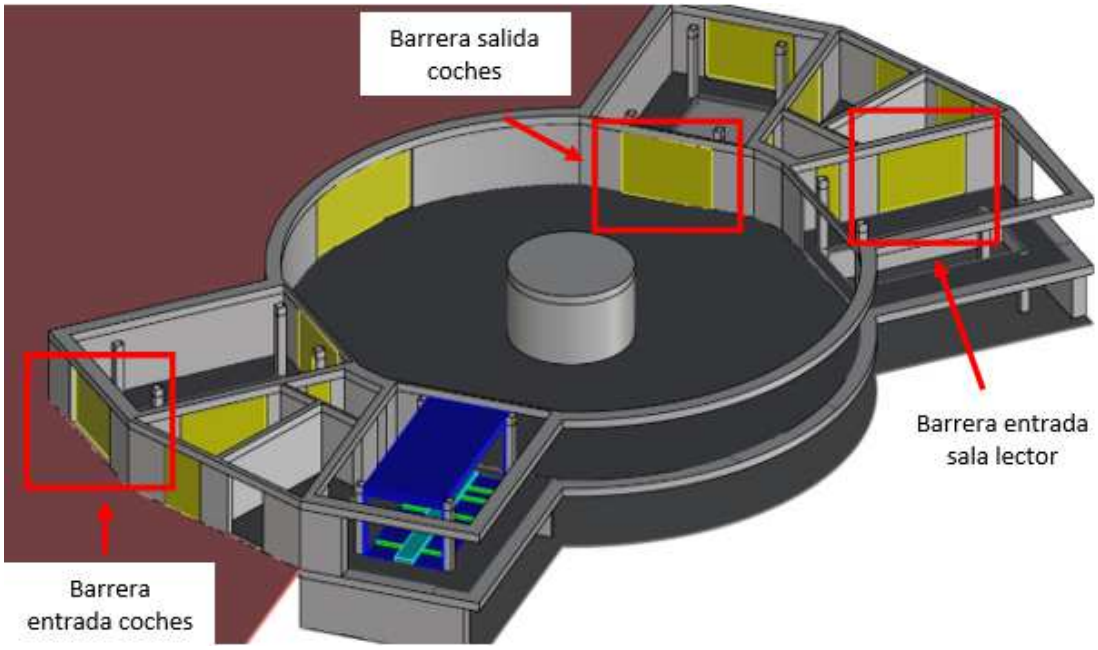


Figura 60. Situación de las barreras de seguridad en el aparcamiento.

4.5.2. ACTIVACIÓN Y RESPUESTA DEL SISTEMA DE SEGURIDAD

Los sensores de presencia se encontrarán conectados y activados en todo momento pues solo emiten una señal que indica la presencia de movimiento. En cambio, las barreras de seguridad necesitarán ser activadas y desactivadas para poder acceder a la plataforma elevadora. En el caso de las barreras, la activación se dará al recibir un nivel alto del microcontrolador principal en una señal de control o activación, dicha señal llegará a un único controlador (el programador digital **NS5-DIG**), que se encargará de controlar la apertura o cierre de las dos barreras y de la puerta corredera.

La respuesta que debe obtener el sistema principal tendrá lógica AND o en serie, es decir, en cuanto uno de los sistemas de seguridad se encuentre desactivado, la respuesta debe ser desfavorable y solo en el caso de que todo esté correctamente activado, la respuesta será favorable.

Para recoger con seguridad el cierre de las puertas o barreras, se colocarán sensores magnéticos codificados **MN200S** (figura 61). Estos sensores cerrarán un circuito si sus partes están lo suficientemente cerca y lo romperán en cuanto la puerta se abra separándolos, por lo que su situación en el conexionado general será normalmente abierto.

Figura 61. Sensores magnéticos MN200S. [22]

Los sensores de movimiento integran en su conexionado la apertura o cierre de un circuito de control. Dicho circuito se romperá cuando se detecte movimiento, por lo que su situación en el conexionado general será normalmente cerrado. La representación del circuito se puede observar en la figura 62.

Figura 62. Representación de la conexión en serie de los diferentes sensores.

Para evitar conectar directamente los sensores al microprocesador, se conectarán previamente a la alimentación de un relé de corriente continua a 24 V, pues los sensores industriales suelen trabajar a este voltaje. Con esto se reducen los picos de tensión y las impedancias y se reducen los tiempos transitorios, obteniendo una respuesta más limpia. Al alimentarse dicho relé, enviará una señal a nivel alto, es decir, a 24 V, la cual se modificará mediante un adaptador de niveles para que su nivel alto sea de a 5 V y será leído por un pin del expansor de puertos 4 (EXP 4). Este circuito se muestra en la figura 63.

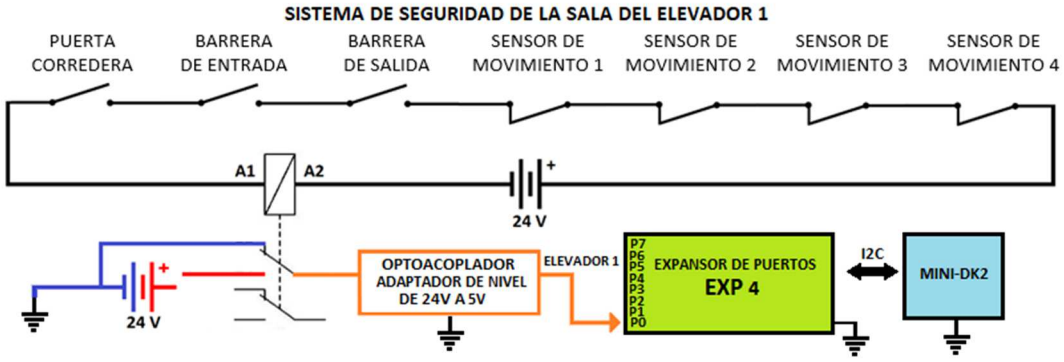


Figura 63. Diagrama de conexión del sistema de seguridad.

CAPÍTULO V

DISEÑO HARDWARE DEL SISTEMA DE CONTROL

5. DISEÑO HARDWARE DEL SISTEMA DE CONTROL

En este apartado se explica el diseño hardware del automatismo del aparcamiento. Los principales problemas que se encuentran son la distancia y el número de patillas del microprocesador, respecto a la distancia, las señales que se obtienen no logran superar los dos metros, pues el ruido y la caída de tensión la distorsionan demasiado, para solventar este problema se utilizan sistemas y protocolos capaces de aumentar hasta en 30 metros la distancia crítica de error. Respecto al número de patillas, la tarjeta de evaluación Mini-DK2, donde va integrado el microcontrolador, solo nos presenta 68, un número bastante escaso para la cantidad de subsistemas integrados en el automatismo, para solventar este problema se usarán expansores de puertos mediante el bus I^2C.

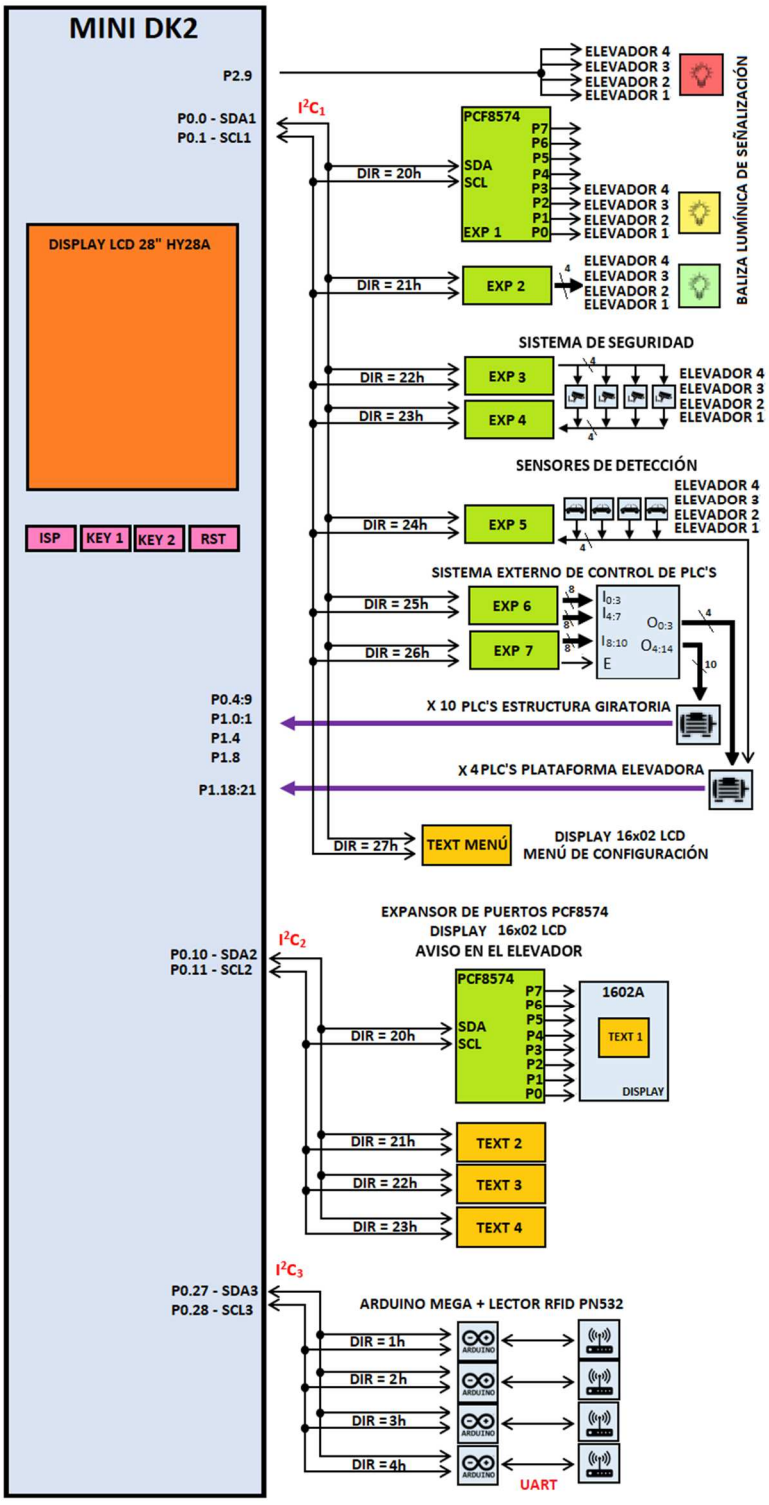


Figura 64. Gráfico resumen del hardware del automatismo.

En la figura 64 se muestra una visión global del hardware antes de adentrarse en cada caso, donde el microprocesador Cortex M3 LPC1768 de ARM sobre la tarjeta de evaluación Mini-DK2 será el centro del automatismo. Este esquema muestra la particularización a 4 plataformas elevadoras y 10 estructuras giratorias que se han presentado con anterioridad.

5.1. TRANSMISIÓN DE DATOS

Como se ha mencionado la transmisión de datos y la escasez de pines o puertos de entrada y salida (GPIO) son uno de los principales problemas de este automatismo, para solventarlos se deben usar protocolos de comunicación que reduzcan el número de patillas y periféricos que hagan efectiva la comunicación a grandes distancias.

En este proyecto se utilizan dos tipos de transmisión de datos: la comunicación I^2C y la comunicación serie asíncrona (UART).

5.1.1. COMUNICACIÓN I^2C

I^2C (Inter Integrated Circuits) es un protocolo de comunicación serial que permite conectar varios dispositivos únicamente por dos cables, un bus de datos (SDA) y un bus reloj (SCL). Esta transmisión de datos es óptima, pues nos permite conectar hasta 129 dispositivos diferentes al mismo bus, donde uno de ellos será el dispositivo maestro, el cual enviará las configuraciones a los demás dispositivos que se denominarán esclavos. Dicho sistema de transmisión tendrá el reposo a nivel alto, por lo que se debe colocar una resistencia pull-up, comúnmente de 4k7 ohmios, en cada cable.

De los 129 dispositivos 128 serán esclavos, este número no se podrá superar, pues cada uno de ellos tendrá una dirección diferente con la que el maestro podrá acceder individualmente dada por 7 bits. Se puede conocer más de dicho protocolo en el apartado 6.1.

En este proyecto se utilizan multitud de componentes, basados en el mismo chip, conectados mediante I^2C, por lo que es común repetir direcciones. Para liberar la carga del bus y evitar tener que elegir otros componentes se implementarán tres buses: el primero conectará siete expansores de puertos de 8 bit y un display LCD 16x2, el segundo cuatro displays LCD 16x2 y el tercero cuatro tarjetas Arduino Mega.

Será muy importante interconectar las tierras, negativo o GND de todos los dispositivos para garantizar la correcta transmisión.

En la tabla 12 se encuentran la conexión entre los pines del **LPC1768** y el bus I^2C

Tabla 12. Pines del microprocesador LPC1768 relacionadas con el bus I²C.

BUS I^2C	SDA	SCL
I^2C_1	P0.0	P0.1
I^2C_2	P0.10	P0.11
I^2C_3	P0.27	P0.28

5.1.2. COMUNICACIÓN SERIE ASÍNCRONA (UART)

UART (*Universal Asynchronous Receiver / Transmitter*) es un protocolo de comunicación asíncrono serial que no requiere la intervención de una señal de reloj, como en el caso del bus I^2C. Este bus únicamente permite conectar dos dispositivos, por lo que para poder conectar más de dos será necesario un sistema externo que controle mediante direcciones los dispositivos. En este protocolo existirán únicamente dos cables, TX o transmisor y RX o receptor. Las conexiones entre los diferentes dispositivos han de hacerse cruzando los cables, es decir, TX de un dispositivo se tendrá que conectar con RX del otro dispositivo. En la figura 65 se representa la interconexión de esta comunicación.

Se puede conocer más de dicho protocolo en el apartado 6.2.

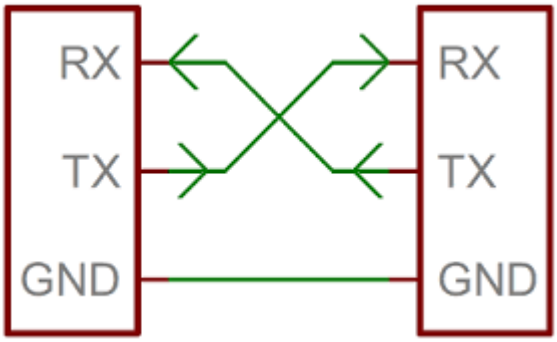


Figura 65. Representación hardware del bus UART. [29]

Para este proyecto se usará la tarjeta de evaluación Arduino Mega para interconectar el lector RFID y el microcontrolador principal (**LPC1768**) pues dispone de varios buses UART, así se podrá conectar el primer bus a PC para monitorear y cualquier otro para la transmisión de datos del lector.

En la tabla 13 se encuentran la conexión entre los pines de Arduino Mega y el bus UART.

Será muy importante interconectar las tierras, negativo o GND de todos los dispositivos para garantizar la correcta transmisión.

Tabla 13. Pines de Arduino Mega relacionadas con el bus UART.

BUS UART	TX	RX
UART 0	P0	P1
UART 1	P18	P19
UART 2	P16	P17
UART 3	P14	P15

5.1.3. COMUNICACIÓN A LARGA DISTANCIA

El inconveniente de este sistema de comunicación de señales es la distancia y el ruido, pues comúnmente se usan cables finos (denominados Dupont o jumper de 22 AWG o 0.644 mm^2 de sección) cuya longitud máxima es de 14,5 metros para una caída de tensión máxima del 3%, teniendo en cuenta que la señal es de 3,3 V y 100 mA.

Según la ley de Ohm:

$$\Delta V = I \cdot R$$

Donde ΔV es la caída de tensión, I la intensidad y R la resistencia. Si tenemos en cuenta que la resistencia de un conductor responde a la siguiente fórmula:

$$R = \frac{L}{c \cdot S}$$

Donde L es la longitud del cable, c la conductividad de este y S su sección. Si se sabe que se envían dos cables, el de señal más la referencia de tierra habrá que multiplicar por dos la resistencia total. Se aplica a la ley de Ohm y se obtiene:

$$\Delta V = I \cdot \frac{L}{c \cdot S} \cdot 2$$

$$L = \frac{\Delta V \cdot c \cdot S}{2 \cdot I}$$

$$L = \frac{0{,}099\ [V] \cdot 45{,}49\left[\frac{m}{mm^2 \cdot \Omega}\right] \cdot 0.644\ [mm^2]}{2 \cdot 0{,}1\ [A]} = 14{,}5m$$

Suponemos una conductividad de 45,49, pues el cable es de cobre y nos remitimos al caso más restrictivo (cable termoestable a 90º).

Se deberá tener en cuenta el ruido, ya que puede generar una distorsión en la señal. Para evitar dichos problemas se propone transportar la señal en conductores de más sección y a 24V en señales de entrada o salida (GPIO), pues simplemente aumentando el voltaje conseguimos una longitud máxima de 105,5 metros, y se usarán módulos especiales (**PCA9615**) para expandir la distancia del bus I^2C.

En las señales de entrada o salida (GPIO) se realiza un cambio de voltajes, donde se utilizarán optoacopladores, recogidos en el módulo **PC817**, el cual se basa en unos ciertos transistores que al recibir una señal en un voltaje predefinido envían otra igual pero de otro voltaje predefinido, en la figura 66 se observa un módulo de optoacopladores que cambian el voltaje de la señal de 24V a 5V.

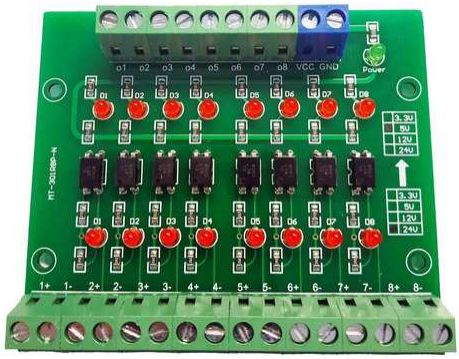


Figura 66. Módulo PC817 optoacoplador de 24V a 5V. [19] [23]

En la comunicación I^2C se propone el uso del módulo **PCA9615** basado en el chip **P82B15**, el cual asegura una distancia efectiva de 20 metros a 400 kHz y de 30 metros a 100 kHz. En la figura 67 se puede observar dicho módulo.

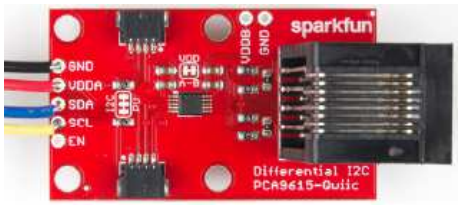


Figura 67. Módulo PCA9615 de Sparkfun. [18] [25]

Se acoplarán tantos módulos como dispositivos haya, es decir, si disponemos de un maestro y ocho esclavos necesitaremos nueve módulos, cada módulo situado al lado del dispositivo. La conexión entre los módulos se da a través de un cable típico de internet, para conectar todos los dispositivos hará falta un conector múltiple que facilite dicho conexión. Se puede observar un diagrama de esta conexión en la figura 68.

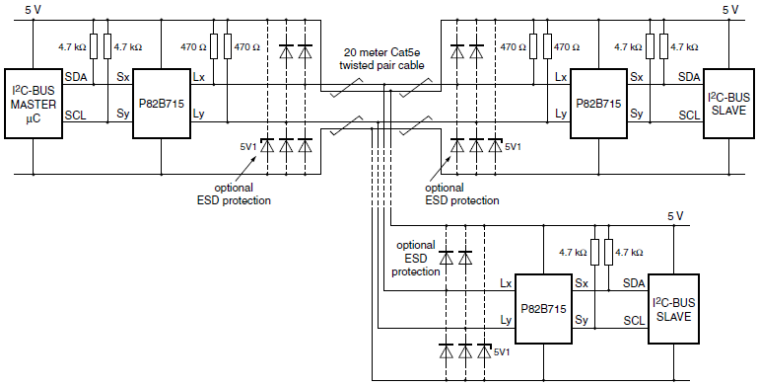


Figura 68. Ilustración de la conexión entre maestro y dos esclavos a gran distancia usando el módulo PCA9615. [18] [26]

5.2. LECTOR RFID PN532

La activación del automatismo se dará gracias a un lector de tarjetas RFID, el cual enviará el UID o identificador único de la tarjeta RFID a Arduino, el cual lo procesará enviando al sistema principal el número de tarjeta. Este lector se basa en el módulo **PN532** que se puede observar en la figura 69.

Dicho lector posee tres tipos de comunicación de datos: HSU, I^2C y SPI. En este caso, conectaremos dicho lector con Arduino mediante el bus HSU (*High Speed UART*). Para realizar la conexión, primero se deben poner los dos *switches* a cero, o como vienen por defecto. TX lo obtendremos de SDA y RX de SCL, ambos en la parte izquierda, conectándolo con RX_1 y TX_1 de Arduino Mega.

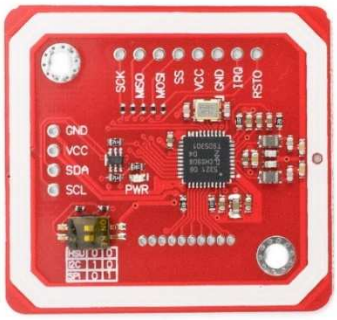


Figura 69. Módulo PN532. [16] [30]

En la siguiente tabla (tabla 14) se relacionan los pines de Arduino con los del lector.

Tabla 14. Pines de Arduino y del lector PN532 interrelacionados con el bus UART.

UART 1 – ARDUINO	TX_1- P18	RX_1- P19
LECTOR PN532	RX- SCL	TX- SDA

5.3. DISPLAY LCD HY28A

La conexión con la tarjeta Mini-DK2 es bastante sencillo, pues dicha tarjeta posee directamente el acople. Las patillas de dicho módulo se pueden observar en las hojas de características, la conexión entre ambos dispositivos se encuentra a continuación en la figura 70.

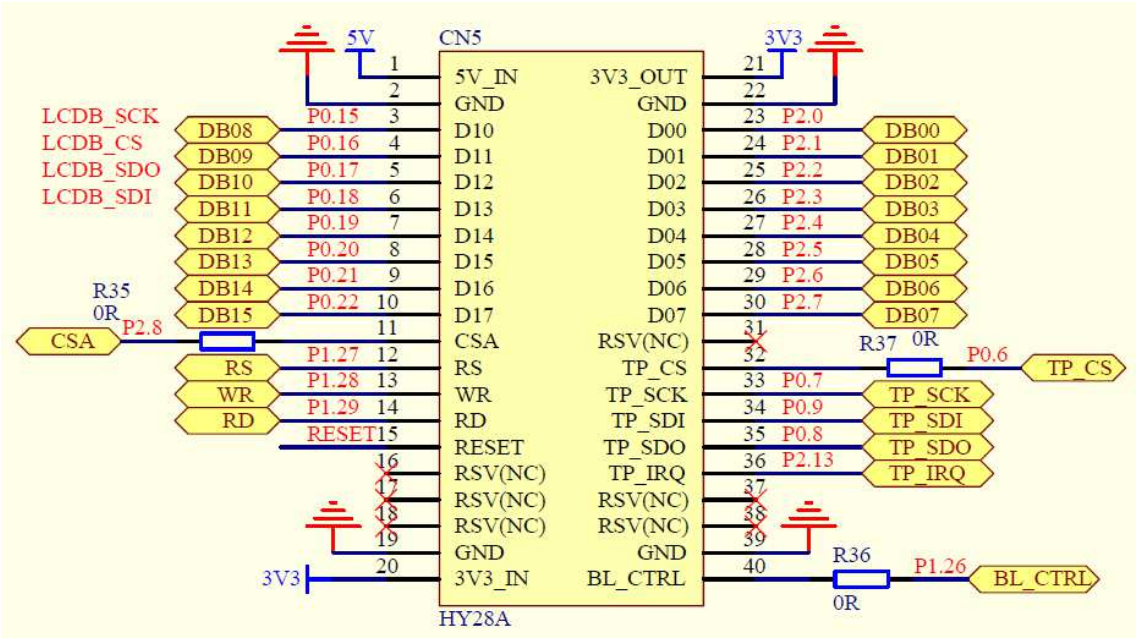


Figura 70. Esquemático del módulo HY28A en la tarjeta Mini-DK2. [12]

5.4. DISPLAY LCD 16X2

El automatismo contará con un display LCD encima de cada lector (un total de cuatro elevadores, lectores y displays), donde se mostrarán mensajes informativos. El componente elegido es el módulo **1602A + PCF8574T**, dicho módulo funciona bajo el protocolo I^2C, por lo que se conectará al microprocesador mediante ese bus.

Adicionalmente se usará un quinto display para visualizar el menú de configuraciones, este display se situará anexa a la tarjeta de evaluación Mini-DK2

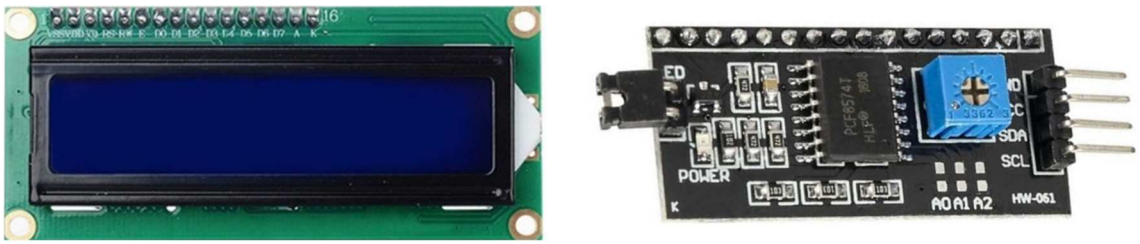


Figura 71. Módulo 1602A + módulo PCF8574. [14] [28]

El módulo **PCF8574T** se basa en el mismo microchip que los expansores de puertos que se verán en el siguiente apartado (apartado 5.5) y posee la misma configuración de la dirección, por ello, se emplearán varios buses I^2C. Para el display del menú de configuraciones se usará el bus I^2C_1, mientras que para los cuatro displays de los elevadores se usará el bus I^2C_2

5.5. EXPANSOR DE PUERTOS

Uno de los principales problemas es la cantidad de patillas disponibles y la multitud de señales que transmitir. Por ello se propone el uso de expansores de puertos mediante el bus I^2C mediante el módulo **PCF8574T**, basado en el chip de mismo nombre.

Dicho módulo expande en ocho el número de patillas, funcionando tanto en lectura como en escritura. Su funcionalidad es bastante sencilla, pues no se necesita configurar previamente el módulo, simplemente se envía la dirección del esclavo, la cual se puede configurar, se remite la petición de lectura o escritura y se lee o se escribe cada una de las patillas con el bit correspondiente (P7 = dato 7 o MSB, … P0 = dato 0 o LSB)

La gran limitación de este módulo es la configuración de la dirección, debido a que solo son configurables mediante un switch los tres últimos bits (A2, A1 y A0), restringiendo a 8 el número máximo de dispositivos de las mismas características. La dirección por defecto se representa en binario: 0 1 0 0 A2 A1 A0.

En este sistema se utilizarán más de ocho dispositivos de estas características (siete expansores de puertos y cinco displays LCD) por lo que se emplearán varios buses I^2C para evitar usar otro dispositivo. Para todos los expansores de puertos usaremos el bus I^2C_1

En la figura 72 se observa dicho módulo y el sistema para configurar la dirección.

Figura 72. Módulo PCF8574T con dirección 20h. [15] [27]

5.6. BALIZAS LUMÍNICAS DE SEÑALIZACIÓN

El aparcamiento acopla un sistema lumínico que avisa del estado de cada una de las plataformas elevadoras mediante tres luces en cada plataforma elevadora y cuatro modos de funcionamiento que se observan en la tabla 15.

Tabla 15. Estado de la plataforma elevadora frente al color de luz y parpadeo de dicho elevador.

ESTADO DE LA PLATAFORMA ELEVADORA	AVISO LUMÍNICO
Disponible, se puede aparcar o retirar un vehículo	Luz verde fija
Modo en ejecución, se está aparcando o retirando un vehículo	Luz amarilla parpadeante
En avería, o bien por error en el PLC del elevador, de la estructura giratoria o por los sistemas de seguridad	Luz amarilla fija
Aparcamiento lleno, solo se pueden retirar coches	Luz roja fija

En este caso, todas las luces rojas de todos los elevadores se conectarán a la misma señal, es decir, si el aparcamiento no tiene plazas libres, aparecerá del mismo color en todos los elevadores.

Dicha señal será generada directamente por el microcontrolador y conectada a las bombillas mediante la patilla P2.9, la cual generará un nivel alto en caso de querer encender la bombilla. Esta suele alimentarse a tensión de red (230V) por lo que habrá que transformar dicha señal a alimentación, para eso se usará un relé, que funciona como un interruptor que cierra o abre el circuito de alimentación de la bombilla. No será necesario usar un relé por cada bombilla, pero en la práctica se utilizará para poder aislar en cuadros eléctricos los diferentes elevadores o salas.

El relé necesitará ser alimentado a 24V por sus terminales DC+ y DC-, por el terminal In1 se introducirá la señal de control. En la salida encontramos el contacto común y dos contactos, uno NC (normalmente cerrado) y otro NA (normalmente abierto).

Figura 73. Relé de 24V de alimentación y señal de control de 3,3V. [24]

En el caso de los colores verde y amarillo no se conectarán todos a la misma señal, sino que será necesaria una señal de control por cada bombilla, para ello se usan expansores de puertos.

El primer expansor de puertos (EXP 1) se encargará de enviar la señal de control de las luces amarillas y el segundo expansor de puertos (EXP2) se encargará de las luces verdes. Se conectarán los ocho pines de cada expansor de puertos con la luz del elevador correspondiente, es decir, el bit menos significativo (LSB) de EXP 0 (P0) se conectará con la bombilla amarilla de la plataforma elevadora 1, mientras que el bit más significativo (MSB) de EXP 1 (P7) se conectará con la bombilla amarilla del octavo elevador, en este caso solo se dispone de cuatro, pero el sistema está preparado para esta ampliación.

En las siguientes tablas (tabla 16 y 17) se puede observar dichas conexiones, donde ELEV se corresponde con el número de plataforma elevadora.

Tabla 16. Relación entre el elevador y cada pin del expansor de puertos 1.

PUERTOS DE EXP 1	P7	P6	P5	P4	P3	P2	P1	P0
BOMBILLA AMARILLA	ELEV. 8	ELEV. 7	ELEV. 6	ELEV. 5	ELEV. 4	ELEV. 3	ELEV. 2	ELEV. 1

Tabla 17. Relación entre el elevador y cada pin del expansor de puertos 2.

PUERTOS DE EXP 2	P7	P6	P5	P4	P3	P2	P1	P0
BOMBILLA VERDE	ELEV. 8	ELEV. 7	ELEV. 6	ELEV. 5	ELEV. 4	ELEV. 3	ELEV. 2	ELEV. 1

Esta señal a 3,3 o 5 voltios es una señal de control, pues las bombillas no trabajan a este voltaje, por lo que se acoplará, al igual que en las bombillas rojas, un relé que permita la llegada del voltaje y corriente correcto a la bombilla.

Dicho expansor y relés se situarán al lado del microprocesador por lo que la distancia no será un problema.

5.7. SISTEMA DE SEGURIDAD

El automatismo requiere de un sistema de seguridad que sea capaz de garantizar la ausencia de personas y animales en el recinto de la plataforma elevadora, para ello se colocarán cuatro sensores de presencia y tres sensores magnéticos que garanticen que las puertas o barreras estén cerradas (apartado 4.5.).

Para activar y bajar o cerrar las barreras de seguridad y la puerta, el sistema enviará una señal de control a cada elevador, donde si dicha señal está a nivel alto las barreras, se bajarán y la puerta se cerrará y si cambia a nivel bajo, el elevador volverá a estar accesible abriendo el paso. Esta señal de control se enviará mediante I^2C hasta un expansor de puertos (EXP 3), que dividirá la señal de control a cada elevador. Para poder garantizar la correcta transmisión a larga distancia, la señal se transformará a 24V, tomando dicha señal como si fuese un indicador GPIO (apartado 5.7.).

Una vez activados dichos sistemas de seguridad se necesita obtener una respuesta por su parte que nos indique si existe alguna anomalía y poder detener el proceso. Esta respuesta se generará en cada uno de los elevadores en forma de señal de control, se recogerá en el cuarto expansor de puertos (EXP 4) y será leída cada cierto tiempo. Para la generación de dicha señal de control o respuesta, los sensores se deberán conectar en serie, de tal manera que en cuanto uno de ellos rompa el circuito (en el caso de los detectores que detecten y en el caso de los sensores magnéticos que dejen de circular) la señal de control esté a nivel bajo. En cambio, si la puerta y barreras están correctamente cerradas y no se detecta movimiento, la señal de control estará a nivel alto. Todo esto se representa en la figura 74.

Figura 74. Representación de la conexión en serie de los diferentes sensores.

Para evitar conectar directamente los sensores al microprocesador se conectarán previamente a la alimentación de un relé de corriente continua a 24 V, pues los sensores industriales suelen trabajar a este voltaje. Con esto se reducen los picos de tensión y las impedancias y disminuyen los tiempos transitorios, obteniendo una respuesta más limpia.

Al alimentarse dicho relé enviará un nivel alto, es decir, 24 V, el cual se modificará a 5 V mediante un optoacoplador adaptador de niveles y será leído por un pin del cuarto expansor de puertos (EXP 4), esta relación de pines del expansor de puertos y el sistema de seguridad de cada plataforma elevadora se recoge en la tabla 18, la representación de la interconexión se observa en la figura 75.

Tabla 18. Relación entre el elevador y cada pin del expansor de puertos 4.

PUERTOS DE EXP 4	P7	P6	P5	P4	P3	P2	P1	P0
SEÑAL DE CONTROL	ELEV. 8	ELEV. 7	ELEV. 6	ELEV. 5	ELEV. 4	ELEV. 3	ELEV. 2	ELEV. 1

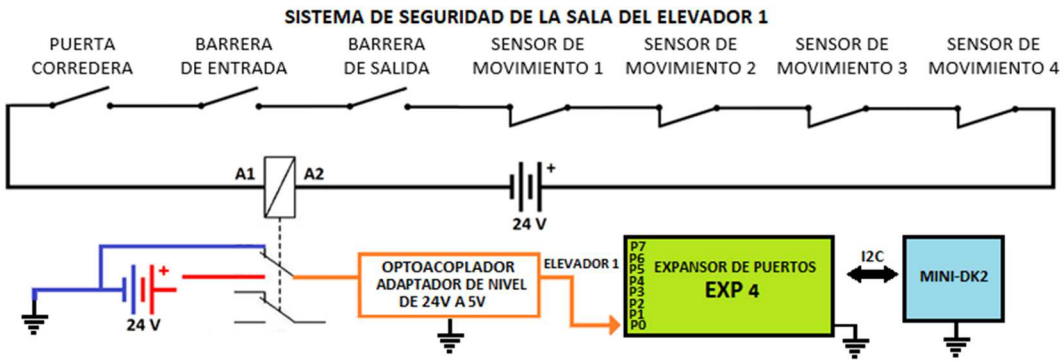


Figura 75. Representación de la respuesta obtenida de los sensores de seguridad.

5.8. SISTEMA DE DETECCIÓN DE VEHÍCULO

El automatismo requiere de un sistema de detección que sea capaz de detectar al vehículo, para ello se colocará un único sensor de presencia (apartado 4.4.).

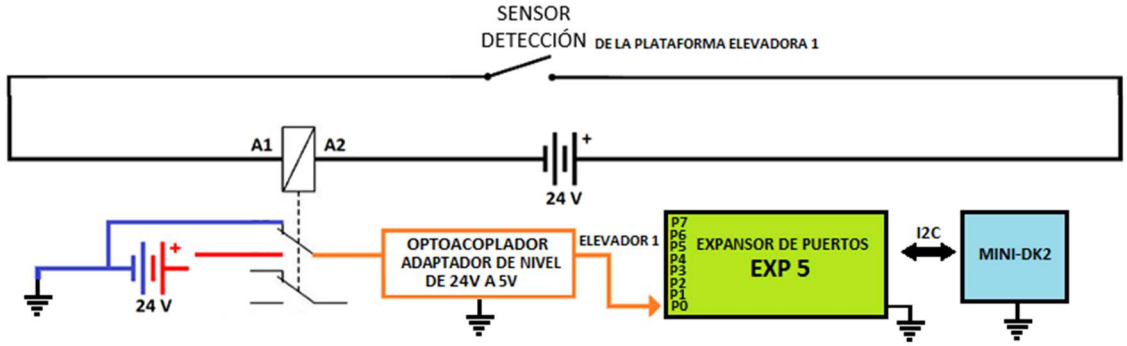


Figura 76. Representación de la respuesta obtenida de los sensores de detección de vehículo.

Al igual que en el apartado 5.8. se conectará dicho sensor a un relé de corriente continua a 24 V el cual al ser activado (el sensor detectará y cerrará el circuito) enviará hacia el expansor de puertos 5 (EXP 5) un nivel alto de 5 V al que se le ha modificado el nivel de voltaje mediante un optoacoplador para poder ser trabajado. Esta relación de pines del expansor de puertos y el sistema de seguridad de cada plataforma elevadora se recoge en la tabla 19, la representación de la interconexión se observa en la figura 76.

Tabla 19. Relación entre el elevador y cada pin del expansor de puertos 5.

PUERTOS DE EXP 5	P7	P6	P5	P4	P3	P2	P1	P0
SEÑAL DE CONTROL	ELEV. 8	ELEV. 7	ELEV. 6	ELEV. 5	ELEV. 4	ELEV. 3	ELEV. 2	ELEV. 1

5.9. CONTROL EXTERNO DE LOS PLC's

Este sistema no integra la posibilidad de controlar los motores únicamente con la tarjeta de evaluación Mini-DK2 y su microcontrolador, sino que propone el uso de PLC's o controladores lógicos programables, pues actualmente hay multitud de empresas que montan plataformas elevadoras y otros proyectos con motores basado en estos sistemas.

En este automatismo habrá 18 PLC's, 10 encargados del giro de las estructuras rotativas y los otros 8 encargados de la plataforma elevadora y de la plataforma de desplazamiento lineal que introducirá o retirará el vehículo (el software está preparado para contener hasta 16 estructuras giratorias y 8 plataformas elevadoras, por lo que este control también lo estará)

El PLC de cada una de las plataformas elevadoras necesita recibir el número de planta a la que se tiene que posicionar, en cambio, el PLC de cada una de las estructuras rotativas necesitará recibir la división de la planta o plaza en la planta y el elevador al cuales tiene que enfrentar. Esto lo conseguimos

combinando dos expansores de puertos (EXP 6 y 7) y dos sistemas multiplexores. Se puede observar en la figura 77.

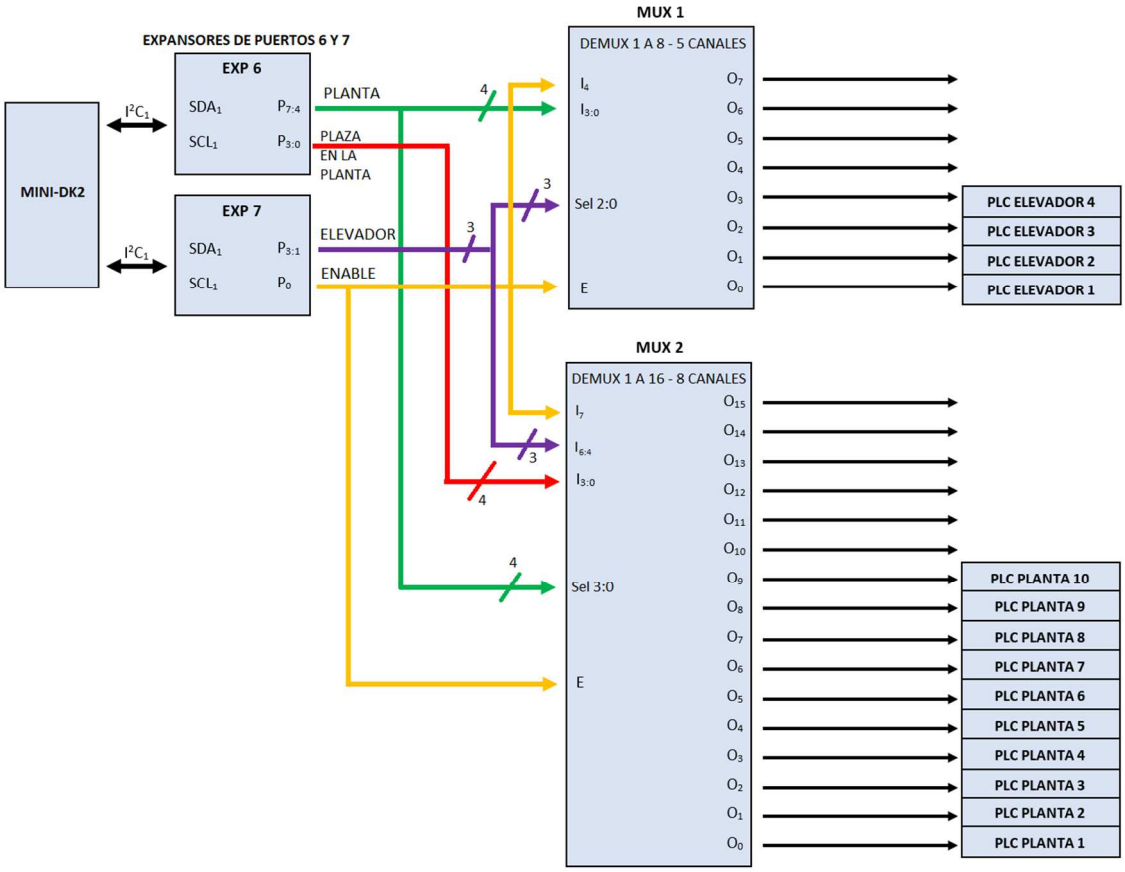


Figura 77. Gráfico de la conexión de los PLC's con el microcontrolador.

El expansor de puertos 6 (EXP 6) se encargará de representar mediante sus cuatro bits más significativos el número de planta a la que el elevador se tiene que dirigir y con sus cuatro bits menos significativos indicará la plaza o división de dicha planta ocupada. Por su parte, EXP 7 mandará una señal de *enable* o activación por el bit menos significativo y el número de elevador por los siguientes 3 bits. En las tablas 20 y 21 se especifica con más detalle.

Tabla 20. Relación entre cada pin del expansor de puertos 6 y su significado.

PUERTOS DE EXP 6	P7	P6	P5	P4	P3	P2	P1	P0
SIGNIFICADO	PLANTA 3	PLANTA 2	PLANTA 1	PLANTA 0	PLAZA 3	PLAZA 2	PLAZA 1	PLAZA 0

Tabla 21. Relación entre cada pin del expansor de puertos 7 y su significado.

PUERTOS DE EXP 6	P7	P6	P5	P4	P3	P2	P1	P0
SIGNIFICADO					ELEV. 2	ELEV. 1	ELEV. 0	ENABLE

Para poder llevar el número de planta, división o plaza y elevador a cada uno de los PLC se necesita un sistema que envíe únicamente dichos datos a un único PLC, para ello se proponen dos demultiplexores complejos, el primero (**MUX1**) de una entrada (**I**) y ocho salidas(**O 7:0**), controladas o seleccionadas por tres señales (**Sel 2:0**), la entrada será de cuatro bits, al igual que las salidas, donde dichos bits se conectarán con los cuatro bits de **PLANTA** (representa el número de estructura rotativa), a cada una de las ocho salidas conectamos cada uno de los cuatro elevadores, dejando las otras cuatro salidas libres por si se necesita hacer alguna ampliación, las entradas de selección serán conectadas a los tres bits de **ELEV**. (representan el número de elevador). Con esto conseguimos mandar el número de planta al elevador deseado. El segundo**, MUX2**, de una entrada (**I**) y 16 salidas (**O 15:0**), controladas o seleccionadas por cuatro señales (**Sel 3:0**), la entrada será de siete bits, al igual que las salidas, donde los primeros cuatro bits se conectarán con los cuatro bits de **PLAZA** (representa el número de división de la planta) y los siguientes tres bits se conectarán a los tres bits de **ELEV**, a cada una de las 16 salidas

conectamos cada una de las 10 estructuras rotativas, dejando libres las otras 6 para futuras ampliaciones, las entradas de selección serán conectadas a los cuatro bits de **PLANTA**. Con esto conseguimos mandar el número de división de la planta y elevador a la panta deseada.

Tanto el sistema de elevación del vehículo como la plataforma de desplazamiento lineal encargado de introducir o retirar los vehículos o el sistema que girará la estructura rotativa de la planta correspondiente, serán controlados mediante PLC's, los cuales enviarán una respuesta si se ha completado con éxito el proceso o si ha ocurrido una avería.

Como se ha comentado, habrá un máximo de 18 autómatas, sin embargo el proyecto está desarrollado para albergar 16 estructuras rotativas y 8 elevadores, por lo que aparecerá el mismo número de entradas o respuestas.

En las siguientes tablas (tabla 22, 23 y 24) se muestra la relación entre los diferentes pines y el correspondiente PLC en el caso estudiado.

Tabla 22. Relación entre el elevador y cada pin del microcontrolador.

PUERTOS DEL MICRO	P1.25	P1.24	P1.23	P1.22	P1.21	P1.20	P1.19	P1.18
PLC ELEVADOR					ELEV. 4	ELEV. 3	ELEV. 2	ELEV. 1

Tabla 23. Relación entre las primeras ocho estructuras rotativas y cada pin del microcontrolador.

PUERTOS DEL MICRO	P1.1	P1.0	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4
PLC ESTRUCT. ROTAT	PLANTA 8	PLANTA 7	PLANTA 6	PLANTA 5	PLANTA 4	PLANTA 3	PLANTA 2	PLANTA 1

Tabla 24. Relación entre las ocho últimas estructuras rotativas y cada pin del microcontrolador.

PUERTOS DEL MICRO	P1.17	P1.16	P1.15	P1.14	P1.10	P1.9	P1.8	P1.4
PLC ESTRUCT. ROTAT.							PLANTA 10	PLANTA 9

En la siguiente figura (figura 78) se muestra las respuestas de los PLC's como entradas a la Mini-DK2.

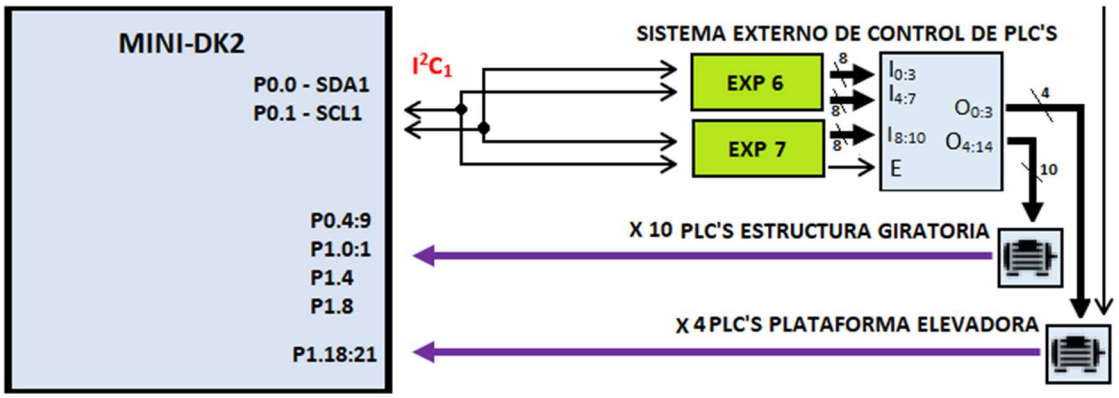


Figura 78. Representación de las respuestas de los PLC's conectadas a la Mini-DK2.

CAPÍTULO VI

SOFTWARE DEL AUTOMATISMO

6. SOFTWARE DEL AUTOMATISMO

El software de este programa se tiene que ajustar a las siguientes especificaciones:

- En reposo, cada elevador dispondrá de una luz verde fija y se informará de que el elevador está disponible en el display del lector.
- El sistema se activará al pasar por un lector RFID una tarjeta válida.
- Dicha tarjeta contendrá un número de plaza, ya que cada tarjeta tiene una plaza asociada.
- Una vez se ha obtenido la lectura de la tarjeta, se buscará si la plaza asociada está libre u ocupada y si se detecta o no vehículo en la plataforma
- Una vez se conoce el estado de la plaza y del sensor de detección se procede a aplicar uno de los siguientes modos de funcionamiento: “0Coches”, “2Coches”, aparcar y retirar vehículo.
- En los casos de “0Coches” y “2Coches” se emitirá por el display del lector un mensaje en el que se muestre que la petición ha sido rechazada, bien porque no hay ningún coche para retirar o bien porque hay un coche aparcado y no es posible aparcar otro.
- En los casos de aparcar o retirar, primero se activará una luz parpadeante amarilla y se mostrará en el display que el elevador está ejecutándose. Después se activarán las medidas de seguridad, cerrando puertas y barreras.
- Transcurrido cierto tiempo, se revisará si las medidas de seguridad están activadas o si ha habido alguna avería o presencia por la zona móvil. En caso de que algo falle, se activará una luz amarilla fija y en el display aparecerá un aviso de avería.
- En caso de que todo salga correctamente, se enviará al sistema de elevación y estructura giratoria los datos necesarios para el proceso.
- El sistema móvil enviará una respuesta cuando haya concluido su trabajo. Dicha respuesta podrá ser o de finalización correcta o de avería, en este último caso se actuará de la misma manera que en los sistemas de seguridad.
- En el caso de que todo sea correcto, habrá finalizado el proceso y volverá a estado de reposo.
- En caso de que todas las plazas estén ocupadas, se activará una luz roja fija y se emitirá el aviso en el display.

6.1. COMUNICACIÓN I²C

I²C (*Inter Integrated Circuits*) es un protocolo de comunicación serial que permite conectar varios dispositivos únicamente por dos cables, un bus de datos (SDA) y un bus reloj (SCL). Esta comunicación de datos es óptima, pues nos permite conectar hasta 129 dispositivos diferentes al mismo bus, donde uno de ellos será el dispositivo maestro, el cual enviará las configuraciones a los demás dispositivos los cuales se denominarán esclavos.

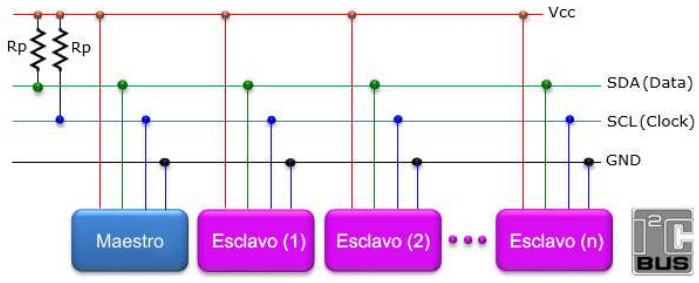


Figura 79. Representación hardware del protocolo I2C. [32]

De los 129 dispositivos 128 serán esclavos, este número no se podrá superar, pues cada uno de ellos tendrá una dirección diferente con la que el maestro podrá acceder individualmente, esta dirección viene dada por 7 bits, lo que restringe el número de dispositivos a conectar.

El protocolo de comunicación consta de los siguientes pasos:

En reposo SDA y SCL estarán a nivel alto, justo cuando se quiera empezar la transmisión el maestro enviará el bit de START, es decir, se envía por SDA un nivel bajo y se mantiene el nivel alto en SCL durante un ciclo de reloj, después SCL pasará a nivel bajo durante otro ciclo de reloj.

Acto seguido se enviará la dirección del esclavo al que se quiere acceder, para ello se enviarán los bits de la dirección de uno en uno empezando por el bit de mayor peso (MSB) y acabando por el de menos peso (LSB). Para poder transmitir ese bit el bus SCL será puesto a nivel alto justo después, manteniendo el nivel de ambos buses durante un ciclo de reloj, transcurrido ese tiempo SCL volverá a nivel bajo y se podrá cambiar el bit a enviar.

Justo después de la dirección se enviará un bit (R/W) que representa la lectura (1) o escritura (0) sobre el esclavo, al igual que con los bits de dirección SCL se pondrá a nivel alto y se mantendrán ambos valores durante un ciclo de reloj. El dispositivo esclavo deberá responder para comprobar que la transmisión de la orden de lectura o escritura se ha recibido, por ello escribirá en SDA un nivel alto, a esta respuesta se le conoce como ACK.

Ahora se enviarán los datos, escribiendo durante un byte de igual manera que la dirección. En el caso de lectura será el esclavo el encargado de escribir y el maestro deberá responder con el bit ACK (1) si se quieren recibir más bytes o NACK (0) si el byte leído era el último. En el caso de escritura será el maestro el encargado de escribir sobre SDA y el esclavo responderá a cada byte con ACK.

El fin de la transmisión se dará cuando el maestro envíe el bit de STOP, el cual dejará a nivel alto SCL y acto seguido SDA, al contrario que en el inicio de la transmisión.

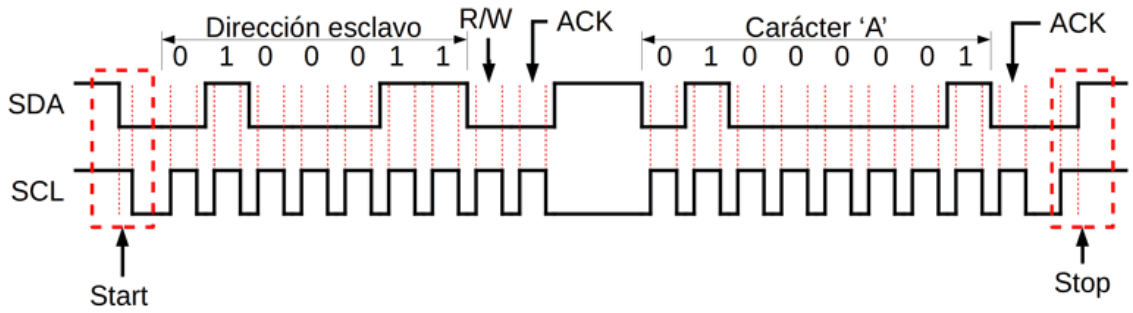


Figura 80. Ilustración del protocolo I²C. [33]

Respecto a las funciones que se usan en software para transmitir o recibir datos están:

- `void I2CSendAddr_x(addr, rw)`
- `void I2CSendByte_x(byte)`
- `unsigned char I2CGetByte_x(ACK)`
- `void I2CSendStop_x(void)`
- `void I2Cdelay_x(void)`

En el caso de querer escribir en el bus I^2C_x primero se debe de enviar la primera función con la dirección del esclavo, **addr**, y el bir **rw** a nivel bajo (read=1, write=0), acto seguido se enviará el byte que queramos escribir, en el caso de querer enviar varios datos o bytes se añaden tantas funciones **I2CSendByte** como datos haya, por último, se escribirá el bit de stop con la cuarta función y la función retardo para asegurar la transmisión.

En el caso de querer leer tan solo habrá que modificar el bit **rw** a nivel alto y cambiar la función **I2CSendByte** por **I2CGetByte(N/ACK)**, donde escribiremos en su argumento el bit **ACK** (**ACK**=1) si queremos seguir recibiendo datos o **NACK** (**NACK**=0) si no queremos recibir más datos. Al igual que en el caso de escritura enviaremos el bit de stop y el retardo.

En este proyecto se va a usar 3 buses I^2C diferentes, el prefijo _X indica el bus 1, 2 o 3, donde la única diferencia son los pines de salida. En Anexo II. se encuentra el código fuente del archivo **I2C_libreria_1.c** que es la librería que contiene estas funciones para el bus I^2C_1

6.2. COMUNICACIÓN SERIE ASÍNCRONA (UART)

UART (*Universal Asynchronous Receiver / Transmitter*) es un protocolo de comunicación asíncrono serial que no requiere la intervención de una señal de reloj, como en el caso del bus I^2C. Este bus únicamente permite conectar dos dispositivos, por lo que para poder conectar más de dos será necesario un sistema externo que controle mediante direcciones los dispositivos.

La transmisión de datos puede ser **simplex** (los datos se enviarán en una sola dirección), **semidúplex** (solo un dispositivo podrá transmitir datos, una vez termine de transmitirlos podrá empezar el otro) o **dúplex completo** (ambos dispositivos podrán transmitir los datos simultáneamente).

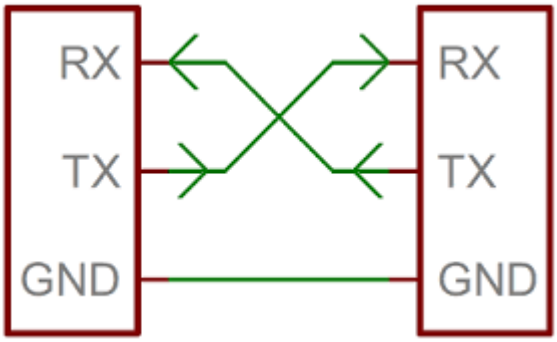


Figura 81. Representación hardware del bus UART. [29]

El protocolo de comunicación UART consta de los siguientes pasos:

El bus TX del emisor estará a nivel alto en reposo. En cuanto la comunicación empiece, se transmitirá el bit START, es decir, el bus TX pasará a un nivel bajo, justo después vendrán entre cinco y ocho bits de datos, un bit de paridad y por último uno o dos bits de STOP, el cual servirá para concluir la comunicación poniendo a nivel alto el bus.

Antes de iniciar la comunicación se tiene que configurar el bus, habrá que elegir el número de bits de datos a transmitir, si se va a enviar o no bit de paridad y el número de bits de STOP que habrá. De igual manera, para poder distinguir entre los diferentes bits se tiene que configurar la velocidad de comunicación la cual se mide en baudios.

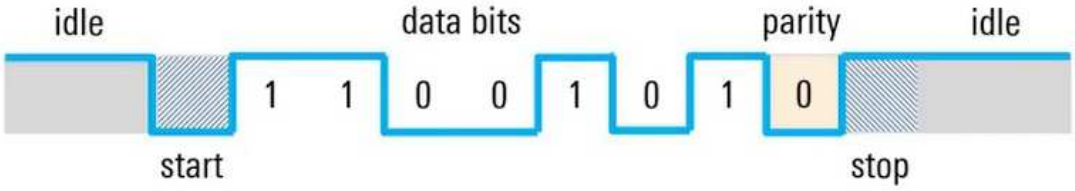


Figura 82. Ilustración del protocolo UART. [34]

6.3. LECTORES RFID

Para activar el automatismo, se acercará una tarjeta RFID al lector. Este le enviará el identificador único de dicha tarjeta a una tarjeta Arduino Mega, que transformará dicho ID en el número de tarjeta, el cual será enviado al cerebro del automatismo (la tarjeta Mini-DK2). Estos lectores se basarán en el módulo **PN532**, que pueden transmitir los datos RFID mediante el bus I^2C, el bus HSU o el bus SPI. En este proyecto, el lector se conectará a la placa Arduino mediante el bus HSU (*High speed UART*) y de ahí al sistema principal mediante el bus I^2C_3.

Para controlar dicho dispositivo se acude al fabricante, *Elechouse*, donde se encuentran 4 librerías: PN532, PN532_HSU, PN532_SPI y PN532_I2C. El primer archivo contendrá ejemplos, en el caso de este trabajo se utiliza de base el archivo ejemplo **iso_14443a_uid**. El software de Arduino (código fuente 1) divide el programa en tres apartados: el primero corresponde con la declaración de variables, el segundo corresponde con la inicialización de las variables y del programa y está bajo la función `void`

`setup(void) {}` y, por último, se encuentra la función principal o main que ejecutará el programa en bucle y se recoge con la función `void loop(void) {}`. Adicionalmente se usan otras funciones que pueden ser llamadas por la función principal o que entrarán a modo de interrupción, en este caso se dispone de la función `void requestEvent() {}`, la cual recogerá la petición de esclavo transmisor de I^{2}C.

Código fuente 1. Programa del lector PN532 para Arduino. [13]

```
/*
 CONEXION CON LECTOR PN532
 1º SPI
 2º HSU <-
 3º I2C
 */
#if 0
  #include <SPI.h>
  #include <PN532_SPI.h>
  #include "PN532.h"
  PN532_SPI pn532spi(SPI, 10);
  PN532 nfc(pn532spi);

/* When the number after #elif set as 1, it will be switch to HSU Mode*/
#elif 1
  #include <PN532_HSU.h>
  #include <PN532.h>
  PN532_HSU pn532hsu(Serial1);
  PN532 nfc(pn532hsu);

/* When the number after #if & #elif set as 0, it will be switch to I2C Mode*/
#else
  #include <Wire.h>
  #include <PN532_I2C.h>
  #include <PN532.h>
  #include <NfcAdapter.h>
  PN532_I2C pn532i2c(Wire);
  PN532 nfc(pn532i2c);
#endif

uint8_t tags[80][4]={ // 1:8 - 9:16 - 17:24 - 25:32 - 33:40 - 41:48 - 49:56 - 57:64 - 65:72 - 73:80
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x1C, 0xD3, 0xEC, 0x5A},{0x00, 0x00, 0x00, 0x00},
{0x33, 0x34, 0x44, 0x92},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x55, 0x7F, 0xA1, 0xA5},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x8A, 0xC6, 0x09, 0xBF},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x9D, 0xBB, 0x34, 0x2D},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0x00, 0x00, 0x00, 0x00},{0xC3, 0x42, 0x61, 0xA7}};
boolean autorizado= false, bloqueo= false;
byte N_TARJETA = 0, cont=0;



/*
 CONEXION CON MINI-DK2
 1º I2C <-
 */
#include <Wire.h>
byte DATO;

void setup(void) {

  // INICIALIZAMOS I2C
  Wire.begin(0x01);              // ARDUINO MODO ESCLAVO DIR=0x01
  Wire.onRequest(requestEvent); // EVENTO DE PETICION DE DATO

  // INICIALIZAMOS VISUALIZADORES
  Serial.begin(9600);            // VISUALIZAMOS POR MOTINOR SERIE
  pinMode(13,OUTPUT);            // VISUALIZAMOS LED

  // INICIALIZAMOS LECTOR PN532
  nfc.begin();
  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
    Serial.print("Didn't find PN53x board");
    while (1); // halt
  }
```

```
  // Got ok data, print it out!
  Serial.print("Found chip PN5"); Serial.println((versiondata>>24) & 0xFF, HEX);
  Serial.print("Firmware ver. "); Serial.print((versiondata>>16) & 0xFF, DEC);
  Serial.print('.'); Serial.println((versiondata>>8) & 0xFF, DEC);

  // Set the max number of retry attempts to read from a card
  // This prevents us from waiting forever for a card, which is
  // the default behaviour of the PN532.
  nfc.setPassiveActivationRetries(0xFF);

  // configure board to read RFID tags
  nfc.SAMConfig();
  Serial.println("ESPERANDO TARJETA");
}

void loop(void) {
  boolean success;
  uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };  // Buffer to store the returned UID
  uint8_t uidLength;                        // Length of the UID (4 or 7 bytes depending on ISO14443A
card type)

  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0], &uidLength);

   if(bloqueo==false){
    if (success) {
      for(int i = 0; i < 81; i++){
        for(int j = 0; j < uidLength; j++){
          if( uid[j] == tags[i][j] ){
            cont++;
            N_TARJETA=(i+1);
          }
        }
      }
      if(cont==4){cont=0; autorizado=true;}
      if(autorizado){
        Serial.println("AUTORIZADO!");
        Serial.println("N_TARJETA: "+(String)N_TARJETA);
        Serial.println("BLOQUEADO");
        digitalWrite(13,HIGH);    //VISUALIZO CON LED QUE SE HA LEIDO
        autorizado=false;
        bloqueo=true;
      }else{
        Serial.println("NO AUTORIZADO");
        bloqueo=false;
      }
      // Wait 1 second before continuing
      delay(1000);
    }
   }
   if(bloqueo==true){
    delay(5000);
    digitalWrite(13,LOW);
    bloqueo=false;
    Serial.println("DESBLOQUEADO");
   }
}
void requestEvent() {
  Wire.write(N_TARJETA); // respond with message of 6 bytes
  Serial.println("ENVIO DATO: "+String(N_TARJETA));
  N_TARJETA=0;
}
```

Lo primero que se hace es escoger el protocolo de comunicación. En este caso, poniendo un 1 en el apartado de HSU, por defecto está seleccionado el bus I^2C. Después se inicializan las variables globales que se usarán, incluyendo el array **tags[80][4]**, la cual guarda 80 identificadores de tarjeta de cuatro bytes cada uno. Lo último será incluir la librería de la comunicación I^2C para el posterior envío.

COMUNICACIÓN UART

El ejemplo **iso_14443a_uid** introduce en la función **setup()** la inicialización de esta comunicación y tanto la búsqueda del lector como su configuración. En la función **loop()** introduce la lectura del identificador único "UID". A esta función se le añade la comparación con el array **tags[j][i]** y se obtiene si la tarjeta es válida o si no pertenece a dicho sistema. Si fuese válida, guarda la posición del tag identificado "**j**" en la variable **N_tarjeta** que representa el número de tarjeta.

Un bloqueo evitará que el lector lea la tarjeta de nuevo hasta pasados cinco segundos.

COMUNICACIÓN I^2C - ESCLAVO

En la función **setup()** se inicializa a la tarjeta Arduino como esclavo I^2C con una dirección fija mediante la función `Wire.begin(0x01)`, en este código la dirección es 1, o 0x01 en hexadecimal. Esta dirección variará de uno a ocho según en qué elevador se coloque dicho lector, pues si se coloca en el cuarto elevador llevará la dirección 0x04.

La función **requestEvent()** se activará al recibir la petición de enviar el dato y mandará el número de tarjeta mediante la función `Wire.write(N_TARJETA)`, después pondrá a cero dicha variable para solo enviarlo una única vez.

COMUNICACIÓN I^2C - MAESTRO

Del lado de la tarjeta Mini-DK2 encontramos el maestro, que, mediante el TIMER 0, mandará la petición de dato (Apartado 6.6.).

El bus utilizado será I^2C_3, pues esta lectura requiere más tiempo que la escritura y se tiene que modificar la librería I2C_libreria_3 (Anexo II)(código fuente 2) añadiendo el siguiente retardo:

Código fuente 2. Función I2Cdelay_3().

```
void I2Cdelay_3(void){
 uint32_t i;
 for(i=0;i<3000;i++);
}
```

El bucle será de 3000 repeticiones en este bus, mientras que en los dos buses anteriores será de 100 ciclos.

6.3. REGISTROS DE PLAZAS LIBRES

El sistema que almacena el estado de las plazas (libre u ocupado) se basa en seis registros de 32 bits, pues el máximo de plazas que el software soporta es de 192, donde cada bit representa una plaza en particular. En la siguiente figura (figura 83) se puede ver la correlación entre el registro y la plaza.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMORIA_6	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161
MEMORIA_5	160	159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129
MEMORIA_4	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97
MEMORIA_3	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65
MEMORIA_2	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
MEMORIA_1	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Figura 83. Plazas del aparcamiento relacionadas con los registros de memoria.

La función **plaza_libre(plaza)** (código fuente 3) se encarga de buscar si una plaza en concreto está libre u ocupada. Para ello, se introduce el número de plaza en su argumento. Al llamarla, la función buscará, en el registro que incluye dicha plaza, un nivel alto o un nivel bajo. Si lo encontrado es un nivel alto, se responderá con un nivel bajo (no, está ocupada) y viceversa (sí, está libre).

La función **sumar_restar_plaza()** (código fuente 3) se encarga de escribir los registros de memoria. Para ello, será necesario introducir el número de plaza y un bit que indique si dicha plaza está libre u ocupada.

Coche aparcado → plaza ocupada → **operación** = 0.

Coche retirado → plaza libre →**operación** = 1.

Dicha función busca el registro de memoria (**Memoria_1:6**), dividiendo la plaza entre el número de bits de los registros, y el bit de dicho registro, restando el número de plaza a los registros completos anteriores, de este modo se obtiene el bit en el que operar.

Código fuente 3. Funciones plaza_libre() y sumar_restar_plaza().

```
uint32_t Memoria_6=0, Memoria_5=0, Memoria_4=0, Memoria_3=0, Memoria_2=0, Memoria_1=0;
        //(192-161)     (160-129)     (128-97)      (96-65)       (64-33)        (32-1)
char plaza_libre(uint8_t plaza){  //PLAZA LIBRE??
   char respuesta;

   if(plaza<33 & plaza>0   ){if(((Memoria_1>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<65 & plaza>32  ){if(((Memoria_2>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<97 & plaza>64  ){if(((Memoria_3>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<129 & plaza>96 ){if(((Memoria_4>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<161 & plaza>128){if(((Memoria_5>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<193 & plaza>160){if(((Memoria_6>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}

   return respuesta; //SI=1 NO=0
}

void sumar_restar_plaza(uint8_t plaza, char operacion){  //operación=0 restar / operación=1 sumar
   char vector_mem, vector_bit,i;

   vector_mem = (plaza/32);            //32 plazas por registro
   vector_bit = plaza-(vector_mem*32);

    if(operacion==1){ //retirado
      Huecoslibres++;
      if(vector_mem==0){Memoria_1 &= ~(1<<vector_bit); }
      if(vector_mem==1){Memoria_2 &= ~(1<<vector_bit); }
      if(vector_mem==2){Memoria_3 &= ~(1<<vector_bit); }
      if(vector_mem==3){Memoria_4 &= ~(1<<vector_bit); }
      if(vector_mem==4){Memoria_5 &= ~(1<<vector_bit); }
      if(vector_mem==5){Memoria_6 &= ~(1<<vector_bit); }
   }
   if(operacion==0){ //aparcado
      Huecoslibres--;
      if(vector_mem==0){Memoria_1 |=  (1<<vector_bit); }
      if(vector_mem==1){Memoria_2 |=  (1<<vector_bit); }
      if(vector_mem==2){Memoria_3 |=  (1<<vector_bit); }
      if(vector_mem==3){Memoria_4 |=  (1<<vector_bit); }
      if(vector_mem==4){Memoria_5 |=  (1<<vector_bit); }
      if(vector_mem==5){Memoria_6 |=  (1<<vector_bit); }
} }
```

6.4. DETECCIÓN DE VEHÍCULO

Este automatismo requiere de un detector de vehículo en cada plataforma elevadora para poder elegir el modo de funcionamiento. Este sistema de detección se basa en un bit por cada plataforma elevadora que indica, con un nivel alto, que se está detectando un vehículo y con un nivel bajo, que se ha dejado de detectar. Para leer dicho bit se dispone de la función **detector_coche(elevador)** (código fuente 4).

El software de este automatismo está diseñado para tener un máximo de 8 elevadores, por lo que cada sistema de detección se recogerá en un expansor de puertos en el bus I^2C_1. Esta función leerá dicho expansor y aislará la patilla correspondiente al elevador, respondiendo con un nivel alto si el sistema está detectando el vehículo y con un nivel bajo si no lo está haciendo. Se puede encontrar la relación entre patilla y elevador en el apartado 5.8.

Código fuente 4. Función detector_coche().

```
char detector_coche(uint8_t elevador){  //COCHE DETECTADO??
   char deteccion, lectura;

   I2CSendAddr_1(DirEP4,1); //leer
   lectura=I2CGetByte_1(1); //NACK, solo ese dato
   I2CSendStop_1();
   I2Cdelay_1();
   if(((lectura >> (elevador-1))&1)==1){deteccion = si;} // SI esta detectando
   else                                {deteccion = no;} // NO esta detectando
   return deteccion;
}
```

6.5. MODOS DE FUNCIONAMIENTO DEL AUTOMATISMO

El sistema puede trabajar en cuatro modos de funcionamiento diferentes: modo aparcar, retirar, “0Coches” y “2Coches”. Los dos primeros son los modos de funcionamiento normales, mientras que los últimos dos son modos de error.

6.5.1. MODOS NORMALES: APARCAR Y RETIRAR

Una vez se conoce el estado de la plaza y del detector, se elige el modo de funcionamiento: "0Coches", "2Coches", aparcar o retirar. En el caso de los dos últimos modos de funcionamiento, se ejecutará de la misma manera, pues el sistema encargado de extraer o introducir el vehículo será externo, el resto del proceso es exactamente igual.

Lo primero que se hará será bloquear el lector para evitar que se ejecute de nuevo la orden. Seguidamente, se actualizará el registro **Ejecut** para que indique que el elevador utilizado está ejecutándose. Este registro se leerá más adelante y activará un aviso lumínico (Apartado 6.11.) y un aviso en el display (Apartado 6.12). Se escribirá un 1 o nivel alto en el bit de **Ejecut** según la siguiente tabla (tabla 25):

Tabla 25. Relación entre los bits del registro EJECUT y los diferentes elevadores.

BITS DE EJECUT	b7	b6	b5	b4	b3	b2	b1	b0
ELEVADOR	ELEV. 8	ELEV. 7	ELEV. 6	ELEV. 5	ELEV. 4	ELEV. 3	ELEV. 2	ELEV. 1

Después, se enviará la petición para activar las medidas de seguridad. Para ello, se llamará a la función **escribir_s_seg(elevador,1)** (código fuente 10), donde se enviará el número de elevador y un bit que active (nivel alto o "1") o desactive (nivel bajo o "0") las medidas de seguridad. Junto con esta función, se activará el flag **flag_marcha[elevador]**, que se encargará de activar un temporizador para detectar si se han aplicado correctamente dichas medidas.

Se carga en el registro global **plaza_En[elevador]** el número de plaza referenciada a su elevador para poder utilizarlo más adelante. También se carga en el registro **movimiento[elevador]** el proceso de aparcar o retirar para poder modificar el valor de las plazas libres, añadiendo la plaza libre o restándola.

6.5.2. MODOS DE ERROR: "0COCHES" Y "2COCHES"

Una vez se conoce el estado de la plaza y del detector, se elige el modo de funcionamiento: "0Coches", "2Coches", aparcar o retirar. En el caso de los dos primeros modos, se responderá enviando a través de el display del lector un pequeño mensaje avisando del error. Para el modo "0Coches", el mensaje será "NO DETECTA COCHE, PLAZA LIBRE" indicado en la primera línea del display que no hay un coche en la plataforma elevadora e indicando en la segunda línea que la plaza referenciada a esa tarjeta está libre, por lo que no se puede retirar el coche. Para el modo "2Coches", el mensaje será "COCHE DETECTADO, PLAZA OCUPADA" indicado en la primera línea del display que hay un coche en la plataforma elevadora e indicando en la segunda línea que la plaza referenciada a esa tarjeta está ocupada, por lo que no se puede aparcar el coche.

Dicho mensaje se enviará mediante la función **ENV_0COCHES(Dirección)** o **ENV_2COCHES(Dirección)** (Apartado 6.12.) y aparecerá en el display un tiempo parametrizable mediante el menú. Dichos valores temporales son 5, 7'5, 10 y 15 segundos (Apartado 6.14.). La temporización de dicho mensaje lo encontramos en el apartado 6.7.

El lector de dicho elevador también será bloqueado, al igual que en los otros modos de funcionamiento, desbloqueándose al finalizar el tiempo de muestra por display.

6.6. TIMER 0

El sistema se activará al pasar la tarjeta RFID por el lector, por lo que se pueden dar cuatro posibles situaciones:

- Se detecta el coche en la plataforma y la plaza donde hay que aparcarlo está libre → Aparcar.
- No se detecta el coche en la plataforma y la plaza donde hay que aparcarlo está ocupada → Retirar.
- Se detecta el coche en la plataforma y la plaza donde hay que aparcarlo está ocupada → Dos coches.
- No se detecta el coche en la plataforma y la plaza donde hay que aparcarlo está libre → Ningún coche.

Estos modos de funcionamiento pondrán en marcha diferentes funciones. En el caso de querer aparcar o retirar el vehículo, primero se activarán las medidas de seguridad, después se transmitirá la información al PLC y, si todo es correcto, se dará por finalizado el proceso. En el caso de no poder retirar o aparcar el vehículo por no detección de este o plaza ocupada, se enviará un mensaje a través del display anexa al lector.

Lo primero que el automatismo hará es leer los registros de las tarjetas Arduino, las cuales contendrán el número de referencia de la tarjeta RFID que ha activado el sistema (Apartado 6.3.). Dicha lectura se hará de manera periódica mediante un temporizador. Una vez se recojan los datos de la lectura (número de plaza) se compara si la plaza a la que se quiere acceder está libre u ocupada y se leerá el registro de detección de coche. Una vez sepamos el estado de la plaza y del sensor de detección, podremos diferenciar entre los cuatro modos de funcionamiento.

Este temporizador se utilizará exclusivamente para la lectura de las tarjetas Arduino y para detectar el modo de funcionamiento, por lo que la velocidad de interrupción no es un requerimiento de gran peso, así que se configurará dicho timer con una frecuencia de interrupción de 10ms.

6.6.1. CONFIGURACIÓN TIMER 0

Cada 10ms, cuando el registro **TC** llegue al valor de *MR0* (249.999), el temporizador interrumpirá y se reiniciará, generando una cadena de interrupciones periódicas.

Se necesitará alimentar al timer (registro **PCONP**) y habilitar la interrupción (registro **NVIC**) para que el programa ejecute las funciones asignadas.

Código fuente 5. Configuración del TIMER 0

```
#include <lpc17xx.h>
#define Fpclk 25e6

uint32_t Ttim0=10000, Ftic0=0;

void config_timer0(void){ //Configuracion TIMER0

   LPC_SC->PCONP|=(1<<1);              //Timer0 ON
   LPC_TIM0->PR=0;                     //Resolucion de 40 ns
   Ftic0=(Fpclk/(LPC_TIM0->PR + 1));
   LPC_TIM0->MCR=0x3;                  //Interrupt ON MR0, Reset ON MR0
   LPC_TIM0->MR0=(Ftic0*Ttim0*1e-6)-1; //Timer0 interrumpe cada 10 ms
   NVIC_EnableIRQ(TIMER0_IRQn);        //Habilitacion Timer0
   NVIC_SetPriority(TIMER0_IRQn,4);    //Timer0 tiene prioridad 1 subprioridad 0
}
```

6.6.2. INTERRUPCIÓN TIMER 0

La lectura de las tarjetas de Arduino se basa en el intercambio de información por el bus I^2C_3. En la variable **plaza** se recogerá un número (en este proyecto estará comprendido entre el 1 y el 80, ambos incluidos), dicha variable será introducida en la función **plaza_libre()** obteniendo en la variable **respuesta** si dicha plaza está ocupada (**respuesta**=si=1) o si está libre (**respuesta**=no=0). Acto seguido, se leerá mediante la función **detector_coche(elevador)** si en dicho elevador se detecta un coche, al igual que en la función anterior obtendremos una respuesta, **deteccion**=si=1, **deteccion**=no=0.

Para poder identificar el elevador correspondiente, se ejecutará la interrupción del TIMER 0 como si de un bucle se tratase. Para ello, se inicializa la variable **cont_TIM0**, la cual se incrementará cada vez que el timer interrumpa. En caso de que dicha variable supere el número de elevadores (**NE**), se reiniciará.

Existirá un bloqueo mediante el registro **bloquear_lector[elevador]** para evitar que se envíe la misma plaza varias veces dentro de la ejecución de la primera.

Código fuente 6. Función interrupción del TIMER 0

```
#include <lpc17xx.h>
#define Fpclk 25e6
#define si 1
#define no 0
uint8_t NE=4; //numero elevadores
uint8_t DirAn[8]={0x01,0x02,0x03,0x04,0x05,0x06, 0x07,0x08};//ARDUINOS
uint8_t cont_TIM0=0,plaza=0,elevador=0;
char respuesta, deteccion;
```

```
char bloquear_lector[8]={0,0,0,0,0,0,0,0};
void TIMER0_IRQHandler(void){ //Interrupcion TIMER0
   static char dir,i;

   LPC_TIM0->IR|=(1<<0); //Borrar flag de MR0
   cont_TIM0_2++;
   if(cont_TIM0_2==200){
      cont_TIM0_2=0;
      cont_TIM0++; //cada cuenta es un lector diferente
      LPC_TIM0->TCR=0x00; //counter disable para garantizar la ejecución

      dir=DirAn[cont_TIM0-1];
      I2CSendAddr_3(dir,1);
      plaza=I2CGetByte_3(1);
      I2CSendStop_3();
      I2Cdelay_3();

      if(bloquear_lector[cont_TIM0-1]==0){
         if(plaza!=0){ //Si el elevador n no ha sido activado dará plaza=0
            elevador=cont_TIM0;
            respuesta=plaza_libre(plaza);
            deteccion=detector_coche(elevador);
         if(respuesta==si & deteccion==no){ //plaza libre y coche no detectado --> 0 coches
            for(i=0;i<8;i++){
               if((elevador-1)==i){
                  bloquear_lector[i]=1;
                  flag_coches[i]=1;
                  ENV_0COCHES(DirPn[i]);
         }}}
         if(respuesta==no & deteccion==si){ //plaza ocupada y coche detectado --> 2 coches
            for(i=0;i<8;i++){
               if((elevador-1)==i){
                  bloquear_lector[i]=1;
                  flag_coches[i] =1;
                  ENV_2COCHES(DirPn[i]);
         }}}
         if((respuesta==no & deteccion==no)|(respuesta==si & deteccion==si)){ //retirar o aparcar
            bloquear_lector[elevador-1]=1;
            Ejecut |= (1<<(elevador-1));
            escribir_s_seg(elevador,1);      //ACTIVO (=1) los sistemas de seguridad
            flag_marcha[elevador-1]=1;
            plaza_En[elevador-1]=plaza;
            if(respuesta==1){movimiento[elevador-1]=0;}//aparcar
            else{movimiento[elevador-1]=1;}//retirar
      }  }}
      if(plaza==0){elevador=0;}
      LPC_TIM0->TCR=0x01;//counter ensable
      if(cont_TIM0==NE){cont_TIM0=0;}
   }
}
```

6.7. MENSAJE DE ERROR TEMPORIZADO

Cuando un elevador se encuentra en los modos “0Coches” o “2Coches”, el sistema deberá de enviar a través del display de su lector un mensaje que avise de ello (Apartado 6.5.2). Este mensaje deberá desaparecer transcurrido un tiempo parametrizable (Apartado 6.14.). Para ello, se recurre a la función de interrupción del TIMER 1 (código fuente 7). Cuando **flag_coches[elevador]** sea igual a 1 (ocurrirá al introducirse en dicho modo de funcionamiento) empezará la cuenta de 5, 7’5, 10 o 15 segundos (**P_mensaje**). Una vez transcurrido el tiempo, se dejará el elevador en modo de reposo o disponible.

Código fuente 7. Fragmento de la función de interrupción de TIMER 1, aviso por display temporizado

```
if(cont_TIM1==1000){  //PARA 0COCHES O 2COCHES --> PONER ERROR Y QUITARLO AL CABO DE UN TIEMPO
      cont_TIM1=0;
      for(i=0;i<8;i++){
         if(flag_coches[i]==1){cont_coches[i]=0;flag_coches[i]=2;}
         if(flag_coches[i]==2){
            cont_coches[i]++;
            if(flag_coches[0]==2){LPC_GPIO0->FIOPIN =(1<<25);}
            if(cont_coches[i]==P_mensaje){
               if(flag_coches[0]==2){LPC_GPIO0->FIOPIN =(~1<<25);}
               bloquear_lector[0]=1;flag_coches[i]=0;modo_disponible[i]=1;}
   }  }  }
```

6.8. SISTEMA DE SEGURIDAD

En cada plataforma elevadora habrá un sistema de seguridad que garantice que ninguna persona o animal entre en el módulo móvil. Dicho sistema de seguridad deberá ser activado o desactivado y deberá enviar una respuesta, que se contrastará para saber si se ha vulnerado la seguridad y evitar daños mayores. Si como máximo hay ocho elevadores, habrá los mismos sistemas de seguridad, por tanto, se usarán dos expansores de 8 puertos, uno para mandar la señal de activación y el otro para recibir la señal de respuesta.

Para activar o desactivar dicho sistema se dispone de la función **escribir_s_seg(elevador, estado)** (código fuente 8), donde al llamar a la función se necesita introducir en su argumento el elevador en cuestión y si se quiere activar o desactivar. Dependiendo de esto último, se sobrescribirá un nivel alto o un nivel bajo en el expansor de puertos mediante el bus I^2C_1.

Código fuente 8. Función escribir_s_seg().

```
void escribir_s_seg(char elevador, char estado){  //ACTIVAR/DESACTIVAR SISTEMAS DE SEGURIDAD
   static char dato;                              //estado=0 -> desactivar  /  estado=1 -> activar

   if(estado){dato |=  (1<<(elevador-1));} //sobrescribir 1
   else       {dato &= ~(1<<(elevador-1));}  //sobrescribir 0
   I2CSendAddr_1(DirEP2,0); //escribir en Expansor Puertos 2
   I2CSendByte_1(dato);
   I2CSendStop_1();
   I2Cdelay_1();
}
```

Para leer dicho sistema se usará la función **leer_s_seg(elevador)** (código fuente 9)donde, al llamar a la función, se necesita introducir en su argumento el elevador. Esta función leerá el otro expansor de puertos mediante el bus I^2C_1 y responderá con un bit recogido en la variable local **aplicado**. Si se corresponde con un nivel alto significa que se han aplicado correctamente dichos sistemas de seguridad, en cambio, si se corresponde con un nivel bajo significa que ha ocurrido algún problema y saltará el estado avería.

Código fuente 9. Función leer_s_seg().

```
char leer_s_seg(char elevador){  //LEER APLICADO O NO SISTEMAS DE SEGURIDAD
   char dato, aplicado;  //aplicado=0 -> no (AVERIA)  /  aplicado=1 -> si (CORRECTO)

   I2CSendAddr_1(DirEP3,1); //lectura en EP3
   dato=I2CGetByte_1(1);    //NACK solo ese dato
   I2CSendStop_1();
   I2Cdelay_1();
   if(((dato>>(elevador))&1)==1){aplicado=1;}    //dato = E7 E6 E5 E4 E3 E2 E1 E0
   else                         {aplicado=0;}    //elevador = (1:8) --> restar 1
   return aplicado;
}
```

6.9. SISTEMA DE CONTROL DE PLC'S

El sistema de PLC'S de los motores necesita que se envíen los datos necesarios para situar el elevador en la planta correcta y alinear la división de la estructura rotativa o hueco con dicho elevador, este responderá con una señal para poder concluir el proceso o para detectar averías.

6.9.1. ENVÍO DE DATOS AL PLC

El PLC del elevador necesita recoger la planta a la que debe situarse, el PLC de la estructura rotativa necesitará recibir el número de elevador y el hueco o división de dicha planta, para enfrentarlos. Para ello se dispone de dos expansores de puertos (**EXP 6** y **EXP 7**) los cuales enviarán según la siguiente imagen los datos:

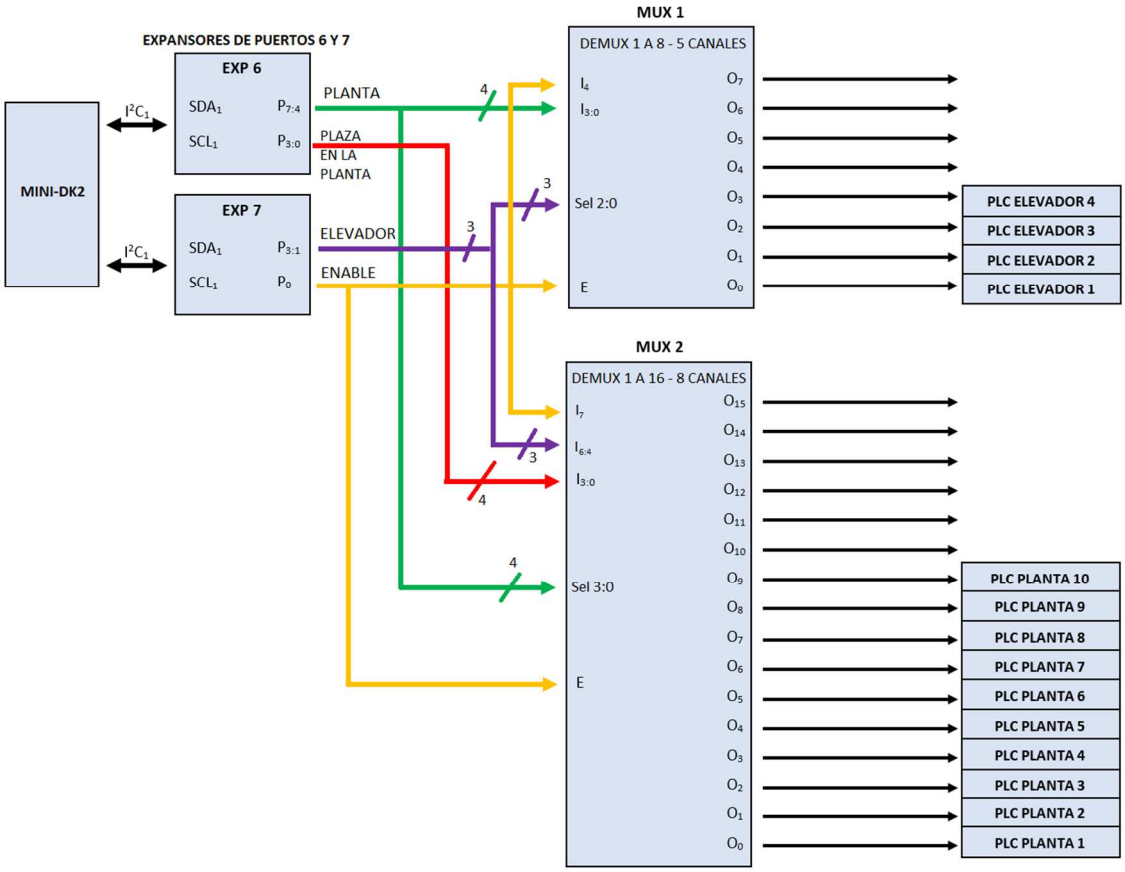


Figura 84. Gráfico de la conexión de los PLC's con el microcontrolador.

La figura 84 recoge la conexión hardware del microcontrolador con los PLC'S, como se observa se enviarán los datos a un sistema de control externo basado en un multiplexor para derivar los datos a los PLC's correspondientes. El software está preparado para recoger hasta 8 elevadores y 16 plataformas giratorias, por lo que, dicho control también lo estará.

Lo primero que se busca es almacenar los datos en cuestión en un registro, para no generar fallos si hubiese dos elevadores que accediesen a la misma planta.

Esto se logra cargando la plaza en concreto y el elevador (el número de plaza tiene implícito el hueco y la planta) en un registro de load llamando a la función **ESPERA(elevador)** (código fuente 10) y cargando la plaza en el registro **plaza_En[elevador]**. Inmediatamente después se llamará a la función **Elevador()**.

Código fuente 10. Función ESPERA().

```
void ESPERA(char elev){ //CARGAR A LOAD EL DATO
   char planta, hueco;
   uint8_t dato;

   planta=(plaza_En[elev]-1)/Huecos_planta;
   hueco=plaza_En[elev]-1-(planta*Huecos_planta);
   LOAD_dato=((planta<<4)|hueco);
   LOAD_elev=elev;
}
```

La función **Elevador()** se encarga de leer el registro Load anterior y comparar la planta a la que se quiere llegar con un registro que muestra que planta está en uso (**PLANTA_USO[planta]**), si la estructura giratoria está libre llamaremos a la función **MARCHA(plaza, elevador)** (código fuente 12), si la estructura estuviese ocupada guardaríamos la plaza en el registro de espera **REG_dato_esp()**.

La función **Elevador()** (código fuente 11) será llamada en cada interrupción de TIMER 1 para sacar cuanto antes las plazas en espera. El registro de espera es un registro circular, lo que quiere decir es que si una plaza que está en espera se vuelve a verificar y sigue en espera esta irá al final del registro para poder leer las demás plazas en espera.

Código fuente 11. Función Elevador().

```
void Elevador(){ //DESCARGAR DE LOAD EL DATO Y PONER EN MARCHA SI LA PLANTA ESTA LIBRE
   char j,planta;
   char a,b;

   if(REG_dato_esp[0]!=0xFF){                          //PRIMERO MIRAMOS SI HAY PLAZAS PENDIENTES
      if(PLANTA_USO[((REG_dato_esp[0]>>4)&0xF)]==0){ //PLANTA LIBRE
         MARCHA(REG_dato_esp[0],REG_elev_esp[0]);
         PLANTA_USO[((REG_dato_esp[0]>>4)&1)]=1;      //PLANTA OCUPADA
         for(j=0; j<8; j++){                          //REPOSICIONAMOS [0]
            REG_dato_esp[j]=REG_dato_esp[j+1];
            REG_elev_esp[j]=REG_elev_esp[j+1];
            if(j==7){REG_dato_esp[j]=0xFF;REG_elev_esp[j]=0xFF;}
   }  }  }

   //mover vector de espera
   a=REG_dato_esp[0];
   b=REG_elev_esp[0];
   for(j=0; j<8; j++){           //REPOSICIONAMOS [0]
      REG_dato_esp[j]=REG_dato_esp[j+1];
      REG_elev_esp[j]=REG_elev_esp[j+1];
      if(j==7){REG_dato_esp[j]=a;REG_elev_esp[j]=b;}
   }

   if(REG_dato[0]!=0xFF){                          //SEGUNDO MIRAMOS SI HAY PLAZAS NUEVAS
      if(PLANTA_USO[((REG_dato[0]>>4)&0xF)]==1){ //PLANTA EN USO
         for(j=0;j<8;j++){
            if(REG_dato_esp[j]==0xFF){REG_dato_esp[j]=REG_dato[0];REG_dato[0]=0xFF;}
            if(REG_elev_esp[j]==0xFF){REG_elev_esp[j]=REG_elev[0];REG_elev[0]=0xFF;}
   }   }  }

   if(REG_dato[0]!=0xFF){
      if(PLANTA_USO[((REG_dato[0]>>4)&0xF)]==0){ //planta libre
         MARCHA(REG_dato[0],REG_elev[0]);
         PLANTA_USO[((REG_dato[0]>>4)&0xF)]=1; }
   }

   for(j=0; j<8; j++){ //desplazamiento a derecha
      REG_dato[j]=REG_dato[j+1];
      REG_elev[j]=REG_elev[j+1];
      if(j==7){REG_dato[j]=0xFF;REG_elev[j]=0xFF;}
   }
   REG_dato[7]=LOAD_dato; // DESCARGAMOS DATO
   LOAD_dato=0xFF;
   REG_elev[7]=LOAD_elev;
   LOAD_elev=0xFF;

}
```

Una vez la estructura giratoria esté liberada y se pueda mandar la orden al PLC se llamará a la función **MARCHA(plaza, elevador)** (código fuente 12), la cual enviará al expansor de puertos 6 la plaza (planta y hueco) y se llamará a la función **Seleccion_elevador(elevador)** para enviar al expansor de puertos 7 el elevador y el bit de enable que mandará los datos a los PLC correspondientes.

Código fuente 12. Funciones MARCHA() y Seleccion_elevador().

```
void MARCHA(uint8_t dato,char elev){ //enviar hueco y planta
   char i;

   I2CSendAddr_1(DirEP5,0); //Write
   I2CSendByte_1(dato);
   I2CSendStop_1();
   I2Cdelay_1();
   Seleccion_elevador(elev); //elev (0:7)
   flag_respE[elev]=1;
}
void Seleccion_elevador(char sel){
   char enable, disable,i;

   enable = (sel<<1)|1;
   disable = (sel<<1)|1;
   I2CSendAddr_1(DirEP6,0); //Write EXP7
   I2CSendByte_1(enable);   //SE ENVIA EL ENABLE CON LA DIRECCIÓN
   I2CSendStop_1();

   for(i=0;i<C_ENABLE;i++){I2Cdelay_1();}

   I2CSendAddr_1(DirEP6,0); //Write EXP7
   I2CSendByte_1(disable);  //SE QUITA EL ENABLE TRAS LOS i CICLOS DE ESPERA
   I2CSendStop_1();
   I2Cdelay_1();
}
```

6.9.2. RESPUESTA DEL PLC

Para obtener la respuesta del PLC y dar por concluido el proceso o avisar de una avería se llamará a la función **respuesta_PLC(elevador)** (código fuente 13) cada cierto tiempo, este tiempo puede ser parametrizable como ya se ha visto anteriormente.

Al llamar a la función se leerá cada una de las entradas o conexiones con los PLC'S, en cuanto se obtenga un nivel alto de una patilla empezará el ciclo de lectura de la señal, se activará un contador que contará 12 ciclos temporales, en dichos 12 ciclos debería de haber o 8 niveles altos para avería o 6 niveles altos para funcionamiento correcto, transcurrido dicho tiempo se clasificará entre correcto, avería de la estructura giratoria, avería de elevador o avería de elevador y estructura, esta función responderá con un número (1, 2, 3 o 4) según la avería o correcto.

Código fuente 13. Función respuesta_PLC()

```
char respuesta_PLC(char i){ // entrará cada 200*P_respE
   static char correcto, planta[8], cont_elev[8], cont_elev_NA[8], correcto_elevador[8],
               cont_planta[8], cont_planta_NA[8], correcto_planta[8];

//LEER PATILLA ELEVADOR   // E8=P1.25 E7=P1.24 E6=P1.23 E5=P1.22 E4=P1.21 E3=P1.20 E2=P1.19 E1=P1.18
   if(cont_elev[i]==0){
      if(((LPC_GPIO1->FIOPIN>>(18+i))&1)==1){
         cont_elev_NA[i]++;      // primer nivel alto
         cont_elev[i]=1;
   }  }
   else{ //ya ha venido el primer nivel alto
      cont_elev[i]++;
      if(((LPC_GPIO1->FIOPIN>>(18+i))&1)==1){
         cont_elev_NA[i]++;      // sumamos cont_correcto cada vez que haya un nivel alto
   }  }

   if(cont_elev[i]==12){
      if(cont_elev_NA[i]==6){cont_elev[i]=0;correcto_elevador[i]=1;}//correcto
      if(cont_elev_NA[i]==8){cont_elev[i]=0;correcto_elevador[i]=2;}//averia
   }

//LEER PATILLA PLANTA
   planta[i]=plaza_En[i]/Huecos_planta; // P15=P1.17  P14=P1.16  P13=P1.15  P12=P1.14
                                        // P11=P1.10  P10=P1.9   P9=P1.8    P8=P1.4
                                        // P3=P0.7    P2=P0.6    P1=P0.5    P0=P0.4
   if(cont_planta[i]==0){
      if(planta[i]<6){ // planta 0,1,2,3,4,5
         if(cont_planta[i]==0){
            if(((LPC_GPIO0->FIOPIN >>(4+planta[i]))&1)==1){
               cont_planta_NA[i]++;// primer nivel alto
               cont_planta[i]=1;
         }  }
         else{ //ya ha venido el primer nivel alto
            cont_planta[i]++;
            if(((LPC_GPIO0->FIOPIN >>(4+planta[i]))&1)==1){
               cont_planta_NA[i]++;         // sumamos cont_correcto cada vez que haya un nivel alto
      }  }  }
      if(planta[i]>5 & planta[i]<8){ // planta 6,7  P7=P1.1    P6=P1.0
         if(cont_planta[i]==0){
            if(((LPC_GPIO1->FIOPIN >>(0+planta[i]-6))&1)==1){
               cont_planta_NA[i]++;         // primer nivel alto
               cont_planta[i]=1;
         }  }
         else{ //ya ha venido el primer nivel alto
            cont_planta[i]++;
            if(((LPC_GPIO1->FIOPIN >>(0+planta[i]-6))&1)==1){
               cont_planta_NA[i]++;         sumamos cont_correcto cada vez que haya un nivel alto
      }  }  }
      if(planta[i]==8){ // planta 8    P8=P1.4
          if(cont_planta[i]==0){
             if(((LPC_GPIO1->FIOPIN >>(4))&1)==1){
                cont_planta_NA[i]++;        // primer nivel alto
                cont_planta[i]=1;
         }  }
         else{ //ya ha venido el primer nivel alto
            cont_planta[i]++;
            if(((LPC_GPIO1->FIOPIN >>(4))&1)==1){
               cont_planta_NA[i]++; // sumamos cont_correcto cada vez que haya un nivel alto
      }  }  }
      if(planta[i]>8 & planta[i]<12){ // planta 9,10,11   P11=P1.10  P10=P1.9  P9=P1.8
         if(cont_planta[i]==0){
            if(((LPC_GPIO1->FIOPIN >>(8+planta[i]-9))&1)==1){
               cont_planta_NA[i]++;         // primer nivel alto
               cont_planta[i]=1;
         }  }
         else{ //ya ha venido el primer nivel alto
            cont_planta[i]++;
            if(((LPC_GPIO1->FIOPIN >>(8+planta[i]-9))&1)==1){
               cont_planta_NA[i]++;  // sumamos cont_correcto cada vez que haya un nivel alto
```

```
      }  }  }
      if(planta[i]>11){ // planta 12,13,14,15  P15=P1.17  P14=P1.16  P13=P1.15  P12=P1.14
         if(cont_planta[i]==0){
            if(((LPC_GPIO1->FIOPIN >>(14+planta[i]-12))&1)==1){
               cont_planta_NA[i]++;         // primer nivel alto
               cont_planta[i]=1;
         }  }
         else{ //ya ha venido el primer nivel alto
            cont_planta[i]++;
            if(((LPC_GPIO1->FIOPIN >>(14+planta[i]-12))&1)==1){
               cont_planta_NA[i]++;  // sumamos cont_correcto cada vez que haya un nivel alto
   }  }  }  }
   if(cont_planta[i]==12){
      if(cont_planta_NA[i]==6){cont_planta[i]=0;correcto_planta[i]=1;}//correcto
      if(cont_planta_NA[i]==8){cont_planta[i]=0;correcto_planta[i]=2;}//averia
   }


   if(correcto_planta[i]==1 & correcto_elevador[i]==1){correcto=1;} //CORRECTO
   if(correcto_planta[i]==1 & correcto_elevador[i]==2){correcto=2;} //AVERIA ELEVADOR
   if(correcto_planta[i]==2 & correcto_elevador[i]==1){correcto=3;} //AVERIA PLANTA
   if(correcto_planta[i]==2 & correcto_elevador[i]==2){correcto=4;} //AVERIA AMBOS

   return correcto; //correcto=1 averiado_ELEVADOR=2 averiado_PLANTA=3 averiado_TODO=4
}
```

6.10. TIMER 1

Una vez se ha detectado la tarjeta RFID, leído el número de plaza y obtenido el modo de funcionamiento, se sigue con los procesos en el TIMER 1, ya que este se encarga de la temporización del aviso en el display de los modos “0Coches” y “2Coches”, además de leer la activación de los sistemas de seguridad y recoger la respuesta del sistema móvil.

6.10.1. CONFIGURACIÓN TIMER 1

Cada 10us, cuando el registro **TC** llegue al valor de *MR0* (249), el temporizador interrumpirá y se reiniciará, generando una cadena de interrupciones periódicas.

Se necesitará alimentar al timer (registro **PCONP**) y habilitar la interrupción (registro **NVIC**) para que el programa ejecute las funciones asignadas.

Código fuente 14. Configuración del TIMER 1.

```
#include <lpc17xx.h>
#define Fpclk 25e6

uint32_t Ttim1=10, Ftic1=0;

void config_timer1(void){ //Configuracion TIMER1

   LPC_SC->PCONP|=(1<<2); //Timer1 ON
   LPC_TIM1->PR=0; //resolucion de 40 ns
   Ftic1=(Fpclk/(LPC_TIM1->PR + 1));
   LPC_TIM1->MCR=0x3; //Interrupt ON MR0, Reset ON MR0
   LPC_TIM1->MR0=(Ftic1*Ttim1*1e-6)-1; //Timer1 interrumpe cada 10 us
   NVIC_EnableIRQ(TIMER1_IRQn); //Habilitacion Timer1
   NVIC_SetPriority(TIMER1_IRQn,4); //Timer1 tiene prioridad 1 subprioridad 0
}
```

6.10.2. INTERRUPCIÓN TIMER 1

La función de interrupción del TIMER 1 recoge la temporización del mensaje de aviso en el modo “0Coches” o “2Coches”, la temporización de la respuesta de los sistemas de seguridad, la recogida de la respuesta de los PLC’s y la ejecución de la función que envía al PLC la plaza para aparcar o retirar.

Lo primero que se encuentra es la parte del código que temporiza la aparición del mensaje de aviso en el modo “0Coches” o “2Coches”. Para ello, cuando **flag_coches[elevador]** del elevador que se esté utilizando esté a nivel alto o “1” (puesto así con anterioridad al seleccionar el modo de funcionamiento), se empezará a contar un tiempo parametrizable (**P_mensaje**) de 5, 7’5, 10 o 15 segundos. Al acabar la cuenta volverá el elevador a estado de reposo.

Lo siguiente que se encuentra es la lectura del sistema de seguridad. En caso de estar en el modo de aparcar o retirar el vehículo, **flag_marcha[elevador]** se pondrá a nivel alto. Pasado un tiempo parametrizable (**P_sseg**) de 5, 10, 15 o 20 segundos, se leerá la respuesta de los sistemas de seguridad. En el caso de que se hayan aplicado correctamente, se continuará con el proceso enviando los datos al

PLC mediante las funciones **ESPERA(elevador)** y **Elevador()**. En caso contrario, se activará el registro de avería que activará la luz amarilla fija y mensaje de avería en el display.

Dentro del mismo bucle "for" encontramos la lectura de la respuesta del PLC. Del mismo modo que en los dos casos anteriores, transcurrirá un tiempo desde que se active **flag_respE[elevador]**. Dicho tiempo será el que haya pasado entre cada lectura de la respuesta del PLC (en la imagen inferior es el tiempo transcurrido entre las barras azules), siendo el periodo de la señal correcta dos veces este tiempo y cuatro veces el de la señal de avería. Dicha lectura de la señal de respuesta se hace llamando a la función **respuesta_PLC(elevador)** (código fuente 13), la cual responde con un byte recogido en la variable **correcto[elevador]** donde, si es igual a uno, significará que todo ha sido correcto, si es igual a dos, se ha producido una avería en el elevador, igual a tres significa avería en la rotación de la estructura de almacenamiento e igual a cuatro es una avería tanto en el elevador como en la planta. Solo en caso de avería del elevador se podrá dejar libre la estructura rotativa para seguir usándola. En caso de que todo sea correcto, se llamará a la función **sumar_restar_plaza()** (código fuente 3) para actualizar el valor de las plazas libres y volverá a estado de reposo.

La respuesta del PLC tendrá la forma de la siguiente figura (figura 85):

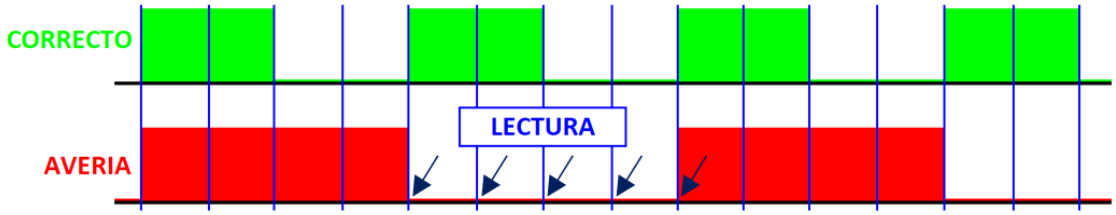


Figura 85. Forma de la señal respuesta del PLC.

PLC y microcontrolador han de sincronizarse. Para ello, dicho tiempo será parametrizable (**P_respE**) entre los valores 0'1, 0'2, 0'5, 1 y 2 segundos.

Por último, se encuentra la llamada a la función **Elevador()** (código fuente 11), la cual se ejecutará cuando el TIMER 1 interrumpa, sin ningún *prescaler*. Esta función se encarga de enviar los datos a los PLC (Apartado 6.9.1.).

Código fuente 15. Función interrupción del TIMER 1

```
void TIMER1_IRQHandler(void){ //Interrupcion TIMER1
   static char i,j,k,planta_elev[8];
   static uint32_t cont_Sseg[8]={0,0,0,0,0,0,0,0},cont_Elev[8]={0,0,0,0,0,0,0,0};
   LPC_TIM1->IR|=(1<<0); //Borrar flag de MR0
   cont_TIM1++;
   cont_TIM1_2++;
   if(cont_TIM1==1000){  //PARA 0COCHES O 2COCHES --> PONER ERROR Y QUITARLO AL CABO DE UN TIEMPO
      cont_TIM1=0;
      for(i=0;i<8;i++){
         if(flag_coches[i]==1){cont_coches[i]=0;flag_coches[i]=2;}
         if(flag_coches[i]==2){
            cont_coches[i]++;
            if(cont_coches[i]==P_mensaje){
               bloquear_lector[0]=1;flag_coches[i]=0;modo_disponible[i]=1;}
   }  }  }
   if(cont_TIM1_2==200){
      cont_TIM1_2=0;
      for(i=0;i<8;i++){
         if(flag_marcha[i]==1){ //LEER SI SE HAN APLICADO LOS S.SEG
            cont_Sseg[i]++;
            if(cont_Sseg[i]==P_sseg){
               cont_Sseg[i]=0;
               flag_marcha[i]=0;
               if(leer_s_seg(i)==si){//se han aplicado
                  ESPERA(i);
                  Elevador();}
               else{ // no se han aplicado
                  escribir_s_seg((i+1),0);      //DESACTIVO (=0) los sistemas de seguridad
                  Averia |= (1<<i);
                  Ejecut &= ~(1<<(i));
         }  }  }
         if(flag_respE[i]==1){ //RECOGER LA RESPUESTA DEL PLC
            cont_Elev[i]++;
            if(cont_Elev[i]==P_respE){
               flag_respE[i]=0;
               cont_Elev[i]=0;
```

```
                correcto[i]=respuesta_PLC(i); // entrará cada 200*P_respE   correcto=1 averiado=2
                planta_elev[i]=plaza_En[i]/Huecos_planta;
                escribir_s_seg((i+1),0);     //DESACTIVO (=0) los sistemas de seguirdad
                if(correcto[i]==1){ //correcto
                   Ejecut &= ~(1<<(i));
                   correcto[i]=0;
                   bloquear_lector[i]=0;
                   PLANTA_USO[planta_elev[i]]=0;
                   sumar_restar_plaza(plaza_En[i], movimiento[i]);
                   modo_disponible[i]=1;
                   FLAGS();
                   modos();}
                if(correcto[i]==2){ //averia  ELEVADOR
                   Ejecut &= ~(1<<i);
                   Averia |=  (1<<i);
                   PLANTA_USO[planta_elev[i]]=0; }
                if(correcto[i]==3){ //averia  PLANTA
                   Ejecut &= ~(1<<i);
                   Averia |=  (1<<i); }
                if(correcto[i]==4){ //averia  PLANTA Y ELEVADOR
                   Ejecut &= ~(1<<i);
                   Averia |=  (1<<i); }
   }  }  }  }
   Elevador(); //EJECUTAR PLAZAS EN ESPERA (DOS PLAZAS CON LA MISMA PLANTA)
}
```

6.11. AVISO LUMÍNICO

Este aparcamiento dispone de tres lámparas por cada plataforma elevadora que indicarán mediante su color y parpadeo los diferentes estados del elevador. En la siguiente tabla (tabla 26) vemos la relación entre estado de funcionamiento y lámpara.

Tabla 26. Estado de la plataforma elevadora frente al color de luz y parpadeo de dicho elevador.

ESTADO DE LA PLATAFORMA ELEVADORA	AVISO LUMÍNICO
Disponible, se puede aparcar o retirar un vehículo	Luz verde fija
Modo en ejecución, se está aparcando o retirando un vehículo	Luz amarilla parpadeante
En avería, o bien por error en el PLC del elevador, de la estructura giratoria o por los sistemas de seguridad	Luz amarilla fija
Aparcamiento lleno, solo se pueden retirar coches	Luz roja fija

Estas bombillas serán accionadas por diferentes señales de control, las cuales se generan mediante un expansor de puertos controlado por I2C1 (en el caso de luz verde y amarilla) y una patilla GPIO en el caso de la luz roja mediante la función **salida()**.

Dicha función simplemente invierte el nivel de salida, pues la señal ha de estar a nivel bajo, y envía al expansor de puertos correspondiente vía I^2C_1 la señal o en el caso de la luz roja por el puerto GPIO.

Código fuente 16. Función salidas().

```
void salidas(void){  //ACTUALIZA EL VALOR DE LAS LUCES

   Luzroja_n=~Luzroja;           //activo a nivel bajo
   Luzamarilla_n=~Luzamarilla;
   Luzverde_n=~Luzverde;

   //ROJO P2.9
   LPC_GPIO2->FIOPIN =((LPC_GPIO2->FIOPIN & ~(1<<9))|((((Luzroja_n)&1) & 0x1) << 9));
   //AMARILLO EXPANSOR 1
   I2CSendAddr_1(DirEP0,0); //direccion, escribir
   I2CSendByte_1(Luzamarilla_n); //enviar salidas de pines
   I2CSendStop_1();
   I2Cdelay_1(); //garantizar la escritura.

   //VERDE EXPANSOR 2
   I2CSendAddr_1(DirEP1,0); //direccion, escribir
   I2CSendByte_1(Luzverde_n); //enviar salidas de pines
   I2CSendStop_1();
   I2Cdelay_1(); //garantizar la escritura.
}
```

6.12. AVISO POR DISPLAY

Existen diferentes modos de funcionamiento del automatismo, para cada estado o modo se enseñará un pequeño mensaje o aviso. Estos mensajes se mostrarán por el display LCD **1602A** que se encontrará anexa al lector y por el display **HY28A** integrado en la tarjeta Mini-DK2.

Respecto al display LCD **1602A** se encuentran seis mensajes posibles: **ENV_LLENO()**, **ENV_DISPONIBLE(dir)**, **ENV_EJECUT(dir)** y **ENV_AVERIA(dir)** que se recogen en la función **modos()** (código fuente 17) llamada por el TIMER 3, y **ENV_0COCHES(dir)** y **ENV_2COCHES(dir)** llamadas por el temporizador 0 cuando entra el elevador en modo "0Coches" o "2Coches". Para hacer más compacto el código principal se ideará un archivo o librería que recoja todas las funciones posibles encargadas de enviar dicho mensaje ("**1602A_Libreria.c**").

Respecto a la función **modos()** (código fuente 17) es llamada en cada interrupción del TIMER 3, en el caso de estar todos las plazas llenas enviará por todos los elevadores el mensaje, en el caso contrario enviará a cada elevador su estado de funcionamiento.

Código fuente 17. Función modos().

```
void modos(){
   char i;

   if(modo_lleno==1){LPC_TIM3-
>TCR=0x00;Huecos_libres(Huecoslibres);ENV_LLENO();modo_lleno=0;LPC_TIM3->TCR=0x01;}

   for(i=0;i<8;i++){
      if(modo_disponible[i]==1)  {LPC_TIM3->TCR=0x00; Huecos_libres(Huecoslibres); menu_lleno();
ENV_DISPONIBLE(DirPn[i]); modo_disponible[i]=0;        LPC_TIM3->TCR=0x01;}

      if(modo_ejecut[i]==1)      {LPC_TIM3->TCR=0x00; Huecos_libres(Huecoslibres); menu_lleno();
ENV_EJECUT(DirPn[i]);              modo_ejecut[i]=0;   LPC_TIM3->TCR=0x01;}

      if(modo_averia[i]==1)      {LPC_TIM3->TCR=0x00; Huecos_libres(Huecoslibres); menu_lleno();
ENV_AVERIA(DirPn[i]);              modo_averia[i]=0;   LPC_TIM3->TCR=0x01;}
          }
}
```

Además del mensaje enviado por el módulo **1602A** se enviará por el display integrada en la tarjeta Mini-DK2, el módulo **HY28A** el estado de cada elevador mediante la llamada de la función **menu_lleno()** (código fuente 18) (dentro de la función anterior). Esta función actualizará el valor del estado de cada elevador. En la siguiente imagen (figura 86) se puede observar dicho display en un ejemplo:

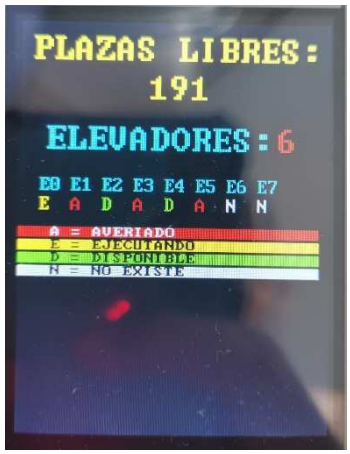


Figura 86. Display HY28A tras un programa simulación.

Código fuente 18. Función menu_lleno().

```
void menu_lleno(void){
   uint8_t i=0,j=0;

   sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
   drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
   drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);
   sprintf(AVERIA_PANTALLA,"%d",Averia);
   sprintf(EJECUT_PANTALLA,"%d",Ejecut);

   for(j=0,i=0;i<8;i++,j=0){
      if(((Averia>>i)&1)==1) {j=15+24*i; drawString(j, 135, "A", RED, BLACK, MEDIUM);}
      else{ if(((Ejecut>>i)&1)==1){j=15+24*i; drawString(j, 135, "E", YELLOW, BLACK, MEDIUM);}
            else                  {j=15+24*i; drawString(j, 135, "D", GREEN, BLACK, MEDIUM);}
      }     }
   for(j=0,i=0;i<8;i++,j=0){
      if(NE<=i){j=15+24*i; drawString(j, 135, "N", WHITE, BLACK, MEDIUM);}
   }
}
```

6.12.1. ARCHIVO "1602A_Libreria.c"

En este apartado no se explicará todo el archivo, sino que se explicarán las funciones que no estén repetidas, el código completo se encuentra en Anexo III.

Se comienza configurando los displays al inicio del programa, primero se añadirá un retardo para garantizar el arranque del display, acto seguido se van enviando las funciones de control que se resumen en lo siguiente:

- Se envía el aviso de que se dispone solo de 4 bits de envío y que el display tiene dos filas.
- Se limpia el display.
- Se retorna a la posición inicial.
- Se habilita el display y con el cursor y parpadeo de cursor se hace lo que se desee.

El display ya está configurada y lista para trabajar. Siempre intercaladas por un retardo para garantizar la escritura.

Código fuente 19. Función init_pantalla().

```
void init_pantalla(char dir){

   delay_pantalla();
   ENV_CONTROL(0x2,0xC,dir); //4 BITS 2 FILAS
   delay_pantalla();
   ENV_CONTROL(0x0,0x1,dir); // CLEAR
   delay_pantalla();
   ENV_CONTROL(0x0,0x2,dir); //RETURN HOME
   delay_pantalla();
   ENV_CONTROL(0x0,0xE,dir); //DISPLAY ON, SI CURSOR, NO BLINK
   delay_pantalla();
}
```

En la siguiente figura (figura 87) se puede ver un extracto de las hojas de características donde se explican dichas configuraciones.

Instruction	Instruction Code										Description	ExecutionTime(f osc=270kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM set DDRAM address to "00H" from AC	1.52ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed	1.52ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display	38 μ s
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D) cursor(C) and blinking of cursor(B) on/off	38 μ s
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data	38 μ s
Function Set	0	0	0	0	1	DL	N	F	-	-	Set interface data length of display line (N: 2line/1line)and, display font type F:5X11dots/5X8dots	38 μ s
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	38 μ s
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	38 μ s
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF The contents of address counter of address counter can also be read	0 μ s
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	38 μ s
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	38 μ s

Figura 87. Extracto de las hojas de características del módulo 1602A. [30]

Para poder enviar los datos se dispone de la función **ENV_CONTROL(1º byte, 2º byte, dirección)** y de **ENV_DATO(char, dirección)** (código fuente 20), estas funciones solo se diferencian por el número de bytes enviados, pues para el control es necesario enviar 2 bytes seguidos y para los caracteres solo se envía el carácter en concreto. Para enviar un carácter se debe enviar dicho carácter más otro byte más, en este caso se envía BT, RW, un bit a cero y RS. Con BT encendemos la luz de fondo, RW sirve para escribir o leer y RS indica en nivel alto que no es un código de control, en la función **ENV_CONTROL()** RS estará a nivel bajo.

Código fuente 20. Función ENV_DATO()

```
void ENV_DATO(unsigned char dato, char dir){ //dato = D7 D6 D5 D4
   unsigned char BT=1, RW=0, RS=1; //FONDO ENCENDIDO, ESCRITURA, DATO
   enviar(((dato & 0xF0)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin E
   enviar(((dato & 0xF0)+(BT<<3)+(1<<2)+(RW<<1)+RS),dir); // con E
   enviar(((dato & 0xF0)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin E
   enviar((((dato & 0x0F)<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin E
   enviar((((dato & 0x0F)<<4)+(BT<<3)+(1<<2)+(RW<<1)+RS),dir); // con E
   enviar((((dato & 0x0F)<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin E
}
```

La función **enviar(dato, dir)** no es más que una escritura en el bus I^2C_2.

Código fuente 21. Función enviar().

```
void enviar(char dato,char dir){ //dato = D7 D6 D5 D4 BT E RW RS
   I2CSendAddr_2(dir,0); //pantalla 1
   I2CSendByte_2(dato);
   I2CSendStop_2();
   I2Cdelay_2();
}
```

A la hora de enviar los mensajes se dispone de una función por cada mensaje, por lo que únicamente se va a mostrar la función **ENV_DISPONIBLE(dir)** (código fuente 22). Dicha función empieza por una limpieza del display, un retardo para asegurar el registro de control y continúa con la función SHIFT, la cual por motivos desconocidos es necesario para que el primer carácter cuadre con el primer display. Después se envían 16 caracteres dependiendo del mensaje, en este caso de centra en el medio la palabra disponible, después se escriben 24 caracteres vacíos para que los siguientes 16 caracteres estén en la fila inferior.

Código fuente 22. Función ENV_DISPONIBLE().

```
void ENV_DISPONIBLE(char dir){
   char i;
   ENV_CONTROL(0x0,0x1,dir); // CLEAR
   delay_pantalla();
   ENV_CONTROL(0x1,0xC,dir); // SHIFT
   ENV_DATO(' ',dir);
   ENV_DATO(' ',dir);
   ENV_DATO(' ',dir);
   ENV_DATO('D',dir);
   ENV_DATO('I',dir);
   ENV_DATO('S',dir);
   ENV_DATO('P',dir);
   ENV_DATO('O',dir);
   ENV_DATO('N',dir);
   ENV_DATO('I',dir);
   ENV_DATO('B',dir);
   ENV_DATO('L',dir);
   ENV_DATO('E',dir);
   ENV_DATO(' ',dir);
   ENV_DATO(' ',dir);
   ENV_DATO(' ',dir);
   for(i=0;i<24;i++){ENV_DATO(' ',dir);}
   ENV_DATO(' ',dir);
   ENV_DATO(' ',dir);
   ENV_DATO('L',dir);
   ENV_DATO('I',dir);
   ENV_DATO('B',dir);
   ENV_DATO('R',dir);
   ENV_DATO('E',dir);
   ENV_DATO('S',dir);
   ENV_DATO(' ',dir);
   ENV_DATO('=',dir);
   ENV_DATO(' ',dir);
   ENV_DATO(HC,dir);
   ENV_DATO(HD,dir);
   ENV_DATO(HU,dir);
   ENV_DATO(' ',dir);
   ENV_DATO(' ',dir);
}
```

Para enviar variables solo se tendrán que llamar, en este caso, a la función **Huecos_libres(libres)** desde el programa principal e introducir en el argumento las plazas libres que hay, esta función lo convertirá en caracteres ASCII y podrá enviarse de tal modo. Las funciones **Plazas_menu(huecos, elevadores, plantas)** o **menú_avanzado_parametros(C_ENABLE, F_parp, T_act, P_mensaje, P_sseg, P_respE)** hacen exactamente lo mismo pero con otras variables para otros casos.

Código fuente 23. Función Huecos_libres().

```
#include <LPC17xx.h>

char addr=0;
uint8_t HC, HD, HU, EU, PD, PU, QD, QU, CED, CEU, TPU, TPd, TPc, TAD, TAU, PMD, PMU, PMd, PSD, PSU,
PRU, PRd;

void Huecos_libres(uint16_t libres){

           HC=(0x3<<4)|(libres/100);
           libres=libres%100;
           HD=(0x3<<4)|(libres/10);
           libres=libres%10;
           HU=(0x3<<4)|(libres/1);
}
```

6.13. TIMER 3

Este timer se encargará de inicializar los demás temporizadores, de actualizar las señales de las luces y de actualizar los displays de los lectores cuando se esté ejecutando el proceso de aparcar o retirar.

6.13.1. CONFIGURACIÓN TIMER 3

Cada 100us, cuando el registro **TC** llegue al valor de *MR0* (2499), el temporizador interrumpirá y se reiniciará, generando una cadena de interrupciones periódicas.

Se necesitará alimentar al timer (registro **PCONP**) y habilitar la interrupción (registro **NVIC**) para que el programa ejecute las funciones asignadas.

La función **Prescaler_T3()** genera dos valores parametrizables que controlarán el tiempo de parpadeo de las luces y el tiempo de actualización de los displays.

Código fuente 24. Configuración del TIMER 3

```
#include <lpc17xx.h>
#define Fpclk 25e6

uint32_t Ttim3=100, Ftic3=0;
uint32_t T_parp=126666, P_parp=0, T_act=2500000, P_act=0;

void Prescaler_T3(void){

   P_parp=T_parp/Ttim3;
   P_act=T_act/Ttim3;
}

void config_timer3(void){ //Configuracion TIMER3

   LPC_SC->PCONP|=(1<<23); //Timer3 ON
   LPC_TIM3->PR=0; //resolucion de 40 ns
   Ftic3=(Fpclk/(LPC_TIM3->PR + 1));
   LPC_TIM3->MCR=0x3; //Interrupt ON MR0, Reset ON MR0
   LPC_TIM3->MR0=(Ftic3*Ttim3*1e-6)-1; //Timer3 interrumpe cada 100 us
   NVIC_EnableIRQ(TIMER3_IRQn); //Habilitacion Timer3
   NVIC_SetPriority(TIMER3_IRQn,4); //Timer3 tiene prioridad 1 subprioridad 0
   Prescaler_T3();
}
```

6.13.2. INTERRUPCIÓN TIMER 3

Este timer se encarga de actualizar las lámparas de aviso lumínico y los displays anexos al lector RFID. Lo primero que se encuentra es la condición para cuando no haya plazas libres, lo que hará el automatismo será activar el flag de luz roja (pues se debe poner la luz de este color) y se actualizan los registros de las lámparas (**Luzroja**, **Luzamarilla** y **Luzverde**). En cuanto vuelva a haber plazas libres se desactivará el flag de la luz roja y se activarán varios flags dependiendo de lo que se quiera conseguir: **flag_amarillo1**, para entrar en el modo avería (luz amarilla fija) **flag_amarillo2**, para entrar en modo

ejecución (luz amarilla parpadeante) y **flag_verde** para entrar en modo disponible o reposo (luz verde fija).

Los registros **modo_lleno**, **modo_averia[elevador]**, **modo_ejecución[elevador]** y **modo_disponible[elevador]** sirven para actualizar más a delante los displays.

Más abajo se encuentra el actualizador de plazas libres para los displays. Si un elevador no cambia de estado no modificará el valor de las plazas que aparecen el su display, por lo que este actualizador lo modificará cada cierto tiempo, un tiempo parametrizable (**P_act**) de 1, 5, 10 o 20 segundos.

Al final del todo se encuentra la inicialización de los timers 0 y 1, para lograr encender las luces y displays antes de iniciar ningún proceso, y la llamada a las funciones **salidas()** y **modos()**, estas funciones se explican en los apartados posteriores y son las encargadas de actualizar las señales de control de las luces y los displays.

Código fuente 25. Función interrupción del TIMER 3

```
void TIMER3_IRQHandler(void){ //Interrupcion TIMER3
   static char i;

   LPC_TIM3->IR|=(1<<0); //Borrar flag de MR0
   cont_TIM3++;
   cont_TIM3_2++;

   if(Huecoslibres == 0) {Luzroja = 0x01;Luzamarilla=0x00; Luzverde=0x00; // NO HAY HUECOS
      if(flag_rojo==0){flag_rojo=1; modo_lleno=1;FLAGS();}
   }
   else {//HAY HUECOS --> ACTUALIZAR LUCES Y PANTALLA
      Luzroja = 0x00;
      if(flag_rojo==1){flag_rojo=0;}
      if(cont_TIM3 == P_parp){
         for(i=0;i<8;i++){
            if(NE>=(i+1)){
               if(((Averia>>i)&1)==1){Luzamarilla |= (1<<i) ;Luzverde &= ~(1<<i);
                  if(flag_amarillo1[i]==0){flag_amarillo1[i]=1;     modo_averia[i]=1;flag_verde[i]=0;
                                                    flag_amarillo2[i]=0;}}   // Averia en E0
               else{
                  if(((Ejecut>>i)&1)==1){Luzamarilla = ((Luzamarilla & ~(1<<i)) |
                     (((~(Luzamarilla>>i))&1)<<i));Luzverde &= ~(1<<i);
                     if(flag_amarillo2[i]==0){
                        flag_amarillo2[i]=1;
                        modo_ejecut[i]=1;
                        flag_verde[i]=0;flag_amarillo1[i]=0;
                     }
                  }
                  else {
                        Luzamarilla &= ~(1<<i) ;Luzverde |=  (1<<i);
                     if(flag_verde[i]==0){
                        flag_verde[i]=1; modo_disponible[i]=1;
            }  }  }
            cont_TIM3=0;
         }
      }
      if(cont_TIM3_2 == P_act){cont_TIM3_2=0;
         if(Huecoslibres_r!=Huecoslibres){Huecoslibres_r=Huecoslibres;
            if(Huecoslibres != 0){FLAGS();}}} // ACTUALIZAR EL NUMERO DE PLAZAS EN LOS ELEVADORES QUE
NO SE HAN USADO

   if(inic_timer01==0){ // PRIMERO SE INICIALIZAN LAS LUCES Y PANTALLAS Y LUEGO SE ACTIVAN LOS
LECTORES Y TIM1
      inic_timer01=1;
      LPC_TIM0->TCR=0x01;
      LPC_TIM1->TCR=0x01;
   }
   salidas();
   modos();
}
```

6.14. MENÚ DE CONFIGURACIÓN

El sistema dispone de un menú de configuraciones que permite parametrizar el automatismo adaptando un único programa a múltiples escenarios.

Dicho menú aparecerá en el display LCD **HY28A** o por el módulo **1602A**.

Nada más arrancar el programa entrará en dicho menú, si al hacer Reset se empieza pulsado el botón KEY 1 se tomará la configuración por defecto.

Se podrá navegar por el menú con los botones integrados ISP, KEY 1 y KEY2, donde KEY 2 se encargará de ascender en el menú, KEY 1 de descender e ISP de seleccionar o marcar el parámetro. El programa empezará entrando en la función **Menu()** (código fuente 28), la cual dibujará mediante las funciones **Uno_menu()** (código fuente 27) y inic_pantalla_menu() los display principales vistas en las siguientes fotografías.

Código fuente 26. Función Menu().

```
void Menu(void){
   Plazas_menu(Huecos_planta,NE,Plantas);
   menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
                            P_respE_menu_2);
   inic_pantalla_menu();
   lcdInitDisplay();
   Uno_menu();

}
```

Figura 89. Display principal en HY28A

Figura 88. Display principal en 1602A.

Código fuente 27. Función Uno_menu().

```
void Uno_menu(void){
   uint8_t i=0,j=0;

   Huecoslibres=Plantas*Huecos_planta;
   fillScreen(BLACK);
   sprintf(borrar_fila,"                          ");
   drawString(10, 10, "PLAZAS LIBRES: ", YELLOW, BLACK, LARGE);
   sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
   drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
   drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);
   sprintf(AVERIA_PANTALLA,"%d",Averia);
   sprintf(EJECUT_PANTALLA,"%d",Ejecut);

   drawString(5, 80, " ELEVADORES:", CYAN, BLACK, LARGE);
   sprintf(NE_PANTALLA,"%d",NE);
   drawString(200, 80, NE_PANTALLA, RED, BLACK, LARGE);
   drawString(5, 110, " PLANTAS:", CYAN, BLACK, LARGE);
   sprintf(Plantas_PANTALLA,"%d",Plantas);
   drawString(185, 110, Plantas_PANTALLA, RED, BLACK, LARGE);
   drawString(5, 140, " HUECOS:", CYAN, BLACK, LARGE);
   sprintf(Huecos_planta_PANTALLA,"%d",Huecos_planta);
   drawString(185, 140, Huecos_planta_PANTALLA, RED, BLACK, LARGE);
   drawString(37, 180, " AVANZADO", YELLOW, BLACK, LARGE);
   drawString(45, 210, " VALIDAR", MAGENTA, BLACK, LARGE);

   drawString(60, 260, "KEY 2" , GREEN, BLACK, MEDIUM);
   drawString(110, 260, " UP "    , BLUE, YELLOW, MEDIUM);
   drawString(60, 280, "KEY 1" , GREEN, BLACK, MEDIUM);
   drawString(110, 280, " DOWN "  , BLUE, YELLOW, MEDIUM);
   drawString(60, 300, " ISP"  , GREEN, BLACK, MEDIUM);
   drawString(110, 300, " OK / SEL ", BLUE, YELLOW, MEDIUM);
}
```

Navegando por el menú con los botones integrados KEY 1, KEY2 e ISP podremos seleccionar que parámetro queremos modificar, si queremos entrar en el menú avanzado o si queremos validar y empezar a trabajar.

Si ahora se pulsase sobre el botón ISP el menú dejaría escoger el valor de dicha variable. Si se volviese a pulsar sobre ISP se volvería al estado anterior.

Figura 91. Menú en HY28A al pulsar una vez KEY2.

Figura 90. Menú en 1602A al pulsar una vez KEY2.

Figura 92. Menú en 1602A al pulsar ISP y dos veces KEY1.

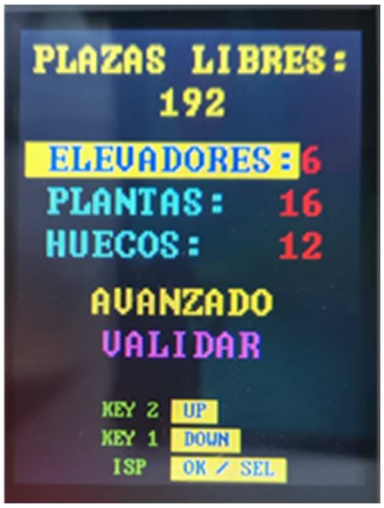


Figura 93. Menú en HY28A al pulsar ISP y dos veces KEY1.

El menú avanzado es exactamente lo mismo pero cambiando los parámetros:

Figura 94. Menú avanzado en HY28A.

Figura 95. Menú avanzado en 1602A.

A continuación se muestra el código (código fuente 28) de las interrupciones (KEY 1, KEY 2 e ISP) y únicamente las funciones para ejecutar el parámetro elevadores, el código completo se puede encontrar en el Anexo I y IV.

Código fuente 28. Interrupciones EINT0, EINT1 y EINT2.

```
void EINT0_IRQHandler(){ //Interrupcion EINT0   ISP

   LPC_SC->EXTINT=(1<<0);
   if(avanzado==0){
      if(selector==0){if(isp2==0){if(isp==0){isp=1;}}}
      if(selector==1){if(isp2==0){if(isp==0){selector_1OK(); isp=1; isp2=1;}}}
      if(selector==2){if(isp2==0){if(isp==0){selector_2OK(); isp=1; isp2=1;}}}
      if(selector==3){if(isp2==0){if(isp==0){selector_3OK(); isp=1; isp2=1;}}}
      if(selector==4){if(isp2==0){if(isp==0){selector_4OK(); isp=0; isp2=1; menu_4();selector=0;}}}
      if(selector==5){if(isp2==0){if(isp==0){selector_5OK(); isp=1; isp2=1; menu_vacio();
                                             Modo_normal();}}} //salir del menu

      if(selector==1){if(isp2==0){if(isp==1){selector_1(); isp=0; isp2=1;}}}
      if(selector==2){if(isp2==0){if(isp==1){selector_2(); isp=0; isp2=1;}}}
      if(selector==3){if(isp2==0){if(isp==1){selector_3(); isp=0; isp2=1;}}}
   }
   if(avanzado==1){
      if(sel_av==0){if(isp2==0){if(isp==0){isp=1;}}}
      if(sel_av==1){if(isp2==0){if(isp==0){sel_av_1OK(); isp=1; isp2=1;}}}
      if(sel_av==2){if(isp2==0){if(isp==0){sel_av_2OK(); isp=1; isp2=1;}}}
      if(sel_av==3){if(isp2==0){if(isp==0){sel_av_3OK(); isp=1; isp2=1;}}}
      if(sel_av==4){if(isp2==0){if(isp==0){sel_av_4OK(); isp=1; isp2=1;}}}
      if(sel_av==5){if(isp2==0){if(isp==0){sel_av_5OK(); isp=1; isp2=1;}}}
      if(sel_av==6){if(isp2==0){if(isp==0){sel_av_6OK(); isp=1; isp2=1;}}}
      if(sel_av==7){if(isp2==0){if(isp==0){sel_av_7OK(); isp=0; isp2=0; sel_av=0; avanzado=0;
                                           Uno_menu();}}}

      if(sel_av==1){if(isp2==0){if(isp==1){sel_av_1(); isp=0; isp2=1;}}}
      if(sel_av==2){if(isp2==0){if(isp==1){sel_av_2(); isp=0; isp2=1;}}}
      if(sel_av==3){if(isp2==0){if(isp==1){sel_av_3(); isp=0; isp2=1;}}}
      if(sel_av==4){if(isp2==0){if(isp==1){sel_av_4(); isp=0; isp2=1;}}}
      if(sel_av==5){if(isp2==0){if(isp==1){sel_av_5(); isp=0; isp2=1;}}}
      if(sel_av==6){if(isp2==0){if(isp==1){sel_av_6(); isp=0; isp2=1;}}}
   }
   isp2=0;
}

void EINT1_IRQHandler(){ //Interrupcion EINT1   KEY 1 DOWN

   LPC_SC->EXTINT=(1<<1);
   if(avanzado==0){
      if(selector==0){if(isp==0){if(key1==0){selector_5(); selector=5; key1=1;
ENV_PPAL_menu(0x27);}}}
      if(selector==1){if(isp==0){if(key1==0){selector_2(); selector=2; key1=1;
ENV_PLANT_menu(0x27);}}}
      if(selector==2){if(isp==0){if(key1==0){selector_3(); selector=3; key1=1;
ENV_HUECO_menu(0x27);}}}
      if(selector==3){if(isp==0){if(key1==0){selector_4(); selector=4; key1=1;
ENV_AVANZ_menu(0x27);}}}
      if(selector==4){if(isp==0){if(key1==0){selector_5(); selector=5; key1=1;
ENV_VALIDAR_menu(0x27);}}}
      if(selector==5){if(isp==0){if(key1==0){selector_1(); selector=1; key1=1;
ENV_ELEV_menu(0x27);}}}

      if(selector==1){if(isp==1){if(key1==0){menu_1(1);                key1=1;}}}
      if(selector==2){if(isp==1){if(key1==0){menu_2(1);                key1=1;}}}
      if(selector==3){if(isp==1){if(key1==0){menu_3(1);                key1=1;}}}
   }
   if(avanzado==1){
      if(sel_av==0){if(isp==0){if(key1==0){sel_av_7(); sel_av=7; key1=1;ENV_SALIR_menu(0x27);}}}
      if(sel_av==1){if(isp==0){if(key1==0){sel_av_2(); sel_av=2; key1=1;ENV_PARP_menu(0x27);}}}
      if(sel_av==2){if(isp==0){if(key1==0){sel_av_3(); sel_av=3; key1=1;ENV_ACT_menu(0x27);}}}
      if(sel_av==3){if(isp==0){if(key1==0){sel_av_4(); sel_av=4; key1=1;ENV_MENSAJE_menu(0x27);}}}
      if(sel_av==4){if(isp==0){if(key1==0){sel_av_5(); sel_av=5; key1=1;ENV_SEG_menu(0x27);}}}
      if(sel_av==5){if(isp==0){if(key1==0){sel_av_6(); sel_av=6; key1=1;ENV_RESP_menu(0x27);}}}
      if(sel_av==6){if(isp==0){if(key1==0){sel_av_7(); sel_av=7; key1=1;ENV_SALIR_menu(0x27);}}}
      if(sel_av==7){if(isp==0){if(key1==0){sel_av_1(); sel_av=1; key1=1;ENV_ENABLE_menu(0x27);}}}

      if(sel_av==1){if(isp==1){if(key1==0){menu_av_1(1);          key1=1;}}}
      if(sel_av==2){if(isp==1){if(key1==0){menu_av_2(1);          key1=1;}}}
      if(sel_av==3){if(isp==1){if(key1==0){menu_av_3(1);          key1=1;}}}
      if(sel_av==4){if(isp==1){if(key1==0){menu_av_4(1);          key1=1;}}}
      if(sel_av==5){if(isp==1){if(key1==0){menu_av_5(1);          key1=1;}}}
      if(sel_av==6){if(isp==1){if(key1==0){menu_av_6(1);          key1=1;}}}
   }
   key1=0;
}

void EINT2_IRQHandler(){ //Interrupcion EINT2   KEY 2 UP

   LPC_SC->EXTINT=(1<<2);
   if(avanzado==0){
      if(selector==0){if(isp==0){if(key2==0){selector_1(); selector=1; key2=1;ENV_ELEV_menu(0x27);}}}
```

```
      if(selector==1){if(isp==0){if(key2==0){selector_5(); selector=5;
key2=1;ENV_VALIDAR_menu(0x27);}}}
      if(selector==2){if(isp==0){if(key2==0){selector_1(); selector=1; key2=1;ENV_ELEV_menu(0x27);}}}
      if(selector==3){if(isp==0){if(key2==0){selector_2(); selector=2;
key2=1;ENV_PLANT_menu(0x27);}}}
      if(selector==4){if(isp==0){if(key2==0){selector_3(); selector=3;
key2=1;ENV_HUECO_menu(0x27);}}}
      if(selector==5){if(isp==0){if(key2==0){selector_4(); selector=4;
key2=1;ENV_AVANZ_menu(0x27);}}}

      if(selector==1){if(isp==1){if(key2==0){menu_1(2);               key2=1;}}}
      if(selector==2){if(isp==1){if(key2==0){menu_2(2);               key2=1;}}}
      if(selector==3){if(isp==1){if(key2==0){menu_3(2);               key2=1;}}}
   }
   if(avanzado==1){
      if(sel_av==0){if(isp==0){if(key2==0){sel_av_1(); sel_av=1; key2=1;ENV_ENABLE_menu(0x27);}}}
      if(sel_av==1){if(isp==0){if(key2==0){sel_av_7(); sel_av=7; key2=1;ENV_SALIR_menu(0x27);}}}
      if(sel_av==2){if(isp==0){if(key2==0){sel_av_1(); sel_av=1; key2=1;ENV_ENABLE_menu(0x27);}}}
      if(sel_av==3){if(isp==0){if(key2==0){sel_av_2(); sel_av=2; key2=1;ENV_PARP_menu(0x27);}}}
      if(sel_av==4){if(isp==0){if(key2==0){sel_av_3(); sel_av=3; key2=1;ENV_ACT_menu(0x27);}}}
      if(sel_av==5){if(isp==0){if(key2==0){sel_av_4(); sel_av=4; key2=1;ENV_MENSAJE_menu(0x27);}}}
      if(sel_av==6){if(isp==0){if(key2==0){sel_av_5(); sel_av=5; key2=1;ENV_SEG_menu(0x27);}}}
      if(sel_av==7){if(isp==0){if(key2==0){sel_av_6(); sel_av=6; key2=1;ENV_RESP_menu(0x27);}}}

      if(sel_av==1){if(isp==1){if(key2==0){menu_av_1(2);          key2=1;}}}
      if(sel_av==2){if(isp==1){if(key2==0){menu_av_2(2);          key2=1;}}}
      if(sel_av==3){if(isp==1){if(key2==0){menu_av_3(2);          key2=1;}}}
      if(sel_av==4){if(isp==1){if(key2==0){menu_av_4(2);          key2=1;}}}
      if(sel_av==5){if(isp==1){if(key2==0){menu_av_5(2);          key2=1;}}}
      if(sel_av==6){if(isp==1){if(key2==0){menu_av_6(2);          key2=1;}}}
   }
   key2=0;
}
```

Las dos siguientes funciones (código fuente 31) se encargan de dar el aspecto a la palabra al seleccionarla.

Código fuente 29. Funciones selector_1() y selector_1OK().

```
void selector_1(){ // elevadores
   char j;

   drawString( 5, 80, " ELEVADORES:", BLUE   , WHITE, LARGE);
   drawString( 5, 110, " PLANTAS:"  , CYAN   , BLACK, LARGE);
   drawString( 5, 140, " HUECOS:"   , CYAN   , BLACK, LARGE);
   drawString(37, 180, " AVANZADO"  , YELLOW , BLACK, LARGE);
   drawString(45, 210, " VALIDAR"   , MAGENTA, BLACK, LARGE);
   for(j=0;j<9;j++){ENV_CONTROL_menu(0x1,0x04,0x27);}// SHIFT
}
void selector_1OK(){ // elevadores
   char j;

   drawString( 5, 80, " ELEVADORES:", BLUE   , YELLOW, LARGE);
   drawString( 5, 110, " PLANTAS:"  , CYAN   , BLACK , LARGE);
   drawString( 5, 140, " HUECOS:"   , CYAN   , BLACK , LARGE);
   drawString(37, 180, " AVANZADO"  , YELLOW , BLACK , LARGE);
   drawString(45, 210, " VALIDAR"   , MAGENTA, BLACK , LARGE);
   for(j=0;j<9;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
```

La siguiente función (código fuente 32) es la encargada de actualizar los valores de **NE** que aparecerán en los displays.

Código fuente 30. Función menu_1()

```
void menu_1(char i){
   static char cont, j;
   if(i==1){ //1=key1
      cont--;
      NE--;
      if(NE<1){NE=1;}
      sprintf(borrar_numero,"    ");
      drawString(200, 80, borrar_numero, BLACK, BLACK, LARGE);
      sprintf(NE_PANTALLA,"%d",NE);
      drawString(200, 80, NE_PANTALLA, RED, BLACK, LARGE);
   }
   if(i==2){ //1=key2
      cont++;
      NE++;
      if(NE>8){NE=8;}
      sprintf(NE_PANTALLA,"%d",NE);
      sprintf(borrar_numero,"    ");
      drawString(200, 80, borrar_numero, BLACK, BLACK, LARGE);
      drawString(200, 80, NE_PANTALLA, RED, BLACK, LARGE);
   }
   Huecoslibres=Plantas*Huecos_planta;
   sprintf(borrar_fila,"                        ");
   sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
```

```
   drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
   drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);

   Plazas_menu(Huecos_planta, NE, Plantas);
   ENV_ELEV_menu(0x27);
   for(j=0;j<9;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
```

Las variables parametrizables son las siguientes:

NE: Número de elevadores: entre 8 y 1, ambos incluidos.

Plantas: Número de plantas de almacenaje: entre 16 y 1, ambos incluidos.

Hueco_plantas: Número de divisiones por planta: entre 12 y 1, ambos incluidos.

C_ENABLE: Ciclos de habilitación (enable) del mux de los PLC's: 1, 3, 5, 7, 10, 12 o 15 ciclos.

F_parp: Frecuencia de parpadeo de las luces: 1, 1.33, 2 y 4 Hz.

T_act: Tiempo entre actualizaciones de los display: 1, 5, 10 o 20 segundos.

P_mensaje: Tiempo de aparición de mensaje en modo "0Coches" o "2Coches": 5, 7.5, 10 o 15 segundos.

P_sseg: Tiempo para leer la respuesta de los sistemas de emergencia: 5, 10, 15 o 20 segundos.

P_respE: Periodo de lectura de la respuesta del PLC: 0.1, 0.2. 0.5, 1 o 2 segundos.

CAPÍTULO VII

CONCLUSIONES Y TRABAJOS FUTUROS

7. CONCLUSIÓN Y TRABAJOS FUTUROS

7.1. CONCLUSIONES

La superficie del aparcamiento propuesto (que dispone de 80 plazas) es aproximadamente de 530 m2, un área más grande comparado con un aparcamiento ejemplo de las mismas plazas repartidas en cuatro plantas (450m^2). Respecto a la altura, el aparcamiento estudiado alcanza los 35 metros de altura, aproximadamente un edificio de 12 plantas (si no se entierra ninguna planta), mientras que las cuatro plantas del aparcamiento ejemplo no superan los 15 metros, y ligeramente menos si no dispone de tejado.

El automatismo está diseñado para moverse a tres velocidades, siendo la velocidad máxima de 0,37 m/s, esto es, ascendería a los 35 metros en 1 minuto y 35 segundos. En el caso de tener que aparcar o recoger un vehículo a la planta más elevada (un recorrido de 27,8 metros) y suponiendo que la plataforma de desplazamiento lineal tarda aproximadamente 30 segundos en desplazar el vehículo, el automatismo tardaría aproximadamente 3 minutos en realizar la tarea. Todo ello, suponiendo un tiempo de 15 segundos para activar las medidas de seguridad, 1 minuto para arrancar y detener con precisión la plataforma elevadora y los 30 segundos de actuación de la plataforma de desplazamiento lineal.

Además, este aparcamiento dispone de 4 elevadores, por lo que, en el supuesto de evacuar el aparcamiento por completo, le llevaría un tiempo aproximado de 49 minutos (se ajusta el tiempo de movimiento de la plataforma elevadora a la quinta planta con 15,3 metros de altura, 2 minutos y 27 segundos). En el caso de un bloque de viviendas es muy poco casual evacuar 80 vehículos en menos de una hora, pero si puede darse el caso de coincidir diez personas, lo que supondría que la última persona tarde unos 10 minutos en obtener el vehículo.

Dentro de la propia estructura de aparcamiento, los espacios vacíos son notables (se puede observar en el apartado 7.2), por lo que se podría añadir un sistema de almacenamiento de vehículos de menor tamaño e incluso un pequeño almacén personal si este aparcamiento se sitúa en una urbanización de viviendas.

Como conclusión, este diseño de aparcamiento automatizado puede ser rentable si se aprovecha mejor la superficie, añadiendo otro sistema de aparcamiento o almacén privado, si se disminuye la altura, situando por debajo del suelo alguna planta de almacenaje, o añadiendo más coches por planta en vez de alturas, siempre manteniendo mínimo cuatro elevadores para evitar las esperas e incorporando un sistema para poder reservar la entrada o retirada del vehículo y así no generar aglomeraciones.

7.2. MODIFICACIONES FÍSICAS.

POSIBILIDAD DE UBICAR LA ZONA DE ENTREGA Y RECOGIDA DE VEHÍCULOS A UNA ALTURA INTERMEDIA, PUDIENDO ENTERRAR VARIAS PLANTAS.

Para evitar que la altura sea un problema se podrá enterrar alguna planta que almacene vehículos. El sistema de PLC'S encargados de la unidad móvil del aparcamiento recoge esta premisa, por lo que solo habría que tenerlo en cuenta a la hora de la construcción y programación de los PLC'S.

POSIBILIDAD DE INCORPORAR APARCAMIENTO DE MOTOCICLETAS Y BICICLETAS EN LOS ESPACIOS MUERTOS

Los espacios muertos de este aparcamiento son grandes, por lo que se propone rellenar dichos espacios con vehículos de menor tamaño. En la siguiente imagen (figura 96) se puede observar en verde un rectángulo de 2,87 metros de largo por 2 metros de ancho donde perfectamente se podría introducir un cuadriciclo ligero.

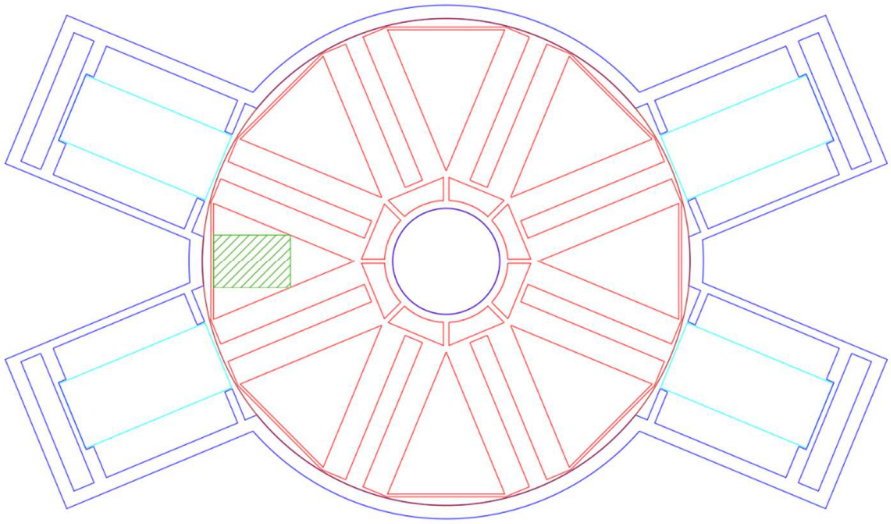


Figura 96. Representación de los espacios muertos del aparcamiento.

POSIBILIDAD DE SITUAR DOS APARCAMIENTOS INDEPENDIENTES DE MENOS PLANTAS UNO SUBTERRÁNEO Y OTRO ELEVADO.

Sería de gran utilidad disponer de dos aparcamientos más pequeños, uno subterráneo y el otro a la vista, con esto se reducirían los tiempos de espera, pero dificultaría el uso de los elevadores, pues habría que distinguir entre ambos aparcamientos para no confundir el método de activación.

7.3. MODIFICACIONES TÉCNICAS.

POSIBILIDAD DE DIMENSIONAR LA TORRE EN NÚMERO DE ALTURAS Y NÚMERO DE COCHES POR PLANTA Y POSIBILIDAD DE AÑADIR MÁS PLATAFORMAS ELEVADORAS.

El sistema está preparado para albergar un máximo de 12 plazas o divisiones por planta, 16 plantas que almacenen vehículos y 8 elevadores, todo lo que sea inferior a esto, como en el caso de estudio, será parametrizable mediante el menú de configuraciones al arranque del sistema.

POSIBILIDAD DE DESVINCULAR LA TARJETA A LA PLAZA.

Esta aplicación conlleva añadir una función al programa. Del lector llega al sistema el número de tarjeta, en este proyecto dicho número es igual al número de plaza, por lo que cada plaza está asociada a cada tarjeta, para poder desvincular el número de tarjeta con el número de plaza se necesita crear una función que recoja el número de tarjeta y asigne la plaza más baja posible, generando una matriz de memoria. Esta matriz de memoria grabará la asignación al aparcar el vehículo y la borrará al retirarlo.

POSIBILIDAD DE UBICAR DICHO APARCAMIENTO EN OTROS ENTORNOS.

Para mayor funcionalidad sería conveniente que el aparcamiento se pueda acoplar a otros entornos, es decir, que pueda funcionar como parking público, como parking privado en un taller o concesionario, etcétera.

En el caso del parking público o centro comercial se sustituiría el sistema lector de tarjetas y se añadiría una impresora y un lector de tarjetas con código de barras o QR, para esta modificación se ha de desvincular la tarjeta a la plaza, recibiendo de igual modo en el microcontrolador el número de tarjeta y asignándolo a la plaza.

En el caso de un parking privado como un taller o concesionario simplemente habría que cambiar el lector de tarjetas por un teclado o una botonera que envíe según el número de plaza, con esto se evita tener una tarjeta por cada coche pero sería necesario elaborar un mapa con la plaza y el coche para evitar confundirse.

CAPÍTULO VIII

PRESUPUESTO

8. PRESUPUESTO

En este apartado se elabora un presupuesto aproximado del proyecto el cual recogerá el gasto de los materiales necesarios para el control del automatismo, el programa descrito anteriormente y la instalación y mantenimiento primario del sistema, sin contemplar costes de del edificio, plataforma elevadora, plataforma de desplazamiento lineal y plataforma rotativa.

HARDWARE

Tabla 27. Tabla de costes de material.

ELEMENTO	CANTIDAD NECESARIA	PRECIO	CANTIDAD USADA	COSTE TOTAL
TARJETA MINI-DK2	1	1x 50€	1	70€
RESISTENCIA 4k7	2x BUS_I2C	1x 0,086€	6	0,52€
EXPANSOR DE DISTANCIA I2C PCA9615	2x ELEVADOR	1x 15,87€	8	126,96€
BALIZA DE COLORES	1x ELEVADOR	1x 47,41€	4	189,64€
RELÉ 24V 2 CONTACTOS	2x ELEVADOR	1x 4,95€	8	39,60€
EXPANSOR DE PUERTOS PCF8574T	7	2x 11,58€	8	46,32€
SENSOR MOV. HUBER Motion 3LV	4x ELEVADOR	1x 14,95€	16	239,20€
SENSOR MAGNÉTICO MN200S	3x ELEVADOR	1x 49,70€	12	596,40€
CONTROLADOR NS5-DIG	1x ELEVADOR	1x 121,90€	4	487,6€
FUENTE 24V S-240-24	1x ELEVADOR	1x 22,99€	4	91,96€
CONVERTIDOR PC817 24V <-> 3,3V	3x ELEVADOR	4x 4,52€	4	18,08€
SENSOR DETECCIÓN E3FA-RN11	1x ELEVADOR	1x 74,70€	4	298,80€
ESPEJO PARA FOTOCÉLULA	1x ELEVADOR	1x 14,85€	4	59,4€
PLC ELEVADOR	1x ELEVADOR	1x 335,77€	4	1.343,08€
PLC PLATAFORMA DESPLAZ. LINEAL	1x ELEVADOR	1x 335,77€	4	1.343,08€
PLC ESTRUCTURA ROTATIVA	1x PLANTA	1x 335,77€	10	3.357,70€
CHIP 74153 y 74155	28 + 35	1x 0,45€	63	28,35€
DISPLAY 28’ LCD HY28A	1	1x 20€	1	20€
DISPLAY 1602A + PCF8574T	1x ELEVADOR + 1	1x 4,89€	5	24,45€
ELEGOO MEGA	1x ELEVADOR	1x 24,99€	4	99,96€
LECTOR PN532	1x ELEVADOR	1x 2,77€	4	11,08€
			TOTAL CON IVA (21%)	**8.492€**

SOFTWARE

Tabla 28. Tabla de costes de software.

PROGRAMA	CANTIDAD	PRECIO UNITARIO	COSTE TOTAL
CONTROL DE PARKING PARA LPC1768	1	5000€	5.000€
CONTROL DE LECTOR PARA ARDUINO	4	100€	400€
		TOTAL SIN IVA	**5.400€**
		TOTAL CON IVA (21%)	**6.534€**

GESTIÓN

Tabla 29. Tabla de costes de mano de obra.

DESCRIPCIÓN	CATEGORÍA	CONCEPTO	COSTE TOTAL
ALEJANDRO VILLALBA HERNANDO	INGENIERO ELECTRÓNICO Y AUTOMÁTICO INDUSTRIAL	SOPORTE TÉCNICO 24h – 1ª semana	5.000€
		INSTALACIÓN Y CONFIGURACIÓN → 35€/h – 80h	2.800€
		TOTAL SIN IVA	**7.800€**
		TOTAL CON IVA (21%)	**9.438€**

TOTAL

Tabla 30. Tabla de costes totales.

CONCEPTO	COSTE TOTAL
COSTE DEL MATERIAL	8.492€
COSTE DEL SOFTWARE	6.534€
COSTE DE MANO DE OBRA	9.438€
TOTAL SIN IVA	**20.218€**
TOTAL CON IVA (21%)	**24.464€**

CAPÍTULO IX

BIBLIOGRAFÍA Y REFERENCIAS

9. BIBLIOGRAFÍA

[1] https://www.motor.es/medidas-coches.

[2] https://www.youtube.com/watch?v=LXIVnl1ezrk.

[3] https://autosolar.es/panel-solar-12-voltios/panel-solar-200w-12v-policristalino

[4] https://autosolar.es/placas-fotovoltaicas/inclinacion-placas-solares

[5] https://www.amb.cat/s/web/amb/administracio-metropolitana/normativa-i-ordenances.html

[6] https://www.equipovertical.com/Poleas-y-polipastos-de-cuerda-para-izar-materiales

[7] https://www.plcmadrid.es/rebt/itc-bt-47-motores/

[8] http://www.cosgra.com/jalmac2.html

[9] https://industrial.omron.es/es/products/e3fa-e3fb

[10]https://www.aprimatic.es/producto/puertas-automaticas-peatonales/automatismos/puertas-correderas-automatismos/ns-120-bs/

[11] https://www.aprimatic.es/producto/barreras-automaticas-3/areas-residenciales-barreras-automaticas-3/barrera-automatica-park-30-plus/

[12] https://es.scribd.com/document/339114191/LPC1768-Mini-DK2-Schematic

[13] https://uelectronics.com/producto/pn532-modulo-rfid-nfc-lectura-y-escritura-v3/

[14] https://datasheetspdf.com/datasheet/LCD-1602A.html

[15] https://datasheetspdf.com/datasheet/search.php?sWord=PCF-8574

[16] https://datasheetspdf.com/datasheet/search.php?sWord=pn-532

[17] https://datasheetspdf.com/datasheet/search.php?sWord=lpc1768

[18] https://datasheetspdf.com/datasheet/search.php?sWord=pca9615

[19] https://datasheetspdf.com/datasheet/search.php?sWord=pc817

10. REFERENCIAS DE FIGURAS

[20]https://www.amazon.es/dp/B0823YJS5G

[21]https://www.aprimatic.es/producto/puertas-automaticas-peatonales/automatismos/vidrio-limpio-pinza/

[22]https://www.ifm.com/es/es/product/MN200S#/

[23]https://practinetmva.mercadoshops.com.mx/MLM-768642126-envio-gratis-optoacoplador-24v-a-5v-8-canales-el817-arduin-_JM

[24]https://es.aliexpress.com/item/32864640628.html

[25]https://www.ptrobotics.com/conversores/7678-sparkfun-differential-i2c-breakout-pca9615-qwiic.html

[26]https://www.arrow.com/es-mx/reference-designs/application-circuit-for-the-p82b715-i2c-bus-extender-for-i2c-bus-extenders-interfacing-standard-i2c-bus-signals-to-twisted-pair-cables-up-to-30m-long/afdcd2c22172a52eb0f02d0e34fdf366

[27]https://www.amazon.es/gp/product/B081JNM3KR/ref=ppx_yo_dt_b_asin_image_o02_s00?ie=UTF8&psc=1

[28] https://es.aliexpress.com/item/32997492326.html

[29] https://www.drouiz.com/blog/2018/06/25/uart-vs-spi-vs-i2c-diferencias-entre-protocolos/

[30]https://es.aliexpress.com/item/1005001561633575.html

[31] https://www.hotmcu.com/lpc1768minidk2-development-board-28-tft-lcd-p-12.html

[32]https://www.google.com/imgres

[33] https://www.google.com/imgres

[34] https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/educational-content/que-es-uart_254524.html

ANEXOS

ANEXOS

ANEXO I. ARCHIVO “Control_parking.c”

Código fuente 31. Código completo Archivo “Control_parking.c”

```
#include <lpc17xx.h>
#define Fpclk 25e6    //Fcpu/4
#define si 1
#define no 0
//PARAMETROS GENERALES
uint8_t NE=4; //numero elevadores
uint8_t Plantas = 10; //numero de plantas
uint8_t Huecos_planta = 8; //numero huecos o plazas por planta

// TIMERS
uint32_t Ftic0=0,                          Ftic1=0,     Ftic3=0;
uint32_t Ttim0=1000, Ttim1=10, Ttim3=100;
uint32_t cont_TIM0_2=0, cont_TIM1=0, cont_TIM1_2=0, cont_TIM3=0,cont_TIM3_2=0,b=0;
uint8_t cont_TIM0=0;
char inic_timer01=0;

//DIRECCIONES I2C
uint8_t DirEP0=0x20,DirEP1=0x21,DirEP2=0x22,DirEP3=0x23,DirEP4=0x24,DirEP5=0x25,DirEP6=0x26,
DirEP7=0x27; //EXPANSOR PUERTOS + PANTALLAS
uint8_t DirPn[8]={0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27}; // PANTALLAS
uint8_t DirAn[8]={0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08}; //ARDUINOS

//MODO FUNCIONAMIENTO
uint8_t Averia = 0x00, Ejecut = 0x00, Huecoslibres = 192, Huecoslibres_r = 192;
uint8_t Luzverde = 0x00,Luzverde_n = 0xFF, Luzamarilla = 0x00, Luzamarilla_n = 0xFF, Luzroja = 0x00,
        Luzroja_n = 0x01;

//FLAGS MODO FUNCIONAMIENTO
char modo_lleno=0, flag_rojo=0;
char flag_verde[8]={0,0,0,0,0,0,0,0}, flag_amarillo1[8]={0,0,0,0,0,0,0,0},
     flag_amarillo2[8]={0,0,0,0,0,0,0,0};
char modo_disponible[8]={0,0,0,0,0,0,0,0}, modo_ejecut[8]={0,0,0,0,0,0,0,0},
     modo_averia[8]={0,0,0,0,0,0,0,0};

// VARIABLES PLAZAS/ELEVADORES
uint8_t elevador=0, plaza=0, plaza_En[8]={0,0,0,0,0,0,0,0};
uint8_t sensor_coche=0; // E7 E6 E5 E4 E3 E2 E1 E0
char respuesta, deteccion;
char PLANTA_USO[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

//MEMORIA PLAZAS
uint32_t Memoria_6=0, Memoria_5=0, Memoria_4=0, Memoria_3=0, Memoria_2=0, Memoria_1=0;
       //(192-161)     (160-129)     (128-97)      (96-65)       (64-33)       (32-1)
char movimiento[8]={0,0,0,0,0,0,0,0};

//PLC
uint8_t REG_dato[8]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
uint8_t REG_elev[8]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
uint8_t REG_dato_esp[8]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
uint8_t REG_elev_esp[8]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
uint8_t LOAD_dato = 0xFF;
uint8_t LOAD_elev = 0xFF;
char flag_respE [8]={0,0,0,0,0,0,0,0};
char correcto[8]={0,0,0,0,0,0,0,0};

//SISTEMA SEGURIDAD
char flag_marcha[8]={0,0,0,0,0,0,0,0};

// BLOQUEO LECTOR
char bloquear_lector[8]={0,0,0,0,0,0,0,0};

// 0COCHES 2COCHES
char flag_coches[8]={0,0,0,0,0,0,0,0};
uint32_t cont_coches[8]={0,0,0,0,0,0,0,0};

//MENU
uint8_t selector=0,sel_av=0;
char key1=0, key2=0, isp=0, isp2=0, avanzado=0;
char C_ENABLE=5,T_act_menu=10,P_sseg_menu=10, F_parp_menu=20,P_respE_menu_2=5, P_mensaje_menu_2=100;
float F_parp = 2.00,P_respE_menu=0.5, P_mensaje_menu=10;
uint32_t T_parp=126666, P_parp=0, T_act=2500000, P_act=0, P_mensaje=36, P_sseg=289, P_respE=15;

//LCD
#include "lcddriver.h"
char AVERIA_PANTALLA[25],EJECUT_PANTALLA[25], HUECOS_PANTALLA[25], NE_PANTALLA[25];
char Plantas_PANTALLA[25], Huecos_planta_PANTALLA[25], C_ENABLE_PANTALLA[25], T_parp_PANTALLA[25];
char T_act_PANTALLA[25], P_mensaje_PANTALLA[25], P_sseg_PANTALLA[25], P_respE_PANTALLA[25];
char borrar_fila[25], borrar_numero[25];
////////////////////////////////////////////////////////////////////////////////////////////////////
                                                LCD MENU
////////////////////////////////////////////////////////////////////////////////////////////////////
void borrar_pantalla(void){
```

```
    uint16_t i=0, j=0;
    for(i=0,j=0;i<14;i++,j=0){
        j=24*i;
        drawString(0, j, borrar_fila, BLACK, BLACK, LARGE);
    }
}
void menu_lleno(void){
    uint8_t i=0,j=0;

    sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
    drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
    drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);
    sprintf(AVERIA_PANTALLA,"%d",Averia);
    sprintf(EJECUT_PANTALLA,"%d",Ejecut);

    for(j=0,i=0;i<8;i++,j=0){
        if(((Averia>>i)&1)==1) {j=15+24*i; drawString(j, 135, "A", RED, BLACK, MEDIUM);}
        else{ if(((Ejecut>>i)&1)==1){j=15+24*i; drawString(j, 135, "E", YELLOW, BLACK, MEDIUM);}
              else                  {j=15+24*i; drawString(j, 135, "D", GREEN, BLACK, MEDIUM);}
                                    }
            }
    for(j=0,i=0;i<8;i++,j=0){
        if(NE<=i){j=15+24*i; drawString(j, 135, "N", WHITE, BLACK, MEDIUM);}
    }
}
void Uno_menu(void){
    uint8_t i=0,j=0;

    Huecoslibres=Plantas*Huecos_planta;
    fillScreen(BLACK);
    sprintf(borrar_fila,"                          ");
    drawString(10, 10, "PLAZAS LIBRES: ", YELLOW, BLACK, LARGE);
    sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
    drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
    drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);
    sprintf(AVERIA_PANTALLA,"%d",Averia);
    sprintf(EJECUT_PANTALLA,"%d",Ejecut);

    drawString(5, 80, " ELEVADORES:", CYAN, BLACK, LARGE);
    sprintf(NE_PANTALLA,"%d",NE);
    drawString(200, 80, NE_PANTALLA, RED, BLACK, LARGE);
    drawString(5, 110, " PLANTAS:", CYAN, BLACK, LARGE);
    sprintf(Plantas_PANTALLA,"%d",Plantas);
    drawString(185, 110, Plantas_PANTALLA, RED, BLACK, LARGE);
    drawString(5, 140, " HUECOS:", CYAN, BLACK, LARGE);
    sprintf(Huecos_planta_PANTALLA,"%d",Huecos_planta);
    drawString(185, 140, Huecos_planta_PANTALLA, RED, BLACK, LARGE);
    drawString(37, 180, " AVANZADO", YELLOW, BLACK, LARGE);
    drawString(45, 210, " VALIDAR", MAGENTA, BLACK, LARGE);

    drawString(60, 260, "KEY 2" , GREEN, BLACK, MEDIUM);
    drawString(110, 260, " UP "    , BLUE, YELLOW, MEDIUM);
    drawString(60, 280, "KEY 1" , GREEN, BLACK, MEDIUM);
    drawString(110, 280, " DOWN "  , BLUE, YELLOW, MEDIUM);
    drawString(60, 300, " ISP"  , GREEN, BLACK, MEDIUM);
    drawString(110, 300, " OK / SEL ", BLUE, YELLOW, MEDIUM);
}
void Avanzado_menu(void){
    uint8_t i=0,j=0;
    sprintf(borrar_fila,"    ");

    drawString(10, 10, "CICLOS ENABLE MUX: ", YELLOW, BLACK, MEDIUM);
    sprintf(C_ENABLE_PANTALLA,"%d ciclos",C_ENABLE);
    drawString(165, 10, borrar_fila, BLACK, BLACK, MEDIUM);
    drawString(165, 10, C_ENABLE_PANTALLA, RED, BLACK, MEDIUM);

    F_parp=2.00;
    drawString(10, 40, "FRECUENCIA PARPADEO: ", YELLOW, BLACK, MEDIUM);
    sprintf(T_parp_PANTALLA,"%.2fHz",F_parp);
    drawString(185, 40, borrar_fila, BLACK, BLACK, MEDIUM);
    drawString(185, 40, T_parp_PANTALLA, RED, BLACK, MEDIUM);

    T_act_menu=10;
    drawString(10, 70, "TIEMPO ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
    sprintf(T_act_PANTALLA,"%d s",T_act_menu);
    drawString(195, 70, borrar_fila, BLACK, BLACK, MEDIUM);
    drawString(195, 70, T_act_PANTALLA, RED, BLACK, MEDIUM);

    P_mensaje_menu=10;
    drawString(10, 100, "TIEMPO MENSAJE: ", YELLOW, BLACK, MEDIUM);
    sprintf(P_mensaje_PANTALLA,"%0.1f s",P_mensaje_menu);
    drawString(145, 100, borrar_fila, BLACK, BLACK, MEDIUM);
    drawString(145, 100, P_mensaje_PANTALLA, RED, BLACK, MEDIUM);

    P_sseg_menu=10;
    drawString(10, 130, "TIEMPO SIST. SEG.: ", YELLOW, BLACK, MEDIUM);
    sprintf(P_sseg_PANTALLA,"%d s",P_sseg_menu);
    drawString(170, 130, borrar_fila, BLACK, BLACK, MEDIUM);
    drawString(170, 130, P_sseg_PANTALLA, RED, BLACK, MEDIUM);

    P_respE_menu=0.5;
    drawString(10, 160, "PERIODO RESP. PLC: ", YELLOW, BLACK, MEDIUM);
```

```
   sprintf(P_respE_PANTALLA,"%0.1f s",P_respE_menu);
   drawString(170, 160, borrar_fila, BLACK, BLACK, MEDIUM);
   drawString(170, 160, P_respE_PANTALLA, RED, BLACK, MEDIUM);

   drawString(70, 210, "SALIR", MAGENTA, BLACK, LARGE);

   drawString(60, 260, "KEY 2" , GREEN, BLACK, MEDIUM);
   drawString(110, 260, " UP "    , BLUE, YELLOW, MEDIUM);
   drawString(60, 280, "KEY 1" , GREEN, BLACK, MEDIUM);
   drawString(110, 280, " DOWN "  , BLUE, YELLOW, MEDIUM);
   drawString(60, 300, " ISP"  , GREEN, BLACK, MEDIUM);
   drawString(110, 300, " OK / SEL ", BLUE, YELLOW, MEDIUM);
}
void menu_vacio(void){
   uint8_t i=0,j=0;
   fillScreen(BLACK);
   sprintf(borrar_fila,"                          ");

   drawString(10, 10, "PLAZAS LIBRES: ", YELLOW, BLACK, LARGE);
   sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
   drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
   drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);
   sprintf(AVERIA_PANTALLA,"%d",Averia);
   sprintf(EJECUT_PANTALLA,"%d",Ejecut);

   drawString(5, 80, " ELEVADORES: ", CYAN, BLACK, LARGE);
   sprintf(NE_PANTALLA,"%d",NE);
   drawString(200, 80, NE_PANTALLA, RED, BLACK, LARGE);
   drawString(15, 120, "E0 E1 E2 E3 E4 E5 E6 E7", CYAN, BLACK, MEDIUM);
   drawString(15, 135, "D  D  D  D  D  D  D  D" , GREEN, BLACK, MEDIUM);

   for(j=0,i=0;i<8;i++,j=0){
      if(NE<=i){j=15+24*i; drawString(j, 135, "N", WHITE, BLACK, MEDIUM);}
   }
   drawString(0, 160, "   A = AVERIADO              ", WHITE, RED, SMALL);
   drawString(0, 170, "   E = EJECUTANDO            ", BLACK, YELLOW, SMALL);
   drawString(0, 180, "   D = DISPONIBLE            ", BLACK, GREEN, SMALL);
   drawString(0, 190, "   N = NO EXISTE             ", BLACK, WHITE, SMALL);
   menu_lleno();
}

////////////////////////////////////////////////////////////////////////////////////////////////////
                                        CONFIG IRQ
////////////////////////////////////////////////////////////////////////////////////////////////////
void config_IRQ(void){ //Configuracion interrupciones EINT1 y EINT2

   LPC_PINCON->PINSEL4|=1<<(10*2); //Configuracion del P2.10 como EINT0
   LPC_PINCON->PINSEL4|=1<<(11*2); //Configuracion del P2.11 como EINT1
   LPC_PINCON->PINSEL4|=1<<(12*2); //Configuracion del P2.12 como EINT2

   LPC_SC->EXTMODE|=7<<0; //Interrupcion EINT0, EINT1 Y EINT2 activa por flanco
   LPC_SC->EXTPOLAR=0x00000000; //Interrupciones EINT0, EINT1 y EINT2 activa por flanco de bajada

   NVIC_SetPriorityGrouping(4);
   NVIC_SetPriority(EINT0_IRQn,0); //EINT0 tiene prioridad 0 subprioridad 0
   NVIC_SetPriority(EINT1_IRQn,0); //EINT1 tiene prioridad 0 subprioridad 0
   NVIC_SetPriority(EINT2_IRQn,0); //EINT2 tiene prioridad 0 subprioridad 0

   NVIC_EnableIRQ(EINT0_IRQn); //Habilitacion EINT0
   NVIC_EnableIRQ(EINT1_IRQn); //Habilitacion EINT1
   NVIC_EnableIRQ(EINT2_IRQn); //Habilitacion EINT2
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                      INICIALIZAR LUCES
////////////////////////////////////////////////////////////////////////////////////////////////////
void inic_luces(void){                          //INICIALIZA CON LAS LUCES VERDES
   LPC_GPIO2->FIOPIN |= (1<<9);  //ROJO P2.9

   I2CSendAddr_1(DirEP0,0); //direccion, escribir
   I2CSendByte_1(0xFF); //enviar salidas de pines
   I2CSendStop_1();
   I2Cdelay_1(); //garantizar la escritura.

   I2CSendAddr_1(DirEP1,0); //direccion, escribir
   I2CSendByte_1(0x00); //enviar salidas de pines
   I2CSendStop_1();
   I2Cdelay_1(); //garantizar la escritura.
}



////////////////////////////////////////////////////////////////////////////////////////////////////
                                      FLAGS / MODOS
////////////////////////////////////////////////////////////////////////////////////////////////////
void FLAGS(void){     //PONE TODOS LOS FLAGS A CERO PARA ACTUALIZAR AVERIA / EJECUT / HUECOS LIBRES
   char i;

   for(i=0;i<8;i++){
      flag_verde[i]=0;
      flag_amarillo1[i]=0;
      flag_amarillo2[i]=0;
```

```
   }
}
void modos(){
   char i;

   if(modo_lleno==1){LPC_TIM3-
>TCR=0x00;Huecos_libres(Huecoslibres);ENV_LLENO();modo_lleno=0;LPC_TIM3->TCR=0x01;}

   for(i=0;i<8;i++){
      if(modo_disponible[i]==1)  {LPC_TIM3->TCR=0x00; Huecos_libres(Huecoslibres); menu_lleno();
ENV_DISPONIBLE(DirPn[i]); modo_disponible[i]=0;        LPC_TIM3->TCR=0x01;}

      if(modo_ejecut[i]==1)      {LPC_TIM3->TCR=0x00; Huecos_libres(Huecoslibres); menu_lleno();
ENV_EJECUT(DirPn[i]);              modo_ejecut[i]=0;   LPC_TIM3->TCR=0x01;}

      if(modo_averia[i]==1)      {LPC_TIM3->TCR=0x00; Huecos_libres(Huecoslibres); menu_lleno();
ENV_AVERIA(DirPn[i]);              modo_averia[i]=0;   LPC_TIM3->TCR=0x01;}
         }
}

////////////////////////////////////////////////////////////////////////////////////////////////////
                                                GPIO
////////////////////////////////////////////////////////////////////////////////////////////////////
void config_pines(void){ //Configuracion pines I/O

   LPC_GPIO0->FIODIR =(7<<23); //Configurar como salida P0.23:25
   LPC_GPIO2->FIODIR =(1<<9); //Configurar como salida P2.9
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                            SALIDA LUCES
////////////////////////////////////////////////////////////////////////////////////////////////////
void salidas(void){  //ACTUALIZA EL VALOR DE LAS LUCES

   Luzroja_n=~Luzroja;           //activo a nivel bajo
   Luzamarilla_n=~Luzamarilla;
   Luzverde_n=~Luzverde;

   //ROJO P2.9
   LPC_GPIO2->FIOPIN =((LPC_GPIO2->FIOPIN & ~(1<<9))|((((Luzroja_n)&1) & 0x1) << 9));
   //AMARILLO EXPANSOR 1
   I2CSendAddr_1(DirEP0,0); //direccion, escribir
   I2CSendByte_1(Luzamarilla_n); //enviar salidas de pines
   I2CSendStop_1();
   I2Cdelay_1(); //garantizar la escritura.

   //VERDE EXPANSOR 2
   I2CSendAddr_1(DirEP1,0); //direccion, escribir
   I2CSendByte_1(Luzverde_n); //enviar salidas de pines
   I2CSendStop_1();
   I2Cdelay_1(); //garantizar la escritura.
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                              MEMORIA
////////////////////////////////////////////////////////////////////////////////////////////////////
char plaza_libre(uint8_t plaza){  //PLAZA LIBRE??
   char respuesta;

   if(plaza<33 & plaza>0   ){if(((Memoria_1>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<65 & plaza>32  ){if(((Memoria_2>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<97 & plaza>64  ){if(((Memoria_3>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<129 & plaza>96 ){if(((Memoria_4>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<161 & plaza>128){if(((Memoria_5>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}
   if(plaza<193 & plaza>160){if(((Memoria_6>>plaza)&1)==1){respuesta = no;} else {respuesta=si;}}

   return respuesta; //SI=1 NO=0
}

void sumar_restar_plaza(uint8_t plaza, char operacion){  //operación=0 restar / operación=1 sumar
   char vector_mem, vector_bit,i;

   vector_mem = (plaza/32);            //32 plazas por registro
   vector_bit = plaza-(vector_mem*32);

    if(operacion==1){ //retirado
      Huecoslibres++;
      if(vector_mem==0){Memoria_1 &= ~(1<<vector_bit); }
      if(vector_mem==1){Memoria_2 &= ~(1<<vector_bit); }
      if(vector_mem==2){Memoria_3 &= ~(1<<vector_bit); }
      if(vector_mem==3){Memoria_4 &= ~(1<<vector_bit); }
      if(vector_mem==4){Memoria_5 &= ~(1<<vector_bit); }
      if(vector_mem==5){Memoria_6 &= ~(1<<vector_bit); }
   }
   if(operacion==0){ //aparcado
      Huecoslibres--;
      if(vector_mem==0){Memoria_1 |=  (1<<vector_bit); }
      if(vector_mem==1){Memoria_2 |=  (1<<vector_bit); }
      if(vector_mem==2){Memoria_3 |=  (1<<vector_bit); }
      if(vector_mem==3){Memoria_4 |=  (1<<vector_bit); }
      if(vector_mem==4){Memoria_5 |=  (1<<vector_bit); }
      if(vector_mem==5){Memoria_6 |=  (1<<vector_bit); }
   }
```

```
}
////////////////////////////////////////////////////////////////////////////////////////////////////////
                                        DETECTOR COCHE
////////////////////////////////////////////////////////////////////////////////////////////////////////
char detector_coche(uint8_t elevador){  //COCHE DETECTADO??
   char deteccion, lectura;

   I2CSendAddr_1(DirEP4,1); //leer
   lectura=I2CGetByte_1(1); //NACK, solo ese dato
   I2CSendStop_1();
   I2Cdelay_1();

   if(((lectura >> (elevador-1))&1)==1){deteccion = si;} // SI esta detectando
   else                                {deteccion = no;} // NO esta detectando
   return deteccion;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////
                                    SISTEMA DE SEGURIDAD
////////////////////////////////////////////////////////////////////////////////////////////////////////
void escribir_s_seg(char elevador, char estado){  //ACTIVAR/DESACTIVAR SISTEMAS DE SEGURIDAD
   static char dato;                               //estado=0 -> desactivar  /  estado=1 -> activar

   if(estado){dato |=  (1<<(elevador-1));} //sobrescribir 1
   else      {dato &= ~(1<<(elevador-1));}  //sobrescribir 0

   I2CSendAddr_1(DirEP2,0); //escribir en Expansor Puertos 2
   I2CSendByte_1(dato);
   I2CSendStop_1();
   I2Cdelay_1();
}
char leer_s_seg(char elevador){  //LEER APLICADO O NO SISTEMAS DE SEGURIDAD
   char dato, aplicado;  //aplicado=0 -> no (AVERIA)  /  aplicado=1 -> si (CORRECTO)

   I2CSendAddr_1(DirEP3,1); //lectura en EP3
   dato=I2CGetByte_1(1);    //NACK solo ese dato
   I2CSendStop_1();
   I2Cdelay_1();

   if(((dato>>(elevador))&1)==1){aplicado=1;}    //dato = E7 E6 E5 E4 E3 E2 E1 E0
   else                         {aplicado=0;}    //elevador = (1:8) --> restar 1
   return aplicado;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////
                                ELEVADOR --> APARCAR O RETIRAR
////////////////////////////////////////////////////////////////////////////////////////////////////////
void Seleccion_elevador(char sel){
   char enable, disable,i;

   enable = (sel<<1)|1;
   disable = (sel<<1)|1;
   I2CSendAddr_1(DirEP6,0); //Write EXP7
   I2CSendByte_1(enable);   //SE ENVIA EL ENABLE CON LA DIRECCIÓN
   I2CSendStop_1();

   for(i=0;i<C_ENABLE;i++){I2Cdelay_1();}

   I2CSendAddr_1(DirEP6,0); //Write EXP7
   I2CSendByte_1(disable);  //SE QUITA EL ENABLE TRAS LOS i CICLOS DE ESPERA
   I2CSendStop_1();
   I2Cdelay_1();
}
void MARCHA(uint8_t dato,char elev){ //enviar hueco y planta
   char i;

   I2CSendAddr_1(DirEP5,0); //Write
   I2CSendByte_1(dato);
   I2CSendStop_1();
   I2Cdelay_1();
   Seleccion_elevador(elev); //elev (0:7)
   flag_respE[elev]=1;
}



void ESPERA(char elev){ //CARGAR A LOAD EL DATO
   char planta, hueco;
   uint8_t dato;

   planta=(plaza_En[elev]-1)/Huecos_planta;
   hueco=plaza_En[elev]-1-(planta*Huecos_planta);
   LOAD_dato=((planta<<4)|hueco);
   LOAD_elev=elev;
}
void Elevador(){ //DESCARGAR DE LOAD EL DATO Y PONER EN MARCHA SI LA PLANTA ESTA LIBRE
   char j,planta;
   char a,b;

   if(REG_dato_esp[0]!=0xFF){                        //PRIMERO MIRAMOS SI HAY PLAZAS PENDIENTES
      if(PLANTA_USO[((REG_dato_esp[0]>>4)&0xF)]==0){ //PLANTA LIBRE
         MARCHA(REG_dato_esp[0],REG_elev_esp[0]);
         PLANTA_USO[((REG_dato_esp[0]>>4)&1)]=1;     //PLANTA OCUPADA
```

```
         for(j=0; j<8; j++){                          //REPOSICIONAMOS [0]
            REG_dato_esp[j]=REG_dato_esp[j+1];
            REG_elev_esp[j]=REG_elev_esp[j+1];
            if(j==7){REG_dato_esp[j]=0xFF;REG_elev_esp[j]=0xFF;}
   }  }  }

   //mover vector de espera
   a=REG_dato_esp[0];
   b=REG_elev_esp[0];
   for(j=0; j<8; j++){           //REPOSICIONAMOS [0]
      REG_dato_esp[j]=REG_dato_esp[j+1];
      REG_elev_esp[j]=REG_elev_esp[j+1];
      if(j==7){REG_dato_esp[j]=a;REG_elev_esp[j]=b;}
   }

   if(REG_dato[0]!=0xFF){                          //SEGUNDO MIRAMOS SI HAY PLAZAS NUEVAS
      if(PLANTA_USO[((REG_dato[0]>>4)&0xF)]==1){ //PLANTA EN USO
         for(j=0;j<8;j++){
            if(REG_dato_esp[j]==0xFF){REG_dato_esp[j]=REG_dato[0];REG_dato[0]=0xFF;}
            if(REG_elev_esp[j]==0xFF){REG_elev_esp[j]=REG_elev[0];REG_elev[0]=0xFF;}
   }   }  }

   if(REG_dato[0]!=0xFF){
      if(PLANTA_USO[((REG_dato[0]>>4)&0xF)]==0){ //planta libre
         MARCHA(REG_dato[0],REG_elev[0]);
         PLANTA_USO[((REG_dato[0]>>4)&0xF)]=1; }
   }

   for(j=0; j<8; j++){ //desplazamiento a derecha
      REG_dato[j]=REG_dato[j+1];
      REG_elev[j]=REG_elev[j+1];
      if(j==7){REG_dato[j]=0xFF;REG_elev[j]=0xFF;}
   }
   REG_dato[7]=LOAD_dato; // DESCARGAMOS DATO
   LOAD_dato=0xFF;
   REG_elev[7]=LOAD_elev;
   LOAD_elev=0xFF;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                        RESPUESTA ELEVADOR
////////////////////////////////////////////////////////////////////////////////////////////////////
char respuesta_PLC(char i){ // entrará cada 200*P_respE
   static char correcto, planta[8], cont_elev[8], cont_elev_NA[8], correcto_elevador[8],
               cont_planta[8], cont_planta_NA[8], correcto_planta[8];

//LEER PATILLA ELEVADOR   // E8=P1.25 E7=P1.24 E6=P1.23 E5=P1.22 E4=P1.21 E3=P1.20 E2=P1.19 E1=P1.18
   if(cont_elev[i]==0){
      if(((LPC_GPIO1->FIOPIN>>(18+i))&1)==1){
         cont_elev_NA[i]++;      // primer nivel alto
         cont_elev[i]=1;
   }  }
   else{ //ya ha venido el primer nivel alto
      cont_elev[i]++;
      if(((LPC_GPIO1->FIOPIN>>(18+i))&1)==1){
         cont_elev_NA[i]++;      // sumamos cont_correcto cada vez que haya un nivel alto
   }  }

   if(cont_elev[i]==12){
      if(cont_elev_NA[i]==6){cont_elev[i]=0;correcto_elevador[i]=1;}//correcto
      if(cont_elev_NA[i]==8){cont_elev[i]=0;correcto_elevador[i]=2;}//averia
   }
//LEER PATILLA PLANTA
   planta[i]=plaza_En[i]/Huecos_planta; // P15=P1.17  P14=P1.16  P13=P1.15  P12=P1.14
                                        P11=P1.10  P10=P1.9   P9=P1.8    P8=P1.4
                                        P3=P0.7    P2=P0.6    P1=P0.5    P0=P0.4
   if(cont_planta[i]==0){
      if(planta[i]<6){ // planta 0,1,2,3,4,5
         if(cont_planta[i]==0){
            if(((LPC_GPIO0->FIOPIN >>(4+planta[i]))&1)==1){
               cont_planta_NA[i]++;// primer nivel alto
               cont_planta[i]=1;
         }  }
         else{ //ya ha venido el primer nivel alto
            cont_planta[i]++;
            if(((LPC_GPIO0->FIOPIN >>(4+planta[i]))&1)==1){
               cont_planta_NA[i]++;         // sumamos cont_correcto cada vez que haya un nivel alto
      }  }  }
      if(planta[i]>5 & planta[i]<8){ // planta 6,7  P7=P1.1    P6=P1.0
         if(cont_planta[i]==0){
            if(((LPC_GPIO1->FIOPIN >>(0+planta[i]-6))&1)==1){
               cont_planta_NA[i]++;         // primer nivel alto
               cont_planta[i]=1;
         }  }
         else{ //ya ha venido el primer nivel alto
            cont_planta[i]++;
            if(((LPC_GPIO1->FIOPIN >>(0+planta[i]-6))&1)==1){
               cont_planta_NA[i]++;         sumamos cont_correcto cada vez que haya un nivel alto
      }  }  }
      if(planta[i]==8){ // planta 8    P8=P1.4
          if(cont_planta[i]==0){
             if(((LPC_GPIO1->FIOPIN >>(4))&1)==1){
                cont_planta_NA[i]++;         // primer nivel alto
```

```
                cont_planta[i]=1;
            }  }
            else{ //ya ha venido el primer nivel alto
               cont_planta[i]++;
               if(((LPC_GPIO1->FIOPIN >>(4))&1)==1){
                  cont_planta_NA[i]++; // sumamos cont_correcto cada vez que haya un nivel alto
         }  }  }
         if(planta[i]>8 & planta[i]<12){ // planta 9,10,11   P11=P1.10  P10=P1.9  P9=P1.8
            if(cont_planta[i]==0){
               if(((LPC_GPIO1->FIOPIN >>(8+planta[i]-9))&1)==1){
                  cont_planta_NA[i]++;         // primer nivel alto
                  cont_planta[i]=1;
            }  }
            else{ //ya ha venido el primer nivel alto
               cont_planta[i]++;
               if(((LPC_GPIO1->FIOPIN >>(8+planta[i]-9))&1)==1){
                  cont_planta_NA[i]++;  // sumamos cont_correcto cada vez que haya un nivel alto
         }  }  }
         if(planta[i]>11){ // planta 12,13,14,15  P15=P1.17  P14=P1.16  P13=P1.15  P12=P1.14
            if(cont_planta[i]==0){
               if(((LPC_GPIO1->FIOPIN >>(14+planta[i]-12))&1)==1){
                  cont_planta_NA[i]++;         // primer nivel alto
                  cont_planta[i]=1;
            }  }
            else{ //ya ha venido el primer nivel alto
               cont_planta[i]++;
               if(((LPC_GPIO1->FIOPIN >>(14+planta[i]-12))&1)==1){
                  cont_planta_NA[i]++;  // sumamos cont_correcto cada vez que haya un nivel alto
      }  }  }  }
      if(cont_planta[i]==12){
         if(cont_planta_NA[i]==6){cont_planta[i]=0;correcto_planta[i]=1;}//correcto
         if(cont_planta_NA[i]==8){cont_planta[i]=0;correcto_planta[i]=2;}//averia
      }
      if(correcto_planta[i]==1 & correcto_elevador[i]==1){correcto=1;} //CORRECTO
      if(correcto_planta[i]==1 & correcto_elevador[i]==2){correcto=2;} //AVERIA ELEVADOR
      if(correcto_planta[i]==2 & correcto_elevador[i]==1){correcto=3;} //AVERIA PLANTA
      if(correcto_planta[i]==2 & correcto_elevador[i]==2){correcto=4;} //AVERIA AMBOS

      return correcto; //correcto=1 averiado_ELEVADOR=2 averiado_PLANTA=3 averiado_TODO=4
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                                TIMER 0
////////////////////////////////////////////////////////////////////////////////////////////////////
void config_timer0(void){ //Configuracion TIMER0

   LPC_SC->PCONP|=(1<<1);              //Timer0 ON
   LPC_TIM0->PR=0;                     //Resolucion de 40 ns
   Ftic0=(Fpclk/(LPC_TIM0->PR + 1));
   LPC_TIM0->MCR=0x3;                  //Interrupt ON MR0, Reset ON MR0
   LPC_TIM0->MR0=(Ftic0*Ttim0*1e-6)-1; //Timer0 interrumpe cada 10 ms
   NVIC_EnableIRQ(TIMER0_IRQn);        //Habilitacion Timer0
   NVIC_SetPriority(TIMER0_IRQn,4);    //Timer0 tiene prioridad 1 subprioridad 0
}
void TIMER0_IRQHandler(void){ //Interrupcion TIMER0
   static char dir,i;

   LPC_TIM0->IR|=(1<<0); //Borrar flag de MR0
   cont_TIM0_2++;
   if(cont_TIM0_2==200){
      cont_TIM0_2=0;
      cont_TIM0++; //cada cuenta es un lector diferente
      LPC_TIM0->TCR=0x00; //counter disable para garantizar la ejecución

      dir=DirAn[cont_TIM0-1];
      I2CSendAddr_3(dir,1);
      plaza=I2CGetByte_3(1);
      I2CSendStop_3();
      I2Cdelay_3();

      if(bloquear_lector[cont_TIM0-1]==0){
         if(plaza!=0){ //Si el elevador n no ha sido activado dará plaza=0
            elevador=cont_TIM0;
            respuesta=plaza_libre(plaza);
            deteccion=detector_coche(elevador);
         if(respuesta==si & deteccion==no){ //plaza libre y coche no detectado --> 0 coches
            for(i=0;i<8;i++){
               if((elevador-1)==i){
                  bloquear_lector[i]=1;
                  flag_coches[i]=1;
                  ENV_0COCHES(DirPn[i]);
         }}}
         if(respuesta==no & deteccion==si){ //plaza ocupada y coche detectado --> 2 coches
            for(i=0;i<8;i++){
               if((elevador-1)==i){
                  bloquear_lector[i]=1;
                  flag_coches[i] =1;
                  ENV_2COCHES(DirPn[i]);
         }}}
         if((respuesta==no & deteccion==no)|(respuesta==si & deteccion==si)){ //retirar o aparcar
            bloquear_lector[elevador-1]=1;
            Ejecut |= (1<<(elevador-1));
            escribir_s_seg(elevador,1);      //ACTIVO (=1) los sistemas de seguridad
```

```
            flag_marcha[elevador-1]=1;
            plaza_En[elevador-1]=plaza;
            if(respuesta==1){movimiento[elevador-1]=0;}//aparcar
            else{movimiento[elevador-1]=1;}//retirar
         }}
      }
      if(plaza==0){elevador=0;}
      LPC_TIM0->TCR=0x01;//counter ensable
      if(cont_TIM0==NE){cont_TIM0=0;}
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                          TIMER 1
////////////////////////////////////////////////////////////////////////////////////////////////////
void config_timer1(void){ //Configuracion TIMER1

   LPC_SC->PCONP|=(1<<2); //Timer1 ON
   LPC_TIM1->PR=0; //resolucion de 40 ns
   Ftic1=(Fpclk/(LPC_TIM1->PR + 1));
   LPC_TIM1->MCR=0x3; //Interrupt ON MR0, Reset ON MR0
   LPC_TIM1->MR0=(Ftic1*Ttim1*1e-6)-1; //Timer1 interrumpe cada 10 us
   NVIC_EnableIRQ(TIMER1_IRQn); //Habilitacion Timer1
   NVIC_SetPriority(TIMER1_IRQn,4); //Timer1 tiene prioridad 1 subprioridad 0
}
void TIMER1_IRQHandler(void){ //Interrupcion TIMER1
   static char i,j,k,planta_elev[8];
   static uint32_t cont_Sseg[8]={0,0,0,0,0,0,0,0},cont_Elev[8]={0,0,0,0,0,0,0,0};

   LPC_TIM1->IR|=(1<<0); //Borrar flag de MR0
   cont_TIM1++;
   cont_TIM1_2++;

   if(cont_TIM1==1000){  //PARA 0COCHES O 2COCHES --> PONER ERROR Y QUITARLO AL CABO DE UN TIEMPO
      cont_TIM1=0;
      for(i=0;i<8;i++){
         if(flag_coches[i]==1){cont_coches[i]=0;flag_coches[i]=2;}
         if(flag_coches[i]==2){
            cont_coches[i]++;
            if(cont_coches[i]==P_mensaje){
               bloquear_lector[0]=1;flag_coches[i]=0;modo_disponible[i]=1;}
   }  }  }

   if(cont_TIM1_2==200){
      cont_TIM1_2=0;
      for(i=0;i<8;i++){

         if(flag_marcha[i]==1){ //LEER SI SE HAN APLICADO LOS S.SEG
            cont_Sseg[i]++;
            if(cont_Sseg[i]==P_sseg){
               cont_Sseg[i]=0;
               flag_marcha[i]=0;
               if(leer_s_seg(i)==si){//se han aplicado
                  ESPERA(i);
                  Elevador();}
               else{ // no se han aplicado
                  escribir_s_seg((i+1),0);      //DESACTIVO (=0) los sistemas de seguridad
                  Averia |= (1<<i);
                  Ejecut &= ~(1<<(i));
         }  }  }

         if(flag_respE[i]==1){ //RECOGER LA RESPUESTA DEL PLC
            cont_Elev[i]++;
            if(cont_Elev[i]==P_respE){
               flag_respE[i]=0;
               cont_Elev[i]=0;
               correcto[i]=respuesta_PLC(i); // entrará cada 200*P_respE   correcto=1 averiado=2
               planta_elev[i]=plaza_En[i]/Huecos_planta;
               escribir_s_seg((i+1),0);     //DESACTIVO (=0) los sistemas de seguirdad
               if(correcto[i]==1){ //correcto
                  Ejecut &= ~(1<<(i));
                  correcto[i]=0;
                  bloquear_lector[i]=0;
                  PLANTA_USO[planta_elev[i]]=0;
                  sumar_restar_plaza(plaza_En[i], movimiento[i]);
                  modo_disponible[i]=1;
                  FLAGS();
                  modos();}
               if(correcto[i]==2){ //averia  ELEVADOR
                  Ejecut &= ~(1<<i);
                  Averia |=  (1<<i);
                  PLANTA_USO[planta_elev[i]]=0; }
               if(correcto[i]==3){ //averia  PLANTA
                  Ejecut &= ~(1<<i);
                  Averia |=  (1<<i); }
               if(correcto[i]==4){ //averia  PLANTA Y ELEVADOR
                  Ejecut &= ~(1<<i);
                  Averia |=  (1<<i); }
   }  }  }  }
   Elevador(); //EJECUTAR PLAZAS EN ESPERA (DOS PLAZAS CON LA MISMA PLANTA)
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                          TIMER 3
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
void Prescaler_T3(void){

   P_parp=T_parp/Ttim3;
   P_act=T_act/Ttim3;
}
void config_timer3(void){ //Configuracion TIMER3

   LPC_SC->PCONP|=(1<<23); //Timer3 ON
   LPC_TIM3->PR=0; //resolucion de 40 ns
   Ftic3=(Fpclk/(LPC_TIM3->PR + 1));
   LPC_TIM3->MCR=0x3; //Interrupt ON MR0, Reset ON MR0
   LPC_TIM3->MR0=(Ftic3*Ttim3*1e-6)-1; //Timer3 interrumpe cada 100 us
   NVIC_EnableIRQ(TIMER3_IRQn); //Habilitacion Timer3
   NVIC_SetPriority(TIMER3_IRQn,4); //Timer3 tiene prioridad 1 subprioridad 0
   Prescaler_T3();
}
void TIMER3_IRQHandler(void){ //Interrupcion TIMER3
   static char i;

   LPC_TIM3->IR|=(1<<0); //Borrar flag de MR0
   cont_TIM3++;
   cont_TIM3_2++;

   if(Huecoslibres == 0) {Luzroja = 0x01;Luzamarilla=0x00; Luzverde=0x00; // NO HAY HUECOS
      if(flag_rojo==0){flag_rojo=1; modo_lleno=1;FLAGS();}
   }
   else {//HAY HUECOS --> ACTUALIZAR LUCES Y PANTALLA
      Luzroja = 0x00;
      if(flag_rojo==1){flag_rojo=0;}
      if(cont_TIM3 == P_parp){
         for(i=0;i<8;i++){
            if(NE>=(i+1)){
               if(((Averia>>i)&1)==1){Luzamarilla |= (1<<i) ;Luzverde &= ~(1<<i);
                  if(flag_amarillo1[i]==0){flag_amarillo1[i]=1;     modo_averia[i]=1;flag_verde[i]=0;
flag_amarillo2[i]=0;}}   // Averia en E0
               else{
                  if(((Ejecut>>i)&1)==1){Luzamarilla = ((Luzamarilla & ~(1<<i)) |
                     (((~(Luzamarilla>>i))&1)<<i));Luzverde &= ~(1<<i);
                     if(flag_amarillo2[i]==0){
                        flag_amarillo2[i]=1;
                        modo_ejecut[i]=1;
                        flag_verde[i]=0;flag_amarillo1[i]=0;
                     }
                  }
                  else {Luzamarilla &= ~(1<<i) ;Luzverde |=  (1<<i);
                     if(flag_verde[i]==0){
                        flag_verde[i]=1; modo_disponible[i]=1;
            }  }  }
            cont_TIM3=0;
         }
      }
      if(cont_TIM3_2 == P_act){cont_TIM3_2=0;
         if(Huecoslibres_r!=Huecoslibres){Huecoslibres_r=Huecoslibres;
            if(Huecoslibres != 0){FLAGS();}}} // ACTUALIZAR EL NUMERO DE PLAZAS EN LOS ELEVADORES QUE
NO SE HAN USADO

   if(inic_timer01==0){ // PRIMERO SE INICIALIZAN LAS LUCES Y PANTALLAS Y LUEGO SE ACTIVAN LOS
LECTORES Y TIM1
      inic_timer01=1;
      LPC_TIM0->TCR=0x01;
      LPC_TIM1->TCR=0x01;
   }
   salidas();
   modos();
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                                MODO MENU
////////////////////////////////////////////////////////////////////////////////////////////////////
void Menu(void){
           Plazas_menu(Huecos_planta,NE,Plantas);
           menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
P_respE_menu_2);
           inic_pantalla_menu();
           lcdInitDisplay();
           Uno_menu();
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                                MODO NORMAL
////////////////////////////////////////////////////////////////////////////////////////////////////
void Modo_normal(void){
   config_pines();
   config_timer0();
   config_timer1();
   config_timer3();
   NVIC_DisableIRQ(EINT0_IRQn); //Habilitacion EINT0
   NVIC_DisableIRQ(EINT1_IRQn); //Habilitacion EINT1
   NVIC_DisableIRQ(EINT2_IRQn); //Habilitacion EINT2
   inic_luces();
   inic_pantalla();
   Plazas_menu(Huecos_planta,NE,Plantas);
   inic_pantalla_menu();
```

```
   lcdInitDisplay();
   menu_vacio();
   LPC_TIM3->TCR=0x01; //counter enable
}
////////////////////////////////////////////////////////////////////////////////////////////////////////
                                                MENU
////////////////////////////////////////////////////////////////////////////////////////////////////////
void menu_1(char i){
   static char cont, j;

   if(i==1){ //1=key1
      cont--;
      NE--;
      if(NE<1){NE=1;}
      sprintf(borrar_numero,"    ");
      drawString(200, 80, borrar_numero, BLACK, BLACK, LARGE);
      sprintf(NE_PANTALLA,"%d",NE);
      drawString(200, 80, NE_PANTALLA, RED, BLACK, LARGE);
   }
   if(i==2){ //1=key2
      cont++;
      NE++;
      if(NE>8){NE=8;}
      sprintf(NE_PANTALLA,"%d",NE);
      sprintf(borrar_numero,"    ");
      drawString(200, 80, borrar_numero, BLACK, BLACK, LARGE);
      drawString(200, 80, NE_PANTALLA, RED, BLACK, LARGE);
   }
   Huecoslibres=Plantas*Huecos_planta;
   sprintf(borrar_fila,"                          ");
   sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
   drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
   drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);

   Plazas_menu(Huecos_planta, NE, Plantas);
   ENV_ELEV_menu(0x27);
   for(j=0;j<9;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_2(char i){
   static char cont, j;

   if(i==1){ //1=key1
      cont--;
      Plantas--;
      if(Plantas<1){Plantas=1;}
      sprintf(borrar_numero,"    ");
      drawString(185, 110, borrar_numero, BLACK, BLACK, LARGE);
      sprintf(Plantas_PANTALLA,"%d",Plantas);
      drawString(185, 110, Plantas_PANTALLA, RED, BLACK, LARGE);
   }
   if(i==2){ //1=key2
      cont++;
      Plantas++;
      if(Plantas>16){Plantas=16;}
      sprintf(borrar_numero,"    ");
      drawString(185, 110, borrar_numero, BLACK, BLACK, LARGE);
      sprintf(Plantas_PANTALLA,"%d",Plantas);
      drawString(185, 110, Plantas_PANTALLA, RED, BLACK, LARGE);
   }
   Huecoslibres=Plantas*Huecos_planta;
   sprintf(borrar_fila,"                          ");
   sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
   drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
   drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);

   Plazas_menu(Huecos_planta, NE, Plantas);
   ENV_PLANT_menu(0x27);
   for(j=0;j<8;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_3(char i){
   static char cont, j;

   if(i==1){ //1=key1
      cont--;
      Huecos_planta--;
      if(Huecos_planta<1){Huecos_planta=1;}
      sprintf(borrar_numero,"    ");
      drawString(185, 140, borrar_numero, BLACK, BLACK, LARGE);
      sprintf(Huecos_planta_PANTALLA,"%d",Huecos_planta);
      drawString(185, 140, Huecos_planta_PANTALLA, RED, BLACK, LARGE);
   }
   if(i==2){ //1=key2
      cont++;
      Huecos_planta++;
      if(Huecos_planta>12){Huecos_planta=12;}
      sprintf(borrar_numero,"    ");
      drawString(185, 140, borrar_numero, BLACK, BLACK, LARGE);
      sprintf(Huecos_planta_PANTALLA,"%d",Huecos_planta);
      drawString(185, 140, Huecos_planta_PANTALLA, RED, BLACK, LARGE);
   }
   Huecoslibres=Plantas*Huecos_planta;
   sprintf(borrar_fila,"                          ");
```

```
   sprintf(HUECOS_PANTALLA,"%d",Huecoslibres);
   drawString(0, 40, borrar_fila, BLACK, BLACK, LARGE);
   drawString(100, 40, HUECOS_PANTALLA, YELLOW, BLACK, LARGE);

   Plazas_menu(Huecos_planta, NE, Plantas);
   ENV_HUECO_menu(0x27);
   for(j=0;j<8;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_4(){
   borrar_pantalla();
   Avanzado_menu();
   avanzado=1;
   ENV_AVANZ_IN_menu(0x27);
}
////////////////////////////////////////////////////////////
void menu_av_1(char i){ //C_ENABLE
   static char cont=3, j;

   if(i==1){cont--;} //1=key1  bajar
   if(i==2){cont++;} //2=key2  subir

   if(cont <1){cont=1;}
   if(cont==1){C_ENABLE=1 ;}
   if(cont==2){C_ENABLE=3 ;}
   if(cont==3){C_ENABLE=5 ;}
   if(cont==4){C_ENABLE=7 ;}
   if(cont==5){C_ENABLE=10;}
   if(cont==6){C_ENABLE=12;}
   if(cont==7){C_ENABLE=15;}
   if(cont >7){cont=7;}

   sprintf(borrar_numero,"       ");
   drawString(165, 10, borrar_numero, BLACK, BLACK, LARGE);
   sprintf(C_ENABLE_PANTALLA,"%d ciclos",C_ENABLE);
   drawString(165, 10, C_ENABLE_PANTALLA, RED, BLACK, MEDIUM);

   menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
                            P_respE_menu_2);
   ENV_ENABLE_menu(0x27);
   for(j=0;j<12;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_av_2(char i){ //T_parp F_parp
   static char cont=3,j;

   if(i==1){cont--;} //1=key1  bajar
   if(i==2){cont++;} //2=key2  subir

   if(cont <1){cont=1;}
   if(cont==1){T_parp=253333;F_parp=1.00;F_parp_menu=10;}
   if(cont==2){T_parp=190000;F_parp=1.33;F_parp_menu=13;}
   if(cont==3){T_parp=126666;F_parp=2.00;F_parp_menu=20;}
   if(cont==4){T_parp=63333 ;F_parp=4.00;F_parp_menu=40;}
   if(cont >4){cont=4;}

   sprintf(borrar_numero,"         ");
   drawString(185, 40, borrar_numero, BLACK, BLACK, MEDIUM);
   sprintf(T_parp_PANTALLA,"%.2fHz",F_parp);
   drawString(185, 40, T_parp_PANTALLA, RED, BLACK, MEDIUM);

   menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
                            P_respE_menu_2);
   ENV_PARP_menu(0x27);
   for(j=0;j<8;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_av_3(char i){ //T_ACT
   static char cont=3,j;

   if(i==1){cont--;} //1=key1  bajar
   if(i==2){cont++;} //2=key2  subir

   if(cont <1){cont=1;}
   if(cont==1){T_act=250000 ; T_act_menu=1 ;}
   if(cont==2){T_act=1250000; T_act_menu=5 ;}
   if(cont==3){T_act=2500000; T_act_menu=10;}
   if(cont==4){T_act=5000000; T_act_menu=20;}
   if(cont >4){cont=4;}

   sprintf(borrar_numero,"         ");
   drawString(195, 70, borrar_numero, BLACK, BLACK, MEDIUM);
   sprintf(T_act_PANTALLA,"%d s",T_act_menu);
   drawString(195, 70, T_act_PANTALLA, RED, BLACK, MEDIUM);

   menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
                            P_respE_menu_2);
   ENV_ACT_menu(0x27);
   for(j=0;j<9;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_av_4(char i){ //P_menasaje
   static char cont=3,j;

   if(i==1){cont--;} //1=key1  bajar
   if(i==2){cont++;} //2=key2  subir
```

```
   if(cont <1){cont=1;}
   if(cont==1){P_mensaje=17; P_mensaje_menu=5  ;P_mensaje_menu_2=50;}
   if(cont==2){P_mensaje=26; P_mensaje_menu=7.5;P_mensaje_menu_2=75;}
   if(cont==3){P_mensaje=36; P_mensaje_menu=10 ;P_mensaje_menu_2=100;}
   if(cont==4){P_mensaje=54; P_mensaje_menu=15 ;P_mensaje_menu_2=150;}
   if(cont >4){cont=4;}

   sprintf(borrar_numero,"         ");
   drawString(145, 100, borrar_numero, BLACK, BLACK, MEDIUM);
   sprintf(P_mensaje_PANTALLA,"%.1f s",P_mensaje_menu);
   drawString(145, 100, P_mensaje_PANTALLA, RED, BLACK, MEDIUM);

   menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
                            P_respE_menu_2);
   ENV_MENSAJE_menu(0x27);
   for(j=0;j<8;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_av_5(char i){ //P_sseg
   static char cont=2,j;

   if(i==1){cont--;} //1=key1  bajar
   if(i==2){cont++;} //2=key2  subir

   if(cont <1){cont=1;}
   if(cont==1){P_sseg=130; P_sseg_menu=5 ;}
   if(cont==2){P_sseg=289; P_sseg_menu=10;}
   if(cont==3){P_sseg=450; P_sseg_menu=15;}
   if(cont==4){P_sseg=600; P_sseg_menu=20;}
   if(cont >4){cont=4;}

   sprintf(borrar_numero,"         ");
   drawString(170, 130, borrar_numero, BLACK, BLACK, MEDIUM);
   sprintf(P_sseg_PANTALLA,"%d s",P_sseg_menu);
   drawString(170, 130, P_sseg_PANTALLA, RED, BLACK, MEDIUM);

   menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
                            P_respE_menu_2);
   ENV_SEG_menu(0x27);
   for(j=0;j<9;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
void menu_av_6(char i){ //P_respE
   static char cont=3,j;

   if(i==1){cont--;} //1=key1  bajar
   if(i==2){cont++;} //2=key2  subir

   if(cont <1){cont=1;}
   if(cont==1){P_respE=4 ; P_respE_menu=0.1;P_respE_menu_2=1;}
   if(cont==2){P_respE=7 ; P_respE_menu=0.2;P_respE_menu_2=2;}
   if(cont==3){P_respE=15; P_respE_menu=0.5;P_respE_menu_2=5;}
   if(cont==4){P_respE=28; P_respE_menu=1  ;P_respE_menu_2=10;}
   if(cont==5){P_respE=56; P_respE_menu=2  ;P_respE_menu_2=20;}
   if(cont >5){cont=5;}

   sprintf(borrar_numero,"         ");
   drawString(170, 160, borrar_numero, BLACK, BLACK, MEDIUM);
   sprintf(P_respE_PANTALLA,"%0.1f s",P_respE_menu);
   drawString(170, 160, P_respE_PANTALLA, RED, BLACK, MEDIUM);

   menu_avanzado_parametros(C_ENABLE, F_parp_menu, T_act_menu, P_mensaje_menu_2, P_sseg_menu,
                            P_respE_menu_2);
   ENV_RESP_menu(0x27);
   for(j=0;j<8;j++){ENV_CONTROL_menu(0x1,0x0,0x27);}// SHIFT
}
////////////////////////////////////////////////////////////////////////////////////////////////////
                                                  EINT
////////////////////////////////////////////////////////////////////////////////////////////////////
void EINT0_IRQHandler(){ //Interrupcion EINT0   ISP

   LPC_SC->EXTINT=(1<<0);
   if(avanzado==0){
      if(selector==0){if(isp2==0){if(isp==0){isp=1;}}}
      if(selector==1){if(isp2==0){if(isp==0){selector_1OK(); isp=1; isp2=1;}}}
      if(selector==2){if(isp2==0){if(isp==0){selector_2OK(); isp=1; isp2=1;}}}
      if(selector==3){if(isp2==0){if(isp==0){selector_3OK(); isp=1; isp2=1;}}}
      if(selector==4){if(isp2==0){if(isp==0){selector_4OK(); isp=0; isp2=1; menu_4();selector=0;}}}
      if(selector==5){if(isp2==0){if(isp==0){selector_5OK(); isp=1; isp2=1; menu_vacio();
                                             Modo_normal();}}} //salir del menu

      if(selector==1){if(isp2==0){if(isp==1){selector_1(); isp=0; isp2=1;}}}
      if(selector==2){if(isp2==0){if(isp==1){selector_2(); isp=0; isp2=1;}}}
      if(selector==3){if(isp2==0){if(isp==1){selector_3(); isp=0; isp2=1;}}}
   }
   if(avanzado==1){
      if(sel_av==0){if(isp2==0){if(isp==0){isp=1;}}}
      if(sel_av==1){if(isp2==0){if(isp==0){sel_av_1OK(); isp=1; isp2=1;}}}
      if(sel_av==2){if(isp2==0){if(isp==0){sel_av_2OK(); isp=1; isp2=1;}}}
      if(sel_av==3){if(isp2==0){if(isp==0){sel_av_3OK(); isp=1; isp2=1;}}}
      if(sel_av==4){if(isp2==0){if(isp==0){sel_av_4OK(); isp=1; isp2=1;}}}
      if(sel_av==5){if(isp2==0){if(isp==0){sel_av_5OK(); isp=1; isp2=1;}}}
      if(sel_av==6){if(isp2==0){if(isp==0){sel_av_6OK(); isp=1; isp2=1;}}}
```

```
      if(sel_av==7){if(isp2==0){if(isp==0){sel_av_7OK(); isp=0; isp2=0; sel_av=0; avanzado=0;
                                          Uno_menu();}}}

      if(sel_av==1){if(isp2==0){if(isp==1){sel_av_1(); isp=0; isp2=1;}}}
      if(sel_av==2){if(isp2==0){if(isp==1){sel_av_2(); isp=0; isp2=1;}}}
      if(sel_av==3){if(isp2==0){if(isp==1){sel_av_3(); isp=0; isp2=1;}}}
      if(sel_av==4){if(isp2==0){if(isp==1){sel_av_4(); isp=0; isp2=1;}}}
      if(sel_av==5){if(isp2==0){if(isp==1){sel_av_5(); isp=0; isp2=1;}}}
      if(sel_av==6){if(isp2==0){if(isp==1){sel_av_6(); isp=0; isp2=1;}}}
   }
   isp2=0;
}

void EINT1_IRQHandler(){ //Interrupcion EINT1   KEY 1 DOWN

   LPC_SC->EXTINT=(1<<1);
   if(avanzado==0){
      if(selector==0){if(isp==0){if(key1==0){selector_5(); selector=5; key1=1;
ENV_PPAL_menu(0x27);}}}
      if(selector==1){if(isp==0){if(key1==0){selector_2(); selector=2; key1=1;
ENV_PLANT_menu(0x27);}}}
      if(selector==2){if(isp==0){if(key1==0){selector_3(); selector=3; key1=1;
ENV_HUECO_menu(0x27);}}}
      if(selector==3){if(isp==0){if(key1==0){selector_4(); selector=4; key1=1;
ENV_AVANZ_menu(0x27);}}}
      if(selector==4){if(isp==0){if(key1==0){selector_5(); selector=5; key1=1;
ENV_VALIDAR_menu(0x27);}}}
      if(selector==5){if(isp==0){if(key1==0){selector_1(); selector=1; key1=1;
ENV_ELEV_menu(0x27);}}}

      if(selector==1){if(isp==1){if(key1==0){menu_1(1);                key1=1;}}}
      if(selector==2){if(isp==1){if(key1==0){menu_2(1);                key1=1;}}}
      if(selector==3){if(isp==1){if(key1==0){menu_3(1);                key1=1;}}}
   }
   if(avanzado==1){
      if(sel_av==0){if(isp==0){if(key1==0){sel_av_7(); sel_av=7; key1=1;ENV_SALIR_menu(0x27);}}}
      if(sel_av==1){if(isp==0){if(key1==0){sel_av_2(); sel_av=2; key1=1;ENV_PARP_menu(0x27);}}}
      if(sel_av==2){if(isp==0){if(key1==0){sel_av_3(); sel_av=3; key1=1;ENV_ACT_menu(0x27);}}}
      if(sel_av==3){if(isp==0){if(key1==0){sel_av_4(); sel_av=4; key1=1;ENV_MENSAJE_menu(0x27);}}}
      if(sel_av==4){if(isp==0){if(key1==0){sel_av_5(); sel_av=5; key1=1;ENV_SEG_menu(0x27);}}}
      if(sel_av==5){if(isp==0){if(key1==0){sel_av_6(); sel_av=6; key1=1;ENV_RESP_menu(0x27);}}}
      if(sel_av==6){if(isp==0){if(key1==0){sel_av_7(); sel_av=7; key1=1;ENV_SALIR_menu(0x27);}}}
      if(sel_av==7){if(isp==0){if(key1==0){sel_av_1(); sel_av=1; key1=1;ENV_ENABLE_menu(0x27);}}}

      if(sel_av==1){if(isp==1){if(key1==0){menu_av_1(1);          key1=1;}}}
      if(sel_av==2){if(isp==1){if(key1==0){menu_av_2(1);          key1=1;}}}
      if(sel_av==3){if(isp==1){if(key1==0){menu_av_3(1);          key1=1;}}}
      if(sel_av==4){if(isp==1){if(key1==0){menu_av_4(1);          key1=1;}}}
      if(sel_av==5){if(isp==1){if(key1==0){menu_av_5(1);          key1=1;}}}
      if(sel_av==6){if(isp==1){if(key1==0){menu_av_6(1);          key1=1;}}}
   }
   key1=0;
}

void EINT2_IRQHandler(){ //Interrupcion EINT2   KEY 2 UP

   LPC_SC->EXTINT=(1<<2);
   if(avanzado==0){
      if(selector==0){if(isp==0){if(key2==0){selector_1(); selector=1; key2=1;ENV_ELEV_menu(0x27);}}}
      if(selector==1){if(isp==0){if(key2==0){selector_5(); selector=5;
key2=1;ENV_VALIDAR_menu(0x27);}}}
      if(selector==2){if(isp==0){if(key2==0){selector_1(); selector=1; key2=1;ENV_ELEV_menu(0x27);}}}
      if(selector==3){if(isp==0){if(key2==0){selector_2(); selector=2;
key2=1;ENV_PLANT_menu(0x27);}}}
      if(selector==4){if(isp==0){if(key2==0){selector_3(); selector=3;
key2=1;ENV_HUECO_menu(0x27);}}}
      if(selector==5){if(isp==0){if(key2==0){selector_4(); selector=4;
key2=1;ENV_AVANZ_menu(0x27);}}}

      if(selector==1){if(isp==1){if(key2==0){menu_1(2);                key2=1;}}}
      if(selector==2){if(isp==1){if(key2==0){menu_2(2);                key2=1;}}}
      if(selector==3){if(isp==1){if(key2==0){menu_3(2);                key2=1;}}}
   }
   if(avanzado==1){
      if(sel_av==0){if(isp==0){if(key2==0){sel_av_1(); sel_av=1; key2=1;ENV_ENABLE_menu(0x27);}}}
      if(sel_av==1){if(isp==0){if(key2==0){sel_av_7(); sel_av=7; key2=1;ENV_SALIR_menu(0x27);}}}
      if(sel_av==2){if(isp==0){if(key2==0){sel_av_1(); sel_av=1; key2=1;ENV_ENABLE_menu(0x27);}}}
      if(sel_av==3){if(isp==0){if(key2==0){sel_av_2(); sel_av=2; key2=1;ENV_PARP_menu(0x27);}}}
      if(sel_av==4){if(isp==0){if(key2==0){sel_av_3(); sel_av=3; key2=1;ENV_ACT_menu(0x27);}}}
      if(sel_av==5){if(isp==0){if(key2==0){sel_av_4(); sel_av=4; key2=1;ENV_MENSAJE_menu(0x27);}}}
      if(sel_av==6){if(isp==0){if(key2==0){sel_av_5(); sel_av=5; key2=1;ENV_SEG_menu(0x27);}}}
      if(sel_av==7){if(isp==0){if(key2==0){sel_av_6(); sel_av=6; key2=1;ENV_RESP_menu(0x27);}}}

      if(sel_av==1){if(isp==1){if(key2==0){menu_av_1(2);          key2=1;}}}
      if(sel_av==2){if(isp==1){if(key2==0){menu_av_2(2);          key2=1;}}}
      if(sel_av==3){if(isp==1){if(key2==0){menu_av_3(2);          key2=1;}}}
      if(sel_av==4){if(isp==1){if(key2==0){menu_av_4(2);          key2=1;}}}
      if(sel_av==5){if(isp==1){if(key2==0){menu_av_5(2);          key2=1;}}}
      if(sel_av==6){if(isp==1){if(key2==0){menu_av_6(2);          key2=1;}}}
   }
   key2=0;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
                                                MAIN
////////////////////////////////////////////////////////////////////////////////////////////////////
int main (void){

   LPC_PINCON->PINSEL4&=~(1<<(11*2)); //Configuracion del P2.11 como GPIO
   LPC_GPIO2->FIODIR&=~(1<<11); //Configurar como entrada
   if(((LPC_GPIO2->FIOPIN >>11) &0x1 )==0){ //Modo normal si ISP no esta pulsado
      Modo_normal();
      while(1);
   }
   if(((LPC_GPIO2->FIOPIN >>11) &0x1 )==1){  //Menu de configuraciones si ISP esta pulsado
      Menu();
      config_IRQ();
      while(1);
   }
}
```

ANEXO II. ARCHIVO "I2C_libreria_1.c"

Código fuente 32. Código completo Archivo "I2C_libreria_1.c"

```
#include <LPC17xx.h>
#define SDA1 0 //pin 0
#define SCL1 1 //pin 1
void I2Cdelay_1(void)//retardo minimo de 4.7
us
{
 uint32_t i;
 for(i=0;i<100;i++); //Modificar límite para
garantizar los tiempos (Bus standar --
>F_max=100kHz)
}
void I2Cdelay_toggle1(void)//retardo para
garantizar la conmutación
{
 unsigned char i;
 for(i=0;i<1;i++); //Modificar límite para
garantizar los tiempos (Bus standar --
>F_max=100kHz)
}

//Genera un pulso de reloj (1 ciclo)
void pulso_SCL1(void)
{
            LPC_GPIO0->FIOSET=(1<<SCL1);    //
Genera pulso de reloj (nivel alto)
            I2Cdelay_1();
            LPC_GPIO0->FIOCLR=(1<<SCL1);    //
Nivel bajo
            I2Cdelay_1();
}

void I2CSendByte_1(unsigned char byte)
{
 unsigned char i;
 for(i=0;i<8;i++){

            if (byte &0x80)  LPC_GPIO0-
>FIOSET=(1<<SDA1); // envia cada bit,
comenzando por el MSB
            else  LPC_GPIO0->FIOCLR=(1<<SDA1);
            byte = byte <<1;                //
siguiente bit
            I2Cdelay_toggle1();
            pulso_SCL1();
           }

//Leer ACK que envía el Slave (el Master ha de
enviar un pulso de reloj)
// CONFIGURAR PIN SDA COMO ENTRADA;
//espera ACK(config. pin como entrada)
 LPC_GPIO0->FIODIR&=~(1<<SDA1);
 pulso_SCL1();

// CONFIGURA PIN SDA COMO SALIDA;
 LPC_GPIO0->FIODIR|=(1<<SDA1);
}


//Función que envía START + Byte de dirección
del Slave (con bit LSB inicando R/W)
void I2CSendAddr_1(unsigned char addr,
unsigned char rw)
{
 //CONFIGURAR PINs SDA, SCL COMO SALIDAS;  //
Por si se nos olvidada en la conf. general.
 LPC_GPIO0->FIODIR|=(1<<SDA1)|(1<<SCL1);

 LPC_GPIO0->FIOSET|=(1<<SDA1)|(1<<SCL1);
// SDA y SCL a nivel alto para garantizar el

 I2Cdelay_1();
 LPC_GPIO0->FIOCLR=(1<<SDA1);
 I2Cdelay_1();
 LPC_GPIO0->FIOCLR=(1<<SCL1);
 I2Cdelay_1();
 I2CSendByte_1((addr=addr<<1) + rw); //envia
byte de direccion


}


// Función para leer un Byte del Slave. El
Master envía al final de la lectura
// el bit ACK o NACK (si es último byte leído)
que se pasa como argumento de la función.
unsigned char I2CGetByte_1(unsigned char ACK)
{
// ACK = 0, para cualquier byte que no sea el
ultimo.
// ACK = 1 (NACK), despues de leer el ultimo
byte
 unsigned char i, byte;
 //CONFIGURAR PIN SDA COMO ENTRADA;
            //configura pin SDA como entrada
 LPC_GPIO0->FIODIR&=~(1<<SDA1);
 for(i=0;i<8;i++){//lee un bit comenzando por
el MSB

                        LPC_GPIO0-
>FIOSET=(1<<SCL1);//mientras SCL=1
                        I2Cdelay_1();
                        byte=byte<<1;
                        if(LPC_GPIO0-
>FIOPIN&(1<<SDA1)) byte++;
            //Si leemos "1" sumamos para
introducir el "1"
                        LPC_GPIO0-
>FIOCLR=(1<<SCL1);//Si leemos "0" solo
desplazamos (se introduce un "0")
                        I2Cdelay_1();
                       }

 //CONFIGURAR PIN SDA COMO SALIDA;    //
Master envía un ACK por cada byte leído.
 LPC_GPIO0->FIODIR|=(1<<SDA1);

 if(ACK)LPC_GPIO0->FIOSET=(1<<SDA1);
// ACK o (NACK) es funcion del último byte
leído
            else LPC_GPIO0->FIOCLR=(1<<SDA1);

 pulso_SCL1(); // Pulso de reloj para su envío
 return (byte);
}

void I2CSendStop_1(void)
{
 LPC_GPIO0->FIOCLR=(1<<SDA1);
 I2Cdelay_1();
 LPC_GPIO0->FIOSET=(1<<SCL1);
// Subir SCL, y después SDA!! para dejar el
bus en reposo
 I2Cdelay_1();
 LPC_GPIO0->FIOSET=(1<<SDA1);
 I2Cdelay_1();
}

// Dejamos SDA de nuevo como salida
```

```
                                                // nivel
```

```
   //condicion de START: Bajar SDA y luego SCL
```

ANEXO III. ARCHIVO “1602A_Libreria.c”

Código fuente 33. Código completo Archivo “1602A_Libreria.c”

```
#include <LPC17xx.h>

char addr=0;
uint8_t HC, HD, HU, EU, PD, PU, QD, QU, CED,
CEU, TPU, TPd, TPc, TAD, TAU, PMD, PMU, PMd,
PSD, PSU, PRU, PRd;

void Huecos_libres(uint16_t libres){

            HC=(0x3<<4)|(libres/100);
            libres=libres%100;
            HD=(0x3<<4)|(libres/10);
            libres=libres%10;
            HU=(0x3<<4)|(libres/1);
}

void Plazas_menu(uint16_t quesitos, uint16_t
elevadores, uint16_t plantas){

            QD=(0x3<<4)|(quesitos/10);
            quesitos=quesitos%10;
            QU=(0x3<<4)|(quesitos/1);

            PD=(0x3<<4)|(plantas/10);
            plantas=plantas%10;
            PU=(0x3<<4)|(plantas/1);

            EU=(0x3<<4)|(elevadores/1);
}

void menu_avanzado_parametros(uint8_t
C_ENABLE, uint8_t F_parp, uint8_t T_act,
uint8_t P_mensaje, uint8_t P_sseg, uint8_t
P_respE){

            CED=(0x3<<4)|(C_ENABLE/10);
            C_ENABLE=C_ENABLE%10;
            CEU=(0x3<<4)|(C_ENABLE/1);

            TPU=(0x3<<4)|(F_parp/10);
            F_parp=F_parp%10;
            TPd=(0x3<<4)|(F_parp/1);
            TPc=(0x3<<4)|(F_parp/1);

            TAD=(0x3<<4)|(T_act/10);
            T_act=T_act%10;
            TAU=(0x3<<4)|(T_act/1);

            PMD=(0x3<<4)|(P_mensaje/100);
            P_mensaje=P_mensaje%100;
            PMU=(0x3<<4)|(P_mensaje/10);
            P_mensaje=P_mensaje%10;
            PMd=(0x3<<4)|(P_mensaje/1);

            PSD=(0x3<<4)|(P_sseg/10);
            P_sseg=P_sseg%10;
            PSU=(0x3<<4)|(P_sseg/1);

            PRU=(0x3<<4)|(P_respE/10);
            P_respE=P_respE%10;
            PRd=(0x3<<4)|(P_respE/1);
}

void delay_pantalla(void){//retardo minimo de
4.7 us
            uint32_t j=0;

            for(j=0;j<5000;j++);
}

void enviar(char dato,char dir){ //dato = D7
D6 D5 D4 BT E RW RS

            I2CSendAddr_2(dir,0); //pantalla 1
            I2CSendByte_2(dato);
            I2CSendStop_2();
            I2Cdelay_2();
}
void enviar_menu(char dato,char dir){ //dato =
D7 D6 D5 D4 BT E RW RS

            I2CSendAddr_1(dir,0); //pantalla 1
            I2CSendByte_1(dato);
            I2CSendStop_1();
            I2Cdelay_1();
}

void ENV_CONTROL(unsigned char datoa, unsigned
char datob, char dir){
            unsigned char BT=1, RW=0, RS=0;
//FONDO ENCENDIDO, ESCRITURA, CONTROL
            char
dato1=(datoa<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS;
            char
dato2=(datoa<<4)+(BT<<3)+(1<<2)+(RW<<1)+RS;
            char
dato3=(datob<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS;
            char
dato4=(datob<<4)+(BT<<3)+(1<<2)+(RW<<1)+RS;

            enviar(dato1,dir); // sin E
            enviar(dato2,dir); // con E
            enviar(dato1,dir); // sin E
            enviar(dato3,dir); // sin E
            enviar(dato4,dir); // con E
            enviar(dato3,dir); // sin E
            }
void ENV_CONTROL_menu(unsigned char datoa,
unsigned char datob, char dir){
            unsigned char BT=1, RW=0, RS=0;
//FONDO ENCENDIDO, ESCRITURA, CONTROL
            char
dato1=(datoa<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS;
            char
dato2=(datoa<<4)+(BT<<3)+(1<<2)+(RW<<1)+RS;
            char
dato3=(datob<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS;
            char
dato4=(datob<<4)+(BT<<3)+(1<<2)+(RW<<1)+RS;

            enviar_menu(dato1,dir); // sin E
            enviar_menu(dato2,dir); // con E
            enviar_menu(dato1,dir); // sin E
            enviar_menu(dato3,dir); // sin E
            enviar_menu(dato4,dir); // con E
            enviar_menu(dato3,dir); // sin E
            }

void ENV_DATO(unsigned char dato, char dir){
//dato = D7 D6 D5 D4
            unsigned char BT=1, RW=0, RS=1;
//FONDO ENCENDIDO, ESCRITURA, DATO

            enviar(((dato &
0xF0)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin
E
            enviar(((dato &
0xF0)+(BT<<3)+(1<<2)+(RW<<1)+RS),dir); // con
E
            enviar(((dato &
0xF0)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin
E
            enviar((((dato &
0x0F)<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); //
sin E
            enviar((((dato &
0x0F)<<4)+(BT<<3)+(1<<2)+(RW<<1)+RS),dir); //
con E
            enviar((((dato &
0x0F)<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); //
sin E
            }
void ENV_DATO_menu(unsigned char dato, char
dir){ //dato = D7 D6 D5 D4
            unsigned char BT=1, RW=0, RS=1;
//FONDO ENCENDIDO, ESCRITURA, DATO

            enviar_menu(((dato &
0xF0)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin
E
            enviar_menu(((dato &
0xF0)+(BT<<3)+(1<<2)+(RW<<1)+RS),dir); // con
E
            enviar_menu(((dato &
0xF0)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); // sin
E
            enviar_menu((((dato &
0x0F)<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); //
sin E
            enviar_menu((((dato &
0x0F)<<4)+(BT<<3)+(1<<2)+(RW<<1)+RS),dir); //
con E
```

```
            enviar_menu((((dato &
0x0F)<<4)+(BT<<3)+(0<<2)+(RW<<1)+RS),dir); //
sin E
            }


void ENV_DISPONIBLE(char dir){
            char i;

            ENV_CONTROL(0x0,0x1,dir); // CLEAR
            delay_pantalla();
            ENV_CONTROL(0x1,0xC,dir); // SHIFT

            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('D',dir);
            ENV_DATO('I',dir);
            ENV_DATO('S',dir);
            ENV_DATO('P',dir);
            ENV_DATO('0',dir);
            ENV_DATO('N',dir);
            ENV_DATO('I',dir);
            ENV_DATO('B',dir);
            ENV_DATO('L',dir);
            ENV_DATO('E',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);

            for(i=0;i<24;i++){ENV_DATO('
',dir);}

            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('L',dir);
            ENV_DATO('I',dir);
            ENV_DATO('B',dir);
            ENV_DATO('R',dir);
            ENV_DATO('E',dir);
            ENV_DATO('S',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('=',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(HC,dir);
            ENV_DATO(HD,dir);
            ENV_DATO(HU,dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
}


void LLENO_sub(char dir){
            char i;

            ENV_CONTROL(0x0,0x1,dir); // CLEAR
            delay_pantalla();
            ENV_CONTROL(0x1,0xC,dir); // SHIFT

            ENV_DATO(' ',dir);
            ENV_DATO('P',dir);
            ENV_DATO('A',dir);
            ENV_DATO('R',dir);
            ENV_DATO('K',dir);
            ENV_DATO('I',dir);
            ENV_DATO('N',dir);
            ENV_DATO('G',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('L',dir);
            ENV_DATO('L',dir);
            ENV_DATO('E',dir);
            ENV_DATO('N',dir);
            ENV_DATO('O',dir);
            ENV_DATO(' ',dir);

            for(i=0;i<24;i++){ENV_DATO('
',dir);}

            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('L',dir);
            ENV_DATO('I',dir);
            ENV_DATO('B',dir);
            ENV_DATO('R',dir);
            ENV_DATO('E',dir);
            ENV_DATO('S',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('=',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(HC,dir);
            ENV_DATO(HD,dir);
            ENV_DATO(HU,dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
}

void ENV_LLENO(char dir){
            LLENO_sub(0x20);
            LLENO_sub(0x21);
            LLENO_sub(0x22);
            LLENO_sub(0x23);
            LLENO_sub(0x24);
            LLENO_sub(0x25);
            LLENO_sub(0x26);
            LLENO_sub(0x27);
}

void ENV_AVERIA(char dir){
            char i;

            ENV_CONTROL(0x0,0x1,dir); // CLEAR
            delay_pantalla();
            ENV_CONTROL(0x1,0xC,dir); // SHIFT

            ENV_DATO('E',dir);
            ENV_DATO('L',dir);
            ENV_DATO('E',dir);
            ENV_DATO('V',dir);
            ENV_DATO('A',dir);
            ENV_DATO('D',dir);
            ENV_DATO('.',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('A',dir);
            ENV_DATO('V',dir);
            ENV_DATO('E',dir);
            ENV_DATO('R',dir);
            ENV_DATO('I',dir);
            ENV_DATO('A',dir);
            ENV_DATO('D',dir);
            ENV_DATO('O',dir);

            for(i=0;i<24;i++){ENV_DATO('
',dir);}

            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('L',dir);
            ENV_DATO('I',dir);
            ENV_DATO('B',dir);
            ENV_DATO('R',dir);
            ENV_DATO('E',dir);
            ENV_DATO('S',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('=',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(HC,dir);
            ENV_DATO(HD,dir);
            ENV_DATO(HU,dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
}
void ENV_EJECUT(char dir){
            char i;

            ENV_CONTROL(0x0,0x1,dir); // CLEAR
            delay_pantalla();
            ENV_CONTROL(0x1,0xC,dir); // SHIFT

            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('E',dir);
            ENV_DATO('N',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('E',dir);
            ENV_DATO('J',dir);
            ENV_DATO('E',dir);
            ENV_DATO('C',dir);
            ENV_DATO('U',dir);
            ENV_DATO('C',dir);
            ENV_DATO('I',dir);
            ENV_DATO('O',dir);
            ENV_DATO('N',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);

            for(i=0;i<24;i++){ENV_DATO('
',dir);}

            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('L',dir);
            ENV_DATO('I',dir);
            ENV_DATO('B',dir);
```

```
            ENV_DATO('R',dir);
            ENV_DATO('E',dir);
            ENV_DATO('S',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('=',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(HC,dir);
            ENV_DATO(HD,dir);
            ENV_DATO(HU,dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
}
void ENV_0COCHES(char dir){
            char i;

            ENV_CONTROL(0x0,0x1,dir); // CLEAR
            delay_pantalla();
            ENV_CONTROL(0x1,0xC,dir); // SHIFT

            ENV_DATO('N',dir);
            ENV_DATO('O',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('D',dir);
            ENV_DATO('E',dir);
            ENV_DATO('T',dir);
            ENV_DATO('E',dir);
            ENV_DATO('C',dir);
            ENV_DATO('T',dir);
            ENV_DATO('A',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('C',dir);
            ENV_DATO('O',dir);
            ENV_DATO('C',dir);
            ENV_DATO('H',dir);
            ENV_DATO('E',dir);

            for(i=0;i<24;i++){ENV_DATO('
',dir);}

            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('P',dir);
            ENV_DATO('L',dir);
            ENV_DATO('A',dir);
            ENV_DATO('Z',dir);
            ENV_DATO('A',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('L',dir);
            ENV_DATO('I',dir);
            ENV_DATO('B',dir);
            ENV_DATO('R',dir);
            ENV_DATO('E',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
}
void ENV_2COCHES(char dir){
            char i;

            ENV_CONTROL(0x0,0x1,dir); // CLEAR
            delay_pantalla();
            ENV_CONTROL(0x1,0xC,dir); // SHIFT

            ENV_DATO('C',dir);
            ENV_DATO('O',dir);
            ENV_DATO('C',dir);
            ENV_DATO('H',dir);
            ENV_DATO('E',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('D',dir);
            ENV_DATO('E',dir);
            ENV_DATO('T',dir);
            ENV_DATO('E',dir);
            ENV_DATO('C',dir);
            ENV_DATO('T',dir);
            ENV_DATO('A',dir);
            ENV_DATO('D',dir);
            ENV_DATO('O',dir);

            for(i=0;i<24;i++){ENV_DATO('
',dir);}

            ENV_DATO(' ',dir);
            ENV_DATO('P',dir);
            ENV_DATO('L',dir);
            ENV_DATO('A',dir);
            ENV_DATO('Z',dir);
            ENV_DATO('A',dir);
            ENV_DATO(' ',dir);
            ENV_DATO(' ',dir);
            ENV_DATO('O',dir);
            ENV_DATO('C',dir);
            ENV_DATO('U',dir);
            ENV_DATO('P',dir);
            ENV_DATO('A',dir);
            ENV_DATO('D',dir);
            ENV_DATO('A',dir);
            ENV_DATO(' ',dir);
}

void ENV_PPAL_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('=',dir);
            ENV_DATO_menu(PD,dir);
            ENV_DATO_menu(PU,dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('V',dir);
            ENV_DATO_menu('=',dir);
            ENV_DATO_menu(EU,dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu('H',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('C',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('X',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('=',dir);
            ENV_DATO_menu(QD,dir);
            ENV_DATO_menu(QU,dir);
}
void ENV_ELEV_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('V',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('D',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu('R',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(EU,dir);
```

```
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);

}
void ENV_PLANT_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(PD,dir);
            ENV_DATO_menu(PU,dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_HUECO_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('H',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('C',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('X',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(QD,dir);
            ENV_DATO_menu(QU,dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_AVANZ_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('V',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('Z',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('D',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('P',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_AVANZ_IN_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('V',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('Z',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('D',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
```

```
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_ENABLE_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('C',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('C',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('B',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(CED,dir);
            ENV_DATO_menu(CEU,dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('c',dir);
            ENV_DATO_menu('i',dir);
            ENV_DATO_menu('c',dir);
  ENV_DATO_menu('l',dir);
            ENV_DATO_menu('o',dir);
            ENV_DATO_menu('s',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_PARP_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('R',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('D',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('O',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(TPU,dir);
            ENV_DATO_menu('.',dir);
```

```
            ENV_DATO_menu(TPd,dir);
            ENV_DATO_menu(TPc,dir);
            ENV_DATO_menu('H',dir);
            ENV_DATO_menu('z',dir);
            ENV_DATO_menu(' ',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_ACT_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('C',dir);
            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('Z',dir);
            ENV_DATO_menu('.',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(TAD,dir);
            ENV_DATO_menu(TAU,dir);
            ENV_DATO_menu('s',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_MENSAJE_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('N',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('J',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(PMD,dir);
            ENV_DATO_menu(PMU,dir);
```

```
            ENV_DATO_menu('.',dir);
            ENV_DATO_menu(PMd,dir);
            ENV_DATO_menu('s',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_SEG_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('.',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('G',dir);
            ENV_DATO_menu('.',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(PSD,dir);
            ENV_DATO_menu(PSU,dir);
            ENV_DATO_menu('s',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_RESP_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu('T',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('M',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('O',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('R',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('.',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('C',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(PRU,dir);
```

```
            ENV_DATO_menu('.',dir);
            ENV_DATO_menu(PRd,dir);
            ENV_DATO_menu('s',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}
void ENV_SALIR_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('R',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('P',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}

void ENV_VALIDAR_menu(char dir){
            char i;

            ENV_CONTROL_menu(0x0,0x1,dir); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x1,0xC,dir); //
SHIFT

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('V',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('L',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('D',dir);
            ENV_DATO_menu('A',dir);
            ENV_DATO_menu('R',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);

            for(i=0;i<24;i++){ENV_DATO_menu('
',dir);}

            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('P',dir);
            ENV_DATO_menu('U',dir);
            ENV_DATO_menu('L',dir);
```

```
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('E',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu('I',dir);
            ENV_DATO_menu('S',dir);
            ENV_DATO_menu('P',dir);
  ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
            ENV_DATO_menu(' ',dir);
}

void init_pantalla_sub(char dir){

            delay_pantalla();
            ENV_CONTROL(0x2,0xC,dir); //4 BITS
2 FILAS
            delay_pantalla();
            ENV_CONTROL(0x0,0x1,dir); // CLEAR
            delay_pantalla();
            ENV_CONTROL(0x0,0x2,dir); //RETURN
HOME
            delay_pantalla();

            ENV_CONTROL(0x0,0xE,dir); //DISPLAY
ON, SI CURSOR, NO BLINK
            delay_pantalla();
}
void inic_pantalla(){

            Huecos_libres(192);
            init_pantalla_sub(0x20);
            init_pantalla_sub(0x21);
            init_pantalla_sub(0x22);
            init_pantalla_sub(0x23);
            init_pantalla_sub(0x24);
            init_pantalla_sub(0x25);
            init_pantalla_sub(0x26);
            init_pantalla_sub(0x27);
            ENV_DISPONIBLE(0x20);
            ENV_DISPONIBLE(0x21);
            ENV_DISPONIBLE(0x22);
            ENV_DISPONIBLE(0x23);
            ENV_DISPONIBLE(0x24);
            ENV_DISPONIBLE(0x25);
            ENV_DISPONIBLE(0x26);
            ENV_DISPONIBLE(0x27);
}

void inic_pantalla_menu(){

            delay_pantalla();
            ENV_CONTROL_menu(0x2,0xC,0x27); //4
BITS 2 FILAS
            delay_pantalla();
            ENV_CONTROL_menu(0x0,0x1,0x27); //
CLEAR
            delay_pantalla();
            ENV_CONTROL_menu(0x0,0x2,0x27);
//RETURN HOME
            delay_pantalla();
            ENV_CONTROL_menu(0x0,0xF,0x27);
//DISPLAY ON, SI CURSOR, SI BLINK
            delay_pantalla();
            ENV_PPAL_menu(0x27);
}
```

ANEXO IV. ARCHIVO “Librería_menu.c”

Código fuente 34. Código completo Archivo “Librería_menu.c”

```
#include <LPC17xx.h>
#include "lcddriver.h"
void selector_0(){ // todo normal
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK, LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK, LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK, LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK, LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK, LARGE);
}
void selector_1(){ // elevadores
          char j;
          drawString( 5, 80, " ELEVADORES:",
BLUE   , WHITE, LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK, LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK, LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK, LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK, LARGE);
          for(j=0;j<9;j++){ENV_CONTROL_menu(0
x1,0x04,0x27);}// SHIFT
}
void selector_2(){ // plantas
          char j;
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK, LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
BLUE   , WHITE, LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK, LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK, LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK, LARGE);
          for(j=0;j<8;j++){ENV_CONTROL_menu(0
x1,0x04,0x27);}// SHIFT
}
void selector_3(){ // huecos
          char j;
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK, LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK, LARGE);
          drawString( 5, 140, " HUECOS:"   ,
BLUE   , WHITE, LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK, LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK, LARGE);
          for(j=0;j<8;j++){ENV_CONTROL_menu(0
x1,0x04,0x27);}// SHIFT
}
void selector_4(){ // avanzado
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK, LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK, LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK, LARGE);
          drawString(37, 180, " AVANZADO"  ,
RED    , WHITE, LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK, LARGE);
}
void selector_5(){ // validar
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK, LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK, LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK, LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK, LARGE);
          drawString(45, 210, " VALIDAR"   ,
RED    , WHITE, LARGE);
}
//////////////////////////////////////
void selector_1OK(){ // elevadores
          char j;
          drawString( 5, 80, " ELEVADORES:",
BLUE   , YELLOW, LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK , LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK , LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK , LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK , LARGE);
          for(j=0;j<9;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void selector_2OK(){ // plantas
          char j;
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK , LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
BLUE   , YELLOW, LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK , LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK , LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK , LARGE);
          for(j=0;j<8;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void selector_3OK(){ // huecos
          char j;
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK , LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK , LARGE);
          drawString( 5, 140, " HUECOS:"   ,
BLUE   , YELLOW, LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK , LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK , LARGE);
          for(j=0;j<8;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void selector_4OK(){ // avanzado
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK , LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK , LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK , LARGE);
          drawString(37, 180, " AVANZADO"  ,
BLACK  , YELLOW, LARGE);
          drawString(45, 210, " VALIDAR"   ,
MAGENTA, BLACK , LARGE);
}
void selector_5OK(){ // validar
          drawString( 5, 80, " ELEVADORES:",
CYAN   , BLACK , LARGE);
          drawString( 5, 110, " PLANTAS:"  ,
CYAN   , BLACK , LARGE);
          drawString( 5, 140, " HUECOS:"   ,
CYAN   , BLACK , LARGE);
          drawString(37, 180, " AVANZADO"  ,
YELLOW , BLACK , LARGE);
          drawString(45, 210, " VALIDAR"   ,
BLACK  , YELLOW, LARGE);
}
//////////////////////////////////////
void sel_av_0(){ // todo normal
          drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
          drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
          drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
          drawString(10, 100, "TIEMPO
MENSAJE: "     , YELLOW, BLACK, MEDIUM);
          drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
          drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
          drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
}
void sel_av_1(){ // elevadores
          drawString(10, 10, "CICLOS ENABLE
MUX: "   , BLUE  , WHITE, MEDIUM);
```

```
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
}
void sel_av_2(){ // plantas
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , BLUE  , WHITE, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
}
void sel_av_3(){ // huecos
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", BLUE  , WHITE, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
}
void sel_av_4(){ // avanzado
  drawString(10, 10, "CICLOS ENABLE MUX: "   ,
YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , BLUE  , WHITE, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
}
void sel_av_5(){ // validar
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , BLUE  , WHITE, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
}
void sel_av_6(){ // validar
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , BLUE  , WHITE, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
}
void sel_av_7(){ // validar
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, RED   , WHITE, LARGE);
}
////////////////////////////////////////
void sel_av_1OK(){ // elevadores
            char j;
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , BLUE  , YELLOW, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
            for(j=0;j<12;j++){ENV_CONTROL_menu(
0x1,0x0,0x27);}// SHIFT
}
void sel_av_2OK(){ // plantas
            char j;
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , BLUE  , YELLOW, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
            for(j=0;j<8;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void sel_av_3OK(){ // huecos
            char j;
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", BLUE  , YELLOW, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
            for(j=0;j<9;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void sel_av_4OK(){ // avanzado
  char j;
            drawString(10, 10, "CICLOS ENABLE
MUX: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , BLUE  , YELLOW, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "  , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "  , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
```

```
            for(j=0;j<8;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void sel_av_5OK(){ // validar
            char j;
            drawString(10, 10, "CICLOS ENABLE
MUX: "    , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "   , BLUE   , YELLOW, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "   , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
            for(j=0;j<9;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void sel_av_6OK(){ // validar
            char j;
            drawString(10, 10, "CICLOS ENABLE
MUX: "    , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "   , BLUE   , YELLOW, MEDIUM);
            drawString(70, 210, "SALIR"
, MAGENTA, BLACK, LARGE);
            for(j=0;j<8;j++){ENV_CONTROL_menu(0
x1,0x0,0x27);}// SHIFT
}
void sel_av_7OK(){ // validar
            drawString(10, 10, "CICLOS ENABLE
MUX: "    , YELLOW, BLACK, MEDIUM);
            drawString(10, 40, "FRECUENCIA
PARPADEO: " , YELLOW, BLACK, MEDIUM);
            drawString(10, 70, "TIEMPO
ACTUALIZACION: ", YELLOW, BLACK, MEDIUM);
            drawString(10, 100, "TIEMPO
MENSAJE: "      , YELLOW, BLACK, MEDIUM);
            drawString(10, 130, "TIEMPO SIST.
SEG.: "   , YELLOW, BLACK, MEDIUM);
            drawString(10, 160, "PERIODO RESP.
PLC: "   , YELLOW, BLACK, MEDIUM);
            drawString(70, 210, "SALIR"
, BLUE   , CYAN, LARGE);
}
```

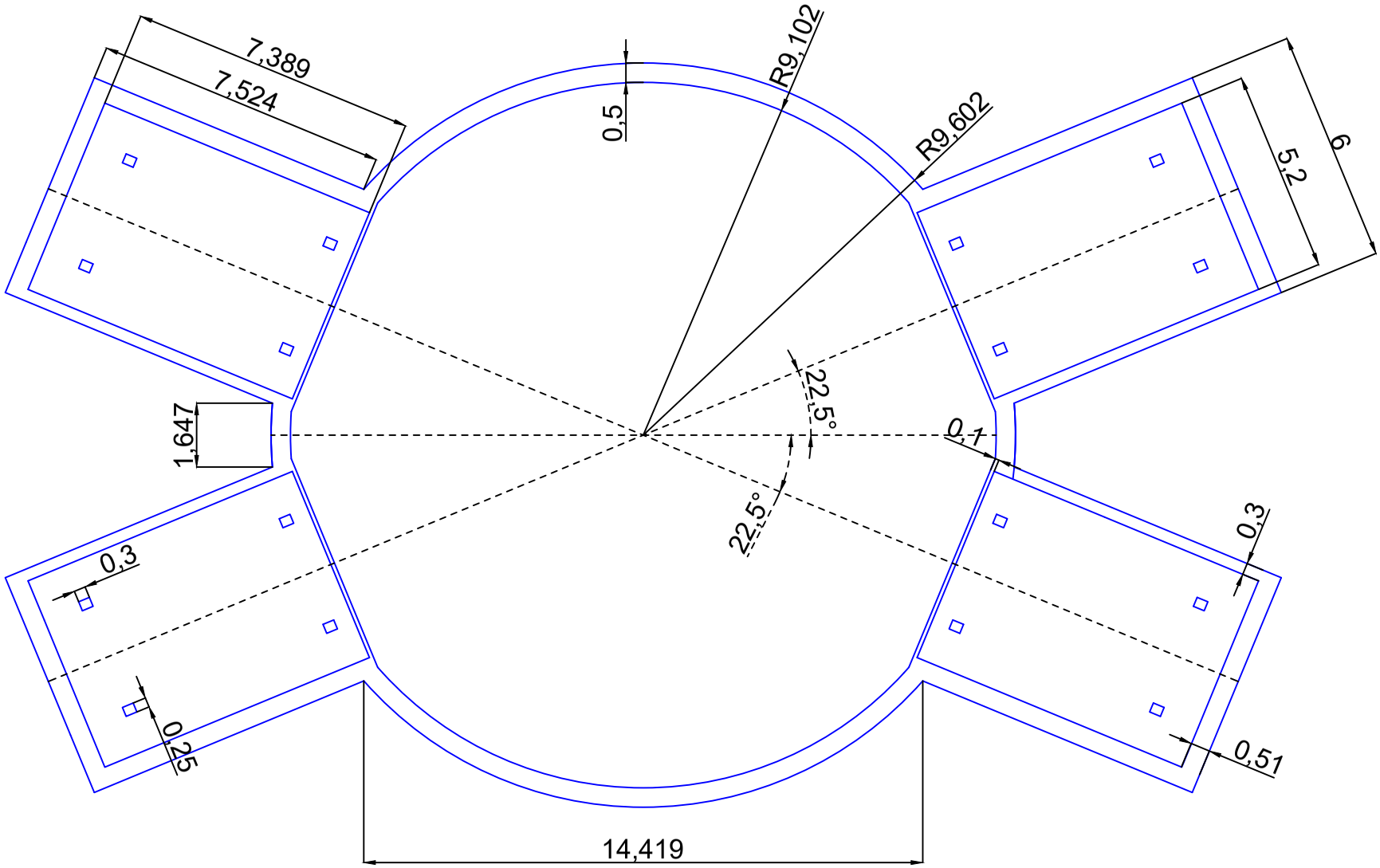
ANEXO V. PLANO DEL FOSO

ANEXO VI. PLANO DE LA PLANTA PRINCIPAL

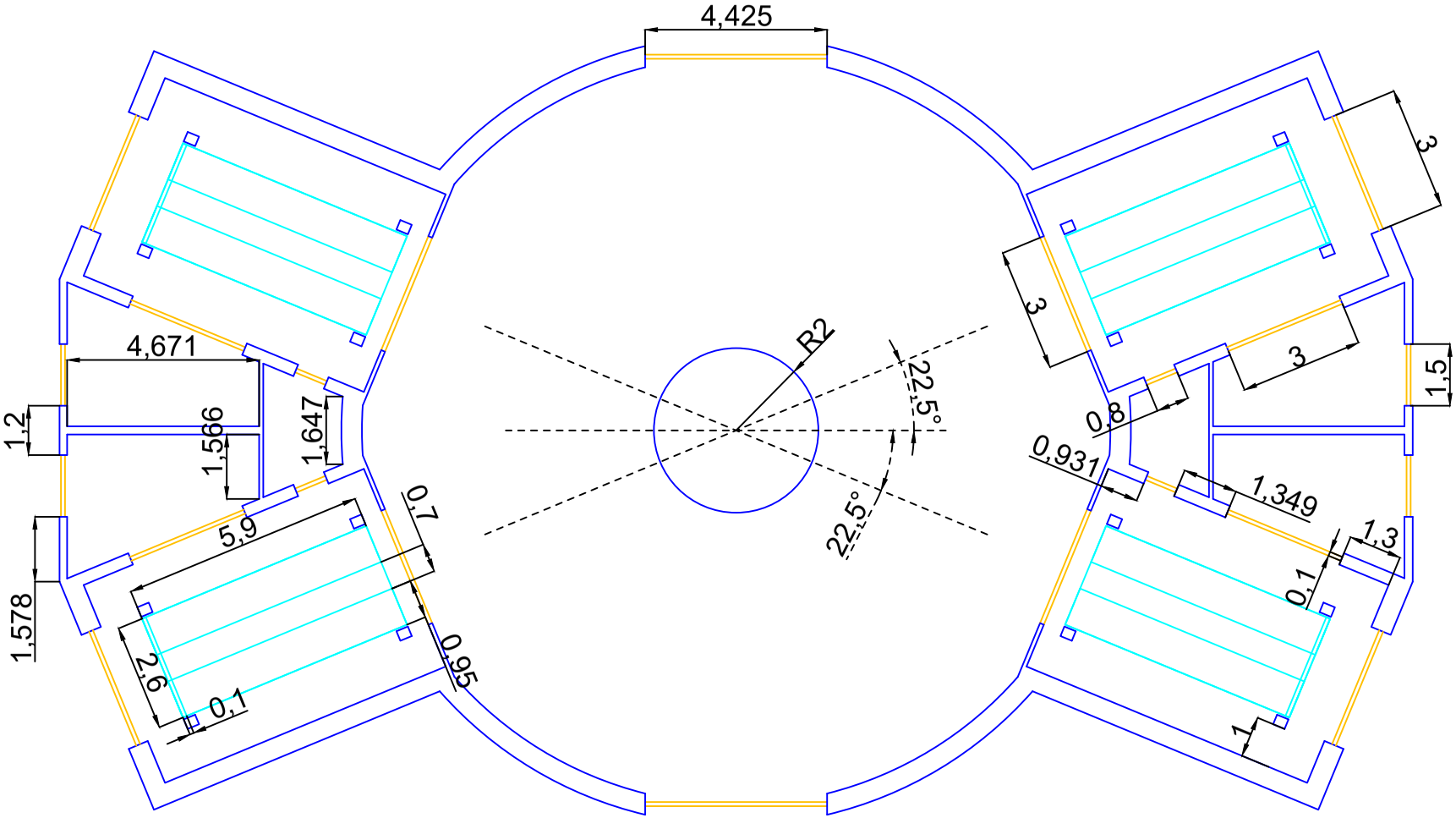
ANEXO VII. PLANO DE LA PLANTA DE CONTROL

ANEXO VIII. PLANO DE LA PLANTA DE ALMACENAJE

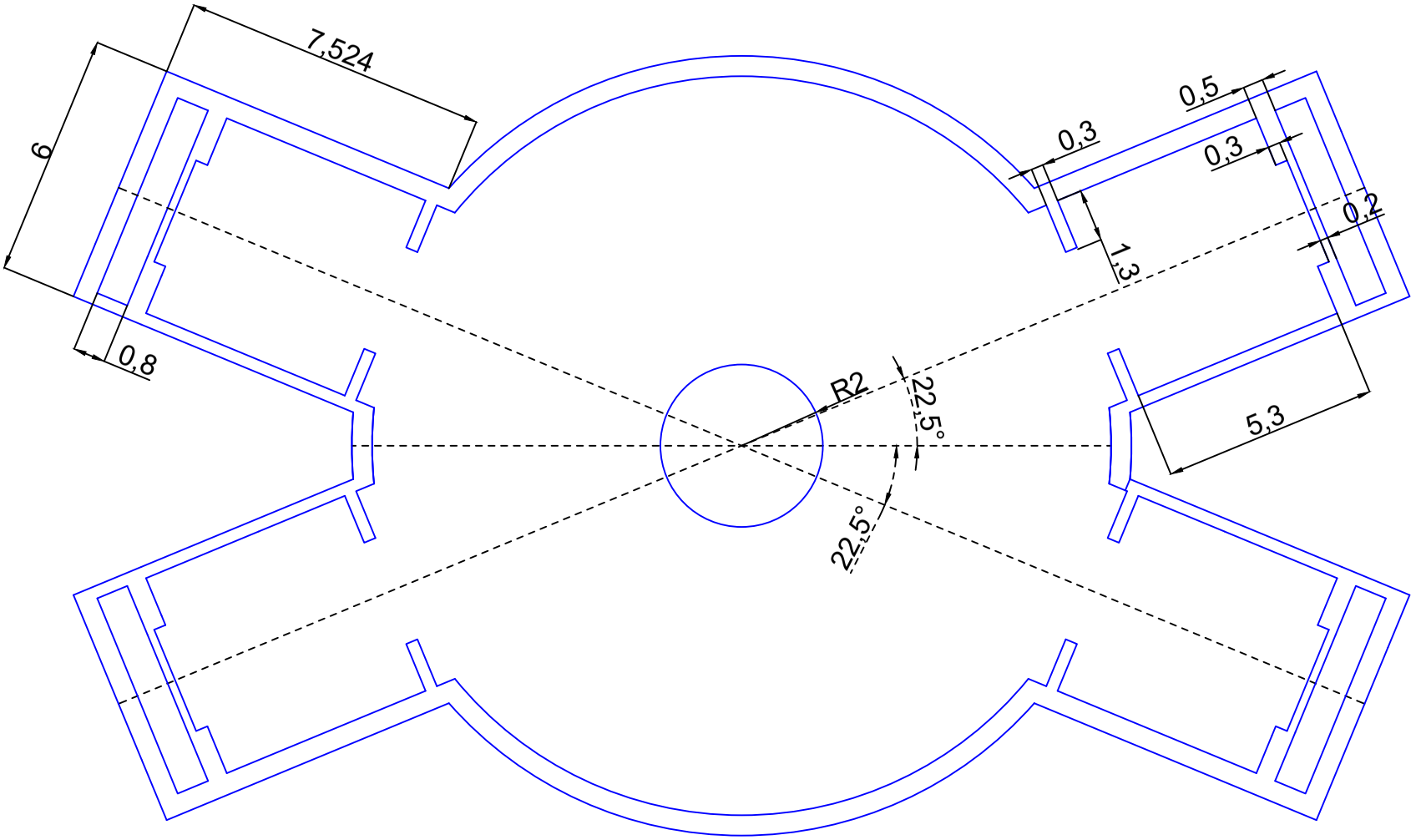
ANEXO IX. PLANO DE LA AZOTEA



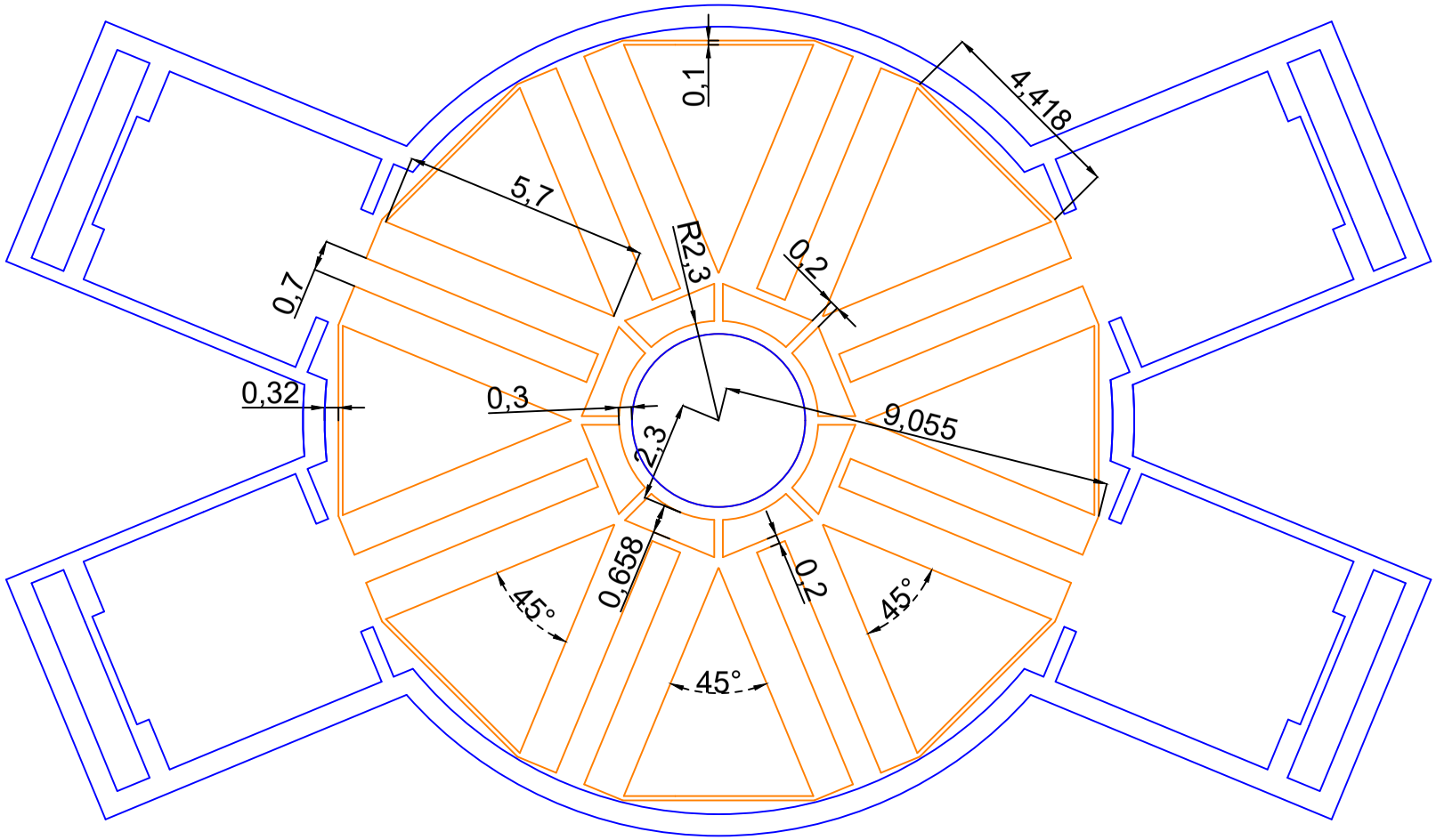
NOMBRE	ESCALA	UNIVERSIDAD DE ALCALÁ DE HERANRES
ALEJANDRO VILLALBA HERNANDO	1:150	Diseño de un sistema automatizado de aparcamiento de vehículos
TÍTULO: FOSO		



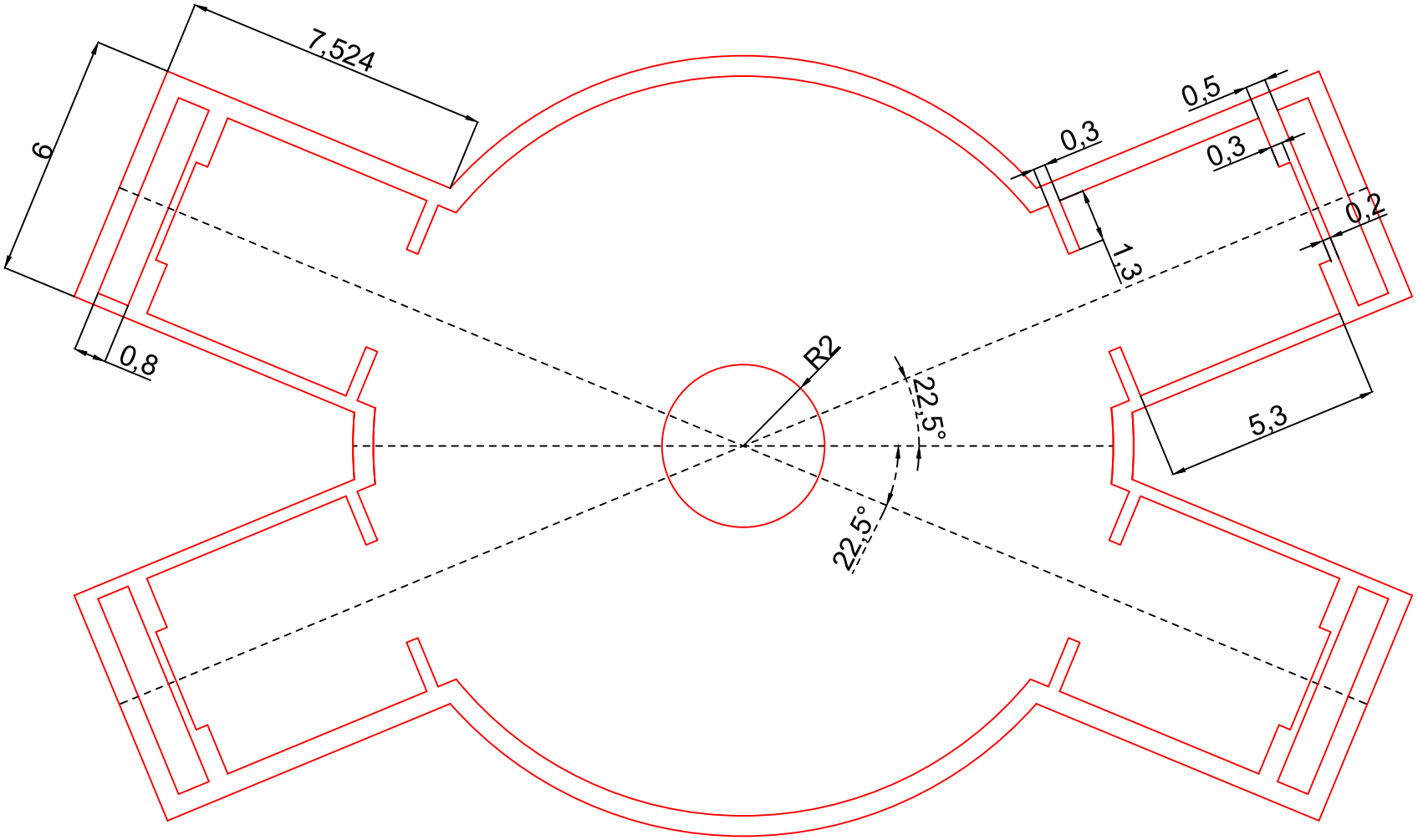
NOMBRE	ESCALA	UNIVERSIDAD DE ALCALÁ DE HERANRES
ALEJANDRO VILLALBA HERNANDO	1:150	Diseño de un sistema automatizado de aparcamiento de vehículos
TÍTULO: PLANTA PRINCIPAL		



NOMBRE	ESCALA	UNIVERSIDAD DE ALCALÁ DE HERANRES
ALEJANDRO VILLALBA HERNANDO	1:150	Diseño de un sistema automatizado de aparcamiento de vehículos
TÍTULO: PLANTA DE CONTROL		



NOMBRE	ESCALA	UNIVERSIDAD DE ALCALÁ DE HERANRES
ALEJANDRO VILLALBA HERNANDO	1:150	Diseño de un sistema automatizado de aparcamiento de vehículos
TÍTULO: PLANTA DE ALMACENAJE DE COCHES		



NOMBRE	ESCALA	UNIVERSIDAD DE ALCALÁ DE HERANRES
ALEJANDRO VILLALBA HERNANDO	1:150	Diseño de un sistema automatizado de aparcamiento de vehículos
TÍTULO: AZOTEA		

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá