PARKING:
DISEÑO DEL SISTEMA

Diego Ramírez Fuente A-109

Rodrigo Martín A-109

Laura Lucía Hernández A-104

Sonia Carrero A-104

Esther Ampudia A-104

Escuela Técnica Superior de Ingeniería y Diseño Industrial
Universidad Politécnica de Madrid

Curso 2020-2021

Introducción

El objetivo de la máquina será detectar la entrada de los coches en un parking, en el que quedarán identificados todos los coches por su matrícula. Este aparcamiento tendrá un aforo máximo y, cuando este sea alcanzado, no se permitirá la entrada de más coches hasta que alguno de los que se encuentren estacionados en el parking salga del mismo.

Este proceso comenzará cuando un coche se aproxime a la barrera del parking, inicialmente bajada y con una luz indicadora de color rojo para no permitir la entrada de ningún vehículo sin ser registrado. Los sensores de movimiento instalados en el sistema de la barrera detectarán la llegada de un coche y, si el aforo no está completo, el conductor pulsará un botón que activará un programa que le pedirá al usuario introducir su matrícula. Una vez introducida la matrícula, la barrera se levantará mediante unos servomotores instalados en ella y la luz cambiará a color verde para indicar al conductor que ya puede entrar en el parking. Además, el contador de aforo sumará una unidad.

El proceso terminará cuando el aforo esté completo ya que, cuando otro conductor se aproxime a la barrera, en lugar de pedir al usuario que introduzca su matrícula, el sistema le informará de que no queda sitio en el aparcamiento. Si algún vehículo estacionado dentro del parking decidiese salir, volverá a accionar el botón y la barrera volverá a funcionar. Después de esto, quedará libre un sitio en el aparcamiento y se permitirá entrar a otro vehículo en su lugar, sin superar nunca el aforo máximo.

La barrera dispone de un sensor de movimiento que evitará que la barrera se baje mientras haya un vehículo cruzando la entrada.

Estructura general de la aplicación que se ejecuta en el PC (Visual Studio)

Gran parte del programa funcionará de manera invisible para el usuario, pues su única función será introducir la matrícula de su coche al entrar en el parking.

El programa funciona mediante un contador que suma una unidad cuando un coche entra y la resta cuando un coche sale. El programa permitirá el acceso a los vehículos cuando el contador sea menor que el aforo máximo (fijado previamente por el programador), es decir, mientras que el aforo no esté completo.

Si el aforo no está completo, cuando el conductor pulse el botón de la barrera, el programa le pedirá al usuario, mediante un mensaje impreso en pantalla, que introduzca su matrícula. Cuando la matrícula sea incorrecta, se le pedirá de nuevo hasta que la digite correctamente. Si la matrícula es correcta, la barrera se levantará y se le indicará al conductor la cantidad de espacios restantes dentro del parking.

Cuando el aforo esté completo se le indicará al usuario, mediante un mensaje impreso en pantalla, que no quedan espacios restantes dentro del aparcamiento y que vuelva más tarde.

CÓDIGO

```
#include<stdio.h>
#include<string.h>
#define aforo 40
struct parking {
	int plaza;
	char matricula[8];
};
int main() {
	struct parking plazas[aforo];
	int contador = 0, saliendo_i = 0;
	char continuar = { 'c' }, opcion;
	char saliendo[8];
	char matricula[8];
	for (int i = 0; i < 40; i++) {
		plazas[i].plaza = 0;
		strcpy_s(plazas[i].matricula, "libreee");
	}
	printf("%s\n", plazas[0].matricula);
	while (continuar == 'c') {
```

```
            printf("Escribe e para entrar s para salir.\n");
            scanf_s("%c", &opcion);
            printf("%d\n", contador);
            printf("%c\n", opcion);
            printf("%d\n", aforo);
            if ((opcion == 'e') && (contador < aforo)) {
                  printf("Escribe tu matricula: ");
                  if (saliendo_i < contador) {
                        scanf_s("%s", matricula);
                        strcpy_s(plazas[contador].matricula, matricula);
                        plazas[saliendo_i].plaza = saliendo_i;
                  }
                  else {
                        scanf_s("%s", matricula);
                        strcpy_s(plazas[contador].matricula, matricula);
                        plazas[contador].plaza = contador;
                        contador++;
                  }
            }
            if ((opcion == 's') && (contador > 0)) {
                  printf("Escribe tu matricula: ");
                  gets_s(saliendo);
                  for (int i = 0; i <= contador; i++) {
                        if (saliendo == plazas[i].matricula) {
                              saliendo_i = plazas[i].plaza;
                        }
                  }
                  strcpy_s(plazas[saliendo_i].matricula, "libreee");
                  if (plazas[saliendo_i].plaza == contador) {
                        contador--;
                  }
            }
            printf("Escribe 'c' para continuar 's' para terminar: ");
            scanf_s("%c", &continuar);
      }
}
```

Estructura general de la aplicación que se ejecuta en la plataforma hardware (Arduino)

La parte más importante del proyecto se llevará a cabo con Arduino. Para comenzar, el usuario pulsará un botón para iniciar el programa de Visual Studio tanto a la entrada como a la salida. El mecanismo de la barrera funcionará mediante servomotores que harán que la barrera se levante y se cierre. Además, en la barrera hay incorporado un sensor que detectará a los coches mientras pasen por debajo de la barrera, para evitar que esta se baje y los vehículos sean dañados. Mientras que la barrera esté abierta, los leds de la barrera emitirán una luz verde, mientras que cuando la barrera no permita el paso a los coches, esta luz será de color rojo.

CÓDIGO

```
#include<Servo.h>
#define Echo 3
#define Trig 2
#define Vel_Son 34000.0
#define LED_rojo 13
#define LED_verde 12
#define boton_entrada 8
#define boton_salida 9
#define aforo 20

Servo servomotor;
float distancia, tiempo;
int Be, Bs, coches = 0;

void setup() {

    Serial.begin(9600);

    servomotor.attach(11);
    servomotor.write(0);
    pinMode(boton_entrada, INPUT);
    pinMode(boton_salida, INPUT);
    pinMode(Echo, INPUT);
    pinMode(Trig, OUTPUT);
    pinMode(LED_rojo, OUTPUT);
    pinMode(LED_verde, OUTPUT);

}

void loop() {
```

```
inicio();
Be = digitalRead(boton_entrada);
Bs = digitalRead(boton_salida);

digitalWrite(LED_rojo, HIGH);
digitalWrite(LED_verde, LOW);

tiempo = pulseIn(Echo, HIGH);
distancia = tiempo * 0.000001 * Vel_Son / 2.0;
Serial.print(distancia);
Serial.print(" cm\n");
Serial.print(coches);
Serial.print(" coches\n");
delay(500);

if (Be == HIGH) {
    delay(100);

    if (Be == HIGH) {

        if (coches == aforo) {
            Serial.print("Aforo completo, por favor dé la vuelta\n");
        }

        else if (coches <= aforo) {
            do {
                servomotor.write(90);
                digitalWrite(LED_rojo, LOW);
                digitalWrite(LED_verde, HIGH);
                inicio();
                tiempo = pulseIn(Echo, HIGH);
                distancia = tiempo * 0.000001 * Vel_Son / 2.0;
                Serial.print(distancia);
                Serial.print(" cm\n");
                delay(500);
            } while (distancia <= 10);

            coches++;

            servomotor.write(0);
            digitalWrite(LED_rojo, HIGH);
            digitalWrite(LED_verde, LOW);
            delay(500);
        }
    }
}
if (Bs == HIGH) {
    delay(100);

    if (Bs == HIGH) {
        do {
            servomotor.write(90);
            digitalWrite(LED_rojo, LOW);
```

```
                digitalWrite(LED_verde, HIGH);
                inicio();
                tiempo = pulseIn(Echo, HIGH);
                distancia = tiempo * 0.000001 * Vel_Son / 2.0;
                Serial.print(distancia);
                Serial.print(" cm\n");
                delay(500);
            } while (distancia <= 10);

            coches--;

            servomotor.write(0);
            digitalWrite(LED_rojo, HIGH);
            digitalWrite(LED_verde, LOW);
            delay(500);
        }
    }

    if (distancia > 10) {
        servomotor.write(0);
        digitalWrite(LED_rojo, HIGH);
        digitalWrite(LED_verde, LOW);
        delay(500);
    }
}

void inicio() {

    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);
}
```

DISEÑO GRÁFICO