**INDICE DEL DOCUMENTO**

[0. Introducción 1](#_Toc453769699)

[1. Memoria 3](#_Toc453769700)

[1.1 Origen del proyecto y motivaciones 4](#_Toc453769701)

[1.2 Justificación de la solución tomada 5](#_Toc453769702)

[1.3 Resumen de los componentes y comportamiento esperado 6](#_Toc453769703)

[1.4 Planificación y programación del proyecto 7](#_Toc453769704)

[2. Planos 10](#_Toc453769705)

[2.1 Planos de maqueta 11](#_Toc453769706)

[2.2 Plano de conexionado 12](#_Toc453769707)

[3. Pliego de condiciones 13](#_Toc453769708)

[3.1 Descripción de la maqueta 14](#_Toc453769709)

[3.2 Componentes 18](#_Toc453769710)

[3.2.1 Sensor ultrasonido HC-SR04 18](#_Toc453769711)

[3.2.2 Sensor de Infrarrojos TCRT 5000/5000L 21](#_Toc453769712)

[3.2.3 Placa Arduino Mega 2560 23](#_Toc453769713)

[3.2.4 Modulo RFID MFRC 522 27](#_Toc453769714)

[3.2.4 Modulo Relé 29](#_Toc453769715)

[3.2.6 Resto de componentes 30](#_Toc453769716)

[3.3 Como se conectaran los componentes 34](#_Toc453769717)

[3.3.1 Tabla de conexionado 34](#_Toc453769718)

[3.3.2 Conexión entre la placa Arduino y los componentes 35](#_Toc453769719)

[3.4 De que se encargará la placa Arduino Mega 2560 41](#_Toc453769720)

[3.4.1 Descripción general sus funciones 41](#_Toc453769721)

[3.4.2 Explicación del código que ejecutara Arduino 42](#_Toc453769722)

[3.5 Interfaz para dispositivos sobremesa 69](#_Toc453769723)

[3.5.1 Descripción y objetivo de la interfaz 69](#_Toc453769724)

[3.5.2 Interfaz y su funcionamiento 70](#_Toc453769725)

[3.5.3 Modo de comunicación entre Arduino y el PC 73](#_Toc453769726)

[4. Presupuesto 74](#_Toc453769727)

[4.1 Material para la maqueta 75](#_Toc453769728)

[4.2 Placa Arduino, sensores y módulos 76](#_Toc453769729)

[4.3 Materiales complementarios para la maqueta 76](#_Toc453769730)

[4.4 Coste total de todas las partes 77](#_Toc453769731)

[5. Conclusiones 78](#_Toc453769732)

[5.1 Problemas e inconvenientes encontrados 79](#_Toc453769733)

[5.2 Alternativas, mejoras y otras cuestiones 81](#_Toc453769734)

[6. Índice de imágenes 84](#_Toc453769735)

[7. Glosario de términos 87](#_Toc453769736)

[8. Referencias 90](#_Toc453769737)

[9. Anexos 92](#_Toc453769738)

[9.I Codigo Arduino 93](#_Toc453769739)

[9.II Código de la Interfaz 123](#_Toc453769740)

# Introducción

El proyecto recogido por este documento nació del interés por el mundo de la programación por parte del autor del documento, Miguel Ángel Sastre Gálvez, que durante los 2 años cursados en el I.E.S Felipe Solís Villechenous (Cabra) ha ido conociendo de la mano de Leonardo Cano Ocaña, tutor de este proyecto.

Por tanto, este proyecto surge de la búsqueda de ampliar y expandir el conocimiento del autor en el mundo de la programación, usando un lenguaje básico como C y C++, para llevar a cabo este proyecto. Para ello, se acompaña por un descubrimiento del mundo de Arduino, con el que pueden hacerse una gran variedad de proyectos, tanto domésticos como algo más profesionales.

Por supuesto, todo esto se acompaña junto al resto de conocimiento adoptados a lo largo de los años cursados en el centro para la elaboración completa del proyecto, donde se incluyen conocimientos para la elaboración de proyectos técnicos de telecomunicaciones, la elaboración de planos con AutoCAD, el conocimiento desde los componentes más básicos de la electrónica hasta los más complejos y como estos interactúan con otros componentes.

En resumidas cuentas, el proyecto desde el punto de vista del autor, cubre sobradamente muchos de los aspectos de los cursados en el grado dando un proyecto variado, completo y a la altura.

# Memoria

## 1.1 Origen del proyecto y motivaciones

La idea original del proyecto nació como una aplicación que fuera capaz de controlar zonas libres para estacionar el coche en una ciudad. La aplicación hubiera sido para móviles bajo el SO de Android y estaría conectada a cada zona de estacionamiento con la tecnología instalada, estas zonas tendrían sensores y enviarían los datos a la aplicación.

Pero esta idea, era complicada de llevar debido a varios factores.

* Se hubiera necesitado conocimiento en lenguaje Java y Android.
* Se hubiera necesitado un estudio exhaustivo de las posibilidades a nivel de calle para la colocación de los sensores sin que estos pudiera sufrir daño.
* Se hubiera necesitado un estudio para conocer el recibimiento a nivel social.
* El entorno para su demostración no era factible a niveles prácticos.

Los 2 últimos puntos eran unos grandes inconvenientes a tener en cuenta, ya que si no se pueden obtener datos fiables, el proyecto se hubiera visto arruinado antes de siquiera iniciarse, lo que me llevo a la solución tomada que se recoge en este documento. A nivel social, se debería comprobar que este sistema no provocara incidentes entre los conductores.

## 1.2 Justificación de la solución tomada

Desarrollar el proyecto en un parking surgió de la necesidad de redimensionarlo, llevándolo a un entorno cerrado y controlado, donde se podría obtener mejores datos sobre la experiencia de los usuarios. Por ello, el otro proyecto quedo a un lado, aunque gracias a él, nació la idea para este.

En casi todos los aspectos es igual, cambiando obviamente, el lenguaje de programación usado. Por el resto, se puede seguir recabando información de la satisfacción de los usuarios y del uso, y por consiguiente, del buen resultado o no, del sistema planteado en el proyecto.

El proyecto entonces recogido en este documento, ha recibido el nombre de "Sistema de control de parking automatizado por software" (SCPAS, como se mencionara de ahora en adelante).

La solución tomada para la realización de este proyecto ha sido influenciada para probar si era posible controlar un parking de manera económica y sin usar sensores y dispositivos de alto valor monetario, así, reduciendo el coste de construcción del sistema, mantenimiento y supervisión de él. Lo que nos lleva a seleccionar unos componentes de bajo valor económico pero con unas prestaciones a la altura para lo que necesitamos.

## 1.3 Resumen de los componentes y comportamiento esperado

En las siguientes líneas se listara brevemente los componentes que conforma el proyecto y como se espera que actúen.

* Placa Arduino Mega 2560, se trata de la pieza central del sistema, será el encargado de ejecutar el código y manejar todos los datos y sensores conectados a él.
* Sensor de tipo ultrasonido modelo HC-SR04, uno de los sensores que controlará cada plaza del parking
* Sensor de tipo infrarrojo modelo TCRT 5000/5000L, el otro encargado de controlar las plazas, gracias a este sensor se asegura que la plaza está ocupada por un vehículo, reduciendo la probabilidades fallo.
* Módulo MFRI 522, se trata de un módulo de identificación de radiofrecuencia (RFID), utilizado para el control de acceso al parking, gracias a él, sabremos que usuarios entra y cuando se van.
* Sensor temperatura, modelo DHT-11, se trata de sensor de temperatura, controlara la temperatura del parking, aparte de ofrecernos datos sobre la temperatura, se usaran estos datos para la activación de la ventilación para la estabilidad del sistema o bien, para activar la alarma de incendios.
* Pantalla LCD 16x2, comunicara las plazas libres, la temperatura y dará la bienvenida a los usuarios o bien, les avisara de su permiso de acceso.
* Módulo Relé, permitirá la conmutación del sistema de ventilación e iluminación.
* Resistencias fijas y potenciómetro, de capacidad variable, se usaran para la protección y buen funcionamiento de los componentes usados.
* LED's, de capacidad variable, de usos variados, los de mayor potencia serán usados para la iluminación, mientras que el resto, indicara la ocupación de las plazas y el acceso al parking.
* El cableado usado para el conexionado se tratará de cable de red, tipo CAT 5.

En el pliego de condiciones se tratará de explicar individualmente cada componente, desde su comportamiento teórico suministrado por el fabricante hasta el práctico recogido de las pruebas.

## 1.4 Planificación y programación del proyecto

El proyecto estará dividido en 5 grandes fases:

1. Elaboración de este documento.

2. Código Arduino.

3. Construcción de la maqueta y montaje.

4. Creación de la interfaz.

5. Revisión y pruebas.

La fecha de iniciación se efectuara 14 de Marzo y teniendo como fecha de finalización 10 de Junio.

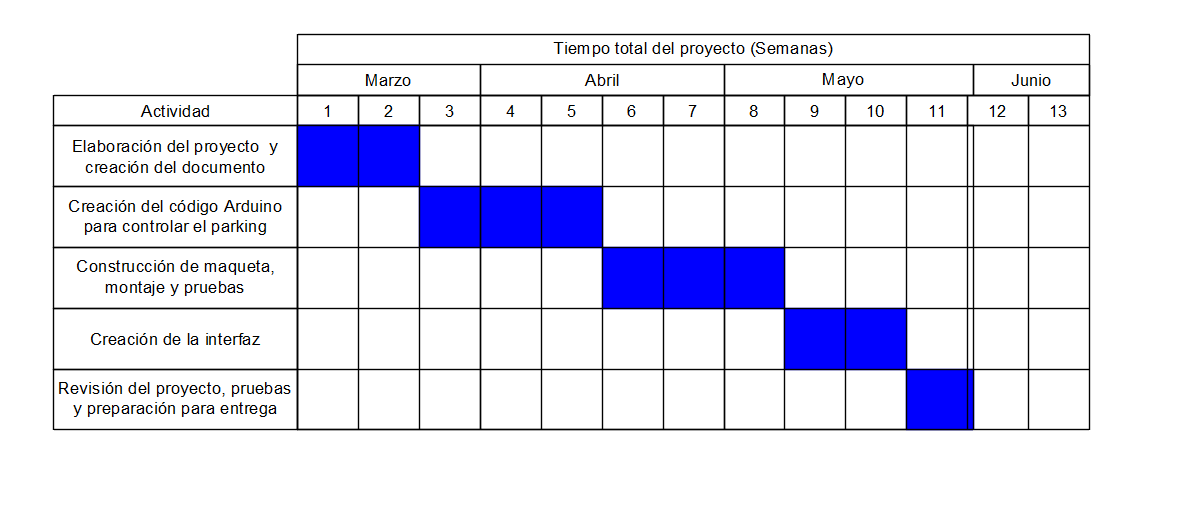
Entre esas fechas, se hará realidad el proyecto recogido por este documento. Dando 13 semanas en total para terminar el proyecto.

Figura 0. Diagrama de Gantt

Se procederá ahora a explicar cada fase individualmente.

**1. Elaboración de este documento**

La primera fase es aquella en la que el proyectista realiza el documento del proyecto, donde se explica en que consiste el proyecto, quien lo recibirá, la finalidad del proyecto, las fases por cuales pasara, la explicación de los componentes, los que se espera de ellos y como deben de ser, como deben de ser montados, etc. En resumen, se trata del documento que recoge como se inició el proyecto y como debe acabar.

El proyectista ocupara entre 1 y 2 semanas para la elaboración completa del documento, aunque este no estará finalizado por completo, hasta las comprobaciones y pruebas pertinentes.

**2. Código Arduino y su conexión futura con la interfaz.**

La segunda fase, se elaborara el código que la placa Arduino ejecutara en su memoria de manera permanente, para ello, la placa Arduino se programa con el lenguaje de programación C. Las características de la placa Arduino serán explicadas detalladamente en el pliego de condiciones.

La elaboración del código ocupará entre 2 y 3 semanas de trabajo.

**3. Construcción de la maqueta y montaje**

La tercera fase se llevara a cabo la construcción, para el montaje de los componentes y poder probar el funcionamiento y hacer pruebas.

La elaboración de la maqueta y montaje ocupara entre 2 y 3 semanas

**4. Creación de la interfaz**

La cuarta fase se llevara a cabo la creación de la interfaz de usuario que conectara Arduino con equipo informático. Creada con el IDE “C++ Builder”, usa lenguaje de programación C++.

La elaboración de la interfaz se estima que dure 2 semanas

**5. Pruebas, revisión y entrega**

La última fase, es de vital importancia, ya que entregar un producto o proyecto a un cliente sin revisar si funciona dentro de los parámetros esperados o si su calidad no es la deseada por el cliente, puede arruinar por completo todo el trabajo realizado. Por ello, antes de la entrega, se realizaran pruebas del funcionamiento de todo el conjunto, se volverá a revisar y preparara para su presentación.

Las pruebas y revisión antes de su entrega ocuparan 1 semana.

# Planos

## 2.1 Planos de maqueta

## 2.2 Plano de conexionado

# Pliego de condiciones

## 

## 3.1 Descripción de la maqueta

La maqueta consiste en la simulación de un parking real a escala, necesitando simular solo una pequeña cantidad de plazas, en este caso 3 plazas. El parking posee una entrada con una barrera, que controlara el acceso y además, contará con una pantalla LCD 16x2 que arrojara datos al usuario que desee entrar, entre los datos que se muestran en ella se encuentra:

* Temperatura y humedad del parking
* Plazas libres y ocupadas
* Acceso permitido o negación del acceso al parking (controlado por el sensor RFID)

Los datos que se obtendrán para comprobar la efectividad de la maqueta y del servicio que se propone complementar son los siguientes:

* Cuando un usuario accede al parking y cuando lo deja
* Que usuario accede al parking y quien lo deja ( Nombre, Apellido, datos bancarios)
* Tiempo de ocupación de las plazas
* El estado de las plazas
* Temperatura del parking
* Humedad del parking
* Sistema de seguridad en caso de incendios
* Sistema de seguridad de control de parking

El material usado para la construcción de la madera es mayormente madera. El suelo estará construido con un tablón de aglomerado con una dimensión de 60 x 40 x 1 cm, el techo de la maqueta y la superficie en la que se apoyara toda la circuitería también será un tablón de aglomerado con una dimensión de 40 x 40 x 1 cm.

Figura 1. Ejemplo de tablon de aglomerado

Las paredes estarán construidas con 3 tableros de contrachapado con unas dimensiones de 40 x 30 x 0,5 cm.

Figura 2. Ejemplo de tablón de contrachapado



Como se mencionaba en el primer párrafo, no toda la maqueta estará construida con madera, se usara por tanto en el techo superior de la maqueta y el lateral derecho unas láminas de metacrilato para poder visualizar la circuitería y el interior, respectivamente, de la maqueta, estas laminas tienen una dimensiones de 40 x 40 x 0,4 cm y 40 x 30 x 0,4 cm respectivamente.



Figura 3. Ejemplo de lámina de metacrilato

En el exterior de la maqueta ira situada la barrera, que controlara el acceso al parking. Esta barrera estará formada por un servomotor, led’s de indicación, un zumbador y un módulo RFID. Para la construcción del bloque de la barrera y el bloque de la pantalla LCD, se usará madera contrachapada de un grosor de 0,5 cm.

Las dimensiones del bloque de la barrera son: 6,8 x 6 x 3,8 cm

Las dimensiones del bloque de la pantalla LCD son: 8 x 4,5 x 2 cm

En el interior del parking, podremos encontrar 3 columnas, a modo de soportes para él entre-techo. En estas columnas irán situados los sensores IR TCRT5000/5000L.

En el entretecho encontraremos toda la circuitería del parking, donde se ubicara los siguientes componentes.

* + Placa Arduino
  + Sensores ultrasonido
  + Relé
  + Fichas de contacto de 6 mm
  + Potenciómetro
  + Resistencias
  + Batería

## 3.2 Componentes

### 3.2.1 Sensor ultrasonido HC-SR04

El módulo HC – SR04 se trata de un transmisor y receptor de ondas ultrasónicas, con un rango de entre 2 cm y 400 cm.

El funcionamiento básico consiste en:

1. Usando un disparador, lanzará una señal de 10us de nivel alto
2. El modulo automáticamente lanzara 8 ondas de 40 KHz y detectara cualquier onda de retorno.
3. La onda de retorno será detectada y con ella, se calculara el tiempo desde que salió hasta que se volvió a detectar. La fórmula usada para la obtención de la distancia será:



Figura 4. Módulo HC-SR04

**Parámetros eléctricos**

|  |  |
| --- | --- |
| **Tensión de trabajo** | 5 V en CC |
| **Corriente de trabajo** | 15 mA |
| **Frecuencia de trabajo** | 40 Hz |
| **Máximo Rango** | 4 m |
| **Mínimo Rango** | 2 cm |
| **Angulo de medida** | 15º |
| **Señal de salida del disparador** | 10uS TTL pulso |
| **Señal eco de entrada** | Proporcional a la señal |
| **Dimensiones** | 45 x 20 x 15 mm |

**Figura 5. Tabla de parámetros HC-SR04**



Figura 6. Rango y dimensiones de HC-SR04

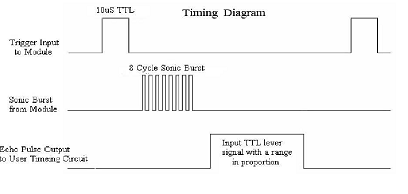
**Diagrama de tiempo**

Figura 7. Diagrama de tiempo

Lo que se observa en el diagrama de tiempo es como funciona exactamente el módulo HC-SR04

Al recibir un pulso de 10uS mediante la tecnología TTL, el modulo iniciara una secuencia de disparo de 8 ciclos de una señal de 40 KHz. Estos pulsos se tratan de una señal inaudible para el ser humano, que según avanzara se ira degradando. Al rebotar con una superficie y volver al módulo, esta señal es recibida y leída. La señal eco, tendrá un nivel dependiendo de la distancia que haya sido capaz de recorrer, y gracias a esto, es como podemos obtener la distancia con la formula ya mencionada.

**Ubicación en la maqueta**

Habrá 3 módulos HC-SR04 en la maqueta que cada uno de ellos ira encima de cada plaza del parking. Estos detectara si hay un cambio en la distancia que normalmente detecta y con ello, se podrá saber si hay algo debajo de ellos.

### 3.2.2 Sensor de Infrarrojos TCRT 5000/5000L

El sensor TCRT5000 y 5000L, se trata de un módulo compuesto por un emisor de infrarrojos y un fototransistor.

El funcionamiento básico consiste en:

1. El emisor de infrarrojos emite una señal en el rango de 950 nm, por lo tanto se trata de una señal invisible al ojo humano.

2. El fototransistor capta la señal provocando una tensión contraria a la de alimentación, la tensión contraria variara dependiendo de la cantidad de luz recibida del emisor lo que nos indicara la distancia.

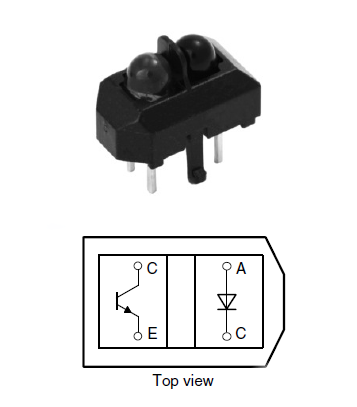


Figura 8. Sensor TCRT 5000/5000L

**Parámetros eléctricos**

|  |  |
| --- | --- |
| **Emisor** |  |
| **Tensión de trabajo** | 1,25 V |
| **Intensidad de radiación** | 21 mW |
| **Longitud de onda** | 950 nm |
| **Corriente de trabajo** | 60 mA |
| **Fototransistor** |  |
| **Potencia disipada** | 200 mW |
| **Tensión de trabajo** | 5 V |
| **Corriente de trabajo** | 100 mA |
| **Rango de temperatura** | -25 a +85 ºC |

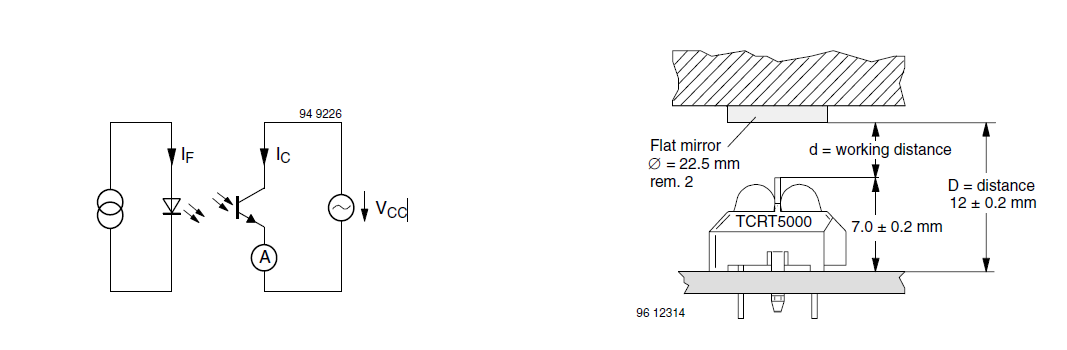
**Figura 9. Tabla de parámetros TCRT 5000/5000L**

Figura 10. Diagrama y esquema de TCRT 5000/5000L

### 3.2.3 Placa Arduino Mega 2560

Arduino Mega 2560 es una placa microcontroladora basada en ATmega2560.

Características:

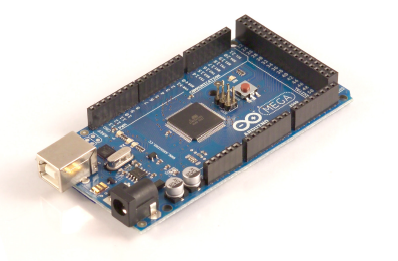
* Posee 54 pines digitales de entrada/salida ( 14 de ellos puede ser usados como salidas PWM)
* 16 pines analógicos de entrada
* 4 UART’s
* Un oscilador de cristal de cuarzo de 16 MHz
* Puerto USB
* Entrada de alimentación tipo Jack
* Botón de reinicio
* Dispositivo **ICSP**

Figura 11. Placa Arduino Mega 2560

**Parámetros**

|  |  |
| --- | --- |
| **Microcontrolador** | ATmega2560 |
| **Tensión de trabajo** | 5 V |
| **Tensión recomendada** | 7 – 12 V |
| **Tensión limite** | 6 – 20 V |
| **Digital I/O pin’s** | 54 |
| **Analógicos I pin’s** | 16 |
| **Corriente por I/O pin** | 40 mA |
| **Corriente por 3,3v pin** | 50 mA |
| **Memoria Flash** | 256 KB (8KB para bootloader) |
| **SRAM** | 8 KB |
| **EEPROM** | 4 KB |
| **Velocidad del Reloj** | 16 MHz |

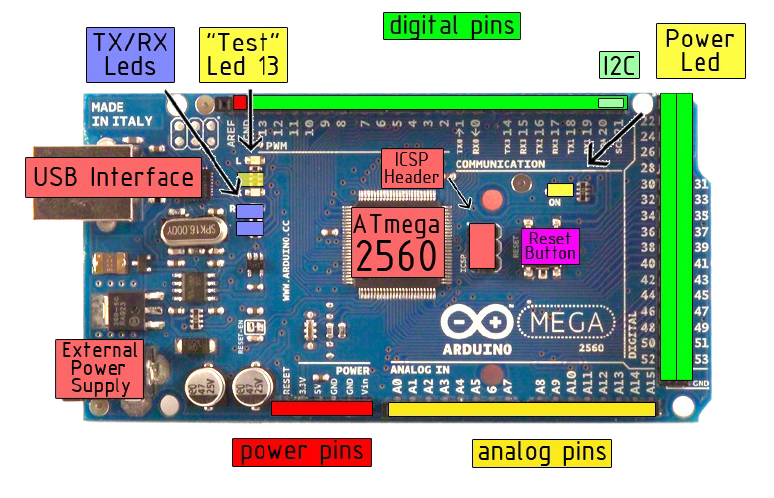
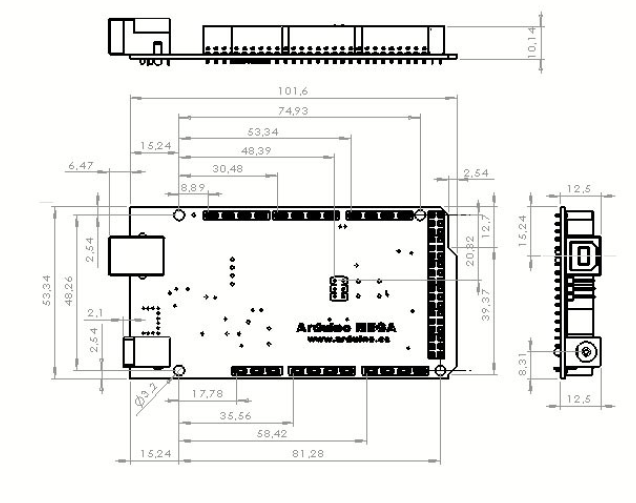
**** **Figura 12. Tabla de parámetros Arduino Mega 2560**

Figura 13. Esquema de Arduino Mega 2560

**Dimensiones de Mega 2560**



**Figura 14. Dimensiones de la placa Arduino Mega 2560 (esquema suministrado por fabricante)**

**IDE de Arduino**

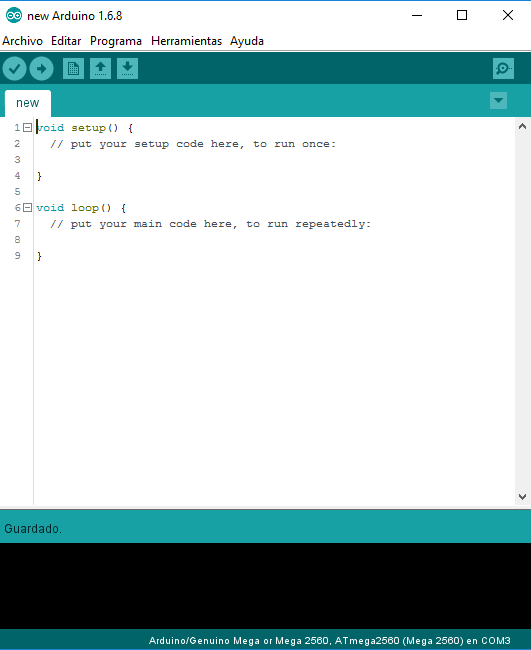
Arduino cuenta con su propia IDE para desarrollar sus programas y cargarlos a una placa Arduino. La sintaxis del lenguaje de programación Arduino es una versión simplificada de C/C++. Para este proyecto se usara esta IDE aunque es posible programar una placa Arduino con otras IDE’s.

Figura 15. IDE de Arduino

### 3.2.4 Modulo RFID MFRC 522

El módulo, modelo MFRC522, es una placa integrada con un lector y escritor IC de comunicación sin contacto de una frecuencia de 13,56 MHz. El módulo RC 522 usa un sistema avanzado de modulación y demodulación para todo tipo de dispositivos pasivos.

El funcionamiento básico consiste en:

1. El módulo RFID posee un campo electromagnético. El modulo siempre está activo a la espera de datos.
2. Al pasar un dispositivo compatible con la tecnológica RFID o NFC, el modulo reacciona y obtiene los datos del dispositivos.
3. Los datos son enviados a la placa Arduino, donde se usaran del modo deseado por el programador.



Figura 16. Modulo RC522

**Parámetros**

|  |  |
| --- | --- |
| **Tensión de trabajo** | 3,3 V |
| **Corriente de trabajo** | 13 – 26 mA |
| **Stand By** | 10 – 13 mA |
| **Corriente máxima** | 30 mA |
| **Frecuencia** | 13,56 MHz |
| **Distancia de lectura** | 0 a 60 mm |
| **Protocolo de comunicación** | SPI |
| **Velocidad de datos máxima** | 10 Mbit/s |

**Figura 17. Tabla de parámetros RC522**

**Sobre los TAG’s**

Los TAGs viene en diferentes modelos, los más comunes son en tarjetas y en llaveros, pero también vienen como etiquetas adhesivas e incluso ya viene incrustados en algunos productos.

Los Tags tienen internamente una antena y un microchip, encargado de realizar todo el proceso de comunicación, la energía lo obtiene de la señal de radiofrecuencia, que si bien la energía en la señal es pequeña, es suficiente para hacer trabajar el microchip, esto es la razón por la que es necesario acercarlos a una pequeña distancia generalmente menor a 10 cm. Pero existen Tags activos, que incorporan baterías, estos tiene alcance de varios metros de distancia.

Figura 18. Tarjeta y llavero RFID

### 3.2.4 Modulo Relé

El relé es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Es la definición básica de un relé común.

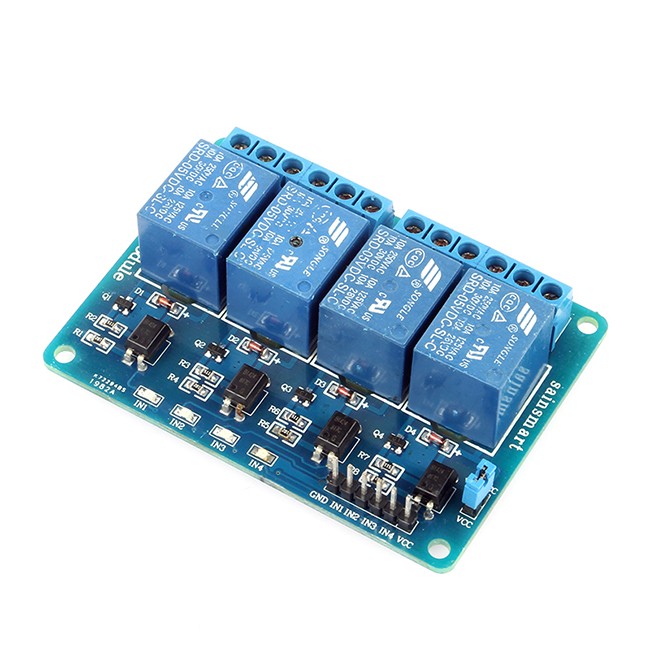
El modulo relé que usaremos tendrá 4 relé controlados por Arduino, mediante valores de tensión Altos y Bajos. Cada relé puede tener posiciones tanto NA como NC. Al recibir un valor de tensión, el relé se abrirá o cerrara.

Figura 19. Módulo x4 relés

### 3.2.6 Resto de componentes

#### Resistencias

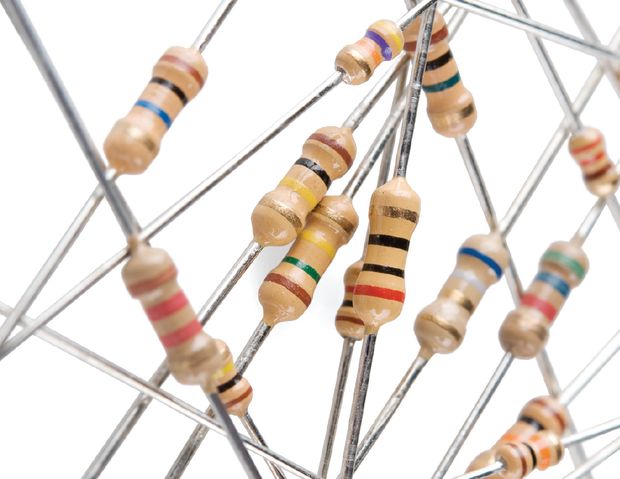
La definición más extendida sobre las resistencias es que una resistencia es un elemento pasivo que disipa energía en forma de calor según la ley de Joule.

Para este proyecto, se necesitara una determinada cantidad de resistencias fijas de un valor determinado, para que ciertos componentes como los sensores de IR o los led’s funcionen correctamente y evitando que se deterioren.

La cantidad de resistencias y su capacidad vienen listadas a continuación:

* Resistencias de 220 Ω 🡪 10
* Resistencias de 4700 Ω 🡪 1
* Resistencias de 10 kΩ 🡪 3

Además se necesita de un potenciómetro, que consiste en una resistencia que puede variar su capacidad, con una capacidad de hasta 100 kΩ





**Figuras 20 y 21. Resistencias fijas y un potenciómetro**

#### LED’S

Un led es un componente opto electrónico pasivo y, más concretamente, un diodo qué emite luz.

En este proyecto, los led’s serán usados con varios fines:

* Iluminación
* Estado de las plazas (libres u ocupadas)
* Indicador de acceso o negación de acceso
* Indicador de funcionamiento de la barrera

Para llevar a cabo cada misión, se ha decidido usar 2 tipos de led’s distintos.

* Para el caso de la iluminación de usaran led’s de color blanco de 5mm que operan con 3,6 v.
* Para los led de indicación, se usan unos led’s de 3mm y de colores verde y rojo.

**Figuras 22 y 23. LED blanco de 5mm de alta luminosidad y LED’s de colores de 3mm**

#### Servomotor

Un servomotor (también llamado servo) es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición.

El servomotor en este proyecto jugará el papel de hacer de barrera. El servomotor usado en la maqueta consta de 3 cables, siendo Positivo (Rojo), Negativo (Negro) y Control (Amarillo).

El control de un servomotor se realiza mediante PWM.



Figura 24. Servomotor

#### Batería Auxiliar y Ventilador

Para la ventilación se usara un ventilador común de 80mm de tamaño, este será alimentado con una batería de 7,8v independiente al sistema de alimentación del resto del circuito.

Figura 25. Ventilador de 80 mm



Figura 26. Batería lipo de 7,8 v

## 3.3 Como se conectaran los componentes

### 3.3.1 Tabla de conexionado

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PWN | PIN |  | DIGITAL | PIN |  | DIGITAL |
|  | 2 |  |  | 22 | Ultra: Trigger1 | PLAZA 1 |
|  | 3 |  |  | 23 | Ultra: Echo1 |
|  | 4 |  |  | 24 | Ultra: Trigger2 | PLAZA 2 |
|  | 5 | RFID - RESET |  | 25 | Ultra: Echo2 |
|  | 6 |  |  | 26 | Ultra: Trigger3 | PLAZA 3 |
|  | 7 |  |  | 27 | Ultra: Echo3 |
|  | 8 |  |  | 28 | LED Rojo | PLAZA 1 |
|  | 9 | Servo |  | 29 | LED Verde |
|  | 10 |  |  | 30 | LED Rojo | PLAZA 2 |
|  | 11 |  |  | 31 | LED Verde |
|  | 12 | Zumbador 1 - Alarma |  | 32 | LED Rojo | PLAZA 3 |
|  | 13 | Zumbador 2 - Negación del acceso |  | 33 | LED Verde |

|  |  |
| --- | --- |
| PIN |  |
| 34 | Sensor Temperatura: Temp1 |
| 35 | Sensor Temperatura: Temp2 |
| 36 | Rele1 -> Ilumnicación |
| 37 | Rele2 -> Ventilación |
| 38 | Rele3 |
| 39 | Rele4 |
| 40 |  |
| 41 |  |
| 42 | LCD - RS |
| 43 | LCD - E |
| 44 | LCD - D4 |
| 45 | LCD - D5 |

|  |  |  |  |
| --- | --- | --- | --- |
| DIGITAL | PIN |  |  |
|  | 46 | LCD - D6 |  |
|  | 47 | LCD - D7 |  |
|  | 48 | LED Acceso |  |
|  | 49 | LED Negación |  |
|  | 50 | RFID - MISO |  |
|  | 51 | RFID - MOSI |  |
|  | 52 | RFID - SCK |  |
|  | 53 | RFID - SDA |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PIN |  | ANALOG | PIN |  |
| A0 | IR1 |  | A14 |  |
| A1 | IR2 |  | A15 |  |
| A2 | IR3 |  |  |  |
| A3 |  |  |  |  |
| A4 |  |  |  |  |
| A5 |  |  |  |  |
| A6 |  |  |  |  |
| A7 |  |  |  |  |
| A8 |  |  |  |  |
| A9 |  |  |  |  |
| A10 |  |  |  |  |
| A11 |  |  |  |  |
| A12 |  |  |  |  |
| A13 |  |  |  |  |

### Conexión entre la placa Arduino y los componentes

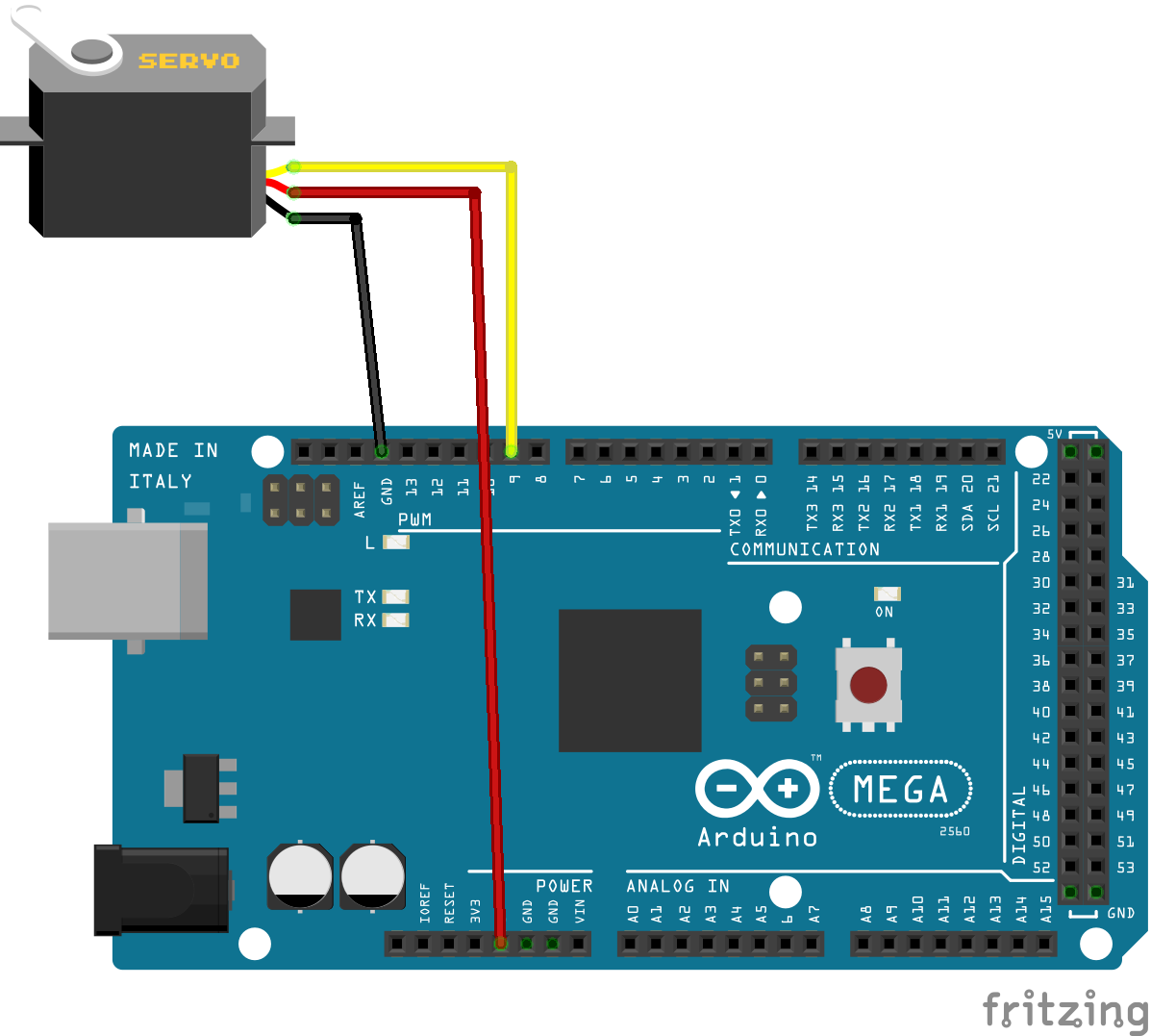


Figura 27. Conexión Placa-Servo

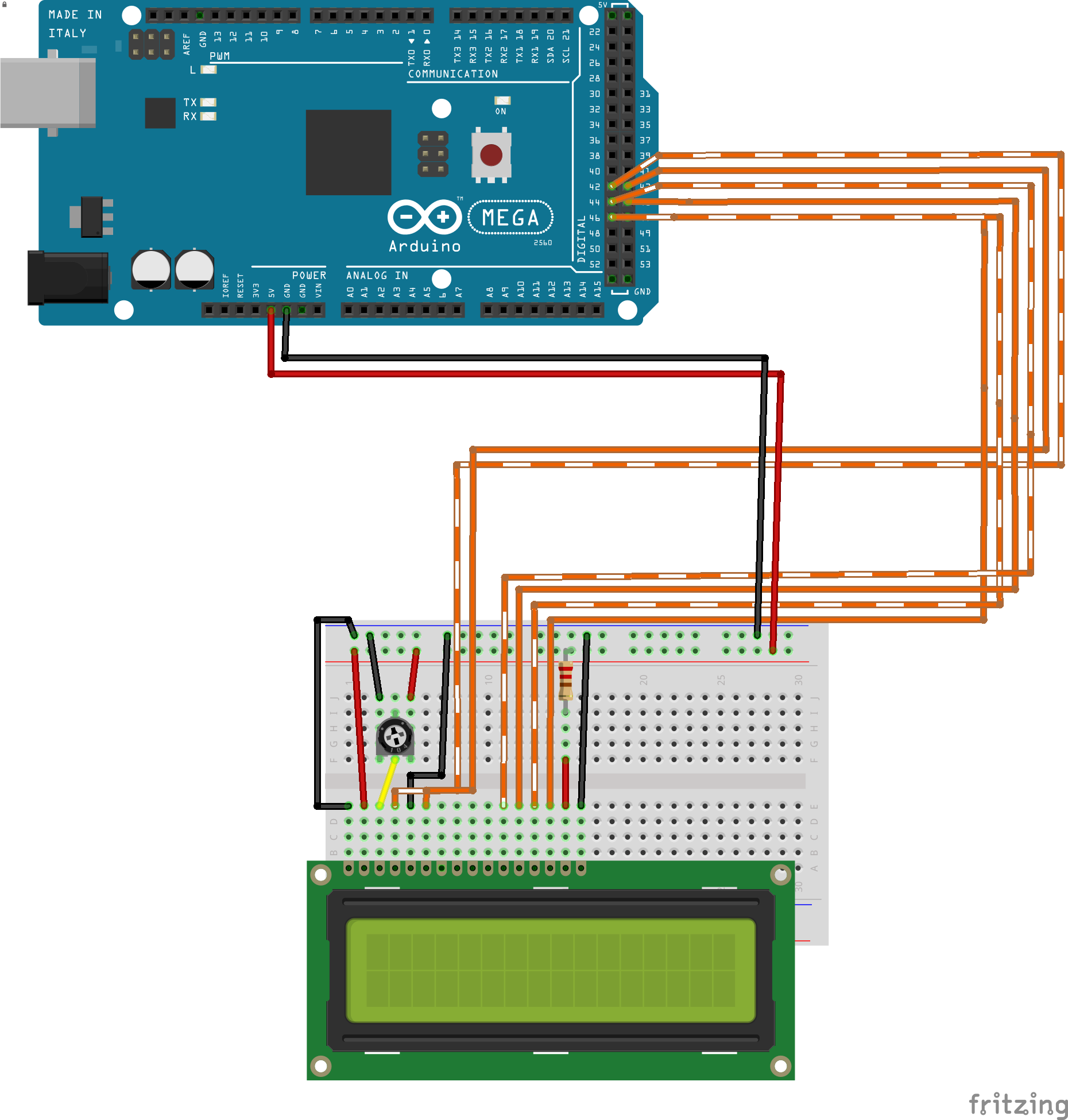


Figura 28. Conexión Placa-LCD

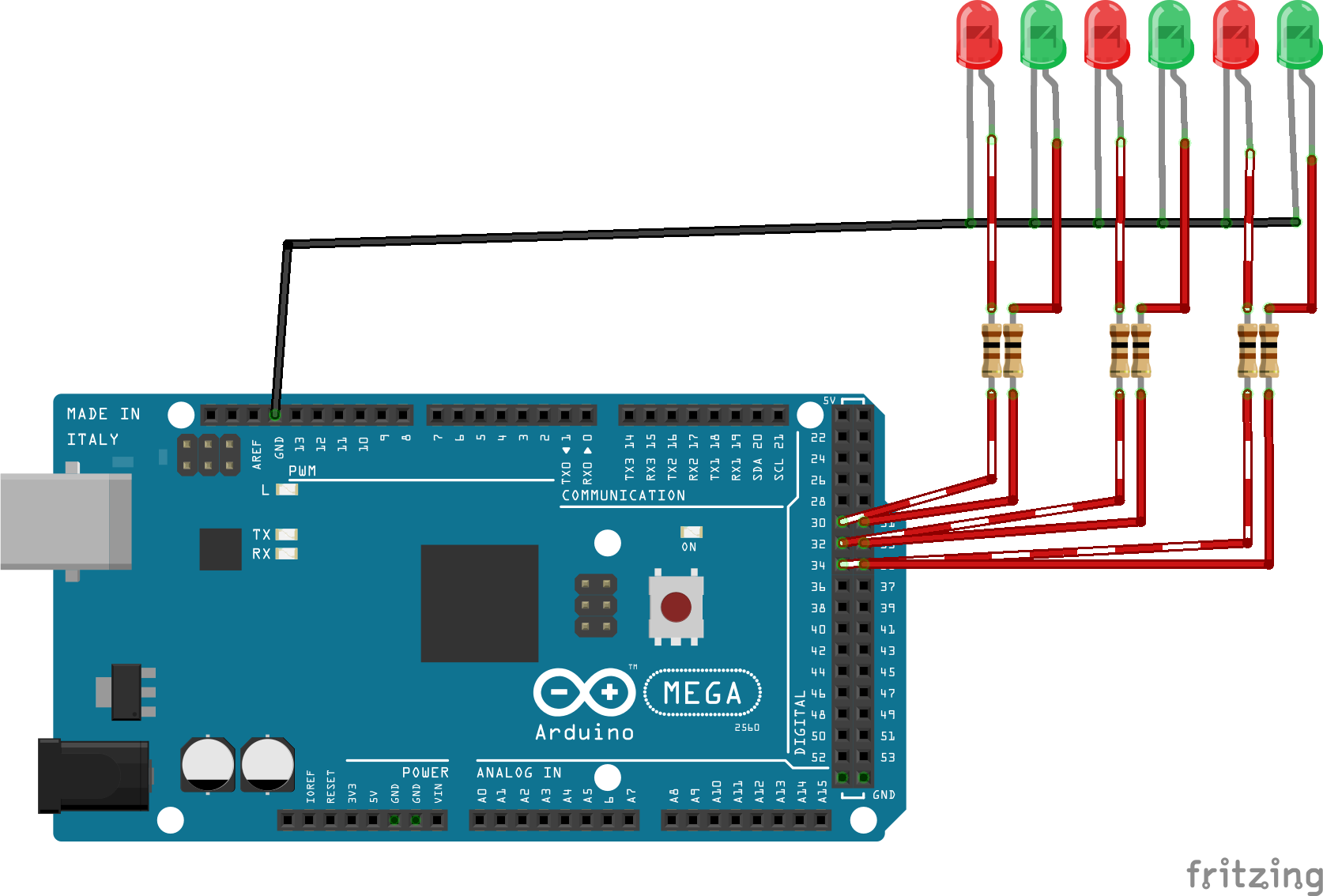


Figura 29. Conexión Placa-Led de ocupación

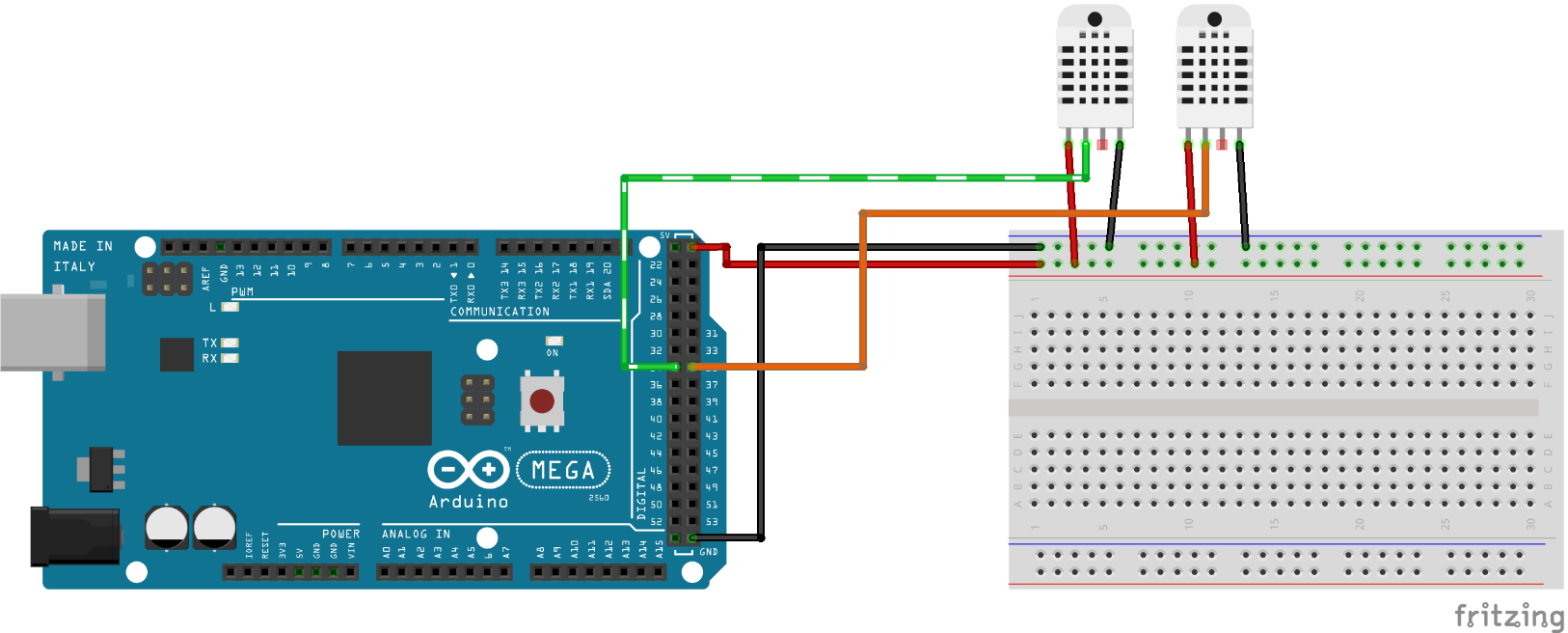


Figura 30. Conexión Placa-DHT11

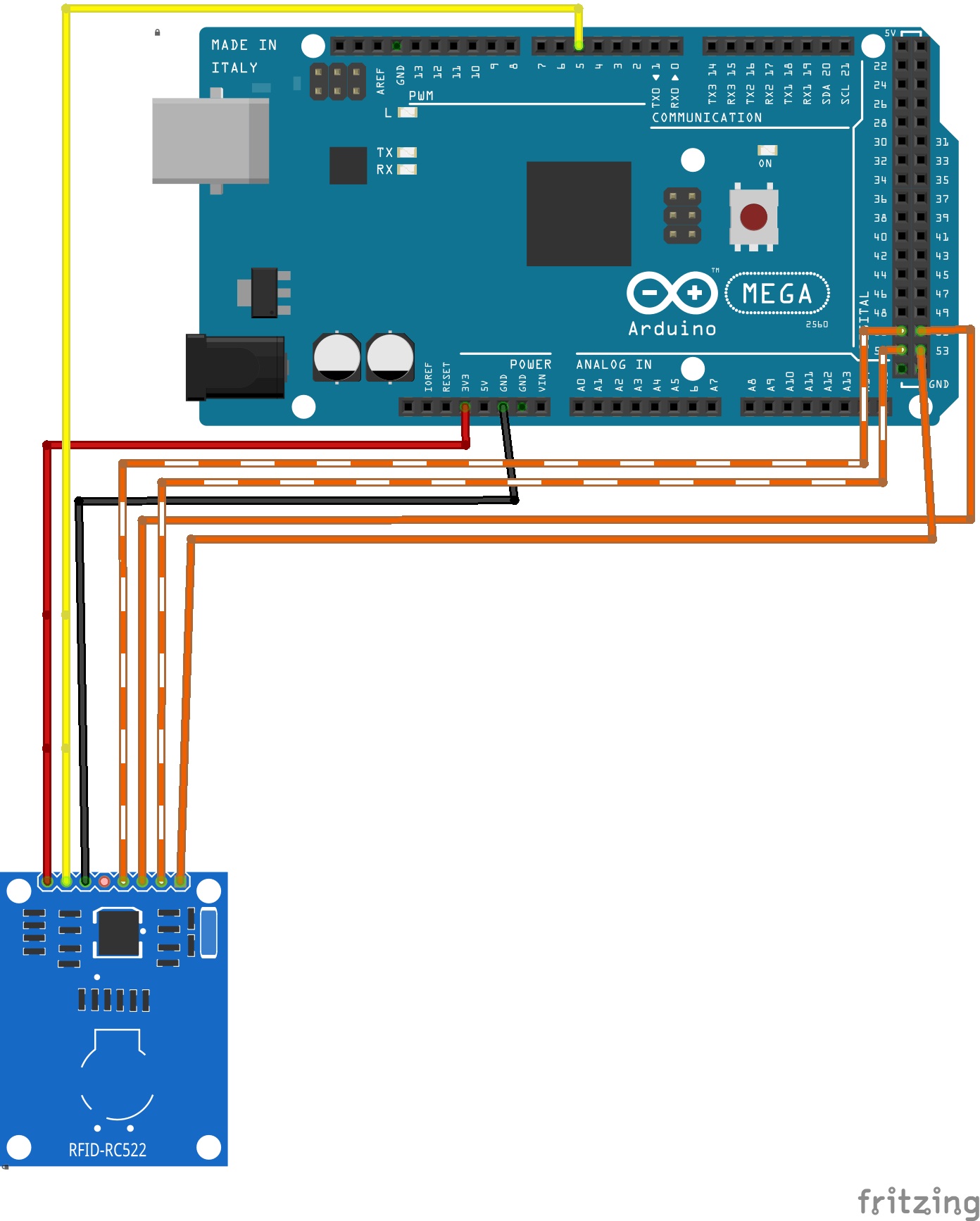


Figura 31. Conexión Placa-Módulo RFID

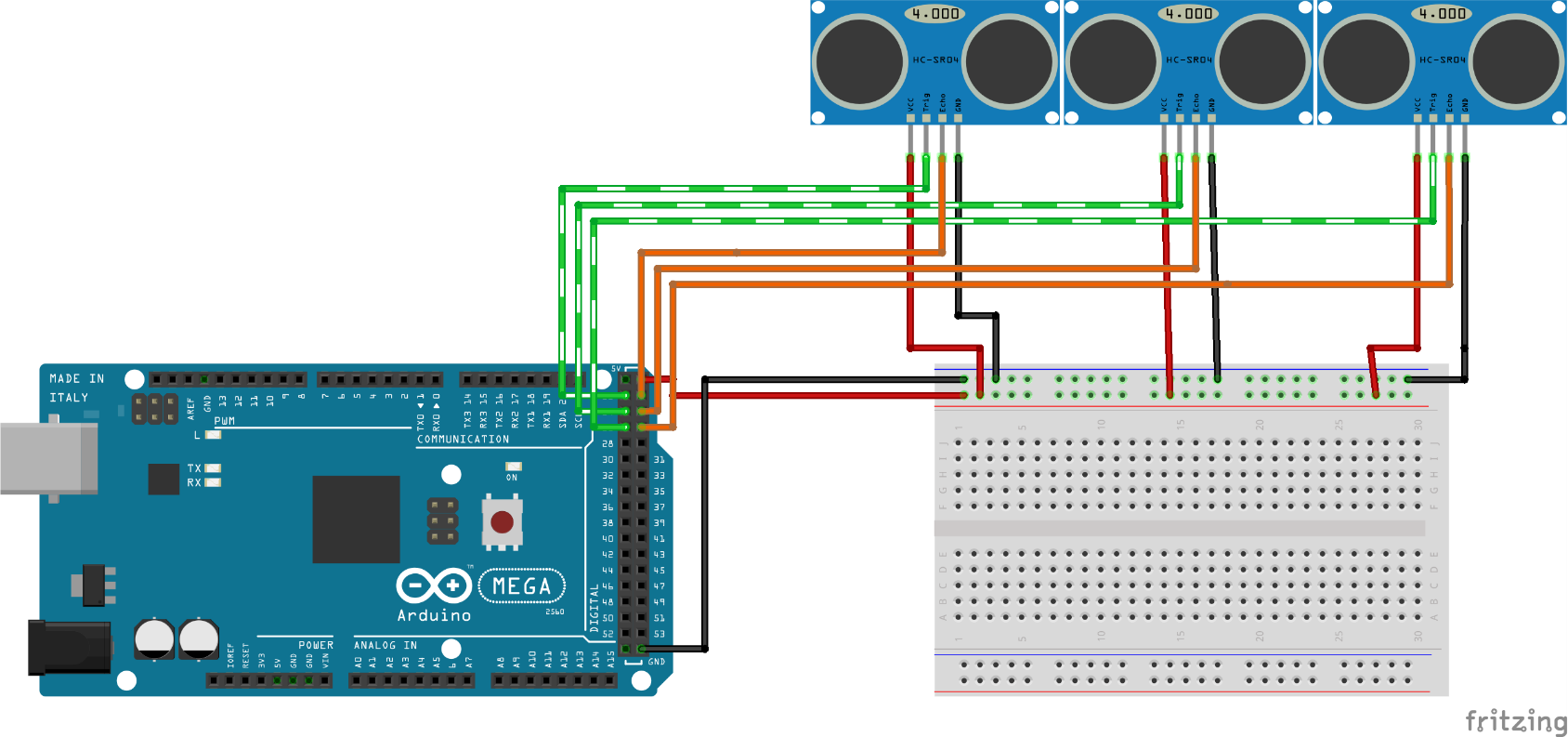
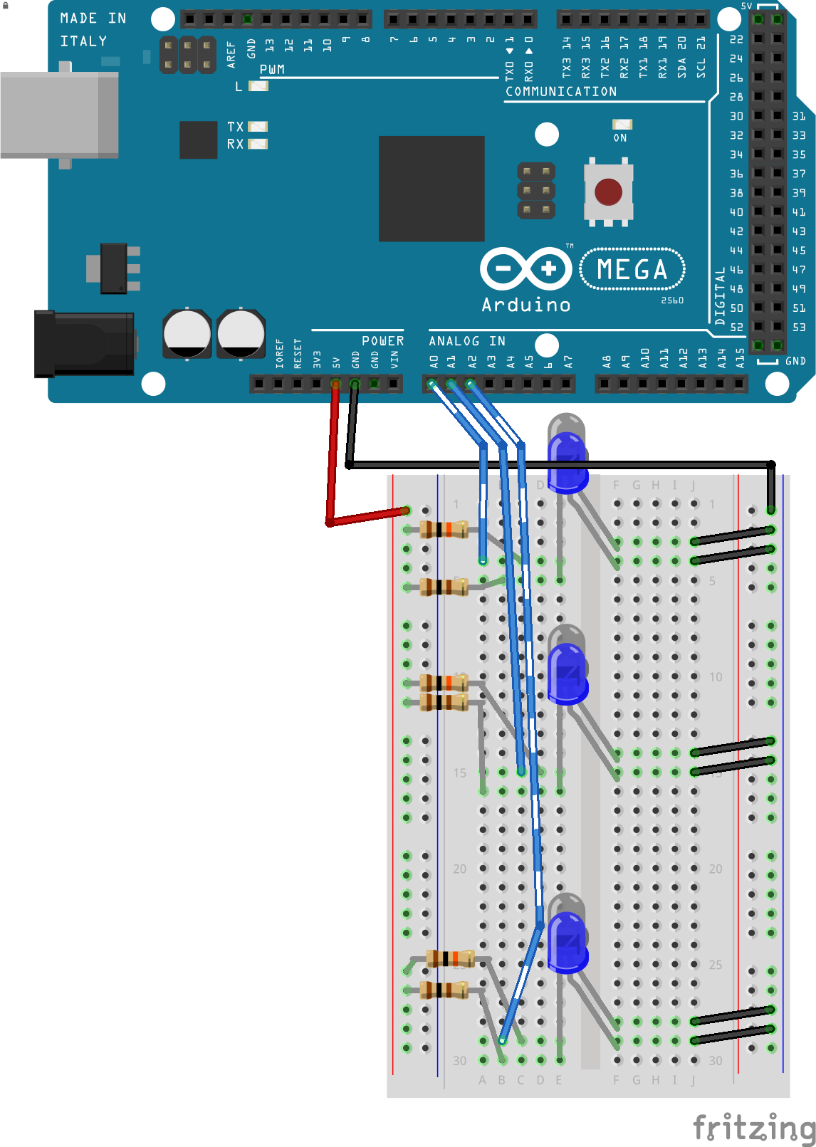


Figura 33. Conexión Placa-TCRT5000/5000L

Figura 32. Conexión Placa-HC-SR04

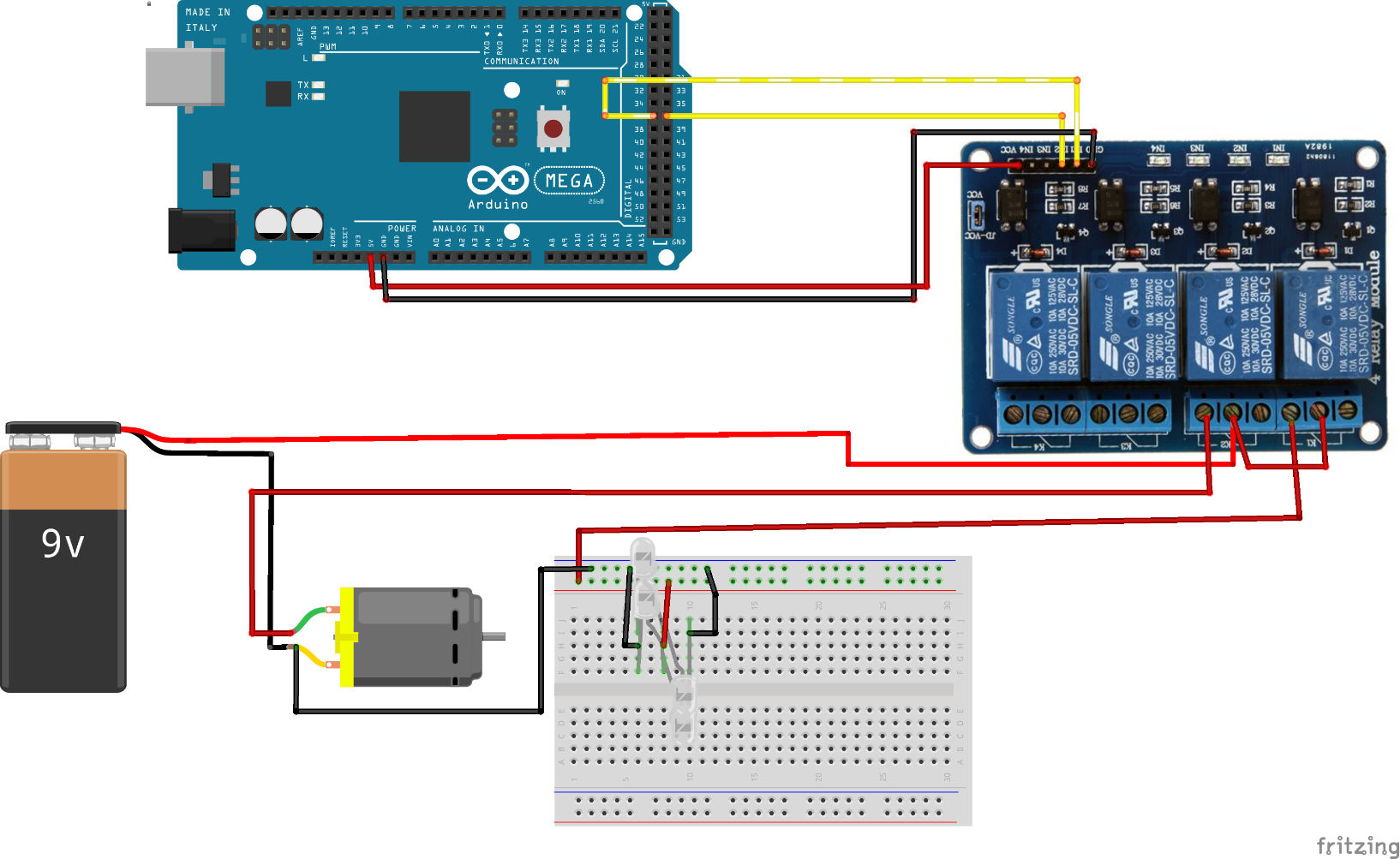
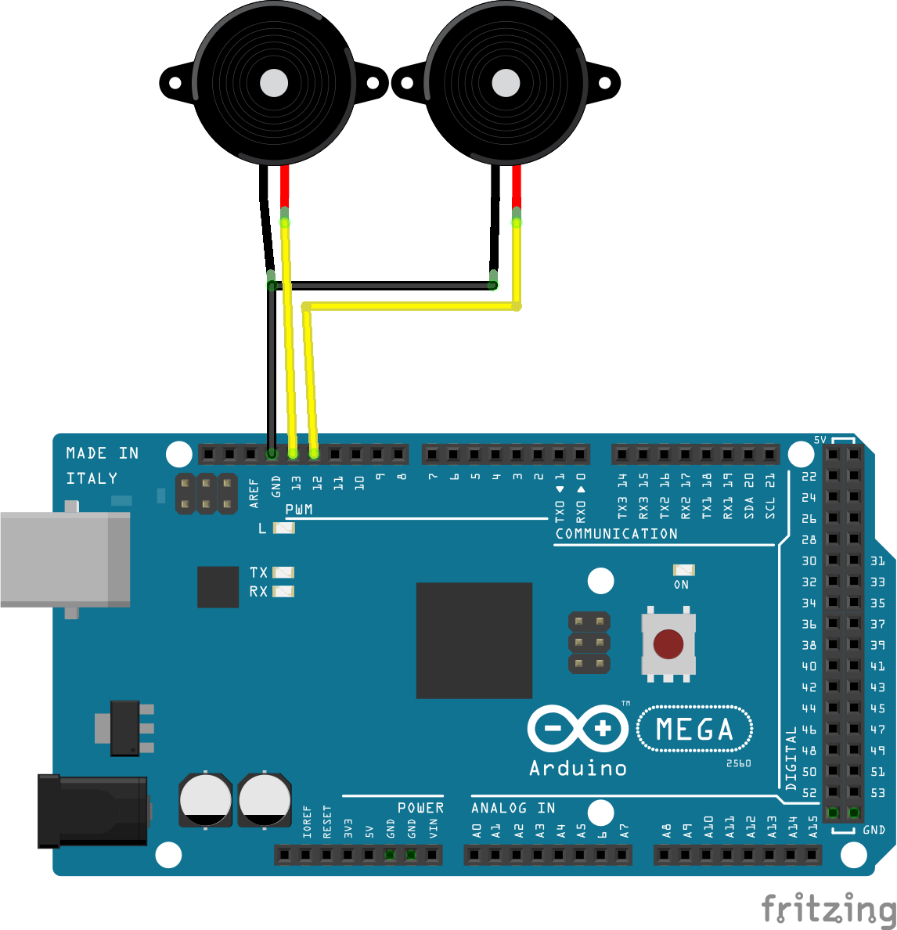


Figura 35. Conexión Placa-Zumbadores

Figura 34. Conexión Placa-Relé

## 3.4 De que se encargará la placa Arduino Mega 2560

### 3.4.1 Descripción general sus funciones

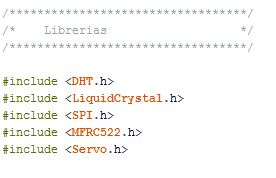
La placa Arduino Mega 2560 que será instalada en la maqueta, será el cerebro del parking. Será la encargada de manejar, controlar y dar energía a los diferentes componentes del sistema.

Sus funciones serán las siguientes:

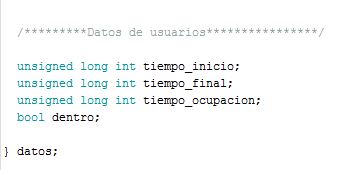
* Control de plazas, mediantes los sensores de ultrasonido (HC-SR04) e IR (TCRT 5000/5000L). Al usar 2 sensores por plazas, se disminuye la probabilidad de una mala lectura.
* Control de temperatura y humedad del parking con el sensor DHT-11. Gracias al sensor DHT-11 el código que ejecutara Arduino se podrá usar para identificar posibles incendios en caso de que la temperatura se eleve por encima de la temperatura máxima estándar de la época.
* Control del acceso al parking, mediante un servomotor, que hará de barrera y un módulo RFID que controlara esta barrera. Gracias al modo RFID, se puede tener una base de datos de usuarios que tenga permitido el acceso al parking, conociendo así, quienes han accedido al parking y durante cuanto han estado en él.
* Control de pantalla LCD. Esta pantalla indicara a los usuarios el estado del parking, sabiendo así, antes de entrar en él, si hay plazas ocupadas. También se mostrara información de la temperatura y la humedad y si se ha concedido o no, el acceso al parking.
* Control de ventilación e iluminación, que mediante un relé se conmutara dada ciertas circunstancias

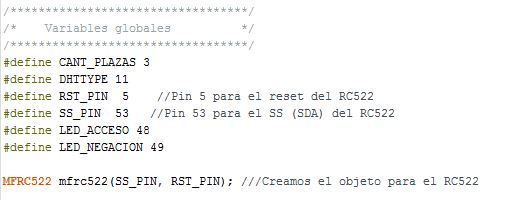
### Explicación del código que ejecutara Arduino

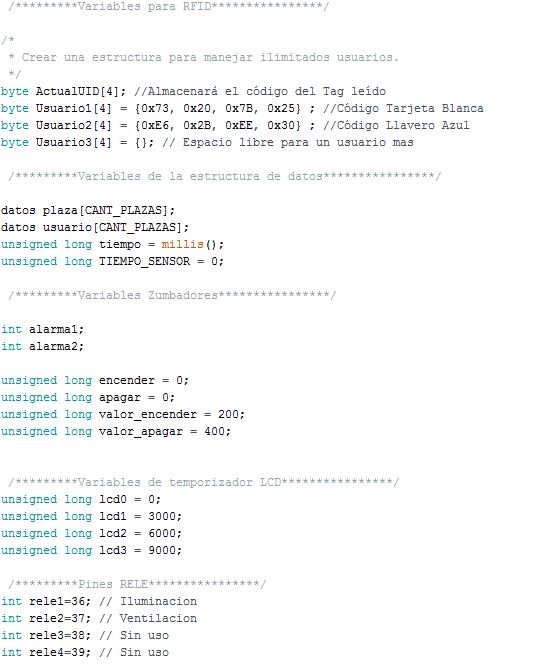
1. **Librerías y Variables globales**

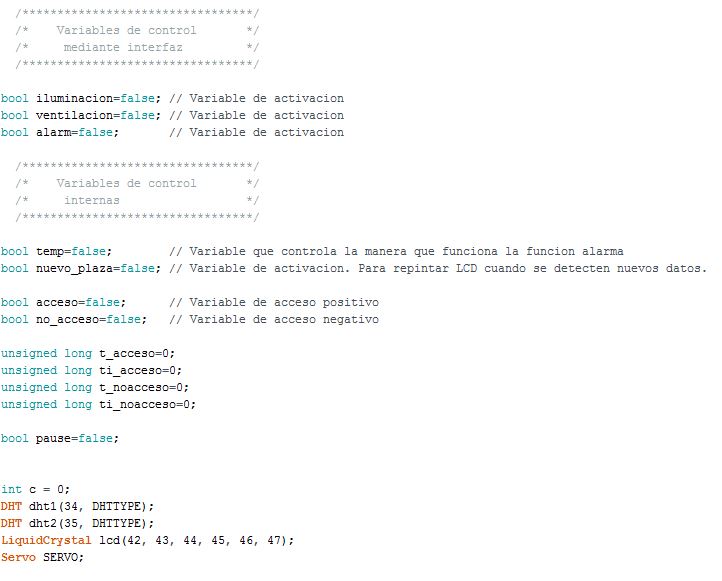












**B. Arranque de Arduino**

La función Setup(), es la primera arrancar, usada para inicializar variables, el modo de los pin, arrancar las librerías y algunas funciones más. Esta parte del código solo se ejecutara 1 vez.

Lo primero que se hará en el Setup, será iniciar la comunicación serie con la función Serial.begin() , con un valor de 9600 baudio.

La siguiente función se verá repetida de ahora en adelante, con el objetivo de mostrar información o saber dónde se encuentra en el código Arduino, esto se realiza mediante la función Serial.print(), esta información se mostrara a través de la consola del IDE.

Como se menciona al principio de este mismo apartado, el objetivo de Setup(), es inicializar y adjudicar valores a las distintas variables y pines de Arduino.

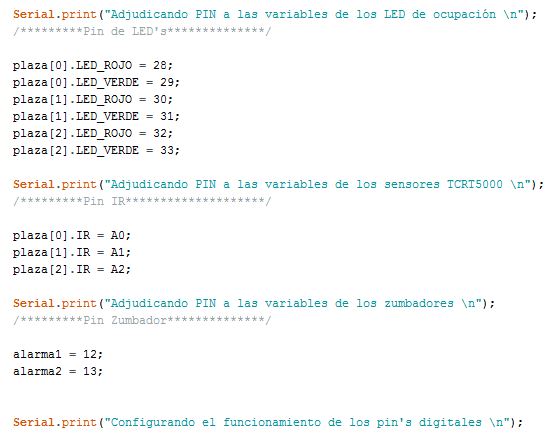
Para los pines se usaran 3 funciones distintas:

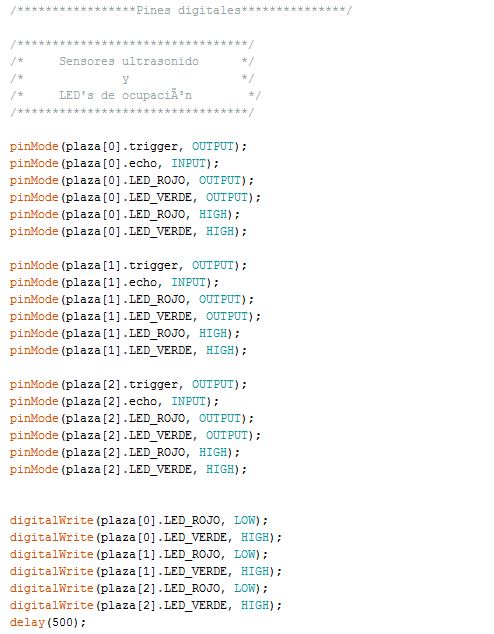
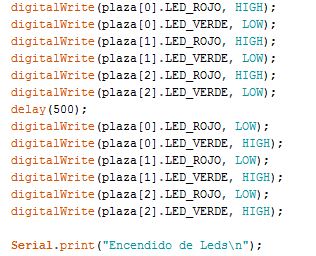
* pinMode(PIN,INPUT/OUTPUT), esta función configurara el modo en el que un determinado pin funcionara, si como salida o entrada.
* digitalWrite() y analogWrite(), estas 2 funciones , dependiendo de valor (HIGH o LOW), alimentaran los diferentes pins.

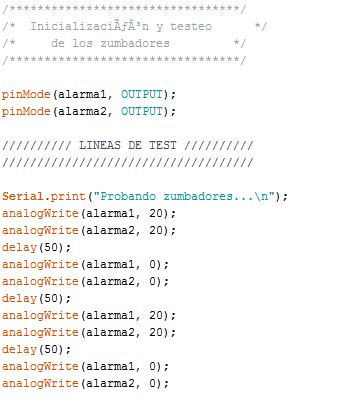
Para el resto de variables se iniciaran a mano, dando los valores de pin al que irán conectado de la placa.

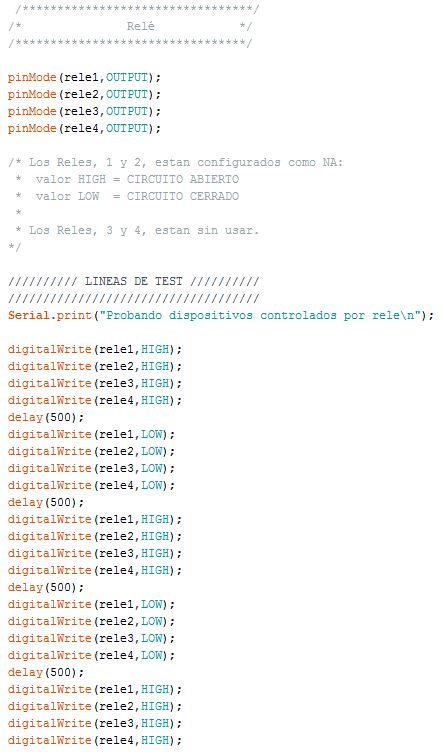
En el Main, además, se realizar varias operaciones de comprobación, para asegurar el buen funcionamiento de cada elemento del sistema.

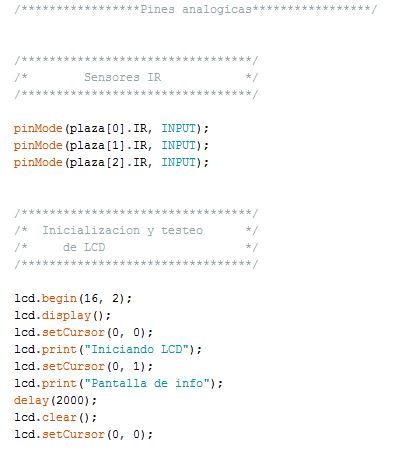


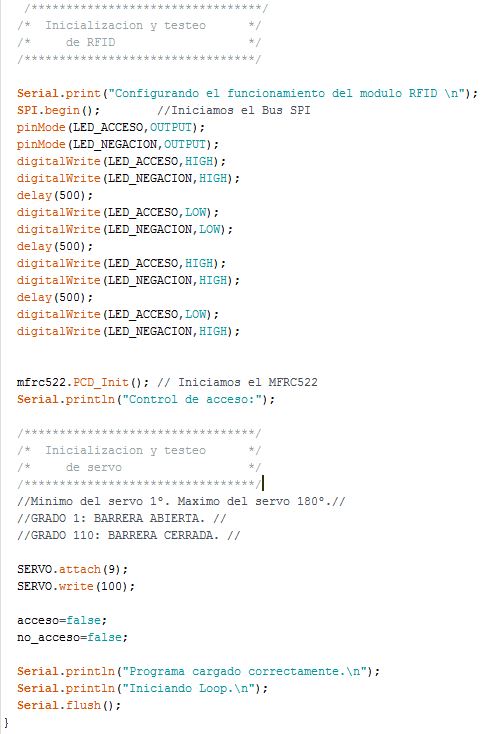








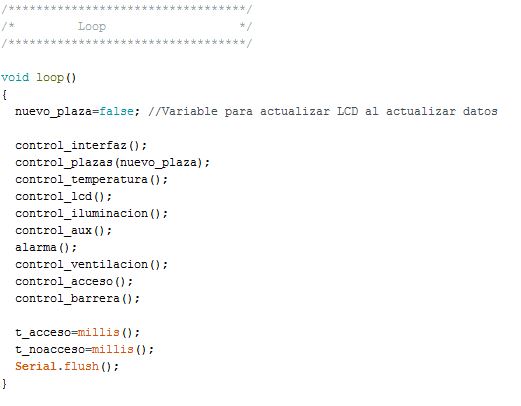




**C.Loop**

Después de ejecutarse la función Setup(), se ejecutara la función loop(), que a diferencia de la función Setup(), se ejecutara una y otra vez, de manera infinita, por tanto, aquí es donde ira todo nuestro programa.

Podemos encontrar las siguientes funciones en el loop:

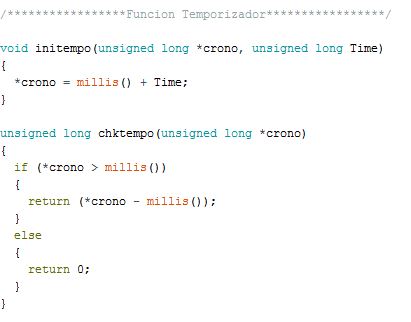
* Control\_plazas (), el código que controlara los sensores de las plazas, tanto el de ultrasonido como el IR.
* Control\_temperatura (), controlara la humedad y la temperatura de parking, y además, puede activarse la ventilación y la alarma de incendios de manera automática.
* Control\_lcd (), manejara la pantalla LCD.
* Control\_iluminación (), función para encender o apagar la iluminación.
* Control\_ventilación (), función para encender o apagar la ventilación.
* Control\_acceso (), función que maneja el modulo RFID
* Control\_barrera (), función que controlara el servomotor

**D. Funciones**

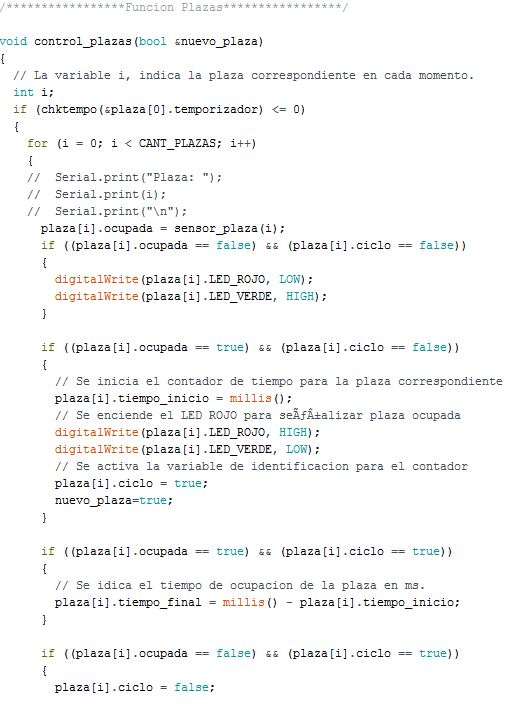
El codigo arduino contará con 14 funciones, que se listaran a continuacion:

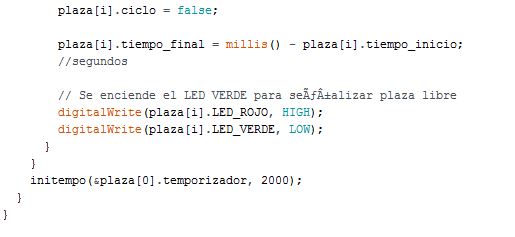
1. void initempo(unsigned long \*crono, unsigned long Time)
2. unsigned long chktempo(unsigned long \*crono)
3. void control\_plazas(bool &nuevo\_plaza)
4. bool sensor\_plaza(int i)
5. void control\_temperatura()
6. void alarma()
7. void control\_lcd()
8. void control\_iluminacion()
9. void control\_ventilacion()
10. void control\_aux()
11. void control\_acceso()
12. boolean compareArray(byte array1[], byte array2[])
13. void control\_barrera()
14. void control\_interfaz()

Las funciones void initempo() y unsigned long chktempo() van juntas, initempo() inicializara nuestros temporizadores pasado X tiempo, un tiempo determinado por nosotros. Chktempo() se encarga de comprobar la variable y ver si se cumple el límite de tiempo impuesto por nosotros.

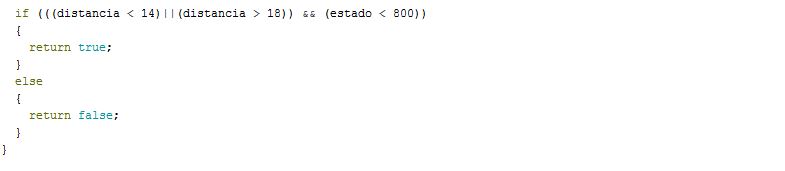


La función void control\_plazas(bool &nuevo\_plaza) será la encargada del control de las plazas. Dentro de ella nos podemos encontrar con la función bool sensor\_plaza(int i) que será la encargada de gobernar los sensores de ultrasonido HC-SR04 y los sensores de IR TCRT 5000/5000L









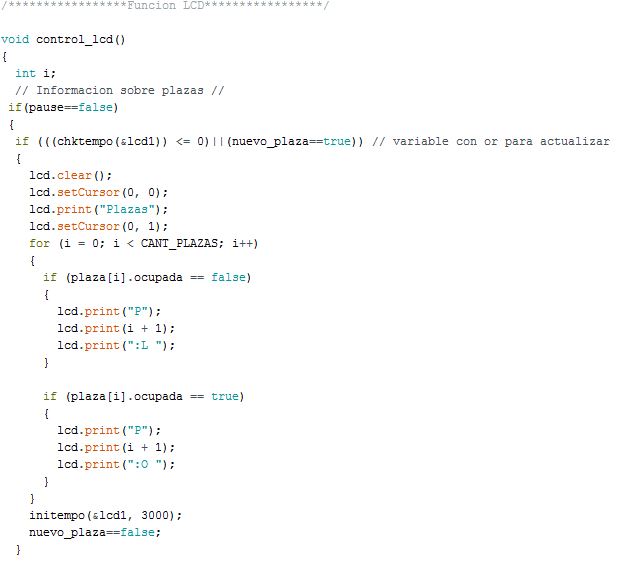
La función void control\_temperatura() será la encargada del control y gestion de los datos obtenidos por los sensores DHT-11, obteniendo asi la temperatura y humedad del parking.

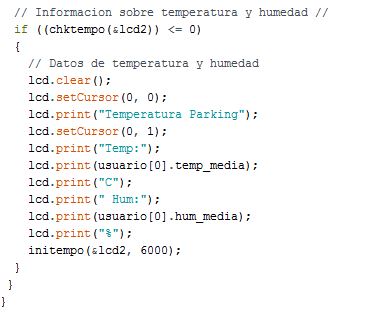


La función void alarma() se encargará del control de la alarma del parking.

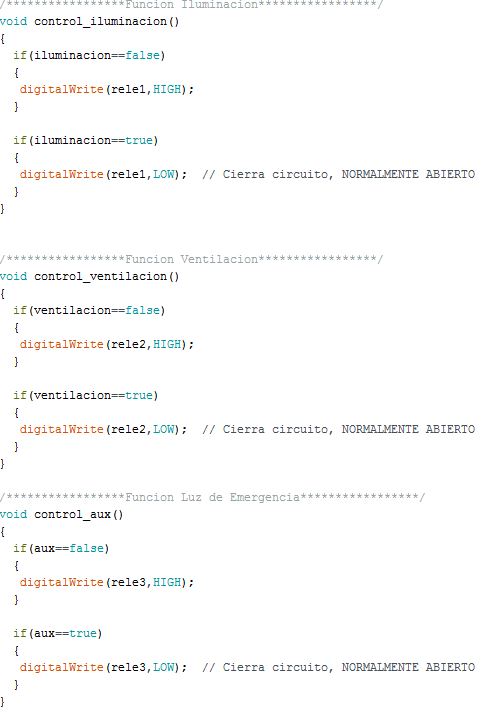


La función void control\_lcd() será la encargada del control de la información mostrada a través de la pantalla LCD.

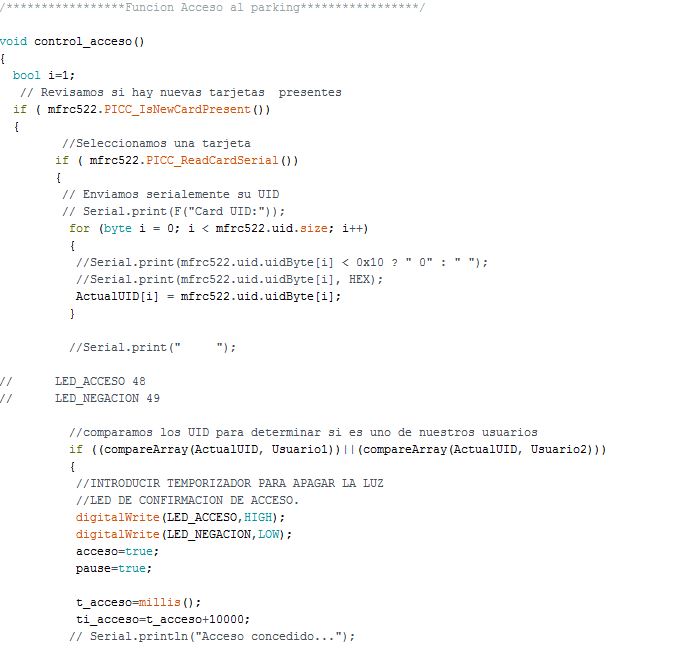


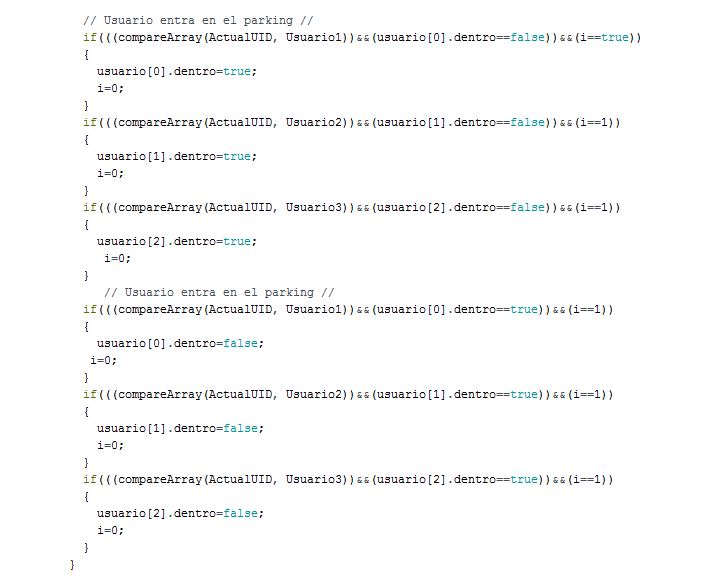


La funciones void control\_iluminacion(), void control\_ventilacion(),void control\_aux() será las encargadas de gobernar el módulo relé que estará instalado en el parking, que a su vez gobiernas los diferentes sistemas del parking.

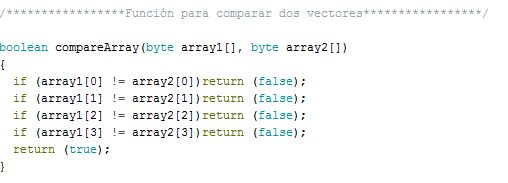


La función void control\_acceso() será la encargada de gobernar el módulo RFID, por lo tanto, nos aportara los datos de que usuario entra o sale del parking, esto se hace mediante la función boolean compareArray(byte array1[], byte array2[]), que compara los datos ID del TAG leído con los datos de ID conocidos. Por último, la función void control\_acceso() controlara la barrera con sus variables en la función La función void control\_barrera()







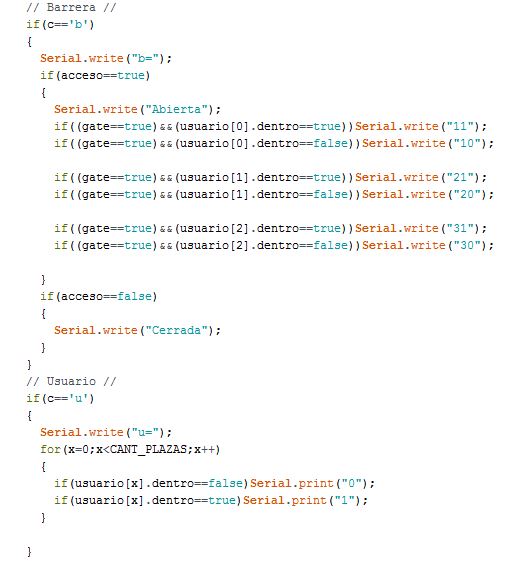


La función void control\_interfaz() será la encargada de la gestión de los datos, enviando los datos a la interfaz para que el usuario final tenga toda la información de lo que está ocurriendo en el parking. Para el funcionamiento de la comunicación serie entre Arduino y la Interfaz se ha creado un lenguaje de comunicación para el entendimiento de ambos lenguajes. El código usado es el siguiente:

* ‘0’ y ‘1’ 🡪 Iluminación
* ‘2’ y ‘3’ 🡪 Ventilación
* ‘4’ y ‘5’ 🡪 Alarma
* ‘6’ y ‘7’ 🡪 Iluminación auxiliar
* ‘t’ 🡪 Temperatura
* ‘h’ 🡪 Humedad
* ‘b’ 🡪 Barrera
* ‘u’ 🡪 Usuarios
* ‘p’ 🡪 Plazas
* ‘m’ , ‘n’ y ‘g’ 🡪 Ocupación de plazas
* ‘s’ 🡪 Sincronización

La información enviada desde Arduino como cadenas de caracteres, en ocasiones, la información ira codificada usando solo caracteres de ‘0’ y ‘1’ y será la interfaz la encargada de decodificar y actuar en consecuencia.









## 3.5 Interfaz para dispositivos sobremesa

### 3.5.1 Descripción y objetivo de la interfaz

La interfaz buscara un manejo simple, rápido y eficaz para gestionar nuestro parking. Aunque la programación que llevara la placa arduino estará pensada para funcionar de manera autónoma, se decidió añadir al proyecto una interfaz para tener un control puntual de ciertas facetas del parking. Además, de obtener datos del parking.

La interfaz será creada con la aplicación C++ Builder 6.

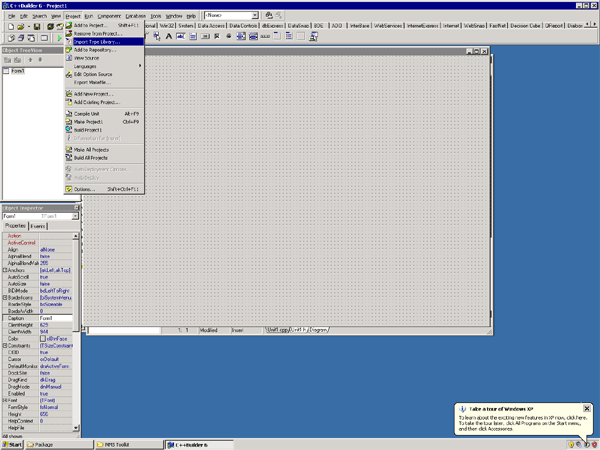


Figura 36. C++ Builder 6

### 3.5.2 Interfaz y su funcionamiento

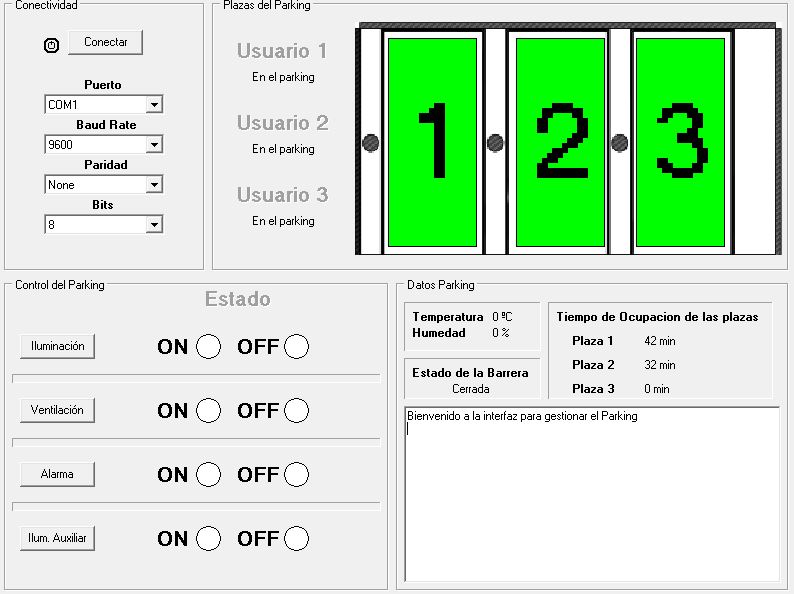


Figura 37. Interfaz para el proyecto

A simple vista se puede diferenciar 4 áreas bien definidas.

* Conectividad
* Plazas del Parking
* Control del Parking
* Datos de Parking

1. **Conectividad**

La 1º área por la que deberá pasar cualquier usuario que use la interfaz será la de conectividad. Desde esta área, se eligiera el puerto de conexión usado para comunicación serial, la velocidad de comunicación en baudios, la paridad y el tamaño de las tramas en bits.



Figura 38. Conectividad

1. **Plazas del Parking**

Esta área tiene como objeto dar una visión rápida del estado de las plazas, su numeración, la ubicación de la plaza y el estado de los usuarios.

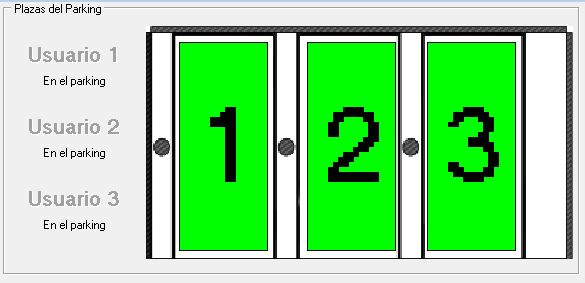


Figura 39. Plazas del Parking

1. **Control del Parking**

Desde el área de control, podremos manejar manualmente el sistema de ventilación, la iluminación y la alarma.



Figura 40. Control Parking

1. **Datos Parking**

En esta área nos encontraremos con diferentes datos a modo de información para el usuario que maneja la interfaz. Como la temperatura y humedad del parking, los usuarios registrados en el parking, el estado de la barrera y el tiempo de ocupación de cada plaza.

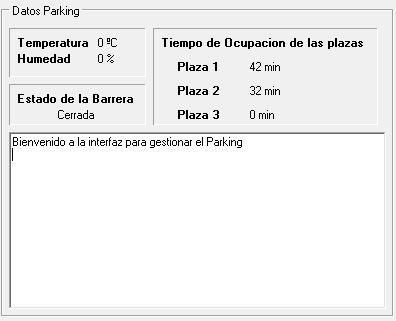


Figura 41. Datos de Parking

### 3.5.3 Modo de comunicación entre Arduino y el PC

La comunicación entre la interfaz y arduino se realiza mediante comunicación serie. Arduino cuenta con sus propias librerías para hacer esto posible, pero en el caso de C++ Builder 6 será necesario instalar la librería **CPortLib** para la gestión de la comunicación serie.

El componente TComPort (desarrollado por Dejan Crnila de Eslovenia) es un componente gratuito para comunicaciones serie para Delphi y C++ Builder. Permite el acceso al puerto serie del PC en Windows XP, Vista y Windows 7. TComPort se basa en la API de Windows por lo que es totalmente compatible con los controladores o drivers pasarela tipo USB - RS232. Es fácil de utilizar para propósitos de comunicación serie básicos. En tcomport.rar está disponible la versión 4.11 y en la página oficial http://sourceforge.net/projects/comport/ estará disponible la última versión.

TComPort encapsula en un componente no visible el acceso al puerto serie del PC. Tiene métodos que permiten abrir y cerrar el puerto, así como escribir y recibir datos a través de él.

# Presupuesto

## 4.1 Material para la maqueta

|  |  |  |  |
| --- | --- | --- | --- |
| CANTIDAD | DESCRIPCIÓN | P. UNIDAD | P. SUBTOTAL |
| 1 | Tablón de madera aglomerada,  dimensiones 120 x 60 x 1 cm | 4,7 | 4,7 |
| 4 | Tablón de madera contrachapada,  Dimensiones 60 x 30 x 0,5 cm | 2,65 | 10,6 |
| 1 | Varilla de haya,  Dimensiones 1000 x 1,5 mm | 0,8 | 0,8 |
| 1 | Moldura de madera,  Dimensiones 1000 x 1 mm | 2,4 | 2,4 |
| 2 | Lamina de Metacrilato,  Dimensiones 40 x 40 x 0,5 cm | 6,5 | 13 |
| 8 | Pintura Acrílica para madera, colores rojo, blanco, negro y gris | 0,99 | 7,92 |
| 50 | Tornillos de cabeza plana,  Dimensiones 3 x 20 mm | 0,0358 | 1,79 |
| 50 | Tornillos de cabeza plana,  Dimensiones 3 x 12 mm | 0,0276 | 1,38 |
| 50 | Barra de silicona térmica | 0,0712 | 3,56 |
| 1 | Sellador Adhesivo para madera, color blanco | 8,5 | 8,5 |
| 1 | Pack de brochas de pintura, distintos tamaños | 4,95 | 4,95 |
| 1 | Laca tapa poros para madera | 7,74 | 7,74 |
| TOTAL PARCIAL | | | 67,34 |

## 4.2 Placa Arduino, sensores y módulos

|  |  |  |  |
| --- | --- | --- | --- |
| CANTIDAD | DESCRIPCIÓN | P. UNIDAD | P. SUBTOTAL |
| 1 | Placa Arduino, modelo Mega 2560 | 10,2 | 10,2 |
| 1 | Módulo RFID (modelo MFRC522) + 2 tags | 7,76 | 7,76 |
| 1 | Módulo LCD 16x2 HD44780 | 2,57 | 2,57 |
| 1 | Módulo x4 Relé | 2,96 | 2,96 |
| 5 | Sensor HC-SR04 | 1,45 | 7,25 |
| 10 | Sensor TCRT 5000/5000L | 0,171 | 1,71 |
| 2 | Sensor DHT-11 | 1,115 | 2,23 |
| TOTAL PARCIAL | | | 34,68 |

## 4.3 Materiales complementarios para la maqueta

|  |  |  |  |
| --- | --- | --- | --- |
| CANTIDAD | DESCRIPCIÓN | P. UNIDAD | P. SUBTOTAL |
| 40 | Cable macho a hembra “Jersey” para Arduino | 0,08275 | 3,31 |
| 1 | Pack de componentes eléctricos ( Resistencias, leds, potenciómetros) | x | 7,8 |
| 75 | Diodos emisores de luz de 3mm, tres colores (RGY) | 0,01813 | 1,36 |
| 50 | Diodos emisores de luz de 5mm , color blanco | 0,03 | 1,5 |
| 1 | 2860 Tiepoint Protoboard + Jumpwires | x | 13,88 |
| 20 | Cable de red CAT 5 | 0,378 | 7,56 |
| 12 | Regleta de conexiones negras | 0,01025 | 1,23 |
| TOTAL PARCIAL | | | 36,64 |

## 4.4 Coste total de todas las partes

|  |  |
| --- | --- |
| DESCRIPCIÓN | TOTAL PARCIAL |
| Material para la maqueta | 67,34 |
| Placa Arduino, sensores y módulos | 34,68 |
| Materiales complementarios para la maqueta | 36,64 |
| COSTE TOTAL DEL PROYECTO | 138,66 |

# Conclusiones

## 5.1 Problemas e inconvenientes encontrados

En este apartado se relatara los inconvenientes aparecidos durante la realización de este proyecto y que deben de servir como lección para futuros proyectos.

El autor de este documento reconoce, que debido a ser el primer proyecto al que se enfrentaba, el modo de actuación a la hora de ejecutar el proyecto no ha sido de los más correctos.

Para empezar, el autor de este proyecto una vez con una idea en la mente, pero no demasiado desarrollada, empezó con la realización de pruebas y la adquisición de algunas materiales para probar su validez en el proyecto. Se empezó a programar una aplicación modelo y se ideo como se complementaria.

Recuerdo al lector, que se trata de un proyecto con fines didácticos, por lo tanto el autor del proyecto y el que lo recibe (en cierto modo) es el mismo. Esto hace que la tarea de elaboración del proyecto sea lenta y este en constante evolución. Aquí es donde reside el primero fallo del autor. El autor debería haber trazado un modo de ejecución, es decir, investigar y elaborar un calendario, comprobar de un modo más extenso las futuras herramientas usadas.

El sensor HC-SR04 del que se ha hablado y aportado documentación técnica en este proyecto, como ya se ha mencionado, se trata de un sensor de ultrasonidos, que a niveles prácticos, es ideal en varios sentidos, tanto económico, sencillez de manejo, tamaño y fácil montaje. Pero que posee un problema grave debido a que se tratan de ondas mecánicas. En el caso de la maqueta del parking, si se estacionaba un vehículo con una formas determinadas, esto podía provocar un rebote de las ondas erróneo o mal dirigido a otros sensores, por lo que es un factor a tener en cuenta, además, de que se necesita de un objeto que tenga una gran superficie para el correcto funcionamiento. Por supuesto, a nivel de maqueta se solventa sabiendo exactamente la altura a la que se encuentra el sensor y por lo tanto, obligando a dar como ocupada la plaza en el momento que no se de esa medida exacta.

El sensor TCRT 5000/5000L usado como sensor de plaza también, al igual que el HC-SR04, en la maqueta funciona correctamente, pero si se quiere extrapolar a otra situación, se trata de un sensor de poca capacidad, llegando solo a los 2mm de detección. Además, al tratarse de un sensor de IR, su comportamiento varía con ciertos colores, como el blanco o el negro, dando unas lecturas distintas.

El cableado en este proyecto, no tiene demasiada importancia en cuanto a estructura u ordenación, pero hubo ciertos momentos en los que se volvió tedioso debido a que un simple fallo, se volvía muy complejo de encontrar debido al gran volumen de conexiones. Uno de estos casos lo provoco el conexionado de la pantalla LCD, que cuenta cerca de 8 cables solo para su funcionamiento.

La elaboración de la interfaz fue relegada a las últimas semanas, el autor asumía que no supondría ninguna dificultad (grave error). Si nos ponemos en situación y recordamos que la aplicación usada para la creación de la interfaz es C++ Builder 6, una aplicación del año 2002 junto a la librería CPortLib, gratuita pero desatendida desde el 2004, nos encontramos en un punto que no preví. Para empezar, la falta de documentación para complementar la librería provoco complicada la realización de la comunicación serie. Esto provocó que el autor del proyecto, tuviera que mal decir al cielo y hacer herejías con el lenguaje de programación que usaría la interfaz para conseguir medianamente un resultado satisfactorio.

## 5.2 Alternativas, mejoras y otras cuestiones

Se va a mencionar ciertas alternativas que se pensaron, aunque podría haber mejorado las prestaciones del proyecto y otras se desecharon por reducir la saturación.

Empezare por las opciones que se desecharon.

Hay varios sensores que se barajaron para incluir en el proyecto, entre ellos se encuentran los siguientes:

* Sensor de humo o CO2
* Sensor de proximidad
* Sensor de movimiento
* Sensor de presión
* Sensor de sonido
* Motor para elevación de la barrera

En el caso del sensor de humo o C02, siendo una maqueta, se hubiera ideado algún método de introducción de humo para la comprobación de datos, pero la principal causa de desecharlo, es su baja complejidad y sobre todo similitud con el sensor DHT-11 de temperatura.

En el caso de los sensores de proximidad, movimiento y presión, no tenían cabida en este proyecto, debido a que se trataba de detectar un objeto inmóvil, de gran tamaño y que fuera viable su exportación a la vida real.

Para la mejora del sistema de detección de las plazas, se pensó en un principio en sensores IR de haz, donde la detección seria de mayor calidad, aunque este sistema era caro, aunque sería una gran opción si se decidiera trasladarlo a la vida real.

Las alternativas a la placa Arduino eran varias. El motivo por el que se optó por Arduino fue por su precio económico y por haberse utilizado en el grado, lo que ayudaba a ampliar los conocimientos adquiridos.

* **Raspberry Pi** se trata de una diminuta placa base de 85 x 54 milímetros (un poco más grande que una cajetilla de tabaco) en el que se aloja un chip Broadcom BCM2835 con procesador ARM hasta a 1 GHz de velocidad, GPU VideoCore IV y hasta 512 Mbytes de memoria RAM.



Figura 42. Raspberry Pi

* **BeagleBone** tienen un procesador ARM A8 cortex a 700 Mhz y una memoria RAM de 256 MB. Mide 9 cms de largo y 5,5 cms de ancho, lee tarjetas microSD, tiene un puerto USB para conectar un periférico o utilizarlo como fuente de energía, un micro USB para conectarlo a otro ordenador, un conector Ethernet y un conector RJ45. En estos momentos hay dos modelos de placa, la BeagleBone Clásica, de 89 dólares, y la BeagleBone Black, algo más barata. Como BeagleBone funciona con Linux, cualquier desarrollador puede hacer su propio software en una gran variedad de lenguajes: C, C++, Java, Python, Ruby, PHP, Javascript, Node.js… Además es compatible con otros sistemas operativos como Android, Ubuntu o Debian. Las posibilidades son enormes.

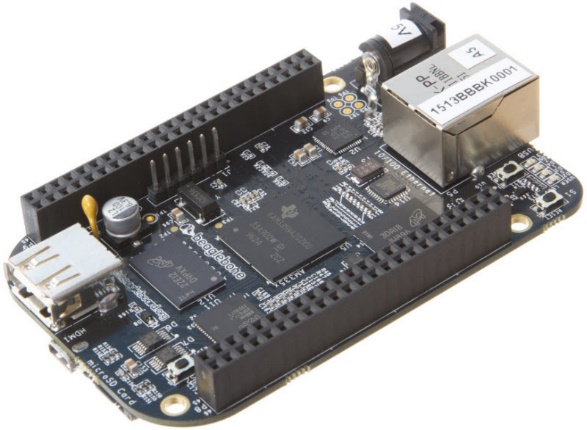


Figura 43. BeagleBone

# Índice de imágenes

**Imágenes en el documento**

Figura 0. Diagrama de Gantt**………………………...........………………...**7

Figura 1. Ejemplo de tablón de aglomerado**………………………............**15

Figura 2. Ejemplo de tablón de contrachapado**………………………....…**16

Figura 3. Ejemplo de lámina de metacrilato**………………….........……..**16

Figura 4. Módulo HC-SR04**………………………………….………………..**18

Figura 5. Tabla de parámetros HC-SR04**….………………………………..**19

Figura 6. Rango y dimensiones de HC-SR**…………………………………..**19

Figura 7. Diagrama de tiempo**………………………………...….………...**20

Figura 8. Sensor TCRT 5000/5000L**….……………………………………..**21

Figura 9. Tabla de parámetros TCRT 5000/5000L**….……………………..**22

Figura 10. Diagrama y esquema de TCRT 5000/5000L**……………….…..**22

Figura 11. Placa Arduino Mega 2560**………………………………………..**23

Figura 12. Tabla de parámetros Arduino Mega 2560**…………………......**24

Figura 13. Esquema Arduino Mega 2560**………………………….………..**24

Figura 14. Dimensiones de la placa Arduino Mega 2560**……………….…**25

Figura 15. IDE de Arduino**…...…………….………………………………..**26

Figura 16. Módulo RC522**………………..…………………………………..**27

Figura 17. Tabla de parámetros RC522**………………..…………………..**28

Figura 18. Tarjeta y llavero RFID**…………….……………………………..**28

Figura 19. Módulo x4 relés**……………………………….…………………..**29

Figuras 20 y 21. Resistencias fijas y un potenciómetro**…………………...**30

Figuras 22 y 23. LED blanco LED’s de colores**……………………………..**31

Figura 24. Servomotor**………………………........................................**32

Figura 25. Ventilador de 80 mm**…………………………….………...........**33

Figura 26. Batería lipo de 7,8 v**…………………………………….............**33

Figura 27. Conexión Placa-Servo**…………………………………………….**35

Figura 28. Conexión Placa-LCD**………………………………………………**36

Figura 29. Conexión Placa-Led de ocupación**………………………………**37

Figura 30. Conexión Placa-DHT11**…………………………………….…….**37

Figura 31. Conexión Placa-Módulo RFID**…………………………………...**38

Figura 32. Conexión Placa-HC-SR04**………………………………………..**39

Figura 33. Conexión Placa-TCRT5000/5000L**………………………………**39

Figura 34. Conexión Placa-Relé**……………………………………...………**40

Figura 35. Conexión Placa-Zumbadores**…………………………….………**40

Figura 36. C++ Builder 6**…………………………………………….……….**65

Figura 37. Interfaz para el proyecto**………………………………..……….**66

Figura 38. Conectividad**………………………………..…………………….**67

Figura 39. Plazas del Parking**……………………………………..………….**67

Figura 40. Control Parking**……………………………………………..…….**68

Figura 41. Datos de Parking**…………………………………………..….….**69

Figura 42. Raspberry Pi**……………………………………………..………..**78

Figura 43. BeagleBone**………………………………………….……………**79

# Glosario de términos

**A**

**Arduino:** Es una compañía de hardware libre, la cual desarrolla placas de desarrollo que integran un microcontrolador y un entorno de desarrollo (IDE), diseñado para facilitar el uso de la electrónica en proyectos multidisciplinarios.

El hardware consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, y puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión (shields) que amplían las características de funcionamiento de la placa arduino.

Por otro lado, el software consiste en un entorno de desarrollo (IDE) basado en el entorno de Processing y lenguaje de programación basado en Wiring, así como en el cargador de arranque (bootloader) que es ejecutado en la placa. El microcontrolador de la placa se programa a través de un computador, haciendo uso de comunicación serial mediante un convertidor de niveles RS-232 a TTL serial.

**B**

**Bootloader:** Gestor de arranque de un sistema.

**C**

**C y C++:** Se tratan de 2 lenguajes de programación. C se trata de un lenguaje de tipos de datos estáticos, tratándose de un lenguaje de bajo nivel. Mientras C++ se trata de un lenguaje orientado a objetos.

**CAT:** Se trata de una categoría estandarizada para los cables de pares trenzados.

**L**

**LCD:** Liquid Crystal Display.

**LED:** Light-Emitting Diode.

**Loop:** El loop en Arduino, es la función que contendrá todo nuestro código y que se ejecutara ilimitadamente.

**I**

**ICSP:** In-Circuit Serial Programming.

**IDE:** Integrated development environment.

**I/O:** Input/Output, Entrada y salida respectivamente.

**IR:** Infrarrojo.

**N**

**NC o NA:** Normalmente Cerrado o Normalmente Abierto.

**P**

**PWM:** Modulación por ancho de pulsos.

**R**

**RFID:** Radio Frecuency IDentification .

**S**

**SCPAS:** Sistema de control de parking automatizado por software.

**SO:** Sistema Operativo.

**SPI:** Bus Serial Peripheral Interface.

**T**

**TTL:** es la sigla en inglés de transistor-transistor logic, es decir, «lógica transistor a transistor». Es una tecnología de construcción de circuitos electrónicos digitales.

**U**

**UART:** universal asynchronous receiver/transmitter.

# Referencias

A continuación se ha redactado las fuentes consultadas para la confección de este proyecto.

* Apuntes de la asignatura “GESTION DE PROYECTOS INTEGRADOS”
* <http://www.cplusplus.com/>
* <http://www.prometec.net/>
* <https://hetpro-store.com/>
* <http://www.educachip.com/>
* <https://www.arduino.cc/>
* https://es.wikipedia.org/
* http://www.leroymerlin.es/
* <https://panamahitek.com/>
* <http://fritzing.org/>
* <http://es.ccm.net/>
* <http://www.carlospes.com/>
* <https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C>
* <http://www.yevol.com/>
* <https://whitehathacking.wordpress.com/>
* http://perso.wanadoo.es/pictob/tcomport.htm

# Anexos

## 9.I Codigo Arduino

// Version 1.0a //

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Librerias \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <DHT.h>

#include <LiquidCrystal.h>

#include <SPI.h>

#include <MFRC522.h>

#include <Servo.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Estructura para las plazas \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct

{

/\*\*\*\*\*\*\*\*\*Datos del programa\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sensor Ultrasonido\*/

int trigger;

int echo;

int IR;

/\*Variable para la funcion temporizador\*/

unsigned long temporizador;

/\*LED de ocupacion

\* Rojo: Ocupado

\* Verde: Libre

\*/

int LED\_ROJO;

int LED\_VERDE;

/\*Variables para la funcion control\_plaza

\* Ocupada --> Indicara si los sensores han marcado una plaza como ocupada o libre.

\* Ciclo --> Variable para controlar cuando empieza la ocupacion de una plaza y cuando termina.

\* Activador --> Sin uso.

\*/

bool ocupada;

bool ciclo;

bool activador;

/\* Mediciones realizadas por los sensores de temparatura

\* Temp = Temperatura

\* Hum = Humedad

\*/

int temp1;

int temp2;

int hum1;

int hum2;

/\*\*\*\*\*\*\*\*\*Datos para el usuario\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int temp\_media;

int hum\_media;

/\*\*\*\*\*\*\*\*\*Datos de usuarios\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned long int tiempo\_inicio;

unsigned long int tiempo\_final;

unsigned long int tiempo\_ocupacion;

bool dentro;

} datos;

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Variables globales \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define CANT\_PLAZAS 3

#define DHTTYPE 11

#define RST\_PIN 5 //Pin 5 para el reset del RC522

#define SS\_PIN 53 //Pin 53 para el SS (SDA) del RC522

#define LED\_ACCESO 48

#define LED\_NEGACION 49

MFRC522 mfrc522(SS\_PIN, RST\_PIN); ///Creamos el objeto para el RC522

/\*\*\*\*\*\*\*\*\*Variables para RFID\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Crear una estructura para manejar ilimitados usuarios.

\*/

byte ActualUID[4]; //Almacenará el código del Tag leído

byte Usuario1[4] = {0x73, 0x20, 0x7B, 0x25} ; //Código Tarjeta Blanca

byte Usuario2[4] = {0xE6, 0x2B, 0xEE, 0x30} ; //Código Llavero Azul

byte Usuario3[4] = {}; // Espacio libre para un usuario mas

/\*\*\*\*\*\*\*\*\*Variables de la estructura de datos\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

datos plaza[CANT\_PLAZAS];

datos usuario[CANT\_PLAZAS];

unsigned long tiempo = millis();

unsigned long TIEMPO\_SENSOR = 0;

/\*\*\*\*\*\*\*\*\*Variables Zumbadores\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int alarma1;

int alarma2;

unsigned long encender = 0;

unsigned long apagar = 0;

unsigned long valor\_encender = 200;

unsigned long valor\_apagar = 400;

/\*\*\*\*\*\*\*\*\*Variables de temporizador LCD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned long lcd0 = 0;

unsigned long lcd1 = 3000;

unsigned long lcd2 = 6000;

unsigned long lcd3 = 9000;

/\*\*\*\*\*\*\*\*\*Pines RELE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int rele1=36; // Iluminacion

int rele2=37; // Ventilacion

int rele3=38; // Luz de emergencias

int rele4=39; // Sin uso

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Variables de control \*/

/\* mediante interfaz \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool iluminacion=false; // Variable de activacion

bool ventilacion=false; // Variable de activacion

bool alert=false; // Variable de activacion de alarma por interfaz

bool gate=false; // Variable de activacon barrera

bool aux=false; // Variable de activacion luces de emergencia

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Variables de control \*/

/\* internas \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool temp=false; // Variable que controla la manera que funciona la funcion alarma

bool nuevo\_plaza=false; // Variable de activacion. Para repintar LCD cuando se detecten nuevos datos.

bool alarm=false; // Variable de activacion para los sensores de temperatura

bool acceso=false; // Variable de acceso positivo

bool no\_acceso=false; // Variable de acceso negativo

unsigned long t\_acceso=0;

unsigned long ti\_acceso=0;

unsigned long t\_noacceso=0;

unsigned long ti\_noacceso=0;

bool pause=false;

int c = 0;

DHT dht1(34, DHTTYPE);

DHT dht2(35, DHTTYPE);

LiquidCrystal lcd(42, 43, 44, 45, 46, 47);

Servo SERVO;

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Funciones \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Temporizador\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initempo(unsigned long \*crono, unsigned long Time)

{

\*crono = millis() + Time;

}

unsigned long chktempo(unsigned long \*crono)

{

if (\*crono > millis())

{

return (\*crono - millis());

}

else

{

return 0;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Plaza\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_plazas(bool &nuevo\_plaza)

{

// La variable i, indica la plaza correspondiente en cada momento.

int i;

if (chktempo(&plaza[0].temporizador) <= 0)

{

for (i = 0; i < CANT\_PLAZAS; i++)

{

// Serial.print("Plaza: ");

// Serial.print(i);

// Serial.print("\n");

plaza[i].ocupada = sensor\_plaza(i);

if ((plaza[i].ocupada == false) && (plaza[i].ciclo == false))

{

digitalWrite(plaza[i].LED\_ROJO, LOW);

digitalWrite(plaza[i].LED\_VERDE, HIGH);

}

if ((plaza[i].ocupada == true) && (plaza[i].ciclo == false))

{

// Se inicia el contador de tiempo para la plaza correspondiente

plaza[i].tiempo\_inicio = millis();

// Se enciende el LED ROJO para seÃƒÂ±alizar plaza ocupada

digitalWrite(plaza[i].LED\_ROJO, HIGH);

digitalWrite(plaza[i].LED\_VERDE, LOW);

// Se activa la variable de identificacion para el contador

plaza[i].ciclo = true;

nuevo\_plaza=true;

}

if ((plaza[i].ocupada == true) && (plaza[i].ciclo == true))

{

// Se idica el tiempo de ocupacion de la plaza en ms.

plaza[i].tiempo\_final = millis() - plaza[i].tiempo\_inicio;

}

if ((plaza[i].ocupada == false) && (plaza[i].ciclo == true))

{

plaza[i].ciclo = false;

plaza[i].tiempo\_final = millis() - plaza[i].tiempo\_inicio;

//segundos

// Se enciende el LED VERDE para seÃƒÂ±alizar plaza libre

digitalWrite(plaza[i].LED\_ROJO, HIGH);

digitalWrite(plaza[i].LED\_VERDE, LOW);

}

}

initempo(&plaza[0].temporizador, 2000);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Sensores\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool sensor\_plaza(int i)

{

float tiempo = 0, distancia = 0, estado = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// - Falta definir los parametros de "distancia" y "estado" para la entrega de datos de la funcion

// mediante "true" o "false".

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

do {

digitalWrite(plaza[i].trigger, LOW); // Por cuestion de estabilizacion del sensor

delayMicroseconds(5);

digitalWrite(plaza[i].trigger, HIGH); // Se realiza un pulso

delayMicroseconds(10);

tiempo = pulseIn(plaza[i].echo, HIGH);

/\* FunciÃƒÂ³n para medir la longitud del pulso entrante. Mide el tiempo que transcurrido entre el envÃƒÂ­o

del pulso ultrasÃƒÂ³nico y cuando el sensor recibe el rebote, es decir: desde que el pin "plaza[i].trigger"

empieza a recibir el rebote, HIGH, hasta que deja de hacerlo, LOW, la longitud del pulso entrante\*/

distancia = int(0.017 \* tiempo);

/\*fÃƒÂ³rmula para calcular la distancia obteniendo un valor entero\*/

} while (distancia <= 0);

estado = analogRead(plaza[i].IR);

//////// LINEAS TEST , MONITOR SERIE ////////

/\*

if (distancia != 0)

{

Serial.print("tiempo: ");

Serial.print(tiempo);

Serial.print("\n");

Serial.print("cm: ");

Serial.print(distancia);

Serial.print("\n");

Serial.print("IR: ");

Serial.print(estado);

Serial.print("\n");

}

/\* Aqui finaliza la funcion, devolviendo "true" o "false".

\*

\* Para que se cumpla la condicion de "true", los 2 sensores deben estar de acuerdo,segun nuestro criterio

\*

\*/

if (((distancia < 14)||(distancia > 18)) && (estado < 800))

{

return true;

}

else

{

return false;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Temperatura\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_temperatura()

{

if (chktempo(&usuario[0].temporizador) <= 0)

{

usuario[0].hum1 = dht1.readHumidity();

usuario[0].temp1 = dht1.readTemperature();

usuario[0].hum2 = dht2.readHumidity();

usuario[0].temp2 = dht2.readTemperature();

usuario[0].temp\_media = ((usuario[0].temp1 + usuario[0].temp2) / 2);

usuario[0].hum\_media = ((usuario[0].hum1 + usuario[0].hum2) / 2);

initempo(&usuario[0].temporizador, 3000);

//////// LINEAS TEST , MONITOR SERIE ////////

/\* Serial.print("humedad: ");

Serial.print(usuario[0].hum\_media);

Serial.print("\n");

Serial.print("temperatura: ");

Serial.print(usuario[0].temp\_media);

Serial.print("\n");

\*/

}

/\*

\* Conflicto, Y si,¿hay mas de 40º grados, pero quiero apagar la alarma?

\* ¿Orden de mayor peso?

\*/

if ((usuario[0].temp\_media >= 40)&&(alarm==false))

{

//ALARMA DE INCENDIOS Y VENTILACION

temp=true;

alarma();

ventilacion=true;

control\_ventilacion();

}

if ((usuario[0].temp\_media <= 40)&&(alarm==true))

{

temp=false;

alarma();

ventilacion=false;

control\_ventilacion();

}

temp=false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Alarma\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void alarma()

{

/\*

\* Alarma1 , sera el zumbador encargado de simular una alarma

\* Alarma2 , sera el zumbador encargado de simular un pitido de negacion de acceso

\* para el modulo RFID

\*/

////////// Valores iniciales ///////////

////////// ENCENDER=200 |-| APAGAR=400 /////////

if((temp==true)||(alert==true))

{

if ((chktempo(&encender)) <= 0)

{

analogWrite(alarma1, 20);

//analogWrite(alarma2, 20);

initempo(&encender, valor\_encender);

}

if (chktempo(&apagar) <= 0)

{

analogWrite(alarma1, 0);

//analogWrite(alarma2, 0);

initempo(&apagar, valor\_apagar);

}

}

/\*

Serial.print("\n");

Serial.print("\n");

Serial.print(no\_acceso);

Serial.print("\n");

Serial.print("\n");

\*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion LCD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_lcd()

{

int i;

// Informacion sobre plazas //

if(pause==false)

{

if (((chktempo(&lcd1)) <= 0)||(nuevo\_plaza==true)) // variable con or para actualizar

{

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Plazas");

lcd.setCursor(0, 1);

for (i = 0; i < CANT\_PLAZAS; i++)

{

if (plaza[i].ocupada == false)

{

lcd.print("P");

lcd.print(i + 1);

lcd.print(":L ");

}

if (plaza[i].ocupada == true)

{

lcd.print("P");

lcd.print(i + 1);

lcd.print(":O ");

}

}

initempo(&lcd1, 3000);

nuevo\_plaza==false;

}

// Informacion sobre temperatura y humedad //

if ((chktempo(&lcd2)) <= 0)

{

// Datos de temperatura y humedad

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temperatura Parking");

lcd.setCursor(0, 1);

lcd.print("Temp:");

lcd.print(usuario[0].temp\_media);

lcd.print("C");

lcd.print(" Hum:");

lcd.print(usuario[0].hum\_media);

lcd.print("%");

initempo(&lcd2, 6000);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Iluminacion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_iluminacion()

{

////////// LINEAS DE TEST //////////

////////////////////////////////////

/\*

if(plaza[0].ocupada==true)

{

iluminacion=true;

}

if(plaza[0].ocupada==false)

{

iluminacion=false;

}

\*/

////////////////////////////////////

if(iluminacion==false)

{

digitalWrite(rele1,HIGH);

}

if(iluminacion==true)

{

digitalWrite(rele1,LOW); // Cierra circuito, NORMALMENTE ABIERTO

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Ventilacion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_ventilacion()

{

////////// LINEAS DE TEST //////////

////////////////////////////////////

/\*

if(plaza[1].ocupada==true)

{

ventilacion=true;

}

if(plaza[1].ocupada==false)

{

ventilacion=false;

}

\*/

////////////////////////////////////

if(ventilacion==false)

{

digitalWrite(rele2,HIGH);

}

if(ventilacion==true)

{

digitalWrite(rele2,LOW); // Cierra circuito, NORMALMENTE ABIERTO

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Luz de Emergencia\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_aux()

{

if(aux==false)

{

digitalWrite(rele3,HIGH);

}

if(aux==true)

{

digitalWrite(rele3,LOW); // Cierra circuito, NORMALMENTE ABIERTO

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funcion Acceso al parking\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_acceso()

{

bool i=1;

// Revisamos si hay nuevas tarjetas presentes

if ( mfrc522.PICC\_IsNewCardPresent())

{

//Seleccionamos una tarjeta

if ( mfrc522.PICC\_ReadCardSerial())

{

// Enviamos serialemente su UID

// Serial.print(F("Card UID:"));

for (byte i = 0; i < mfrc522.uid.size; i++)

{

//Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");

//Serial.print(mfrc522.uid.uidByte[i], HEX);

ActualUID[i] = mfrc522.uid.uidByte[i];

}

//Serial.print(" ");

// LED\_ACCESO 48

// LED\_NEGACION 49

//comparamos los UID para determinar si es uno de nuestros usuarios

if ((compareArray(ActualUID, Usuario1))||(compareArray(ActualUID, Usuario2)))

{

//INTRODUCIR TEMPORIZADOR PARA APAGAR LA LUZ

//LED DE CONFIRMACION DE ACCESO.

digitalWrite(LED\_ACCESO,HIGH);

digitalWrite(LED\_NEGACION,LOW);

acceso=true;

pause=true;

t\_acceso=millis();

ti\_acceso=t\_acceso+10000;

// Serial.println("Acceso concedido...");

// Usuario entra en el parking //

if(((compareArray(ActualUID, Usuario1))&&(usuario[0].dentro==false))&&(i==true))

{

usuario[0].dentro=true;

i=0;

}

if(((compareArray(ActualUID, Usuario2))&&(usuario[1].dentro==false))&&(i==1))

{

usuario[1].dentro=true;

i=0;

}

if(((compareArray(ActualUID, Usuario3))&&(usuario[2].dentro==false))&&(i==1))

{

usuario[2].dentro=true;

i=0;

}

// Usuario entra en el parking //

if(((compareArray(ActualUID, Usuario1))&&(usuario[0].dentro==true))&&(i==1))

{

usuario[0].dentro=false;

i=0;

}

if(((compareArray(ActualUID, Usuario2))&&(usuario[1].dentro==true))&&(i==1))

{

usuario[1].dentro=false;

i=0;

}

if(((compareArray(ActualUID, Usuario3))&&(usuario[2].dentro==true))&&(i==1))

{

usuario[2].dentro=false;

i=0;

}

}

else

{

no\_acceso=true;

t\_noacceso=millis();

ti\_noacceso=t\_noacceso+2000;

// Serial.println("Acceso denegado...");

}

// Terminamos la lectura de la tarjeta tarjeta actual

////////// LINEAS DE TEST //////////

////////////////////////////////////

/\* Serial.println("He llegado aqui, Valor=");

Serial.println(no\_acceso);\*/

////////////////////////////////////

mfrc522.PICC\_HaltA();

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Función para comparar dos vectores\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

boolean compareArray(byte array1[], byte array2[])

{

if (array1[0] != array2[0])return (false);

if (array1[1] != array2[1])return (false);

if (array1[2] != array2[2])return (false);

if (array1[3] != array2[3])return (false);

return (true);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Función para elevar/bajar barrera y control de LED's de acceso\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control\_barrera()

{

//Minimo del servo 1º. Maximo del servo 180º.//

//GRADO 1: BARRERA ABIERTA. //

//GRADO 110: BARRERA CERRADA. //

if(((acceso==true)&&(compareArray(ActualUID, Usuario1))))

{

if(usuario[0].dentro==true)

{

SERVO.write(10);

gate=true;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(" Bienvenido");

lcd.setCursor(0, 1);

lcd.print(" Usuario1");

}

if(usuario[0].dentro==false)

{

SERVO.write(10);

gate=true;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Barrera Abierta");

lcd.setCursor(0, 1);

lcd.print(" Adios Usuario1");

}

}

if(((acceso==true)&&(compareArray(ActualUID, Usuario2))))

{

if(usuario[1].dentro==true)

{

SERVO.write(10);

gate=true;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(" Bienvenido");

lcd.setCursor(0, 1);

lcd.print(" Usuario2");

}

if(usuario[1].dentro==false)

{

SERVO.write(10);

gate=true;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Barrera Abierta");

lcd.setCursor(0, 1);

lcd.print(" Adios Usuario2");

}

}

if((t\_acceso>=ti\_acceso)&&(acceso==true))

{

SERVO.write(100);

pause=false;

acceso=false;

gate=false;

digitalWrite(LED\_ACCESO,LOW);

digitalWrite(LED\_NEGACION,HIGH);

}

if(no\_acceso==true)

{

analogWrite(alarma2, 20);

pause=true;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Acceso denegado");

}

if((t\_noacceso>=ti\_noacceso)&&(no\_acceso==true))

{

analogWrite(alarma2,0);

pause=false;

no\_acceso=false;

}

}

void control\_interfaz()

{

int x;

if(Serial.available())

{

char c=Serial.read();

// Iluminación //

if(c=='0'){iluminacion=false;}

if(c=='1'){iluminacion=true; }

// Ventilación //

if(c=='2'){ventilacion=false;}

if(c=='3'){ventilacion=true;}

// Alarma //

if(c=='4'){alert=false;}

if(c=='5'){ alert=true;}

// Iluminacion de emergencia //

if(c=='6'){aux=false;}

if(c=='7'){aux=true;}

// Temperatura //

if(c=='t')

{

Serial.write("t=");

Serial.print(usuario[0].temp\_media);

}

// Humedad //

if(c=='h')

{

Serial.write("h=");

Serial.print(usuario[0].hum\_media);

}

// Barrera //

if(c=='b')

{

Serial.write("b=");

if(acceso==true)

{

Serial.write("Abierta");

if((gate==true)&&(usuario[0].dentro==true))Serial.write("11");

if((gate==true)&&(usuario[0].dentro==false))Serial.write("10");

if((gate==true)&&(usuario[1].dentro==true))Serial.write("21");

if((gate==true)&&(usuario[1].dentro==false))Serial.write("20");

if((gate==true)&&(usuario[2].dentro==true))Serial.write("31");

if((gate==true)&&(usuario[2].dentro==false))Serial.write("30");

}

if(acceso==false)

{

Serial.write("Cerrada");

}

}

// Usuario //

if(c=='u')

{

Serial.write("u=");

for(x=0;x<CANT\_PLAZAS;x++)

{

if(usuario[x].dentro==false)Serial.print("0");

if(usuario[x].dentro==true)Serial.print("1");

}

}

// Plaza //

if(c=='p')

{

Serial.write("p=");

for(x=0;x<CANT\_PLAZAS;x++ )

{

if(plaza[x].ocupada==false)Serial.print("0");

if(plaza[x].ocupada==true)Serial.print("1");

}

}

// Ocupacion Plaza

if(c=='m')

{

if(plaza[0].ocupada==true)

{

Serial.print("p1=");

Serial.print((plaza[0].tiempo\_final/1000)/60);

}

else

{

Serial.print("Li1");

}

}

if(c=='n')

{

if(plaza[1].ocupada==true)

{

Serial.print("p2=");

Serial.print((plaza[1].tiempo\_final/1000)/60);

}

else

{

Serial.print("Li2");

}

}

if(c=='g')

{

if(plaza[2].ocupada==true)

{

Serial.print("p3=");

Serial.print((plaza[2].tiempo\_final/1000)/60);

}

else

{

Serial.print("Li3");

}

}

// Sincronizacion //

if(c=='s')

{

if(iluminacion==false)Serial.print(0);

if(iluminacion==true)Serial.print(1);

if(ventilacion==false)Serial.print(0);

if(ventilacion==true)Serial.print(1);

if(alert==false)Serial.print(0);

if(alert==true)Serial.print(1);

if(aux==false)Serial.print(0);

if(aux==true)Serial.print(1);

}

}//Serial.available()

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup()

{

int iniciador;

Serial.begin(9600);

Serial.print("Inicializando programa...\n");

Serial.print("Adjudicando valor a variables\n");

plaza[0].ocupada = false;

plaza[0].ciclo = false;

plaza[0].activador = false;

plaza[1].ocupada = false;

plaza[1].ciclo = false;

plaza[1].activador = false;

plaza[2].ocupada = false;

plaza[2].ciclo = false;

plaza[2].activador = false;

for (iniciador = 0; iniciador < CANT\_PLAZAS; iniciador++)

{

TIEMPO\_SENSOR = TIEMPO\_SENSOR + 2000;

plaza[iniciador].temporizador = TIEMPO\_SENSOR;

TIEMPO\_SENSOR = TIEMPO\_SENSOR - 1000;

usuario[iniciador].temporizador = TIEMPO\_SENSOR;

TIEMPO\_SENSOR = TIEMPO\_SENSOR + 1000;

usuario[iniciador].dentro=false;

}

Serial.print("Adjudicando PIN a las variables de los sensores HC-SR04 \n");

/\*\*\*\*\*\*\*\*\*Pin HC-SR04\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

plaza[0].activador = true;

plaza[0].trigger = 22;

plaza[0].echo = 23;

plaza[1].trigger = 24;

plaza[1].echo = 25;

plaza[2].trigger = 26;

plaza[2].echo = 27;

Serial.print("Adjudicando PIN a las variables de los LED de ocupación \n");

/\*\*\*\*\*\*\*\*\*Pin de LED's\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

plaza[0].LED\_ROJO = 28;

plaza[0].LED\_VERDE = 29;

plaza[1].LED\_ROJO = 30;

plaza[1].LED\_VERDE = 31;

plaza[2].LED\_ROJO = 32;

plaza[2].LED\_VERDE = 33;

Serial.print("Adjudicando PIN a las variables de los sensores TCRT5000 \n");

/\*\*\*\*\*\*\*\*\*Pin IR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

plaza[0].IR = A0;

plaza[1].IR = A1;

plaza[2].IR = A2;

Serial.print("Adjudicando PIN a las variables de los zumbadores \n");

/\*\*\*\*\*\*\*\*\*Pin Zumbador\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

alarma1 = 12;

alarma2 = 13;

Serial.print("Configurando el funcionamiento de los pin's digitales \n");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pines digitales\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Sensores ultrasonido \*/

/\* y \*/

/\* LED's de ocupaciÃ³n \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pinMode(plaza[0].trigger, OUTPUT);

pinMode(plaza[0].echo, INPUT);

pinMode(plaza[0].LED\_ROJO, OUTPUT);

pinMode(plaza[0].LED\_VERDE, OUTPUT);

pinMode(plaza[0].LED\_ROJO, HIGH);

pinMode(plaza[0].LED\_VERDE, HIGH);

pinMode(plaza[1].trigger, OUTPUT);

pinMode(plaza[1].echo, INPUT);

pinMode(plaza[1].LED\_ROJO, OUTPUT);

pinMode(plaza[1].LED\_VERDE, OUTPUT);

pinMode(plaza[1].LED\_ROJO, HIGH);

pinMode(plaza[1].LED\_VERDE, HIGH);

pinMode(plaza[2].trigger, OUTPUT);

pinMode(plaza[2].echo, INPUT);

pinMode(plaza[2].LED\_ROJO, OUTPUT);

pinMode(plaza[2].LED\_VERDE, OUTPUT);

pinMode(plaza[2].LED\_ROJO, HIGH);

pinMode(plaza[2].LED\_VERDE, HIGH);

digitalWrite(plaza[0].LED\_ROJO, LOW);

digitalWrite(plaza[0].LED\_VERDE, HIGH);

digitalWrite(plaza[1].LED\_ROJO, LOW);

digitalWrite(plaza[1].LED\_VERDE, HIGH);

digitalWrite(plaza[2].LED\_ROJO, LOW);

digitalWrite(plaza[2].LED\_VERDE, HIGH);

delay(500);

digitalWrite(plaza[0].LED\_ROJO, HIGH);

digitalWrite(plaza[0].LED\_VERDE, LOW);

digitalWrite(plaza[1].LED\_ROJO, HIGH);

digitalWrite(plaza[1].LED\_VERDE, LOW);

digitalWrite(plaza[2].LED\_ROJO, HIGH);

digitalWrite(plaza[2].LED\_VERDE, LOW);

delay(500);

digitalWrite(plaza[0].LED\_ROJO, LOW);

digitalWrite(plaza[0].LED\_VERDE, HIGH);

digitalWrite(plaza[1].LED\_ROJO, LOW);

digitalWrite(plaza[1].LED\_VERDE, HIGH);

digitalWrite(plaza[2].LED\_ROJO, LOW);

digitalWrite(plaza[2].LED\_VERDE, HIGH);

Serial.print("Encendido de Leds\n");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* InicializaciÃƒÂ³n y testeo \*/

/\* de los zumbadores \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pinMode(alarma1, OUTPUT);

pinMode(alarma2, OUTPUT);

////////// LINEAS DE TEST //////////

////////////////////////////////////

Serial.print("Probando zumbadores...\n");

analogWrite(alarma1, 20);

analogWrite(alarma2, 20);

delay(50);

analogWrite(alarma1, 0);

analogWrite(alarma2, 0);

delay(50);

analogWrite(alarma1, 20);

analogWrite(alarma2, 20);

delay(50);

analogWrite(alarma1, 0);

analogWrite(alarma2, 0);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Relé \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pinMode(rele1,OUTPUT);

pinMode(rele2,OUTPUT);

pinMode(rele3,OUTPUT);

pinMode(rele4,OUTPUT);

/\* Los Reles, 1 y 2, estan configurados como NA:

\* valor HIGH = CIRCUITO ABIERTO

\* valor LOW = CIRCUITO CERRADO

\*

\* Los Reles, 3 y 4, estan sin usar.

\*/

////////// LINEAS DE TEST //////////

////////////////////////////////////

Serial.print("Probando dispositivos controlados por rele\n");

digitalWrite(rele1,HIGH);

digitalWrite(rele2,HIGH);

digitalWrite(rele3,HIGH);

digitalWrite(rele4,HIGH);

delay(500);

digitalWrite(rele1,LOW);

digitalWrite(rele2,LOW);

digitalWrite(rele3,LOW);

digitalWrite(rele4,LOW);

delay(500);

digitalWrite(rele1,HIGH);

digitalWrite(rele2,HIGH);

digitalWrite(rele3,HIGH);

digitalWrite(rele4,HIGH);

delay(500);

digitalWrite(rele1,LOW);

digitalWrite(rele2,LOW);

digitalWrite(rele3,LOW);

digitalWrite(rele4,LOW);

delay(500);

digitalWrite(rele1,HIGH);

digitalWrite(rele2,HIGH);

digitalWrite(rele3,HIGH);

digitalWrite(rele4,HIGH);

////////////////////////////////////

Serial.print("Configurando el funcionamiento de los pin's analógicos \n");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pines analogicas\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Sensores IR \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pinMode(plaza[0].IR, INPUT);

pinMode(plaza[1].IR, INPUT);

pinMode(plaza[2].IR, INPUT);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Inicializacion y testeo \*/

/\* de LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

lcd.begin(16, 2);

lcd.display();

lcd.setCursor(0, 0);

lcd.print("Iniciando LCD");

lcd.setCursor(0, 1);

lcd.print("Pantalla de info");

delay(2000);

lcd.clear();

lcd.setCursor(0, 0);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Inicializacion y testeo \*/

/\* de RFID \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Serial.print("Configurando el funcionamiento del modulo RFID \n");

SPI.begin(); //Iniciamos el Bus SPI

pinMode(LED\_ACCESO,OUTPUT);

pinMode(LED\_NEGACION,OUTPUT);

digitalWrite(LED\_ACCESO,HIGH);

digitalWrite(LED\_NEGACION,HIGH);

delay(500);

digitalWrite(LED\_ACCESO,LOW);

digitalWrite(LED\_NEGACION,LOW);

delay(500);

digitalWrite(LED\_ACCESO,HIGH);

digitalWrite(LED\_NEGACION,HIGH);

delay(500);

digitalWrite(LED\_ACCESO,LOW);

digitalWrite(LED\_NEGACION,HIGH);

mfrc522.PCD\_Init(); // Iniciamos el MFRC522

Serial.println("Control de acceso:");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Inicializacion y testeo \*/

/\* de servo \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Minimo del servo 1º. Maximo del servo 180º.//

//GRADO 1: BARRERA ABIERTA. //

//GRADO 110: BARRERA CERRADA. //

SERVO.attach(9);

SERVO.write(100);

acceso=false;

no\_acceso=false;

Serial.println("Programa cargado correctamente.\n");

Serial.println("Iniciando Loop.\n");

Serial.flush();

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Loop \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop()

{

nuevo\_plaza=false; //Variable para actualizar LCD al actualizar datos

control\_interfaz();

control\_plazas(nuevo\_plaza);

control\_temperatura();

control\_lcd();

control\_iluminacion();

control\_aux();

alarma();

control\_ventilacion();

control\_acceso();

control\_barrera();

t\_acceso=millis();

t\_noacceso=millis();

Serial.flush();

}

## 9.II Código de la Interfaz

**Unit1.h**

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ExtCtrls.hpp>

#include <jpeg.hpp>

#include "CPort.hpp"

#include "CPortCtl.hpp"

#include <Graphics.hpp>

#include <Menus.hpp>

#include <DBCtrls.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TGroupBox \*GroupBox1;

TGroupBox \*GroupBox2;

TGroupBox \*GroupBox3;

TButton \*Button1;

TButton \*Button2;

TButton \*Button3;

TGroupBox \*GroupBox4;

TButton \*Button5;

TComPort \*ComPort1;

TLabel \*Label2;

TLabel \*Label6;

TButton \*Button4;

TShape \*Shape1;

TBevel \*Bevel1;

TBevel \*Bevel2;

TBevel \*Bevel3;

TShape \*Shape2;

TLabel \*Label7;

TLabel \*Label8;

TLabel \*Label9;

TShape \*Shape3;

TLabel \*Label10;

TShape \*Shape4;

TLabel \*Label11;

TShape \*Shape5;

TLabel \*Label12;

TShape \*Shape6;

TLabel \*Label13;

TShape \*Shape7;

TLabel \*Label14;

TShape \*Shape8;

TImage \*Image1;

TImage \*Image2;

TComComboBox \*ComComboBox1;

TLabel \*Label3;

TComComboBox \*ComComboBox2;

TLabel \*Label4;

TComComboBox \*ComComboBox3;

TComComboBox \*ComComboBox4;

TLabel \*Label1;

TImage \*Image3;

TShape \*Shape9;

TShape \*Shape10;

TShape \*Shape11;

TLabel \*Label5;

TLabel \*Label15;

TBevel \*Bevel4;

TLabel \*Label16;

TLabel \*Label17;

TBevel \*Bevel5;

TLabel \*Label18;

TLabel \*Label19;

TBevel \*Bevel6;

TLabel \*Label20;

TLabel \*Label21;

TLabel \*Label22;

TLabel \*Label23;

TLabel \*Label24;

TLabel \*Label25;

TLabel \*Label26;

TLabel \*Label27;

TLabel \*Label28;

TLabel \*Label29;

TMemo \*Memo1;

TTimer \*Timer1;

TTimer \*Timer4;

TTimer \*Timer6;

TTimer \*Timer7;

TTimer \*Timer8;

TTimer \*Timer9;

TTimer \*Timer10;

TLabel \*Label30;

TTimer \*Timer2;

TLabel \*Label31;

TLabel \*Label32;

TLabel \*Label33;

TLabel \*Label34;

TLabel \*Label35;

TLabel \*Label36;

void \_\_fastcall Button5Click(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

void \_\_fastcall Button3Click(TObject \*Sender);

void \_\_fastcall Button4Click(TObject \*Sender);

void \_\_fastcall Timer6Timer(TObject \*Sender);

void \_\_fastcall Timer7Timer(TObject \*Sender);

void \_\_fastcall Timer4Timer(TObject \*Sender);

void \_\_fastcall Timer1Timer(TObject \*Sender);

void \_\_fastcall Timer8Timer(TObject \*Sender);

void \_\_fastcall Timer9Timer(TObject \*Sender);

void \_\_fastcall Timer10Timer(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall Timer2Timer(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**Unit1.cpp**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma link "CPort"

#pragma link "CPortCtl"

#pragma resource "\*.dfm"

TForm1 \*Form1;

bool conexion=false;

bool iluminacion=false;

bool ventilacion=false;

bool alarma=false;

bool emergencia=false;

bool sincro=true;

int ciclo=3;

AnsiString cadena;

bool temp=false;

bool hum=false;

bool usuario1=false;

bool usuario2=false;

/////////////////////////////////////

bool plaza1=false;

bool plaza2=false;

bool plaza3=false;

/\*LOS VALORES ESTAN NORMALIZADO PARA "0" Y "1", SIENDO 0=FALSE Y 1=TRUE

EN ARDUINO

\*/

//---------------------------------------------------------------------------

void sincronizacion()

{

Form1->Cursor=crHourGlass;

ShowMessage("Sincronizando Arduino con Interfaz");

Form1->ComPort1->WriteStr('s');

Sleep(500);

Form1->ComPort1->ReadStr(cadena,4);

if(cadena.SubString(1,1)=='0')

{

iluminacion=false;

Form1->Shape1->Brush->Color=clWhite;

Form1->Shape2->Brush->Color=clRed;

}

if(cadena.SubString(1,1)=='1')

{

iluminacion=true;

Form1->Shape1->Brush->Color=clLime;

Form1->Shape2->Brush->Color=clWhite;

}

if(cadena.SubString(2,1)=='0')

{

ventilacion=false;

Form1->Shape3->Brush->Color=clWhite;

Form1->Shape4->Brush->Color=clRed;

}

if(cadena.SubString(2,1)=='1')

{

ventilacion=true;

Form1->Shape3->Brush->Color=clLime;

Form1->Shape4->Brush->Color=clWhite;

}

if(cadena.SubString(3,1)=='0')

{

alarma=false;

Form1->Shape5->Brush->Color=clWhite;

Form1->Shape6->Brush->Color=clRed;

}

if(cadena.SubString(3,1)=='1')

{

alarma=true;

Form1->Shape5->Brush->Color=clLime;

Form1->Shape6->Brush->Color=clWhite;

}

if(cadena.SubString(4,1)=='0')

{

emergencia=false;

Form1->Shape7->Brush->Color=clWhite;

Form1->Shape8->Brush->Color=clRed;

}

if(cadena.SubString(4,1)=='1')

{

emergencia=true;

Form1->Shape7->Brush->Color=clLime;

Form1->Shape8->Brush->Color=clWhite;

}

Form1->ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

Form1->Memo1->Lines->Add("Bienvenido a la interfaz para gestionar el Parking");

Form1->Memo1->Lines->Add("Sesion abierta con fecha de "+DateToStr(Date())+" a la "+Time());

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormClose(TObject \*Sender, TCloseAction &Action)

{

Form1->Memo1->Lines->Add("Sesión finalizada "+DateToStr(Date())+" a la "+Time());

Form1->Memo1->Lines->Add("");

Form1->Memo1->Lines->SaveToFile("texto.txt");

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button5Click(TObject \*Sender)

{

if(conexion==false)

{

// ComPort1->ShowSetupDialog();

ComPort1->Open();

conexion=true;

Form1->Button5->Caption="Desconectar";

Form1->Image1->Visible=true;

Form1->Image2->Visible=false;

Form1->Memo1->Lines->Add("Conectado al dispositivo Arduino usando el puerto "+ComPort1->Port);

Form1->Memo1->Lines->Add("Iniciando la función para sincronizar la Interfaz con Arduino");

Form1->Memo1->Lines->Add("Por favor, tenga paciencia");

sincronizacion();

// Plazas

Form1->Timer1->Enabled=true;

// Usuarios

Form1->Timer2->Enabled=true;

// Barrera

Form1->Timer4->Enabled=true;

// Temperatura y humedad

Form1->Timer6->Enabled=true;

Form1->Timer7->Enabled=true;

// Tiempo Plazas

Form1->Timer8->Enabled=true;

Form1->Timer9->Enabled=true;

Form1->Timer10->Enabled=true;

return;

}

if(conexion==true)

{

ComPort1->Close();

conexion=false;

Form1->Button5->Caption="Conectar";

Form1->Image1->Visible=false;

Form1->Image2->Visible=true;

Form1->Memo1->Lines->Add("Desconectado");

Form1->Memo1->Lines->Add("");

// Plazas

Form1->Timer1->Enabled=false;

// Usuarios

Form1->Timer2->Enabled=false;

// Barrera

Form1->Timer4->Enabled=false;

// Temperatura y humedad

Form1->Timer6->Enabled=false;

Form1->Timer7->Enabled=false;

// Tiempo Plazas

Form1->Timer8->Enabled=false;

Form1->Timer9->Enabled=false;

Form1->Timer10->Enabled=false;

return;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

// ILUMIINACIÓN

// Valores asignados "0" y "1"

if(iluminacion==false)

{

ComPort1->WriteStr('1');

ComPort1->ClearBuffer(true, true);

iluminacion=true;

Form1->Shape1->Brush->Color=clLime;

Form1->Shape2->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Encendiendo la iluminacion del parking //");

return;

}

if(iluminacion==true)

{

ComPort1->WriteStr('0');

ComPort1->ClearBuffer(true, true);

Form1->Shape2->Brush->Color=clRed;

Form1->Shape1->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Apagando la iluminacion del parking //");

iluminacion=false;

return;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

// VENTILACIÓN

// Valores asignados "2" y "3"

if(ventilacion==false)

{

ComPort1->WriteStr('3');

ComPort1->ClearBuffer(true, true);

Form1->Shape3->Brush->Color=clLime;

Form1->Shape4->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Encendiendo ventilación del parking //");

ventilacion=true;

return;

}

if(ventilacion==true)

{

ComPort1->WriteStr('2');

ComPort1->ClearBuffer(true, true);

Form1->Shape4->Brush->Color=clRed;

Form1->Shape3->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Apagando ventilación del parking //");

ventilacion=false;

return;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

// Alarma

// Valores asignados "4" y "5"

if(alarma==false)

{

ComPort1->WriteStr('5');

ComPort1->ClearBuffer(true, true);

Form1->Shape5->Brush->Color=clLime;

Form1->Shape6->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Alarma Activada //");

alarma=true;

return;

}

if(alarma==true)

{

ComPort1->WriteStr('4');

ComPort1->ClearBuffer(true, true);

Form1->Shape6->Brush->Color=clRed;

Form1->Shape5->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Alarma Desactivada //");

alarma=false;

return;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

// iluminacion de emergencia

// Valores asignados "6" y "7"

if(emergencia==false)

{

ComPort1->WriteStr('7');

ComPort1->ClearBuffer(true, true);

Form1->Shape7->Brush->Color=clLime;

Form1->Shape8->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Encendiendo iluminacion de emergencia //");

emergencia=true;

return;

}

if(emergencia==true)

{

ComPort1->WriteStr('6');

ComPort1->ClearBuffer(true, true);

Form1->Shape8->Brush->Color=clRed;

Form1->Shape7->Brush->Color=clWhite;

Form1->Memo1->Lines->Add("// Apagando iluminacion de emergencia //");

emergencia=false;

return;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer1Timer(TObject \*Sender)

{

//Plaza

ComPort1->WriteStr('p');

Sleep(100);

ComPort1->ReadStr(cadena,5);

Form1->Memo1->Lines->Add("// Actualizando estado de las plazas //");

if(cadena.SubString(0,1)=='p')

{

// Plaza 1 //

if(cadena.SubString(3,1)=='0')

{

Form1->Label27->Color=clLime;

Form1->Shape9->Brush->Color=clLime;

}

if(cadena.SubString(3,1)=='1')

{

Form1->Label27->Color=clRed;

Form1->Shape9->Brush->Color=clRed;

}

// Plaza 2 //

if(cadena.SubString(4,1)=='0')

{

Form1->Label28->Color=clLime;

Form1->Shape10->Brush->Color=clLime;

}

if(cadena.SubString(4,1)=='1')

{

Form1->Label28->Color=clRed;

Form1->Shape10->Brush->Color=clRed;

}

// Plaza 3 //

if(cadena.SubString(5,1)=='0')

{

Form1->Label29->Color=clLime;

Form1->Shape11->Brush->Color=clLime;

}

if(cadena.SubString(5,1)=='1')

{

Form1->Label29->Color=clRed;

Form1->Shape11->Brush->Color=clRed;

}

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer2Timer(TObject \*Sender)

{

// Usuarios

ComPort1->WriteStr('u');

Sleep(100);

ComPort1->ReadStr(cadena,4);

Form1->Memo1->Lines->Add("// Actualizando estado de los usuarios //");

if(cadena.SubString(0,1)=='u')

{

// USUARIO 1

if(cadena.SubString(3,1)=='1')

{

Form1->Label34->Caption="En el parking";

usuario1=true;

}

if(cadena.SubString(3,1)=='0')

{

Form1->Label34->Caption="Fuera del parking";

usuario1=false;

}

// USUARIO 2

if(cadena.SubString(4,1)=='1')

{

Form1->Label35->Caption="En el parking";

usuario2=true;

}

if(cadena.SubString(4,1)=='0')

{

Form1->Label35->Caption="Fuera del parking";

usuario2=false;

}

// USUARIO 3

Form1->Label36->Caption="Usuario Libre";

/\*

if(cadena.SubString(5,1)=='1')

{

Form1->Label36->Caption="En el parking";

usuario2=true;

}

if(cadena.SubString(5,1)=='0')

{

Form1->Label36->Caption="Fuera del parking";

usuario2=false;

}

\*/

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer4Timer(TObject \*Sender)

{

// Barrera

ComPort1->WriteStr('b');

Sleep(100);

ComPort1->ReadStr(cadena,10);

Form1->Memo1->Lines->Add("// Actualizando estado de la barrera //");

if(cadena.SubString(0,1)=='b')

{

Form1->Label19->Caption=cadena.SubString(3,7);

if(cadena.SubString(11,2)=="11")

{

Form1->Memo1->Lines->Add("Usuario 1 está accediendo al parking");

Form1->Memo1->Lines->Add("ID: 73207B25");

usuario1=true;

}

if(cadena.SubString(11,2)=="10")

{

Form1->Memo1->Lines->Add("Usuario 1 está abandonando el parking");

Form1->Memo1->Lines->Add("ID: 73207B25");

usuario1=false;

}

if(cadena.SubString(11,2)=="21")

{

Form1->Memo1->Lines->Add("Usuario 2 está accediendo al parking");

Form1->Memo1->Lines->Add("ID: E62BEE30");

usuario2=true;

}

if(cadena.SubString(11,2)=="20")

{

Form1->Memo1->Lines->Add("Usuario 2 está abandonando el parking");

Form1->Memo1->Lines->Add("ID: E62BEE30");

usuario2=false;

}

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer6Timer(TObject \*Sender)

{

// Temperatura

ComPort1->WriteStr('t');

Sleep(100);

ComPort1->ReadStr(cadena,4);

Form1->Memo1->Lines->Add("// Actualizando temperatura del parking //");

if(cadena.SubString(0,1)=='t')

{

Form1->Label16->Caption=cadena.SubString(3,2)+" ºC";

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer7Timer(TObject \*Sender)

{

// Humedad

ComPort1->WriteStr('h');

Sleep(100);

ComPort1->ReadStr(cadena,4);

Form1->Memo1->Lines->Add("// Actualizando humedad del parking //");

if(cadena.SubString(0,1)=='h')

{

Form1->Label17->Caption=cadena.SubString(3,2)+" %";

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer8Timer(TObject \*Sender)

{

// Plaza 1

ComPort1->WriteStr('m');

Sleep(100);

ComPort1->ReadStr(cadena,6);

if(cadena.SubString(0,2)=="p1")

{

Form1->Label21->Caption=cadena.SubString(4,4)+" minutos";

}

if(cadena.SubString(0,3)=="Li1")

{

Form1->Label21->Caption="Libre";

Form1->Memo1->Lines->Add("1º plaza libre");

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer9Timer(TObject \*Sender)

{

// Plaza 2

ComPort1->WriteStr('n');

Sleep(100);

ComPort1->ReadStr(cadena,6);

if(cadena.SubString(0,2)=="p2")

{

Form1->Label22->Caption=cadena.SubString(4,4)+" minutos";

}

if(cadena.SubString(0,3)=="Li2")

{

Form1->Label22->Caption="Libre";

Form1->Memo1->Lines->Add("2º plaza libre");

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer10Timer(TObject \*Sender)

{

// Plaza 3

ComPort1->WriteStr('g');

Sleep(100);

ComPort1->ReadStr(cadena,6);

if(cadena.SubString(0,2)=="p3")

{

Form1->Label23->Caption=cadena.SubString(4,4)+" minutos";

}

if(cadena.SubString(0,3)=="Li3")

{

Form1->Label23->Caption="Libre";

Form1->Memo1->Lines->Add("3º plaza libre");

}

ComPort1->ClearBuffer(true, true);

}

//---------------------------------------------------------------------------

## 9. III Documento digital

Todo este documento se encuentra en formato digital para su visualización en cualquier momento.

Carpeta en Dropbox con los documentos 🡪 https://goo.gl/3xLDEO

* Documento completo 🡪 https://goo.gl/Zu1F8W
* Plano 1/11 🡪 https://goo.gl/If3qKg
* Plano 2/11 🡪 https://goo.gl/FqHnI0
* Plano 3/11 🡪 https://goo.gl/XrJMfO
* Plano 4/11 🡪 https://goo.gl/MOJnX9
* Plano 5/11 🡪 https://goo.gl/eQJDkW
* Plano 6/11 🡪 https://goo.gl/VOtRaj
* Plano 7/11 🡪 https://goo.gl/blmjFb
* Plano 8/11 🡪 https://goo.gl/f8HTrM
* Plano 9/11 🡪 https://goo.gl/6JDbHC
* Plano 10/11 🡪 https://goo.gl/74qhsq
* Plano 11/11 🡪 https://goo.gl/LQ6AfH
* Plano de conexionado del proyecto 🡪 https://goo.gl/weWNvk