CRUCES DE SEMÁFOROS

Introducción

En la presente documentación se van a desarrollar una serie de ejemplos para instruirse en el manejo de la controladora CNICE utilizando el lenguaje de programación **Logo**, empleando el compilador MSWLogo que distribuye el **Instituto de Tecnologías Educativas** a través de su Observatorio y el lenguaje C usando el compilador DevC++. Con ellos se tratará de que el "alumno" vaya aprendiendo de forma gradual a programar esta controladora con estos lenguajes, por ello, y como se podrá comprobar, hay ejemplos que se desarrollan en varias fases. No obstante, hay que destacar que se presuponen unos conocimientos previos en estos lenguajes de programación y en las primitivas que se emplean para la controladora.

En primer lugar se definirán los conceptos de programación más importantes y que se han empleado para implementar la mayor parte de los ejemplos para facilitar la comprensión del código desarrollado.

A continuación se explicará qué pasos se deben seguir para ejecutar en el compilador de **MSWLogo** cualquiera de los ejemplos proporcionados para poder ver cómo funcionan con la controladora CNICE.

Seguidamente se procederá a explicar en detalle cómo se ha construido el dispositivo que va a permitir probar los programas desarrollados. Se explicará también en detalle en qué consiste cada uno de los cruces de semáforos.

Por último se pasará a exponer el funcionamiento de cada uno de los ejemplos y las conexiones que se deben realizar en la controladora junto con el código correspondiente. Para obtener más información acerca de cómo se ha realizado la programación se debe consultar los amplios comentarios que se han introducido en el código de cada ejemplo.

Consideraciones previas de programación y de implementación

Como ya se ha mencionado anteriormente, se considera que el usuario debe tener conocimientos previos en el lenguaje de programación Logo y en las primitivas que se utilizan para manipular la controladora CNICE. No obstante se van a dar unas definiciones básicas para facilitar la comprensión del código desarrollado para implementar los programas proporcionados.

Vectores.

Un vector o array es una colección no ordenada de elementos a los que se puede acceder a través de un índice.

En Logo para definir un vector se emplea la función predefinida:

```
matriz tamaño [origen]
```

donde **tamaño** indica el número de elementos de los que consta el vector y **origen** es un parámetro opcional que indica el valor a partir del cual se comenzará a indexar en el vector.

Para introducir elementos en el vector se utiliza el procedimiento predefinido:

ponelemento indice nomb_vector valor

donde **indice** indica la posición en el vector en la que se va a introducir el elemento, **nomb_vector** es el nombre del vector en el que se quiere poner el elemento y **valor** es el objeto que se pondrá en la posición indicada del vector.

Por ejemplo, supongamos que queremos crear un vector de dos elementos al que queremos llamar **mivector** cuyo índice empiece en 0. Una vez creado queremos introducir los valores 5 y 7 en las posiciones 0 y 1 del vector. Por último queremos que se muestre por pantalla el resultado obtenido:

```
haz "mivector (matriz 2 0)
ponelemento 0 :mivector 5
ponelemento 1 :mivector 7
muestra :mivector
{5 7}
```

Construcción

Este dispositivo simula el funcionamiento de un cruce de semáforos. Tenemos una calle que será la principal y en la que el semáforo permanecerá más tiempo en verde. La secuencia de colores entre semáforos es la siguiente:

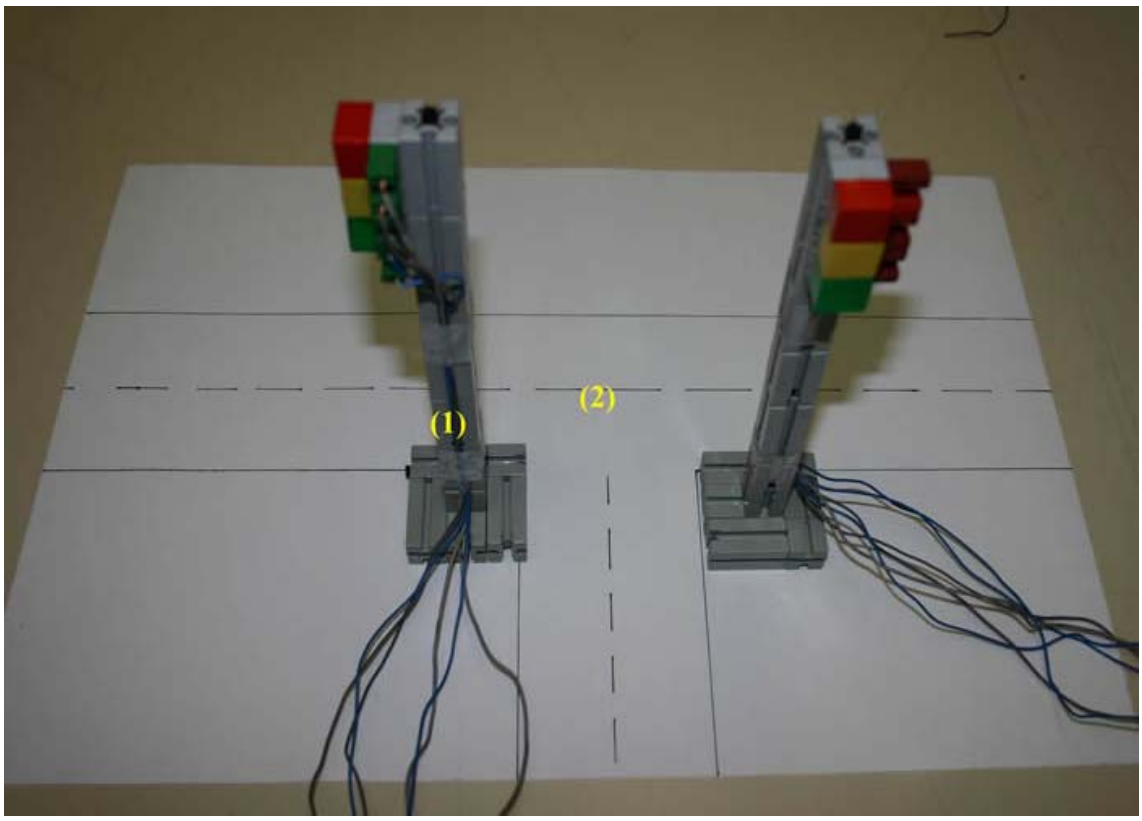
Secuencia de colores entre semáforos	
Semáforo 1 (calle principal)	**Semáforo 2 (calle secundaria)**
Rojo	Verde
Verde	Rojo
Ámbar	Rojo

Descripción de elementos

El cruce de semáforos esta formado por los siguientes elementos:

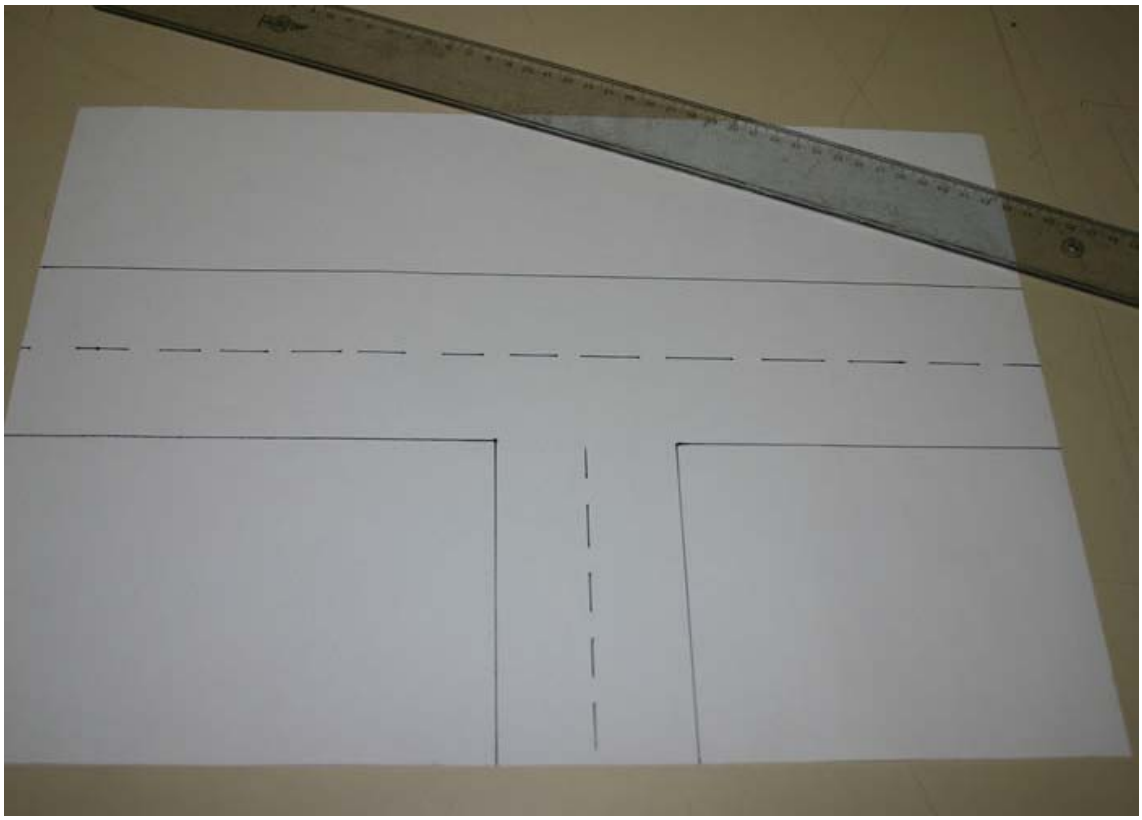
- 2 Semáforos (1): Se utilizan los semáforos que son la parte fundamental de esta construcción.
- Plataforma (2): Los semáforos se van a situar sobre una plataforma, sobre la cual se pinta un cruce de carreteras.

Montaje del cruce de semáforos

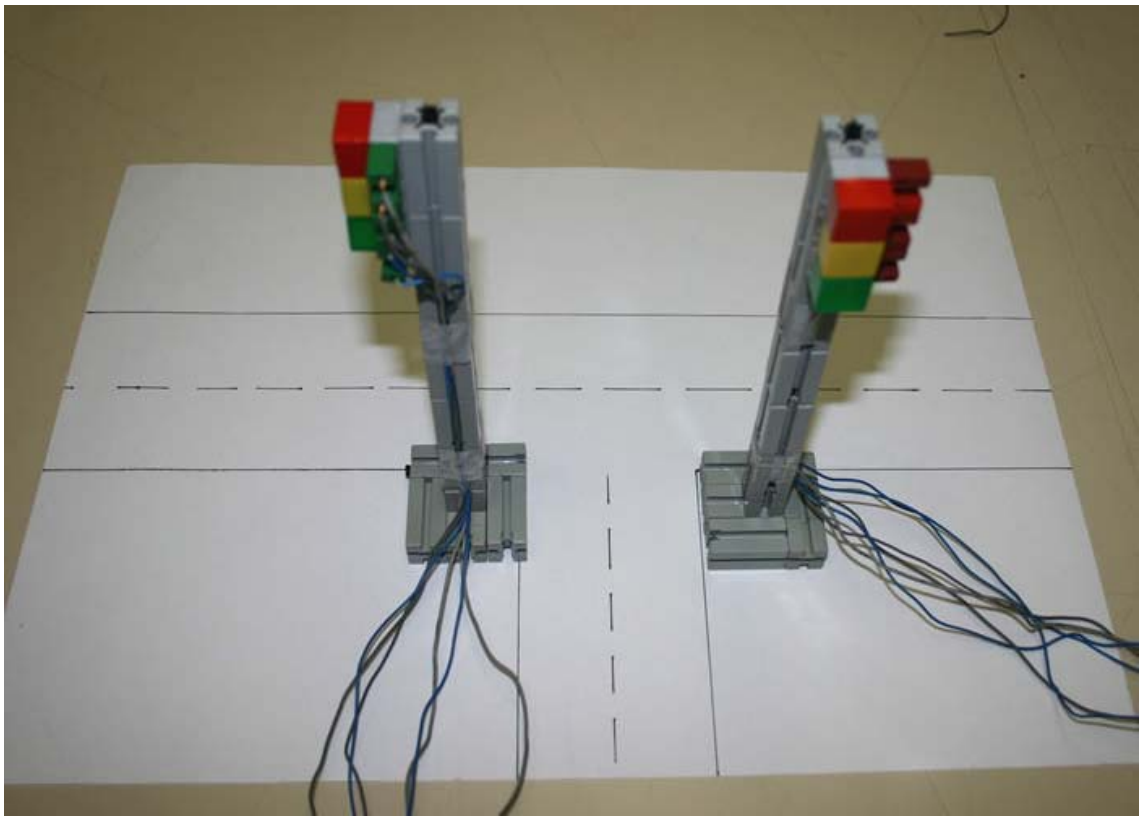


El semáforo se ha montado a partir de piezas fisher. Los pasos que se han seguido se describen a continuación:

1. Se utiliza una cartulina o una hoja como plataforma (2), sobre la cual se pintarán un cruce de carreteras.

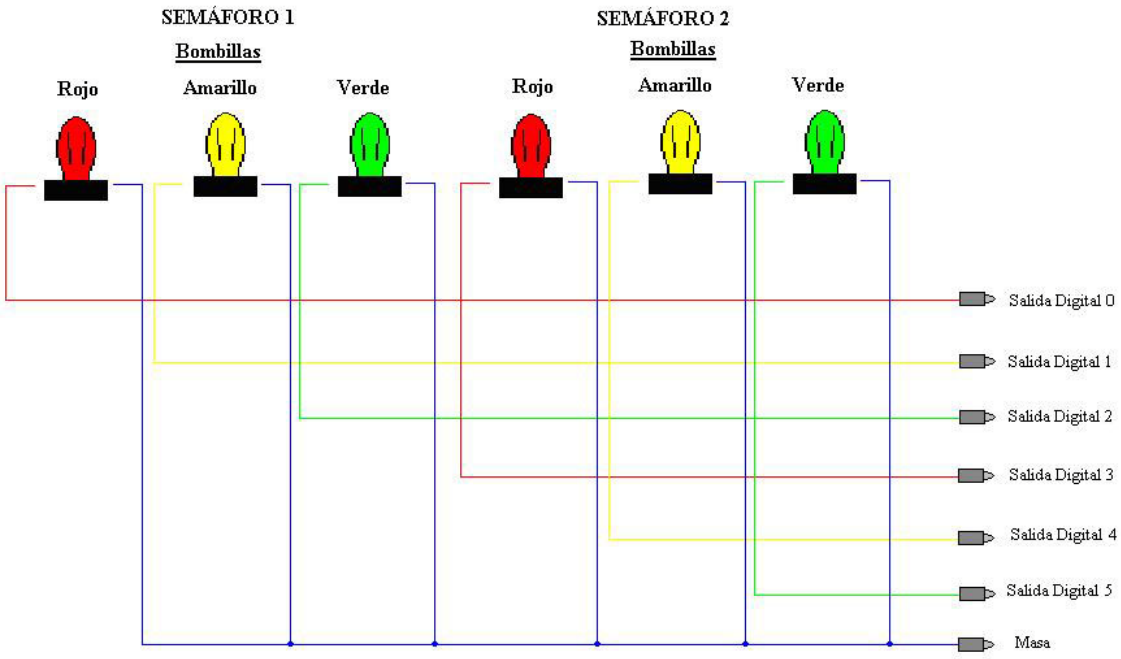


2. Se sitúan los semáforos (1) sobre la plataforma.



Esquema eléctrico:

Para poder alimentar las bombillas es necesario 1 par de hilos eléctricos para cada una de ellas. Un cable se conectará a las salidas digitales y el otro cable se conectará a masa. La forma en la que se conectan las bombillas de los semáforos a la controladora se muestra en el siguiente esquema:

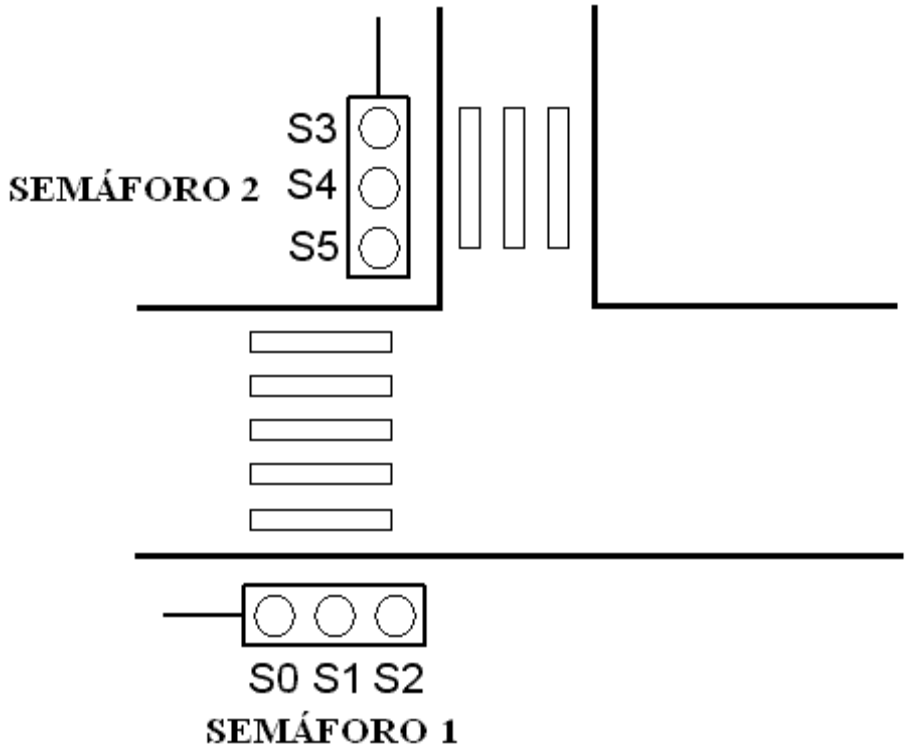


Descripción de los cruces de semáforos

Los conceptos de programación más importantes que se van a manejar con esta serie de ejemplos son el uso de contadores, la captura de eventos a través del teclado y la creación de interfaces. Con respecto a la controladora se aprenderá a manipular sus salidas digitales.

El cruce de semáforos que vamos a simular tendrá las siguientes características:

1. El semáforo_1 regula el tráfico de la calle principal.
2. El semáforo_2 regula el tráfico de la calle secundaria.



Se debe dar mayor prioridad a los coches que circulan por la calle principal que a los coches de la calle secundaria, por ello el semáforo_1 debe permanecer menos tiempo en ROJO que el semáforo_2.

Los tiempos que se han empleado en la tabla que se muestra a continuación son un ejemplo de cómo conseguir la regulación del cruce.

Semáforo_1	correspondencia	Semáforo_2
ROJO	= >	VERDE
ROJO	= >	AMBAR
VERDE	= >	ROJO
AMBAR	= >	ROJO

La secuencia de colores que deben mostrar los dos semáforos es la siguiente:

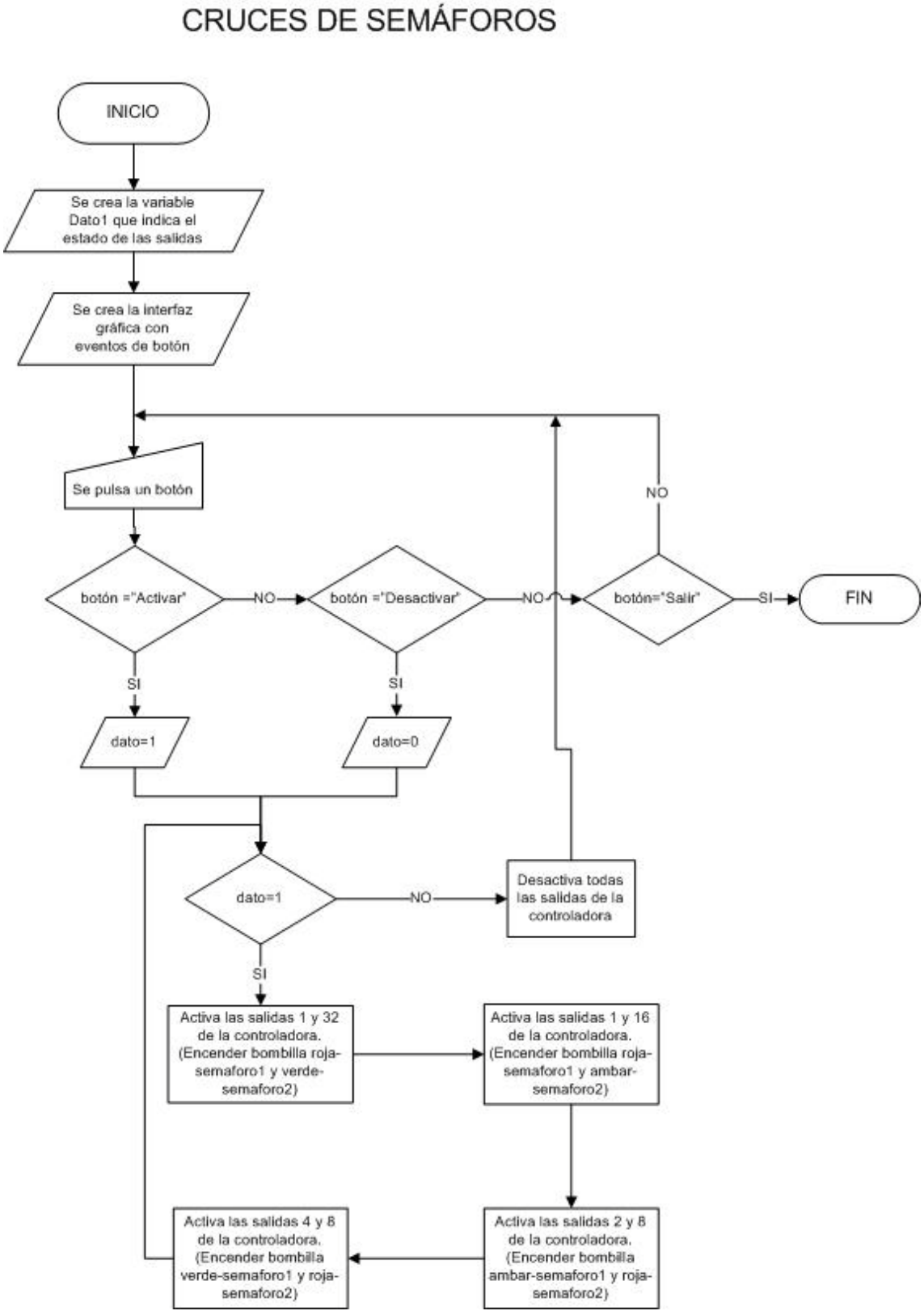
1. ROJO_1 - VERDE_2 (R1_V2)
2. ROJO_1 - AMBAR_2 (R1_A2)
3. VERDE_1 – ROJO_2 (V1_R2)
4. AMBAR_1 – ROJO_2 (A1_R2)

En todos los programas de cruces de semáforos para comprobar el funcionamiento en la controladora se deberán emplear seis salidas digitales, 3 para cada semáforo, en concreto:

- S0 = > ROJO del semáforo_1
- S1 = > ÁMBAR del semáforo_1
- S2 = > VERDE del semáforo_1
- S3 = > ROJO del semáforo_2
- S4 = > ÁMBAR del semáforo_2
- S5 = > VERDE del semáforo_2

PROGRAMACIÓN

Diagrama de flujo



Programación en MSWLogo

La programación de este caso se estructura en los siguientes pasos:

1. Se crea un procedimiento para crear la ventana gráfica principal del programa. Para ello se utiliza la función **creaventana**. Dentro de la ventana se crean los botones con la función **creaboton**. Dentro de cada botón se establecerán entre los corchetes las funciones que se han de ejecutar una vez presionado el botón. Uno de los botones creados será el que se utiliza para salir de la aplicación, para lo cual se utiliza la orden adios. Además se mostrarán las imágenes de la barrera con la función **cargadib**.

```
cargadib("blancoblanco.bmp)
creaventana " "Principal [Cruce de Semaforos ] 100 42 150 100 []
creaboton "Principal "Activar "Activar 18 20 50 20 [Luces]
creaboton "Principal "Desactivar "Desactivar 80 20 50 20 [proc_salir]
creaboton "Principal "Salir "Salir 50 50 50 20 [proc_salir adios]
```

En este procedimiento se crea y se inicializa a 0 una variable que llamaremos **bucle** que nos servirá para poder leer las entradas digitales de manera continuada.

2. El procedimiento **Luces** hace las llamadas correspondientes al procedimiento **Encender** pasándole como parámetros el color de la luz y el tiempo que se quiere tener encendida cada luz. Se va a establecer como tiempo de encendido de cada luz 3 segundos excepto el parpadeo del color ámbar que serán 5 veces. Con estos parámetros determina qué salida o salidas digitales se deben encender en cada momento.

```
PARA Encender :luz :tiempo
  SISINO (:luz = "ROJOVERDE) [SALIDA 33 cargadib("rojoverde.bmp) ESPERA :tiempo ][
    SISINO (:luz = "ROJOAMBAR) [Parpadear :tiempo 1 17 "rojoblanco.bmp
"rojoambar.bmp] [
      SISINO (:luz = "VERDEROJO) [SALIDA 12 cargadib("verderojo.bmp) ESPERA :tiempo]
[
       SI (:luz = "AMBARROJO) [Parpadear :tiempo 8 10 "blancorojo.bmp "ambarrojo.bmp]
      ]
    ]
  ]
FIN
```

3. El procedimiento **Parpadear** hace que la salida digital S1 y S4, que es donde deben conectarse las luces ámbar de los dos semáforos, se enciendan y apaguen alternativamente cuando les corresponda. El tiempo que debe permanecer encendida/apagada la luz ámbar depende del valor del parámetro tiempo:

```
PARA Parpadear :tiempo :salida1 :salida2 :foto1 :foto2
;Procedimiento que hace parpadear la luz ÁMBAR 5 veces
  DESDE [i 1 5 1] [SALIDA :salida1 cargadib(:foto1) ESPERA (:tiempo/6) SALIDA :salida2
cargadib(:foto2) ESPERA (:tiempo/6)]
FIN
```

4. Se llama al procedimiento que crea la ventana gráfica fuera de cualquier procedimiento para que se cargue la aplicación gráfica nada más cargar el fichero de logo.

Descargar el archivo programado en **MSWLogo**, descomprímalo y guárdelo en un directorio aparte. Contiene el fichero de código en MSWLogo (cruce.lgo y las imágenes de los semaforos).

Ejecute el compilador **MSWLogo versión 6.5a** en castellano.

Vaya al menú del programa, Archivo/Abrir y seleccione el fichero **cruce.lgo** que se descargó previamente.

Se visualizará la siguiente pantalla:

Programación en C

La programación de este caso se estructura en los siguientes pasos:

1.- Se crea un nuevo proyecto

2.- Se añaden al proyecto los archivos io.h, io.cpp, Primitivas_CNICE.CPP y Primiticas_CNICE.HPP

3.- Se crea el archivo main.c donde se incluirán las funciones necesarias para crear las ventanas

4.- Dentro del archivo main.c creado anteriormente, se añade la declaración a las funciones de la biblioteca io.dll de la siguiente manera:

```
#include "io.h"
```

También se añade la declaración a las funciones de la biblioteca SDL.dll de la siguiente manera:

```
#include <SDL.h>
```

5.- En nuestro archivo main.c se define la siguiente función que permite activar o desactivar las entradas digitales de la controladora.

```
void Escribir_Salidas_Digitales(int Dato)
{
   LoadIODLL();
   PortOut(0x37A,0x7);
   PortOut(0x378,Dato);
}
```

6.- Se crearán dos botones en nuestra ventana, uno para activar el funcionamiento del cruce y otro para detenerlo. También hay un botón para salir de la aplicación.

7.- Para activar el funcionamiento del cruce se crea un hilo de control, para así detener el funcionamiento cuando se pulse el botón “Desactivar”. La creación del hilo se hace de la siguiente manera:

```
Hilo_Control=CreateThread(NULL, 0, Activar, 0,0, NULL);
if (Hilo_Control==NULL)
{
        MessageBox(0,"Error al crear el hilo","Welcome Message",1);
        exit(0);
 }
```

8.- La función que ejecuta el hilo de control que se ha creado ejecutará la secuencia a seguir por los semáforos que forman el cruce. Para ello, se hacen llamadas a la función Escribir_Salidas_Digitales definida anteriormente. Hay que tener en cuenta que he pueden activar dos salidas a la vez, por lo que debemos pasar a la función Escribir_Salidas_Digitales el valor de la suma de las salidas que queremos que se activen. La función Activar será de la siguiente manera:

```
DWORD WINAPI Activar(LPVOID)
{
    SDL_Surface *rojoverde, *verderojo, *rojoblanco, *blancorojo, *rojoambar, *ambarrojo;
    SDL_Surface *screen;
    SDL_Rect rect;
    for(;;){
       if(dato==1){
            rojoverde = SDL_LoadBMP("rojoverde.bmp");
            screen = SDL_SetVideoMode( 70, 220, 0, SDL_NOFRAME );
            if( screen == NULL ) {
               printf( "Error al entrar a modo grafico: %s\n", SDL_GetError() );
               SDL_Quit();
            }
            rect = (SDL_Rect) {0, 0, 200, 200};
            SDL_BlitSurface(rojoverde, NULL,screen,&rect);
            SDL_Flip(screen);
            Escribir_Salidas_Digitales(33);
            Sleep(1800);
            for(int i=0; i<5;i++){
                 rojoblanco = SDL_LoadBMP("rojoblanco.bmp");
                 screen = SDL_SetVideoMode( 70, 220, 0, SDL_NOFRAME );
                 if( screen == NULL ) {
                    printf( "Error al entrar a modo grafico: %s\n", SDL_GetError() );
                    SDL_Quit();
                 }
                 rect = (SDL_Rect) {0, 0, 200, 200};
                 SDL_BlitSurface(rojoblanco, NULL,screen,&rect);
                 SDL_Flip(screen);
                 Escribir_Salidas_Digitales(1);
                 Sleep(500);

                 rojoambar = SDL_LoadBMP("rojoambar.bmp");
                 screen = SDL_SetVideoMode( 70, 220, 0, SDL_NOFRAME );
                 if( screen == NULL ) {
                    printf( "Error al entrar a modo grafico: %s\n", SDL_GetError() );
                    SDL_Quit();
                 }
                 rect = (SDL_Rect) {0, 0, 200, 200};
                 SDL_BlitSurface(rojoambar, NULL,screen,&rect);
                 SDL_Flip(screen);
                 Escribir_Salidas_Digitales(17);
                 Sleep(500);
            }

            verderojo = SDL_LoadBMP("verderojo.bmp");
```

```
                screen = SDL_SetVideoMode( 70, 220, 0, SDL_NOFRAME );
                if( screen == NULL ) {
                   printf( "Error al entrar a modo grafico: %s\n", SDL_GetError() );
                   SDL_Quit();
                }
                rect = (SDL_Rect) {0, 0, 200, 200};
                SDL_BlitSurface(verderojo, NULL,screen,&rect);
                SDL_Flip(screen);
                Escribir_Salidas_Digitales(12);
                Sleep(1800);
                for(int i=0; i<5;i++){
                     blancorojo = SDL_LoadBMP("blancorojo.bmp");
                     screen = SDL_SetVideoMode( 70, 220, 0, SDL_NOFRAME );
                     if( screen == NULL ) {
                        printf( "Error al entrar a modo grafico: %s\n", SDL_GetError() );
                        SDL_Quit();
                     }
                     rect = (SDL_Rect) {0, 0, 200, 200};
                     SDL_BlitSurface(blancorojo, NULL,screen,&rect);
                     SDL_Flip(screen);
                     Escribir_Salidas_Digitales(8);
                     Sleep(500);

                     ambarrojo = SDL_LoadBMP("ambarrojo.bmp");
                     screen = SDL_SetVideoMode( 70, 220, 0, SDL_NOFRAME );
                     if( screen == NULL ) {
                        printf( "Error al entrar a modo grafico: %s\n", SDL_GetError() );
                        SDL_Quit();
                     }
                     rect = (SDL_Rect) {0, 0, 200, 200};
                     SDL_BlitSurface(ambarrojo, NULL,screen,&rect);
                     SDL_Flip(screen);
                     Escribir_Salidas_Digitales(10);
                     Sleep(500);
                }
            }  else
                Escribir_Salidas_Digitales(0);
        }
    }
```

Explicación de las funciones de la librería SDL.

Cada vez que se quiera mostrar una imagen se indicara en nuestro proyecto lo siguiente:

```
                foto = SDL_LoadBMP("imagen.bmp");
                screen = SDL_SetVideoMode(200, 308, 0, SDL_NOFRAME );
                if( screen == NULL ) {
                printf( "Error al entrar a modo grafico: %s\n", SDL_GetError() );
                            SDL_Quit();
                            return -1;
                }
                rect.x=0;
                rect.y=0;
                rect.w=primera1->w;
                rect.h=primera1->h;
                destino.x=0;
                destino.y=0;
                SDL_BlitSurface(primera1, &rect, screen, &destino);
                SDL_Flip(screen);
```

donde **foto** y **screen** son del tipo SDL_Surface y **rect** y **destino** es del tipo SDL_Rect.

- SDL_LoadBMP: carga la imagen .bmp que queramos

- SDL_SetVideoMode (int width, int height, int bpp, Uint32 flags): configura un modo de video con una anchura (width), una altura (height) y unos bits-por-pixeles. El parámetro flags indica el tipo de ventana que se quiere. En nuestro caso una ventana sin titulo no borde.
- SDL_BlitSurface(imagen, &rect, screen, &destino): pega desde la imagen, la porción seleccionada por rect sobre la superficie screen en el destino indicado por destino.
- SDL_Flip(screen): muestra la imagen que se ha seleccionado.

7.- Una vez creados los botones con la función que les corresponden, se compila comprobando que no hay ningún error.

8.- Una vez que se ha comprobado que no hay ningún error en nuestro código, se ejecuta y se comprueba el funcionamiento del cruce. Al ejecutar el proyecto se creará el fichero **cruce_semaforos.exe**

Descargue los diferentes archivos que forman todo el proyecto programado en C, descomprímalos y guárdelos en un directorio aparte. Ejecute el fichero **cruce_semaforos.exe**. Se visualizará la siguiente pantalla:

Nota:

En la aplicación programada con C, la imagen puede no aparecer al lado de la ventana. En este caso basta con mover nuestra ventana y se verá correctamente la imagen.