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, es decir, mientras que el aforo no esté completo. El aforo máximo lo establecerá el usuario, con el fin de crear un programa válido para diversos parkings.

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.

Dependiendo del tiempo que el coche permanezca estacionado, se calculará un precio acorde con las tarifas establecidas. El precio a pagar por el usuario se almacenará en un fichero junto con la matrícula del vehículo.

CÓDIGO

```
#define TARIFA 0.06
#define DIM 8
#define MAX_BUFFER 200
#define MAX_INTENTOS_READ 4
```

```
#define MS_ENTRE_INTENTOS 200
#define SI 1
#define NO 0

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
#include <windows.h>
#include <string.h>
#include <conio.h>
#include "SerialClass/SerialClass.h"
#include <Windows.h>

typedef struct {
	int plaza;
	float tiempo;
	char matricula[DIM];
}COCHE;

void registro(char matricula[], int coches);
void entrada(int coches, COCHE*);
int salida(int coches, COCHE*);
void tarifa(COCHE*, int coches);

// Funciones prototipo
//int menu_principal(void);
void configura(void);

void Talk_with_Arduino(Serial* Arduino);
void Send_to_hw(Serial*, char*);
int Receive_from_hw(Serial* Arduino, char* BufferEntrada);
int Send_and_Receive(Serial* Arduino, const char* msg_out, int
valor_out, char* msg_in, int* valor_in);
void monitorizar_aforo(Serial*);

int main() {

	Serial* Arduino;
	char puerto[] = "COM3"; //Puerto serie al que está conectado
Arduino
	char BufferEntrada[MAX_BUFFER];
	int bytesReceive, numero_recibido;

	// Tareas de configuración y carga
	configura();
	Arduino = new Serial((char*)puerto);  // Establece la conexión
con Arduino
```

```
	printf("Introduce el numero de plazas de tu parking: ");
	int n;
	scanf_s("%d", &n);

	bytesReceive = Send_and_Receive(Arduino, "GET_AFORO_MAX", n,
BufferEntrada, &numero_recibido);
	if (bytesReceive == 0)
		printf("No se ha recibido respuesta al mensaje
enviado\n");
	else
		printf("Mensaje recibido %s %d\n", BufferEntrada,
numero_recibido);

	COCHE* plazas;
	plazas = (COCHE*)calloc(n, sizeof(COCHE));

	int  coches = 0, mat, error = 0;
	float precio = 0, importe_introducido = 0;
	char opc;

	do {

		printf("\t\t\tBIENVENIDO, PRESIONE EL BOTON PARA
CONTINUAR\n");

		do {
			bytesReceive = Send_and_Receive(Arduino,
"Bienvenido", coches, BufferEntrada, &numero_recibido);
			if (bytesReceive != 0)
				printf("Mensaje recibido %s %d\n",
BufferEntrada, numero_recibido);

		} while (numero_recibido != 1 && numero_recibido != 2);


		switch (numero_recibido) {

		case 1:

			if (coches == n) {
				printf("\t\t\tAFORO COMPLETO, POR FAVOR DE LA
VUELTA\n\n\n");
				Sleep(2000);
				system("cls");
			}
			else {
				entrada(coches, plazas);

				do {
```

```
                              bytesReceive = Send_and_Receive(Arduino,
"Registro_completado_e", 1, BufferEntrada, &numero_recibido);
                              if (bytesReceive != 0)
                                    printf("Mensaje recibido %s %d\n",
BufferEntrada, numero_recibido);

                        } while (numero_recibido != 10);

                        Sleep(2000);

                        system("cls");

                        coches++;
                  }
                  break;


            case 2:

                  if (coches == 0) {

                        printf("No hay coches dentro");
                        error++;

                        Sleep(2000);
                        system("cls");
                  }

                  if (coches > 0) {

                        mat = salida(coches, plazas);

                        if (mat == 1) {
                              do {

                                    bytesReceive =
Send_and_Receive(Arduino, "Registro_completado_s", 1, BufferEntrada,
&numero_recibido);
                                    if (bytesReceive != 0)
                                          printf("Mensaje recibido %s
%d\n", BufferEntrada, numero_recibido);

                              } while (numero_recibido != 11);
                              coches--;
                        }

                        else
                              error++;
```

```
                    Sleep(2000);

                    system("cls");
               }

               break;

          default:
               break;
          }

          printf("Presione c para continuar: ");
          scanf_s(" %c", &opc);

     } while ((opc == 'c' || opc == 'C') && error == 0);

     if (error > 0)
          printf("\nSe ha producido un error");

     printf("\nHa salido del programa\n\n");
}

void entrada(int coches, COCHE* plazas) {

     printf("Introduzca su matricula: ");
     getchar();
     gets_s(plazas[coches].matricula);

     plazas[coches].tiempo = clock();

     registro(plazas[coches].matricula, coches);

     printf("\n\nOk, puede entrar\n\n");

}

void registro(char matricula_e[], int coches) {

     FILE* pf;

     errno_t err; // Open for read (will fail if file "registro.txt"
doesn't exist)

     err = fopen_s(&pf, "registro.txt", "a+");

     if (err == 0) {

          printf("El archivo registro.txt esta abierto\n");
```

```
            fprintf_s(pf, "%s\n", matricula_e);
      }

      else
            printf("El archivo registro.txt NO está abierto\n");

      if (fclose(pf) == NULL)
            printf("\n Archivo cerrado correctamente");
      else
            printf("\n Error en el cierre del archivo");
}

int salida(int coches, COCHE* plazas) {

      int i = 0, mat;
      char matricula_sal[DIM];

      printf("Introduzca su matricula: ");
      getchar();
      gets_s(matricula_sal);

      do {  //comprobar si la matricula coincide con alguna del
registro strcmp(matricula_sal, plazas[i].matricula) == 0

            if (strcmp(matricula_sal, plazas[i].matricula) == 0) {
//si coincide:

                  tarifa(plazas, i);
                  printf("\nBuen viaje\n");
            }

            else
                  i++;

      } while ((strcmp(matricula_sal, plazas[i].matricula) != 0) && i
< coches);

      mat = 1;

      if (i = coches && (strcmp(matricula_sal, plazas[i].matricula)
!= 0)) {
            printf("Matricula introducida no encontrada.\n");
            mat = 0;
      }
      return mat;
}

void tarifa(COCHE* plazas, int i) {
```

```
	float precio, tiempo, imp_i = 0.00;

	tiempo = (float)(clock() - plazas[i].tiempo) / 60000;

	precio = tiempo * TARIFA;

	do {

		printf("\nLe falta introducir %.2f EUROS, introduzcalos
por favor: ", (precio - imp_i));
		scanf_s("%f", &imp_i);

	} while (precio > (imp_i + 0.009));

	if ((precio + 0.009) < imp_i)
		printf("\nSe le han devuelto %.2f EUROS, recojalos por
favor\n", (imp_i - precio));

}

void configura(void)
{
	// Establece juego de caracteres castellano
	// Para que funcione hay que partir de un proyecto vacío
	// No utilice la plantilla Aplicación de consola C++
	setlocale(LC_ALL, "spanish");
}

// Ejemplo de función de intercambio de datos con Arduino
void Talk_with_Arduino(Serial* Arduino)
{
	//char BufferSalida[MAX_BUFFER];
	char BufferEntrada[MAX_BUFFER];
	int bytesReceive, numero_recibido;

	if (Arduino->IsConnected()) // Si hay conexión con Arduino
	{

		// Para enviar un mensaje y obtener una respuesta se
utiliza la función Send_and_Receive
		// El mensaje está formado por un texto y un entero
		// El mensaje que se recibe está formado también por un
texto y un entero.
		// Parámetros de la función:
		// El primero es la referencia a Arduino
		// El segundo es el mensaje que se desea enviar
		// El tercero es un entero que complementa al mensaje que
se desea enviar
```

```
		// El cuarto es el vector de char donde se recibe la
respuesta
		// El quinto es la referencia donde se recibe el entero
de la respuesta
		// IMPORTANTE: El mensaje de respuesta que emite Arduino
debe incluir un espacio en blanco separando respuesta de valor
		// La función devuelve un entero con los bytes recibidos.
Si es cero no se ha recibido nada.

		bytesReceive = Send_and_Receive(Arduino, "GET_AFORO_MAX",
-1, BufferEntrada, &numero_recibido);
		if (bytesReceive == 0)
			printf("No se ha recibido respuesta al mensaje
enviado\n");
		else
			printf("Mensaje recibido %s %d\n", BufferEntrada,
numero_recibido);
	}
	else
		printf("La comunicación con la plataforma hardware no es
posible en este momento\n"); // Req 3
}

// Protocolo de intercambio mensajes entre Pc y platforma hardware
// Envío Mensaje valor
// Recibe Mensaje valor
		// IMPORTANTE: El mensaje de respuesta que emite Arduino
debe incluir un espacio en blanco separando respuesta de valor
// Retorna bytes de la respuesta (0 si no hay respuesta)
int Send_and_Receive(Serial* Arduino, const char* msg_out, int
valor_out, char* msg_in, int* valor_in)
{
	char BufferSalida[MAX_BUFFER];
	char BufferEntrada[MAX_BUFFER];
	char* ptr;
	int bytesReceive;

	sprintf_s(BufferSalida, "%s\n%d\n", msg_out, valor_out);
	Send_to_hw(Arduino, BufferSalida);
	bytesReceive = Receive_from_hw(Arduino, BufferEntrada);
	if (bytesReceive != 0)
	{
		ptr = strpbrk(BufferEntrada, " ");
		if (ptr == NULL)
			*valor_in = -1;
		else
		{
			*valor_in = atoi(ptr);
			*ptr = '\0';
```

```
            }
            strcpy_s(msg_in, MAX_BUFFER, BufferEntrada);
      }
      return bytesReceive;
}


// Envía un mensaje a la plataforma hardware
void Send_to_hw(Serial* Arduino, char* BufferSalida)
{
      Arduino->WriteData(BufferSalida, strlen(BufferSalida));
}

//Recibe (si existe) un mensaje de la plataforma hardware
//Realiza MAX_INTENTOS_READ para evitar mensajes recortados
int Receive_from_hw(Serial* Arduino, char* BufferEntrada)
{
      int bytesRecibidos, bytesTotales = 0;
      int intentos_lectura = 0;
      char cadena[MAX_BUFFER];

      BufferEntrada[0] = '\0';
      while (intentos_lectura < MAX_INTENTOS_READ)
      {
            cadena[0] = '\0';
            bytesRecibidos = Arduino->ReadData(cadena, sizeof(char) *
(MAX_BUFFER - 1));
            if (bytesRecibidos != -1)
            {
                  cadena[bytesRecibidos] = '\0';
                  strcat_s(BufferEntrada, MAX_BUFFER, cadena);
                  bytesTotales += bytesRecibidos;
            }
            intentos_lectura++;
            Sleep(MS_ENTRE_INTENTOS);
      }
      return bytesTotales;
                                    }
```

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

Servo servomotor;
float distancia, tiempo;
int Be, Bs, coches = 1, aforo;

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();

    if (Serial.available() > 0)  // Si hay mensajes procedentes del
PC
```

```
        aforo = procesar_aforo();

    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;
    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) {

                int numero;
                String cadena = Serial.readStringUntil('\n'); // Lee
mensaje
                //Serial.print(cadena);

                String valor = Serial.readStringUntil('\n');  // Lee
valor
                numero = valor.toInt();  // Transforma valor a
entero

                if (cadena.equals("Bienvenido")) {
                    Serial.println("Proceder_al_registro ");
                    Serial.println(1);
                }

                delay(5000);

                cadena = Serial.readStringUntil('\n'); // Lee
mensaje
                //Serial.print(cadena);

                valor = Serial.readStringUntil('\n');  // Lee valor
                numero = valor.toInt();  // Transforma valor a
entero
```

```
                if (cadena.equals("Registro_completado_e")) {
                    do {
                        Serial.println("Coche_en_barrera ");
                        Serial.println(distancia);
                        servomotor.write(90);
                        digitalWrite(LED_rojo, LOW);
                        digitalWrite(LED_verde, HIGH);
                        inicio();
                        tiempo = pulseIn(Echo, HIGH);
                        distancia = tiempo * 0.000001 * Vel_Son /
2.0;
                        delay(500);
                    } while (distancia <= 10);

                    Serial.println("El_coche_ha_entrado ");
                    Serial.println(10);

                    //coches++;

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

            }
        }
    }

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

        if (Bs == HIGH) {

            if (coches > 0) {

                int numero;
                String cadena = Serial.readStringUntil('\n'); // Lee
mensaje
                //Serial.print(cadena);

                String valor = Serial.readStringUntil('\n');  // Lee
valor
                numero = valor.toInt();  // Transforma valor a
entero

                if (cadena.equals("Bienvenido")) {
                    Serial.println("Proceder_al_registro ");
                    Serial.println(2);
```

```
                }

                delay(10000);

                cadena = Serial.readStringUntil('\n'); // Lee
mensaje
                //Serial.print(cadena);

                valor = Serial.readStringUntil('\n');  // Lee valor
                numero = valor.toInt();  // Transforma valor a
entero

                if (cadena.equals("Registro_completado_s")) {
                    do {
                        Serial.println("Coche_en_barrera ");
                        Serial.println(distancia);
                        servomotor.write(90);
                        digitalWrite(LED_rojo, LOW);
                        digitalWrite(LED_verde, HIGH);
                        inicio();
                        tiempo = pulseIn(Echo, HIGH);
                        distancia = tiempo * 0.000001 * Vel_Son /
2.0;
                        delay(500);
                    } while (distancia <= 10);

                    Serial.println("El_coche_ha_salido ");
                    Serial.println(11);

                    //coches--;

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

void inicio(void) {

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

```
int procesar_aforo(void) {
    int numero;
    String cadena = Serial.readStringUntil('\n'); // Lee mensaje
    //Serial.print(cadena);

    String valor = Serial.readStringUntil('\n');  // Lee valor
    numero = valor.toInt();  // Transforma valor a entero

    if (cadena.equals("GET_AFORO_MAX")) // Entre las comillas se
pone el texto del mensaje que se espera
    {

        Serial.println("Aforo_fijado ");
        Serial.println(numero);
        return numero;
    }
}
```

DISEÑO GRÁFICO