TRABAJO INFORMATICA: CRUCE DE SEMÁFOROS

Antonio Casado Hierro (53871)

Ignacio de Benito Gómez (53908)

Objetivo del proyecto: en nuestro proyecto trataremos de mejorar la circulación tanto de vehículos como peatones en una vía de dos carriles cada uno en un sentido. Para ello sincronizaremos dos semáforos para los vehículos y dos semáforos para peatones uno a cada lado de la vía y los cuales tendrán un pulsador cada uno. El semáforo de peatones permanecerá siempre en rojo hasta que un peatón que quiera cruzar pulse cualquiera de los dos pulsadores dispuestos a cada lado de la vía.

EXPLICACIÓN DEL CÓDIGO EN VISUAL STUDIO

CÓDIGO:

```
#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 *);
voidsemaforo(void);
void espera(void);

voidmain(void)
{
	int pulsador = 0, *ppulsador, dim = 1, contador = 0;
	//Con asignaciondinamica de memoria reservo 4 bytes de memoria
(equivalente del int)
	ppulsador = (int*)malloc(dim * sizeof(int));
	if (ppulsador == NULL)
	{
		printf("No hay espacio");
		exit(-1);
	}
	ppulsador = &pulsador;
	do
	{
		Estado1(ppulsador);
		if (*ppulsador != 0)
		{
			espera();
			semaforo();
			contador++;
		}
	} while (*ppulsador != 0);
	printf("\n\nSe ha pulsado un total de %d veces el pulsador del
semaforo", contador);
	//Libero el espacio reservado
	free(ppulsador);
	printf("\n\n");
	system("pause");
}

void Estado1(int *x)
{
	printf("\nSemaforo 1\nVerde\n\nSemaforo 2\nVerde\n\nSemaforo 3\nRojo");
	printf("\n\n\nPulsepeaton\n");
	scanf_s("%d", x);
	system("cls");//comando para borrar lo impreso en pantalla anteriormente
}

voidsemaforo()
{
	int color = 2, seg;
	while (color != 0)
	{
		if (color == 2)
		{
			for (seg = 10; seg>= 0; seg--)
			{
```

```
                        printf("\t\nSemaforo 1\nRojo %d\t\n\nSemaforo
2\nRojo %d\t\n\nSemaforo 3\nVerde %d", seg, seg, seg);
                        Sleep(999);//999 milisegundos
                        system("cls");
                }
        }
        elseif (color == 1)
        {
                for (seg = 5; seg>= 0; seg--)
                {
                        printf("\t\nSemaforo 1\nAmbar %d\t\n\nSemaforo
2\nAmbar %d\t\n\nSemaforo 3\nVerde Intermitente %d", seg, seg, seg);
                        Sleep(999);
                        system("cls");
                }
        }
        color--;
    }
}

void espera(void)
{
    intseg;
    for (seg = 5; seg>= 0; seg--)
    {
        printf("\t\nSemaforo 1\nAmbar %d\t\n\nSemaforo 2\nAmbar
%d\t\n\nSemaforo 3\nRojo %d", seg, seg, seg);
        Sleep(999);
        system("cls");
    }
}
```

EXPLICACIÓN LÍNEA A LÍNEA:

Para comenzar, agregamos a nuestro programa las bibliotecas “stdio.h” y “Windows.h”, las cuales utilizamos en todos nuestros ejercicios, y a su vez incluimos la biblioteca “stdlib.h” (la encargada de permitirnos incluir la asignación dinámica de memoria), y la “conio.h” (fundamental para implantación de temporizadores).

A continuación, introducimos los prototipos de las funciones empleadas. Como bien podemos ver, utilizaremos un total de 3 funciones. La función Estado1 no devolverá ninguna variable, pero sí recibirá un puntero del tipo int. La siguiente, semáforo, no devolverá ni recibirá ninguna variable, exactamente igual que la tercera, espera.

Posteriormente abrimos el main, con el cuerpo voidmain(void), evitando el warning que nos produciría iniciándolo como intmain().

Dentro del main irá el esqueleto principal de nuestro programa.

Comenzamos inicializando la variable pulsador a 0, la variable dim a 1 y la variable contador a 0. También declaramos el punteroppulsador.

Gracias a la asignación dinámica de memoria, con malloc, reservamos el espacio equivalente a una variable int (4 bytes) para el puntero declarado anteriormente. El bloque del if que viene justo después tiene la función de saltar por si se produce cualquier tipo de error durante el proceso de reserva.

Seguimos con el código inicializando el puntero ppulsador a la dirección de memoria de lavariable pulsador declarada e inicializada anteriormente.

Abrimos el bucle do-while, el cual se repetirá siempre que el contenido residente en ppulsador sea distinto de 0 (como veremos posteriormente, el valor 0 está reservado para cerrar el programa).

A continuación, nos encontramos con la llamada a la función Estado1, con la cual enviamos el puntero ppulsador.

Esta función tiene un funcionamiento muy sencillo. Lo que hace es imprimir en pantalla mediante un printf el estado de los 3 semáforos de forma estática, y también nos permite dar un valor a x (contenido de ppulsador), que al ser una función paso por referencia nos permite variar el contenido del puntero sin necesitar que esta devuelva nada (en este caso le daremos un valor numérico distinto de 0, ya que el puntero es del tipo int). Por primera vez nos encontramos con el comando system("cls"), un comando perteneciente a la biblioteca "Windows.h", y el cual nos permite borrar lo impreso anteriormente en pantalla.

Al haber variado el valor de ppulsador, volvemos al main, y entramos en el if, ya que el valor en ppulsador será distinto de 0 (como fue explicado anteriormente el 0 es nuestra marca para salir del programa).

Seguido de esto nos encontramos con la llamada a la función espera.

En esta función, inicializamos como int la variable seg (será utilizada como el número total de segundos en esta función). Abrimos un bucle for el cual vaya desde el valor de la variable seg que hayamos iniciado (en nuestro caso 5) hasta que sea 0, y reduciendo de uno en uno su valor. Dentro de este bucle for, con un printf volvemos a mostrar el estado de los semáforos, y a su vez el valor de la variable seg (mediante el %d). En la siguiente línea utilizamos el comando Sleep(999), que nos permite pasar a la siguiente línea 999 milisegundos después, que será el equivalente de un segundo, y volvemos a borrar lo impreso en pantalla con el comando system("cls"). Gracias a este bucle, en nuestro programa observaremos una cuenta atrás.

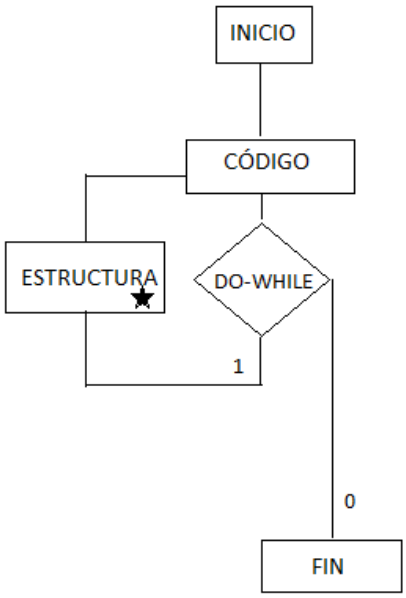
Volviendo al main saltamos a la función semáforo, la cual tiene un funcionamiento prácticamente idéntico que la función espera, con la única diferencia de que declaramos e inicializamos la variable color a 2, y que mediante un bucle while, que se ejecutará mientras que esta variable sea distinta de 0, imprimiremos en pantalla el estado de los semáforos según nos marque la variable color. Cuando la variable color valga 2,entraremos en el if el cual nos enseña que el estado de los semáforos 1, 2 y 3 será rojo, rojo, verde simultáneamente. Cuando el temporizador realizado mediante el bucle for marque 0 saldremos tanto del bucle for, como del if, y la variable color disminuirá una unidad. Se repite el bucle while, en este caso con el valor de la variable color igual a 1, y los bucles if y for serán idénticamente iguales, solo que ahora los estados de los semáforos serán ambar, ambar y verde intermitente. Por último, cuando la variable color disminuya otra unidad, color valdrá 0 y saldremos de esta función.

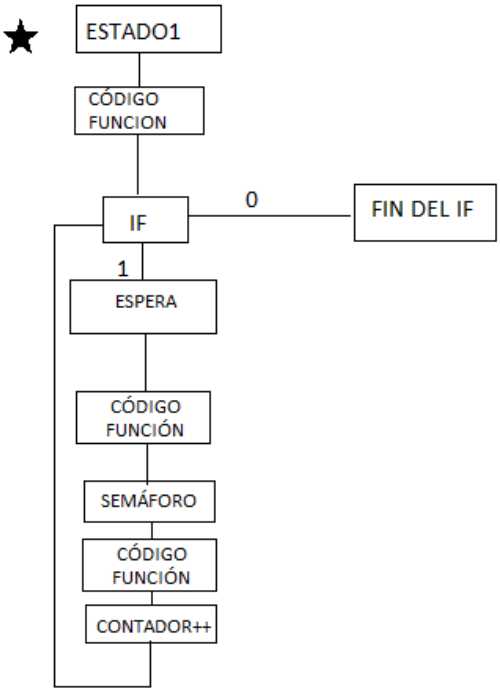
Volviendo al main, aumentamos en una unidad la variable contador.

Repetiremos tantas veces como el usuario quiera el bucle do-while, y cuando se desee cerrar el programa, saliendo del bucle, el programa nos mostrará el número de veces que repetimos el bucle.

Por último, liberamos el espacio reservado al puntero ppulsador y cerramos el programa.

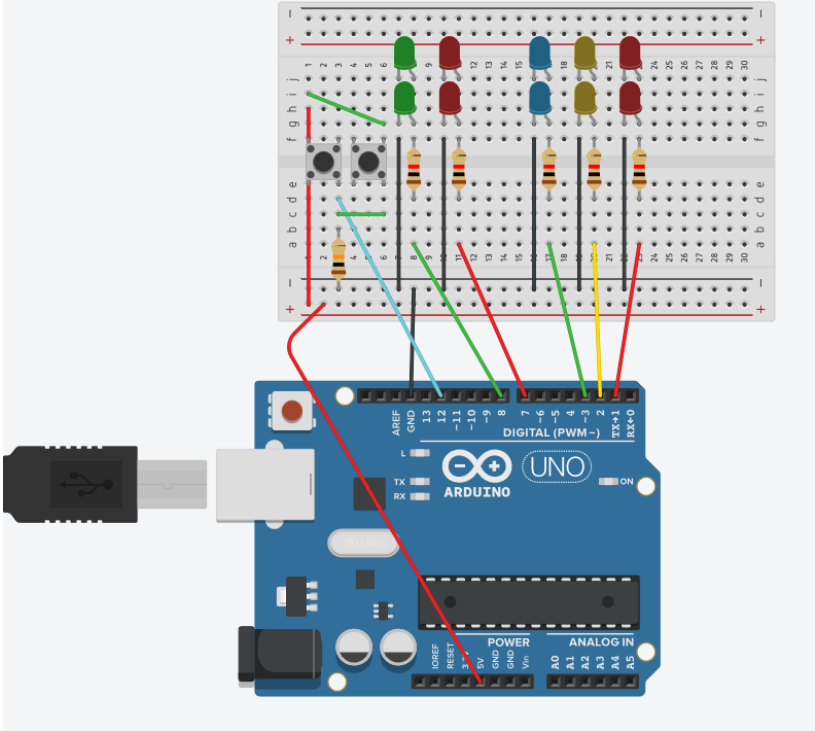
FLUJOGRAMA:


INICIO
CÓDIGO
ESTRUCTURA
DO-WHILE
1
0
FIN


ESTADO1
CÓDIGO FUNCION
IF
0
FIN DEL IF
1
ESPERA
CÓDIGO FUNCIÓN
SEMÁFORO
CÓDIGO FUNCIÓN
CONTADOR++

CODIGO Y MONTAJE DE ARDUINO

- El código en arduino le hemos conseguido gracias a la búsqueda de información en internet así como de la ayuda de un programa llamado mblock el cual ayuda a visualizar el flujo del programa de arduino que quieres montar y de una página de internet llamada tinkercad que te permite ver si tu programa hace que la placa arduino ejecute las acciones que tu quieres que haga mediante una simulación. En esta última página se realiza el montaje en la protoboard para que mediante la simulación quede reflejado si tu código es correcto o no y de esta manera ahorrar tiempo a lo hora de realizar el montaje final. Quedando el montaje de nuestro trabajo de la siguiente forma:



- Material utilizado:
  -Cables
  -10 leds
  -6 resistencias
  -2 pulsadores
  -Protoboard
  -Placa arduino

- A continuación se presenta el código en arduino que hemos realizado en este trabajo:

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

doubleangle_rad = PI/180.0;
doubleangle_deg = 180.0/PI;
voidNormal();
double tiempo;
doubletiempoactual;
double Intervalo;
double Pulsador1;
doublecurrentTime = 0;
doublelastTime = 0;
void Pasopeatones1();

voidNormal()
{
  tiempo = currentTime;
while(!((tiempo) > ((tiempoactual) + (Intervalo))))
  {
    _loop();
digitalWrite(1,1);
digitalWrite(2,0);
digitalWrite(3,0);
    tiempo = currentTime;
if(digitalRead(12)){
       Pulsador1 = 1;
    }
  }
while(!((tiempo) > ((tiempoactual) + ((2) * (Intervalo)))))
  {
    _loop();
digitalWrite(1,0);
digitalWrite(2,0);
```

```
digitalWrite(3,1);
      tiempo = currentTime;
if(digitalRead(12)){
         Pulsador1 = 1;
      }
   }
while(!((tiempo) > ((tiempoactual) + ((3) * (Intervalo)))))
   {
      _loop();
digitalWrite(1,0);
digitalWrite(2,1);
digitalWrite(3,0);
      tiempo = currentTime;
if(digitalRead(12)){
         Pulsador1 = 1;
      }
   }
tiempoactual = currentTime;
}

void Pasopeatones1()
{
   tiempo = currentTime;
while(!((tiempo) > ((tiempoactual) + (Intervalo))))
   {
      _loop();
digitalWrite(1,1);
digitalWrite(2,0);
digitalWrite(3,0);
digitalWrite(7,0);
digitalWrite(8,1);
      tiempo = currentTime;
   }
digitalWrite(7,1);
digitalWrite(8,0);
```

```
while(!((tiempo) > ((tiempoactual) + ((2) * (Intervalo)))))
  {
    _loop();
digitalWrite(1,0);
digitalWrite(2,0);
digitalWrite(3,1);
    tiempo = currentTime;
  }
while(!((tiempo) > ((tiempoactual) + ((3) * (Intervalo)))))
  {
    _loop();
digitalWrite(1,0);
digitalWrite(2,1);
digitalWrite(3,0);
    tiempo = currentTime;
  }
tiempoactual = currentTime;
}

voidsetup(){
pinMode(1,OUTPUT);
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(12,INPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
lastTime = millis()/1000.0;
  Intervalo = 5;
tiempoactual = 0;
digitalWrite(7,1);
digitalWrite(8,0);
}

voidloop(){
Normal();
```

```
if(digitalRead(12)){
      Pulsador1 = 1;
   }
if(((Pulsador1)==(1))){
      Pasopeatones1();
   }
   Pulsador1 = 0;
   _loop();
}

void _delay(floatseconds){
longendTime = millis() + seconds * 1000;
while(millis() <endTime)_loop();
}

void _loop(){
currentTime = millis()/1000.0 - lastTime;
}
```

EXPLICACIÓN DEL CÓDIGO EN ARDUINO:

Como primera medida se inicializan las librerías,así como a continuación se inicializan las variables que se utilizan en el programa y las funciones Normal y Pasopeatones1.

Después utilizamos un cronómetro que va registrando el tiempo en cada momento que junto con las variables tiempo, tiempoactualy intervalo nos ayudan a coordinar la secuencia que llevaran los semáforos. Para llevar a cabo la sincronización de los semáforos utilizamos bucles while donde las condiciones son operaciones de tiempo con las tres variables citadas anteriormente. Dentro de cada bucle while asignamos a cada salida digital (leds) su estado HIGH si esta activada o LOW si esta desactivada.

Todo esto se utiliza en las funciones Normal y Pasopeatones1, funciones que luego forman parte del programa principal en el cual se realiza la función Normal siempre a no ser que la entrada digital 12 que representa a los pulsadores sea activada y se le asigne el valor 1 que en ese caso acabara el ciclo de la función Normal y entrara en la función Pasopeatones para ejecutarlo 1 vez. También disponemos de una parte del código denominada voidsetup en la cual queden registradas y asociadas a cada entrada y salida un pin en la placa arduino.