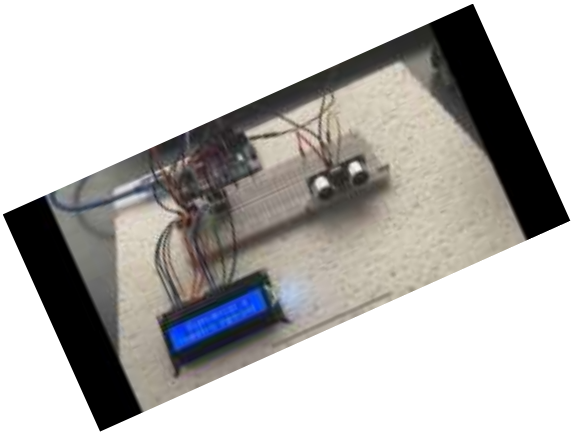
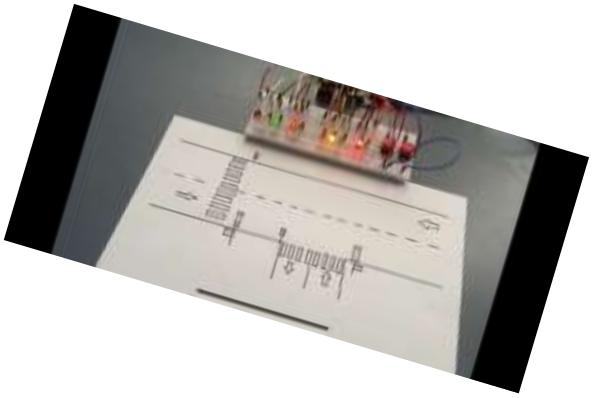
PARKING PÚBLICO





GRUPO: A-104

ALUMNOS:

VICTOR ARAGONÉS PIMENTEL

PABLO MORENO MARTÍN

IGNACIO DE BENITO GÓMEZ

- **Descripción del trabajo:**

Nuestro trabajo consiste en el control de un parking público mediante una serie de funcionalidades que hemos implementado.
Estas funcionalidades que hemos implementado son las siguientes:

1. Gestión del parking: hemos desarrollado un programa que nos permita acceder como empleado de nuestro parking a un sistema de control de matrículas para gestionar tanto el aforo de nuestro parking como para informar a los usuarios que salen cuál es el precio que tienen que pagar por su estacionamiento.

2. Guiado: se ha implementado también un sistema de guiado del usuario que le indica si entra al parking si este se encuentra completo o por el contrario le dirige hacia la plaza más cercana. Así como un control de salidas para controlar que plazas se quedan libres.

3. Aparcamiento: hemos desarrollado un sistema de ayuda al estacionamiento que mediante sensores ultrasonidos se evalúa la posición del vehículo cuando esta aparcando y le informa mediante leds y sonidos cuanto le queda para llegar a tocar con la pared.

4. Control de acceso a la vía pública con semáforos: por último se ha establecido el control de acceso a la vía pública mediante la elaboración de un patrón de funcionamiento de los semáforos tanto de la vía como de salida del parking. De la misma manera se han añadido semáforos para los peatones tanto de la vía como del parking así como pulsadores para que los peatones puedan pulsarlo cuando deseen cruzar el paso de peatones.

Una vez descritas nuestras funcionalidades cabe destacar que hemos desarrollado de forma paralela tanto el hardware con Arduino como el software con el código en Visual Studio. La única diferencia es que debido a que como es obvio no contamos con suficientes sensores o actuadores como realmente podrían hacer falta para controlar todos los estacionamientos de un parking, nuestro software es algo más completo y trae mayores ventajas y funcionalidades que nuestro hardware. Por ello presentaremos a continuación los códigos tanto en Visual Studio como en Arduino.

- # Gestion del parking

1. Código en Visual Studio:

En esta parte del código, comenzaremos la experiencia del Parking desde el principio, con nuestro operario “Bernardo”. Se saludará al operario mientras se le pide una contraseña para la cual dispone de 3 intentos en total. Si acierta, le pediremos el día de hoy, para guardar un registro con las matriculas de coche en cada uno de los días (el titulo del fichero será el día de la apertura). Una vez creado, daremos opción a entrada y salida de los coches. En la entrada, los coches meterán su matrícula, que será guardada en el fichero anterior. En la salida, los coches dejarán constancia de las horas que han pasado en el parking, y su matrícula, ya que la primera letra de la matricula será clave en su antigüedad y, por lo tanto, un impuesto superior respecto a los coches más modernos. Además tendremos en cuenta si es día de diario o fin de semana. También contaremos las personas que entren y salgan del parking, ya que, si las plazas se completan, no dejará meter más coches.
En esta parte hemos utilizado de forma completa los contenidos abordados en la teoría de una manera clara, utilizando estructuras, asignación dinámica de memoria, ficheros y funciones.

El código en visual es:

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#define DIM 200   //plazas del parking

struct coche
{
	char letra_matricula[4];
	int num_matricula;

};

void meter_matricula(FILE* fp, char dia_actual[], struct coche* cars);
float seleccionarprecio(char dia_actual[], struct coche* cars);
int preciobase(char dia_actual[]);

int main()
{
	struct coche* cars;
	FILE* fp = 0;
	int* p, correcto, intento, i, j = 0, flag = 0, opc, gente = 0, n, numeros;
	char pass[10], dia_actual[30];
	const char contra[] = "A";
	float precio_total = 0;
	//contraseña para inicializar el parking
	{
		intento = 1;
		correcto = 0;
```

```
		printf("Hola Bernardo, buenos dias.\nIntroduce la password de
operario(tiene tres intentos)\n");
		gets_s(pass);
		if (strcmp(pass, contra) == 0) correcto = 1;
		while ((correcto == 0) && (intento < 3))
		{
			intento++;
			printf("Contrasena incorrecta. %i intento: \n", intento);
			gets_s(pass);
			if (strcmp(pass, contra) == 0) correcto = 1;
		}
		if (correcto == 0)
		{
			printf("Has fallado los tres intentos, llamame al telefono
666 666 666\n\n");
		}
		else
		{
			printf("Clave generada correctamente!\n\n ");

			printf("Introduzca el numero de coches que podrian pasar al
parking hoy\n");
			scanf_s("%d", &n);
			cars = (coche*)malloc(sizeof(coche) * n);

			printf("Hola, bienvenidos al parking, introduzca el dia de
hoy(diasemana-dia-mes-año)\nACLARACION: en diasemana poner:\nlunes->1\nmartes-
>2\nmiercoles->3\njueves->\nviernes->5\nsabado->6\ndomingo->7\n");
			getchar();
			gets_s(dia_actual);

			//empieza la jornada laboral, el parking tiene 500 plazas.
Serán guardadas las matriculas en un fichero. Esto es un parking moderno, y cada
coche, según su año de matriculación(que obtendremos con las matriculas)
			//y el numero de horas que esté en el parking, tendra un
precio u otro.

			//tendremos que elegir, metemos coche o sacamos coche
			do {
				printf("Ha llegado un coche, o se va un coche?\n1)HA
LLEGADO\n2)SE VA\n3)SALIR(solo operarios)\n");
				scanf_s("%d", &opc);
				switch (opc)
				{
				case 1: //ha llegado->guardar matricula en un fichero
				{
					if (gente <= DIM)  //si queda espacio en el
parking, meteremos un coche
					{
						printf("Hola, cliente. Introduzca su
matricula NNNN LLL (donde las N son numeros y las L letras(mayusculas))\n");
						do
						{
							getchar();
							printf("Primero los
numeros\nIMPORTANTE: 4 CIFRAS\n");
							scanf_s("%d", &cars-
>num_matricula);

						} while (cars->num_matricula < 999 ||
cars->num_matricula>10000); //para que el numero sea de cuatro cifras
```

```
                                        getchar();
                                        printf("\nAhora las letras\nIMPORTANTE:
3 CIFRAS\n");
                                        gets_s(cars->letra_matricula);
                                        meter_matricula(fp, dia_actual, cars);
//ahora que tenemos la matricula, la guardaremos en nuestro fichero de antes
                                        gente++;  //indica que se ha metido una
persona
                                }
                                else   //si no queda espacio en el parking,
diremos al cliente que esta lleno
                                {
                                        printf("EL PARKING ESTA LLENO");
                                }
                                break;
                        }
                        case 2: //se va-> tiene que pagar
                        {

                                precio_total = seleccionarprecio(dia_actual,
cars);
                                printf("\nPRECIO A PAGAR:%f\n", precio_total);
                                gente--;  //indica que se ha ido una persona
                                break;
                        }
                        case 3:
                                printf("Adios Bernardo! Tenga un buen dia\n");
                                break;
                        }


                } while (opc != 3);
        }
    }
}


void meter_matricula(FILE* fp, char dia_actual[], struct coche* cars)
{
    fopen_s(&fp, dia_actual, "at");
    if (fp == NULL)
    {
        printf("ERROR 02\n");
    }
    else
    {
        printf("SE HA ABIERTO CORRECTAMENTE\n");
        fprintf(fp, "%d %s\n", cars->num_matricula, cars->letra_matricula,
3);
    }
    fclose(fp);
}

float seleccionarprecio(char dia_actual[], struct coche* cars)  //falta
determinar precios
{
    float horas = 0, precio = 0;
    int precio_bas = 0;
    printf("Esperemos que haya disfrutado la estancia, cuantas horas ha
estado?\n");  //preguntamos las horas
    scanf_s("%d", &horas);
    //de lunes a jueves, el precio base sera: 3euros/h
```

```
	//viernes,sabados y domingos, el precio base sera: 5euros/h
	precio_bas = preciobase(dia_actual);
	printf("\nprecio base= %d, Ahora tocara añadirle el impuesto por
antiguedad del coche\n", precio_bas);

	//ahora como el vehiculo sale, necesitamos otra vez el numero de matricula

	printf("\ndigame SOLO las letras de su matricula:\n");
	gets_s(cars->letra_matricula);
	getchar();
	printf("%c", cars->letra_matricula[0]);

	if (cars->letra_matricula[0] == 'A' || cars->letra_matricula[0] == 'B')
//--> hasta 2002
	{
		precio = (float)precio_bas + (precio_bas * 0.5);  //50%
	}
	else if (cars->letra_matricula[0] == 'C' || cars->letra_matricula[0] ==
'D')  //--> hasta 2006
	{
		precio = (float)precio_bas + (precio_bas * 0.4);  //40%
	}
	else if (cars->letra_matricula[0] == 'E' || cars->letra_matricula[0] ==
'F' || cars->letra_matricula[0] == 'G') //--> hasta 2010
	{
		precio = (float)precio_bas + (precio_bas * 0.3);  //30%
	}
	else if (cars->letra_matricula[0] == 'H' || cars->letra_matricula[0] ==
'I' || cars->letra_matricula[0] == 'J') //--> hasta 2017
	{
		precio = (float)precio_bas + (precio_bas * 0.2);  //20%
	}
	else if (cars->letra_matricula[0] == 'K' || cars->letra_matricula[0] ==
'L') //--> hasta 2021
	{
		precio = (float)precio_bas;  //0%
	}
	return precio;
	printf("%f\n", precio);
}


int preciobase(char dia_actual[])
{
	int precio_base;

	if (dia_actual[0] >= 49 && dia_actual[0] <= 52)  //mirando en el codigo
ascii, estos seran los numeros del 1 al 4
	{
		precio_base = 3;
	}

	else if (dia_actual[0] >= 53 && dia_actual[0] <= 56) //mirando en el
codigo ascii, estos seran los numeros del 5 al 7
	{
		precio_base = 5;
	}
	return precio_base;
	}
```

2. Código en Arduino:

Al tratarse de la entrada del Parking, en este apartado abarcaremos la barrera inicial a cualquier parking. Su funcionamiento será muy sencillo. Utilizaremos un servomotor, un infrarrojo PIR, y una LED. Todo esto estará conectado en una placa ARDUINO UNO, totalmente capaz de realizar las etapas. Como se muestra en el video mandado, al pasar un coche, el sensor PIR lo detectará y mandará una señal al servo (que girará 90 grados), y a la LED. Gracias al delay de 7000ms del código le daremos tiempo al coche a que pase, para su posterior bajada.

El código:

```
#include <Servo.h>

int PIR=3;

int LED=6;

Servo barrera;

void setup() {

barrera.attach(2);

pinMode(PIR,INPUT);

pinMode(LED,OUTPUT);

}

void loop() {


int entrada=digitalRead(PIR);


 if(entrada==HIGH)

 {

  barrera.write(90);

  digitalWrite(LED,HIGH);

  delay(7000);

 }
```

```
else

{

  barrera.write(0);

    digitalWrite(LED,LOW);

}

}
```

- **Guiado**

1. Código en Visual Studio:

   Como se mostrará a continuación en nuestro código dentro de esta parte lo que hemos hecho es:

   Con asignación dinámica de memoria hemos creado un puntero llamado psensor que va a almacenar el estado (“Ocupado-1”,”Libre-0”) de cada uno de los estacionamientos de nuestro parking.
   Para facilitar la inicialización del estado de nuestro parking hemos establecido el tamaño a algo reducido como es 5 mediante #define.
   Hemos creado una función llamada situación() que lo que nos hace es pedir al usuario que nos indique el estado inicial de cada sensor para poder empezar nuestro dia de trabajo en el parking.
   Tenemos otra función llamada dirección() que lo que nos hace es recorres nuestro puntero comprobando el estado de nuestros sensores y una vez detecta que uno de ellos esta en “0”,es decir libre, redirige a nuestro usuario a esa plaza que es gracias al bucle que utilizamos la primera plaza disponible más cercana.
   Por último tenemos la función lleno() que simplemente evalúa si el parking esta completo y si es así asigna un 1 a nuestro variable completo mediante su puntero.

```
#include<stdio.h>
#include<windows.h>
#include<malloc.h>
#define TAM 5

void situacion(int*);
void direccion(int*);
void lleno(int*,int*);

int main()
{
	int
*psensor,cocheentrada=0,cierre=0,completo=0,*pcompleto,cochesalida=0,plazasalida;
```

```
	psensor = (int*)malloc(TAM * sizeof(int));
	if (psensor == NULL)
	{
		printf("No hay memoria suficiente\n\n");
	}
	else
	{
		printf("Memoria asignada correctamente\n\n");
	}
	pcompleto = &completo;
	situacion(psensor);
	lleno(psensor,pcompleto);
	if (*pcompleto == 0)
	{
		do
		{
			printf("\n%cHa salido algun coche%c\n1-Si\n0-No\n",168,63);
			scanf_s("%d", &cochesalida);
			if (cochesalida == 1)
			{
				printf("%cEn que plaza estaba%c\n",168,63);
				scanf_s("%d",&plazasalida);
				*(psensor + (plazasalida - 1)) = 0;
			}
			printf("\n%cHay coche a la entrada%c\n1-Si\n0-No\n",168,63);
			scanf_s("%d", &cocheentrada);
			lleno(psensor, pcompleto);
			if (*pcompleto == 0)
			{
				if (cocheentrada == 1)
				{
					direccion(psensor);
				}
			}
			else
			{
				printf("\nLo sentimos nuestro parking esta
comleto\n");
			}
			printf("\n%cEs la hora del cierre del parking%c:", 168, 63);
			scanf_s("%d", &cierre);
			printf("\n\n");
		} while (cierre != 1 && *pcompleto != 1);
	}
	else
	{
		printf("El parking esta completo\n");
	}
}

void situacion(int* psensor)
{
	int i;
	printf("\nIntroduzca el estado de cada sensor situado en cada plaza\n");
	for (i = 0; i < TAM; i++)
	{
		printf("Plaza %d:",i+1);
		scanf_s("%d", psensor+i);
		printf("\n");
	}
}
void direccion(int* psensor)
```

```
{
	int i=0,flag=0;

	do
	{
		if (*(psensor + i) == 0)
		{
			printf("\nDirijase a la plaza numero %d\n", i + 1);
			*(psensor + i) = 1;
			flag = 1;
			i++;
		}
		i++;
	} while (flag != 1);
}
void lleno(int* psensor, int* pcompleto)
{
	int i=0,flag=0;
	do
	{
		if (*(psensor + i) == 1)
		{
			*pcompleto = 1;
			i++;
		}
		else if (*(psensor + i) == 0)
		{
			*pcompleto = 0;
			i++;
			flag = 1;
		}
	} while (flag != 1 && i<TAM);
		}
```

2. Código en Arduino:

Para el caso del hardware con arduino lo que se ha hecho es instalar un sensor ultrasonidos simulando el sensor de la primera plaza de nuestro parking, así como también se ha instalado una pantalla LCD en la que se da la bienvenida a nuestro parking cada vez que es ejecuta nuestra función loop(), después manda una señal de ultrasonidos desde nuestro sensor con la que obtenemos el tiempo que tarda en rebotar y volver dicha señal y tiempo con el que mediante una conversión obtenemos la distancia a la que se encuentra el objeto. Una vez hecho esto evaluamos la distancia detectada y si se considera que es una distancia tal como para que un vehículo este ocupando la plaza en nuestra pantalla LCD se muestra el mensaje de que nuestro parking esta completo. Por el contrario si se detecta que no hay ningún vehículo ocupando la plaza se muestra en el LCD un mensaje que dice “diríjase a la plaza numero 1”.

```
#include <LiquidCrystal.h>
```

```
int TRIG=12;
int ECHO=11;
int V0=3;
LiquidCrystal lcd(4, 5, 6, 7, 8, 9);
long tiempo;
long distancia;

void setup()
{
  pinMode(TRIG,OUTPUT);
  pinMode(ECHO,INPUT);
  pinMode(1,INPUT);
  Serial.begin(9600);
  analogWrite(V0,130);
  lcd.begin(16, 2);
}
void loop()
{
  lcd.setCursor(0,0);
  lcd.print("  Bienvenido a");
  lcd.setCursor(0,1);
  lcd.print("nuestro parking");
  //lcd.clear();
  delay(1500);
```

```
digitalWrite(TRIG,0);
delayMicroseconds(5);
digitalWrite(TRIG,HIGH);
tiempo=pulseIn(ECHO,HIGH);
distancia=tiempo/58.2;
Serial.println(distancia);

if(distancia<10)
{
  lcd.setCursor(0,0);
  lcd.print("  Lo sentimos ");
  lcd.setCursor(0,1);
  lcd.print("  No hay plazas");
  delay(2000);
}
else
{
  lcd.setCursor(0,0);
  lcd.print("Dirijase a la ");
  lcd.setCursor(0,1);
  lcd.print("Plaza numero 1 ");
  delay(2000);
}
lcd.clear();
delay(1500);
```

}

- **Aparcamiento**

1. Código en Visual Studio:

La parte del control de aparcamiento es la más sencilla, puesto que es el mismo sistema para todas las plazas, solo ha habido que modelarlo una vez. Como ya hablamos en la memoria preliminar del proyecto, el funcionamiento es muy sencillo. En este caso hemos utilizado dos funciones void, a las que les pasamos enteramente valores por referencia. la razón de esto es que, al estar emulando una máquina de estados, Todas las variables tendrán valores lógicos (0 ó 1), a excepción de la variable *buzz, que es de tipo char, y de la variable distancia, que tendrá valores enteros desde 30 hasta 0, simulando el alcance del sensor de ultrasonidos utilizado para el diseño del hardware. Por lo que sumado a la eficiencia de este método de programación en contraposicón al paso por valor. Nos permite realizar un código mucho más compacto y eficiciente. En la función main solo tendremos la inicialización de las variables, y un while en el cual el compilador entrará siempre y cuando la distancia sea mayor que 0 (<0 el sensor no funciona, recordemos que el visual lo usamos para emular el funcionamiento del hardware) dentro de este while se encontraran las llamadas a las funciones que hemos utilizado, void parking_aid, y void estado_plaza.

La función void parking_aid le pasamos la variable distancia (el while se irá encargando de hacerla más pequeña con una variable decremento (i--)
Y se producirá una llamada a la función para cada valor de esta. En el prototipo, tenemos una serie de ifs con intervalos de distancia, cada uno representan un escenario diferente en cuanto al encendido de los leds y al valor de la variable buzz, representado como una nota músical con un carácter (#A, #B, #C…), las variables que valgan 1, representan un led encendido y 0 representa el mismo led apagado. En la impresión en pantalla se puede ver como cada vez hay menos 0 y más 1, lo que representa un descenso progresivo de la distancia, señal de que hay un vehículo aparcando. en el último condicional de todos, se produce la llamada a la siguiente función; void estado_plaza.

La función estado plaza representa, como su nombre indica, el cambio de estado de una plaza. De libre a ocupada. Información que será de interés después para el funcionamiento del sistema de guiado. la variable aux nos ayuda a representar mejor ese cambio de estado, ya que la variable libre ha de pasar de valer 1 a 0, y la variable ocupada, pasa de 0 a 1, representando lo que en Arduino sería el encendido y apagado simultaneo del led verde (se apaga cuando se aparca y se enciende el rojo). Para que sea más fácil entender lo que

se imprime en pantalla y no tener que poner tantas variables, en este caso en pantalla se escribirá “la plaza está ocupada”

```
void parking_aid(int, int*, int*, int*, int*, char*);
void estado_plaza(int*,int*);
int main()
{
	char buzz='A';//la primera nota que oiremos por el altavoz será un Do (#A)
	int distancia=30;//a partir de esta distancia (cm) entramos en el rango de
funcionamiento del sensor y se activa el sistema
	int ledblanco = 0, ledverde = 0, ledamarillo = 0, ledrojo = 0;
	int libre = 1;
	int ocupada = 0;
	while (distancia > 0)//simulamos con esto el coche aparcando y la lectura
del sensor que cada vez será menor
	{
		parking_aid(distancia, &ledblanco, &ledverde, &ledamarillo,
&ledrojo, &buzz);
		distancia--;
	}
	estado_plaza(&libre, &ocupada);
	//en el arduino, los leds no están fijos, si no que parpadean y en el
último tramo siguen una secuencia. esto no se puede simular en visual.
}
void parking_aid(int distancia, int *ledblanco, int *ledverde, int *ledamarillo,
int *ledrojo, char *buzz)
{
	if (distancia < 30 && distancia>=20)
	{
		*ledblanco = 1;
		*buzz = 'B';//ahora tenemos un re (#B)
		printf("%d %d %d %d %c\n", *ledblanco, *ledverde, *ledamarillo,
*ledrojo, *buzz);
	}
	if (distancia < 20 && distancia >= 10)
	{
		*ledverde = 1;
		*buzz = 'C';//#C
		printf("%d %d %d %d %c\n", *ledblanco, *ledverde, *ledamarillo,
*ledrojo, *buzz);
	}
	if (distancia < 10 && distancia >= 5)
	{
		*ledamarillo = 1;
		*buzz = 'D';//#D
		printf("%d %d %d %d %c\n", *ledblanco, *ledverde, *ledamarillo,
*ledrojo, *buzz);
	}
	if (distancia < 5 && distancia>0)//no hace falta >=0 porque ya eliminamos
esa posibilidad con el while
	{
		*ledrojo = 1;
		*buzz = 'E';//#E
		printf("%d %d %d %d %c\n", *ledblanco, *ledverde, *ledamarillo,
*ledrojo, *buzz);
	}
	return;
}
void estado_plaza(int *libre, int *ocupada)
{
```

```
int aux = 0;
aux = *libre;
*libre = *ocupada;
*ocupada = aux;
if (*libre == 0)
{
	printf("la plaza esta ocupada\n");
}
return;
	}
```

2. Código en Arduino:

En este caso, el funcionamiento es muy similar al ya explicado con el código de visual. Solo que en este caso ya tenemos leds reales y un pequeño altavoz que emite el sonido que necesitamos. En este caso, a parte de usar las funciones necesarias para Arduino, Hemos implementado una más para calcular la distancia al sensor, metiéndoles las variables de distancia y duración, también por referencia, pero esta vez si que nos devuelve un valor, que es el que utilizaremos para los condicionales que irán encendiendo y apagando los leds.

El sensor de ultrasonido funciona por pulsos de tensión emitidos a alta frecuencia. Utiliza la función pulseIn para calcular una duración estimada desde que emite un pulso hasta que vuelve al rebotar con el objeto que se acerque, dividimos entre 58,2 para obtener una distancia en cm que nosotros podamos interpretar con mayor facilidad.

El buzzer y los leds funcionan de una manera muy parecida, con las funciones digitalWrite para encender (HIGH) o apagar (LOW) los leds. Esta función recibe la constante asignada a cada led y HIGH o LOW en función de lo que se necesite hacer.

La misma función la usamos para el buzzer, mismo funcionamiento. Pero aquí añadimos una función extra, tone, a la cual le pasamos la frecuencia a la que queremos que suene, la duración del sonido en ms y el valor de la constante que le hemos asignado al buzzer en nuestro programa (en nuestro caso const int 4)

Aquí si contamos con variable temporal, por lo que podemos representar secuencias de luces, y la sincronización del sonido emitido por el altavoz.

```
const int ocupada=8;
const int libre=12;
const int LEDblanco=11;
const int LEDverde=10;
const int LEDamarillo=7;
```

```
 const int LEDrojo=5;
 const int buzz=4;
 float duracion;
 float distancia;
 const int trigger=2;
 const int echo=3;
 float dist(float*,float*);
void setup()
{
 pinMode(ocupada, OUTPUT);
 pinMode(libre, OUTPUT);
 pinMode(LEDblanco, OUTPUT);
 pinMode(LEDverde,OUTPUT);
 pinMode(LEDamarillo, OUTPUT);
 pinMode(LEDrojo, OUTPUT);
 pinMode(trigger, OUTPUT);
 pinMode(echo, INPUT);
 pinMode(buzz,OUTPUT);
 Serial.begin(9600);
}

void loop() {


 digitalWrite(LEDverde, LOW);
 digitalWrite(LEDamarillo, LOW);
 digitalWrite(LEDrojo, LOW);
 digitalWrite(buzz,LOW);
 digitalWrite(trigger, HIGH);
 delay(1);
 distancia=dist(&distancia, &duracion);
 Serial.println("distancia:");
 Serial.println(distancia);
 delay(250);
 if(distancia>30)
 {
    digitalWrite(libre, HIGH);
    digitalWrite(ocupada, LOW);
 }
 if(distancia<30 && distancia>20)
 {
  digitalWrite(libre, HIGH);
  digitalWrite(ocupada, LOW);
  digitalWrite(LEDblanco, HIGH);
  digitalWrite(buzz, HIGH);
```

```
  tone(4,100,400);
  delay(400);
  digitalWrite(LEDblanco, LOW);
  digitalWrite(buzz, LOW);
  digitalWrite(LEDverde, LOW);
  digitalWrite(LEDamarillo, LOW);
  digitalWrite(LEDrojo, LOW);
 }
 if(distancia<20 && distancia>10)
 {
  digitalWrite(libre, HIGH);
  digitalWrite(ocupada, LOW);
  digitalWrite(LEDblanco, HIGH);
  digitalWrite(LEDverde, HIGH);
  digitalWrite(buzz,HIGH);
  tone(4,200,200);
  delay(200);
  digitalWrite(LEDblanco, LOW);
  digitalWrite(buzz, LOW);
  digitalWrite(LEDverde, LOW);
  digitalWrite(LEDamarillo, LOW);
  digitalWrite(LEDrojo, LOW);
 }
 if(distancia<10&&distancia>5)
 {
  digitalWrite(libre, HIGH);
  digitalWrite(ocupada, LOW);
   digitalWrite(LEDblanco, HIGH);
   digitalWrite(LEDverde, HIGH);
   digitalWrite(LEDamarillo, HIGH);
   digitalWrite(buzz, HIGH);
   tone(4,300,100);
   delay(100);
   digitalWrite(LEDblanco, LOW);
   digitalWrite(LEDverde, LOW);
   digitalWrite(LEDamarillo, LOW);
   digitalWrite(LEDrojo, LOW);
   digitalWrite(buzz,LOW);
 }
 if(distancia<5&&distancia>0)
 {
   digitalWrite(ocupada, HIGH);
   digitalWrite(libre, LOW);
   digitalWrite(buzz, HIGH);
   tone(4,380,0);
```

```
    delay(250);
    digitalWrite(LEDblanco, HIGH);
    delay(250);
    digitalWrite(LEDblanco, LOW);
    digitalWrite(LEDverde, HIGH);
    delay(250);
    digitalWrite(LEDverde, LOW);
    digitalWrite(LEDamarillo,HIGH);
    delay(250);
    digitalWrite(LEDamarillo, LOW);
    digitalWrite(LEDrojo, HIGH);
    delay(250);
    digitalWrite(LEDrojo, LOW);
    delay(250);
  }


}
float dist(float *distancia, float *duracion)
{
  digitalWrite(trigger, HIGH);
  delay(1);
  digitalWrite(trigger, LOW);
  *duracion=pulseIn(echo, HIGH);
  *distancia=*duracion/58.2;
  return *distancia;
}
```

- **Semáforos**

1. Código en Visual Studio:

   En este código básicamente hemos ido creando funciones en función de los estados que queríamos para los semáforos y para las secuencias.
   De esta manera podemos ver que tenemos la función Estado1() que nos indica el estado de funcionamiento normal de nuestros semáforos si no sucede nada y pregunta al usuario si algún peatón quiere pulsar el botón para cruzar o si está prevista la salida del parking de un coche.
   Una vez hecho se evalúa si hay algún coche que vaya a salir del parking y si es así se crea una secuencia de estados de los semáforos para permitir la salida

controlada del vehículo a la vía pública, de lo contrario pasaríamos a evaluar los estados de los pulsadores de los peatones para saber que secuencia de estados debemos realizar.

Para las secuencias de estados se han creado una serie de funciones llamadas semáforo(),semaforo2(),espera(),espera2() y salida() que cada uno de ellos contiene una sucesión de estados.

Por último se ha implementado un contador que una vez finalizado el programa nos indica cuantas veces se ha pulsado los botón para permitir a los peatones cruzar.

```
#include<stdio.h>
#include<windows.h>
#include<stdlib.h>//Biblioteca para asignación dinámica de memoria
#include<conio.h>//Biblioteca para el temporizador

void Estado1(int*,int*,int*);
void semaforo(void);
void semaforo2(void);
void salida(void);
void espera(void);
void espera2(void);

int main()
{
	int pulsadorvia = 0, * ppulsadorvia,pulsadorparking=0, * ppulsadorparking,
dim = 1, contador = 0,salidacoche=0, * psalida;
	//Con asignacion dinamica de memoria reservo 4 bytes de memoria
(equivalente del int)
	ppulsadorvia = (int*)malloc(dim * sizeof(int));
	ppulsadorparking = (int*)malloc(dim * sizeof(int));
	psalida= (int*)malloc(dim * sizeof(int));
	if (ppulsadorvia == NULL)
	{
		printf("No hay espacio");
		exit(1);
	}
	if (ppulsadorparking == NULL)
	{
		printf("No hay espacio");
		exit(1);
	}
	if (psalida == NULL)
	{
		printf("No hay espacio");
		exit(1);
	}

	ppulsadorvia = &pulsadorvia;
	ppulsadorparking = &pulsadorparking;
	psalida = &salidacoche;
	do
	{
		Estado1(ppulsadorvia,ppulsadorparking,psalida);
		if (*psalida == 0)
		{
			if (*ppulsadorvia != 0 || *ppulsadorparking != 0)
```

```
                        {
                                if (*ppulsadorvia != 0 && *ppulsadorparking == 0)
                                {
                                        espera(); //Llamada a las funciones espera y
semaforo
                                        semaforo();
                                        contador++;
                                }
                                if (*ppulsadorparking != 0 && *ppulsadorvia == 0)
                                {
                                        espera2(); //Llamada a las funciones espera y
semaforo
                                        semaforo2();
                                        contador++;
                                }
                                if (*ppulsadorparking != 0 && *ppulsadorvia != 0)
                                {
                                        espera();
                                        semaforo();
                                        espera2();
                                        semaforo2();
                                        contador = contador + 2;

                                }
                        }
                }
                if (*psalida != 0)
                {
                        if (*ppulsadorparking != 0 || *ppulsadorvia != 0)
                        {
                                if (*ppulsadorvia != 0 && *ppulsadorparking == 0)
                                {
                                        espera();
                                        salida();
                                        espera(); //Llamada a las funciones espera y
semaforo
                                        semaforo();
                                        contador++;
                                }
                                if (*ppulsadorparking != 0 && *ppulsadorvia == 0)
                                {
                                        espera();
                                        salida();
                                        espera2(); //Llamada a las funciones espera y
semaforo
                                        semaforo2();
                                        contador++;
                                }
                                if (*ppulsadorparking != 0 && *ppulsadorvia != 0)
                                {
                                        espera();
                                        salida();
                                        espera();
                                        semaforo();
                                        espera2();
                                        semaforo2();
                                        contador = contador + 2;
                                }
                        }
                }
        } while(*ppulsadorvia != 0 ||*ppulsadorparking!=0 || *psalida!=0);
```

```
	printf("\n\nSe han pulsado un total de %d veces los pulsadores de los
semaforos", contador);

	free(ppulsadorvia); //Libero el espacio reservado
	getchar();
	printf("\n\n");
	system("pause");
}
void Estado1(int* x,int* y,int* z)
{
	int pulsavia,pulsaparking,salida;
	printf("\nSemaforo Via\nVerde\n\nSemaforo Parking\nRojo\n\nSemaforo Peaton
Parking\nRojo\n\nSemaforo Peaton Via\nRojo");
	printf("\n\n\nPulse peaton de via\n");
	scanf_s("%d", &pulsavia);
	printf("\n\n\nPulse peaton de parking\n");
	scanf_s("%d", &pulsaparking);
	printf("\n\n\nPrevista salida de un coche del parking\n");
	scanf_s("%d", &salida);
	*x = pulsavia;
	*y = pulsaparking;
	*z = salida;
	system("cls");  //comando para borrar lo impreso en pantalla anteriormente
}

void espera(void)
{
	int seg;
	for (seg = 5; seg >= 0; seg--)
	{
		printf("\t\nSemaforo Via\nAmbar %d\t\n\nSemaforo Parking\nRojo
%d\t\n\nSemaforo Peaton Parking\nRojo %d\t\n\nSemaforo Peaton Via\n Rojo %d",
seg, seg, seg, seg);
		Sleep(999);
		system("cls");
	}
}

void semaforo(void)
{
	int color = 2, seg;
	while (color != 0)
	{
		if (color == 2) {
			for (seg = 10; seg >= 0; seg--) {
				printf("\t\nSemaforo Via\nRojo %d\t\n\nSemaforo
Parking\nVerde %d\t\n\nSemaforo Peaton Parking\nRojo %d\t\n\nSemaforo Peaton
Via\nVerde %d", seg, seg, seg, seg);
				Sleep(999);  //999 milisegundos
				system("cls");
			}
		}
		else if(color == 1)
		{
			for (seg = 5; seg >= 0; seg--) {
				printf("\t\nSemaforo Via\nAmbar intermitente
%d\t\n\nSemaforo Parking\nAmbar %d\t\n\nSemaforo Peaton Parking\nRojo
%d\t\n\nSemaforo Peaton Via\nVerde Intermitente %d", seg, seg, seg, seg);
				Sleep(999);
				system("cls");
			}
		}
```

```
			color--;
		}
}

void espera2(void)
{
		int seg;
		for (seg = 5; seg >= 0; seg--)
		{
			printf("\t\nSemaforo Via\nVerde %d\t\n\nSemaforo Parking\nRojo
%d\t\n\nSemaforo Peaton Parking\nRojo %d\t\n\nSemaforo Peaton Via\n Rojo %d",
seg, seg, seg, seg);
			Sleep(999);
			system("cls");
		}
}

void semaforo2(void)
{
		int color = 2, seg;
		while (color != 0)
		{
			if (color == 2) {
				for (seg = 10; seg >= 0; seg--) {
					printf("\t\nSemaforo Via\nVerde %d\t\n\nSemaforo
Parking\nRojo %d\t\n\nSemaforo Peaton Parking\nVerde %d\t\n\nSemaforo Peaton
Via\nRojo %d", seg, seg, seg, seg);
					Sleep(999);  //999 milisegundos
					system("cls");
				}
			}
			else if (color == 1)
			{
				for (seg = 5; seg >= 0; seg--) {
					printf("\t\nSemaforo Via\nVerde %d\t\n\nSemaforo
Parking\nAmbar intermitente %d\t\n\nSemaforo Peaton Parking\nVerde intermitente
%d\t\n\nSemaforo Peaton Via\nRojo %d", seg, seg, seg, seg);
					Sleep(999);
					system("cls");
				}
			}
			color--;
		}

}

void salida(void)
{
		int color = 2, seg;
		while (color != 0)
		{
			if (color == 2) {
				for (seg = 10; seg >= 0; seg--) {
					printf("\t\nSemaforo Via\nRojo %d\t\n\nSemaforo
Parking\nVerde %d\t\n\nSemaforo Peaton Parking\nRojo %d\t\n\nSemaforo Peaton
Via\nRojo %d", seg, seg, seg, seg);
					Sleep(999);  //999 milisegundos
					system("cls");
				}
			}
			else if (color == 1)
			{
```

```
                    for (seg = 5; seg >= 0; seg--) {
                           printf("\t\nSemaforo Via\nAmbar intermitente
%d\t\n\nSemaforo Parking\nAmbar %d\t\n\nSemaforo Peaton Parking\nRojo
%d\t\n\nSemaforo Peaton Via\nRojo %d", seg, seg, seg, seg);
                           Sleep(999);
                           system("cls");
                    }
             }
             color--;
      }
             }
```

2. Código en Arduino:

Para esta parte de Arduino lo que se ha hecho es mediante led y pulsadores simular el funcionamiento de los semáforos tanto para vehículos como para peatones. De una medida muy similar pero sin la funcionalidad de saber si un coche podía salir del parking debido a la falta de sensor ultrasonidos para esta parte. Una vez indicado esto como podemos comprobar hemos creado las función Normal() que ejecutará una secuencia de colores para nuestros leds asociados a cada uno de los pines de nuestra placa arduino uno, y que solo cambiará en el momento en que uno de los pulsadores, ya sea el de los peatones dela vía o el de los peatones del parking, sea activado. En ese momento entrara en la función de cual de los pulsadores haya sido activado en la función Pasopeatones1() o Pasopeatones2(). Una vez ejecutada una de estas funciones los semáforos volverán a funcionar según el estado Normal().

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

void Normal();
double Pulsador1;
double Pulsador2;
void Pasopeatones1();
void Pasopeatones2();

void Normal()
{
  while(Pulsador1!=1 || Pulsador2!=1)
  {
    digitalWrite(1,0);
    digitalWrite(2,0);
    digitalWrite(3,1);
    digitalWrite(4,1);
    digitalWrite(5,0);
    digitalWrite(6,0);
```

```
      digitalWrite(7,1);
      digitalWrite(8,0);
      digitalWrite(9,1);
      digitalWrite(10,0);

      if(digitalRead(11))
      {
        Pasopeatones1();
      }
      else if(digitalRead(12))
      {
        Pasopeatones2();
      }
   }

}

void Pasopeatones1()
{
 digitalWrite(1,0);
 digitalWrite(2,0);
 digitalWrite(3,1);
 digitalWrite(4,1);
 digitalWrite(5,0);
 digitalWrite(6,0);
 digitalWrite(7,1);
 digitalWrite(8,0);
 digitalWrite(9,1);
 digitalWrite(10,0);
 delay(5000);

 digitalWrite(1,0);
 digitalWrite(2,1);
 digitalWrite(3,0);
 digitalWrite(4,1);
 digitalWrite(5,0);
 digitalWrite(6,0);
 digitalWrite(7,1);
 digitalWrite(8,0);
 digitalWrite(9,1);
 digitalWrite(10,0);
 delay(5000);

 digitalWrite(1,1);
 digitalWrite(2,0);
```

```
 digitalWrite(3,0);
 digitalWrite(4,0);
 digitalWrite(5,1);
 digitalWrite(6,0);
 digitalWrite(7,0);
 digitalWrite(8,1);
 digitalWrite(9,1);
 digitalWrite(10,0);
 delay(5000);
}

void Pasopeatones2()
{
 digitalWrite(1,0);
 digitalWrite(2,0);
 digitalWrite(3,1);
 digitalWrite(4,1);
 digitalWrite(5,0);
 digitalWrite(6,0);
 digitalWrite(7,1);
 digitalWrite(8,0);
 digitalWrite(9,1);
 digitalWrite(10,0);
 delay(5000);

 digitalWrite(1,0);
 digitalWrite(2,0);
 digitalWrite(3,1);
 digitalWrite(4,1);
 digitalWrite(5,0);
 digitalWrite(6,1);
 digitalWrite(7,0);
 digitalWrite(8,0);
 digitalWrite(9,0);
 digitalWrite(10,1);
 delay(5000);
}

void setup()
{
 pinMode(1,OUTPUT);
 pinMode(2,OUTPUT);
 pinMode(3,OUTPUT);
 pinMode(4,OUTPUT);
 pinMode(5,OUTPUT);
```

```
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,INPUT);
  pinMode(12,INPUT);
}

void loop()
{
  Normal();
  delay(1000);
}
```

- **<u>Conclusión del trabajo:</u>**

  Estamos satisfechos con el gran trabajo realizado para desarrollar este proyecto, así como de los conocimientos transversales que hemos obtenido ya que hemos aprendido a utilizar el hardware de Arduino de una manera muy útil así como el lenguaje que se utiliza para mandar directrices y ordenes a nuestra placa Arduino.
  De la misma manera el trabajo también nos ha ayudado para la parte de programación en C ya que el código que hemos conseguido implementar contiene todos los contenidos vistos durante el curso.