Área de Educación Tecnológica
Educación Tecnológica II
2 Año

Sistema Intersección de Calles
Manual de Instrucciones

Alumno:

Curso:

Índice

1. Cómo escribir un nuevo programa

Para comenzar con un nuevo programa, deberemos seleccionar:

Archivo ► Nuevo

Al hacer esto, aparece un código vacío con lo siguiente:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figura 1: Código vacío (nuevo programa).

Para todo programa que hagamos, siempre se deberán realizar los siguientes pasos previos:

- Incluir la librería: Para usar las funciones/instrucciones de la librería, es necesario incluirla al principio del programa. Para esto, seleccionar:

  Menú Programa ► Incluir Librería ► LibSemaforo

  O bien, escribir esta línea al comienzo:

  #include <LibSemaforo.h>

- Inicialización: Dentro de "setup", escribir la instrucción inicializar_sistema();

Instrucción	**Descripción**
inicializar_sistema();	Esta instrucción es la que configura e inicializa al sistema. La misma deberá escribirse en la sección "setup" del código.

Una vez hecho esto, tendremos lo siguiente:

```
#include <LibSemaforo.h>

void setup() {
  // put your setup code here, to run once:
  inicializar_sistema();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figura 2: Código vacío con pasos previos.

2. Comentarios

Los comentarios son líneas en el programa que se usan para informar a uno mismo o a otros sobre la manera en la que el programa trabaja. Los comentarios son ignorados por el compilador (no producen ningún tipo de error), no formarán parte del código de máquina. Es decir, no ocupan ningún espacio en la memoria del controlador.

El propósito de los comentarios es ayudar a entender (o recordar), o informar a otros sobre qué es lo que nuestro código realiza.

Todo comentario comienza con una doble barra (//) y termina cuando se cambia a la siguiente línea de código.

```
//Esto es un comentario.

//Esto también.
```

Figura 3: Ejemplo de comentarios.

3. Manejo de Actuadores

- **Semáforo 1**

Instrucción	Descripción
semaforo1.encender(ROJO);	Enciende luz roja del semáforo 1.
semaforo1.encender(AMARILLO);	Enciende luz amarilla del semáforo 1.
semaforo1.encender(VERDE);	Enciende luz verde del semáforo 1.
semaforo1.apagar(ROJO);	Apaga luz roja del semáforo 1.
semaforo1.apagar(AMARILLO);	Apaga luz amarilla del semáforo 1.
semaforo1.apagar(VERDE);	Apaga luz verde del semáforo 1.

- **Semáforo 2**

Instrucción	Descripción
semaforo2.encender(ROJO);	Enciende luz roja del semáforo 2.
semaforo2.encender(AMARILLO);	Enciende luz amarilla del semáforo 2.
semaforo2.encender(VERDE);	Enciende luz verde del semáforo 2.
semaforo2.apagar(ROJO);	Apaga luz roja del semáforo 2.
semaforo2.apagar(AMARILLO);	Apaga luz amarilla del semáforo 2.
semaforo2.apagar(VERDE);	Apaga luz verde del semáforo 2.

- **Buzzer 1**

Instrucción	Descripción
buzzer1.encender();	Enciende buzzer 1.
buzzer1.apagar();	Apaga buzzer 1.

- **Buzzer 2**

Instrucción	Descripción
buzzer2.encender();	Enciende buzzer 2.
buzzer2.apagar();	Apaga buzzer 2.

- **Luz de Calle**

Instrucción	Descripción
luz_calle.encender();	Enciende luz de calle.
luz_calle.apagar();	Apaga la luz de calle.

4. Espera de Tiempo

Instrucción	Descripción
delay(TIME);	Espera de tiempo de TIME milisegundos. Se debe reemplazar la palabra TIME por la cantidad de milisegundos a esperar.

Ejemplo: delay(1000); es una espera de tiempo de 1 segundo.

5. Declaración de Variables

Una variable es un elemento que usamos para guardar datos en memoria. Podemos pensar a la memoria como un armario con cajones. Cada uno de estos cajones es una variable, y dentro de cada uno de estos cajones podemos guardar un dato distinto.

Dentro de Arduino, existen distintos tipos de variables. Nosotros solo vamos a usar las variables de tipo entero (integer en inglés). Un entero es un número que puede tomar los valores 0, 1, 2, 3, etc. De la manera en la que lo utilizaremos, el número máximo que podremos guardar es el 32.767 (no es necesario saber el por qué de esto, pero hay que tenerlo en cuenta para los ejercicios que hagamos).

En nuestra librería están definidas las siguientes variables para usar:

- estado_sensor_luz: Para leer el sensor de luz (si es de día o de noche).
- estado_pulsador1: Para leer el pulsador 1.
- estado_pulsador2: Para leer el pulsador 2.
- estado_sensor1: Para leer el sensor de calle 1.
- estado_sensor2: Para leer el sensor de calle 2.
- numero_ingresado: Para ingresar un número o una clave por teclado.
- valor_sensor_luz: Para leer el valor analógico del sensor de luz (avanzado).
- contador: Para llevar la cuenta de algún evento.

6. Condicional

Antes de ver las instrucciones de los sensores, aquí una explicación del condicional en Arduino. Cuando se lee el valor de un sensor y se lo guarda en una de las variables de arriba, hay que preguntar por el valor de esa variable. Para esto usamos la estructura if / else.

Instrucción	**Descripción**

if(CONDICIÓN)
{
//Instrucciones a ejecutar si se
//cumple la condición
}
else
{
//Instrucciones a ejecutar si no se
//cumple la condición
}
	Si se cumple la condición, va a ejecutar las instrucciones que se encuentran entre llaves a continuación de "If".
	En caso de que la condición no se cumpla, se ejecutarán las instrucciones que se encuentran entre llaves luego del "else".
	A esta estructura se la conoce como If/Else.

Condición:

Las condiciones se pueden armar con los operadores de igualdad, desigualdad, mayor y menor. A continuación, se explica cada uno.

- x == y (x es igual a y)
- x != y (x no es igual a y)
- x < y (x es menor que y)
- x > y (x es mayor que y)
- x <= y (x es menor o igual que y)
- x >= y (x es mayor o igual que y)
- && (para hacer un AND de dos condiciones, que se cumplan las dos al mismo tiempo).
- || (para hacer un OR de dos condiciones, que se cumpla al menos una de las dos).

7. Manejo de Sensores y elementos de ingreso de datos (EID)

Para utilizar los sensores deberemos leer el estado o valor del sensor y guardarlo en una variable ya declarada, o una que nosotros declaremos al principio del programa. Para esto vamos a usar las variables que se mencionaban en el punto 5.

- **Pulsador 1**

Instrucción	**Descripción**
estado_pulsador1 = pulsador1.leer();	Lee el valor del pulsador 1 y lo guarda en "estado_pulsador1".

Los valores posibles del Pulsador 1 son:

Valor del Pulsador 1	**Descripción**
PRESIONADO	El pulsador 1 ha sido presionado.
NO_PRESIONADO	El pulsador 1 no ha sido presionado.

Todo esto lo mezclamos con la estructura if / else, como se muestra en la siguiente figura:

```
estado_pulsador1 = pulsador1.leer();

if(estado_pulsador1 == PRESIONADO)
{
  //Instrucciones a ejecutar si se presiona el pulsador 1.
}
else
{
  //Instrucciones a ejecutar si no se presiona el pulsador 1.
}
```

Figura 4: Lectura del pulsador 1 y uso del condicional.

- **Pulsador 2**

Instrucción	Descripción
estado_pulsador2 = pulsador2.leer();	Lee el valor del pulsador 2 y lo guarda en "estado_pulsador2".

Los valores posibles del Pulsador 2 son:

Valor del Pulsador 2	Descripción
PRESIONADO	El pulsador 2 ha sido presionado.
NO_PRESIONADO	El pulsador 2 no ha sido presionado.

```
estado_pulsador2 = pulsador2.leer();

if(estado_pulsador2 == PRESIONADO)
{
  //Instrucciones a ejecutar si se presiona el pulsador 2.
}
else
{
  //Instrucciones a ejecutar si no se presiona el pulsador 2.
}
```

Figura 5: Lectura del pulsador 2 y uso del condicional.

- **Sensor de Calle 1**

Instrucción	Descripción
estado_sensor1 = sensor_calle1.leer();	Lee el valor del sensor de calle 1 y lo guarda en "estado_sensor1".

Los valores posibles que devuelve la función "leer" del Sensor de Calle 1 son:

Valor del Sensor de Calle 1	Descripción
ACTIVADO	Sensor detecta presencia de un objeto.
DESACTIVADO	Sensor detecta ausencia de un objeto.

```
estado_sensor1 = sensor_calle1.leer();

if(estado_sensor1 == ACTIVADO)
{
  //Instrucciones a ejecutar si se activa el sensor de calle 1.
}
else
{
  //Instrucciones a ejecutar si no se activa el sensor de calle 1.
}
```

Figura 6: Lectura del sensor de calle 1 y uso del condicional.

- **Sensor de Calle 2**

Instrucción	Descripción
estado_sensor2 = sensor_calle2.leer();	Lee el valor del sensor de calle 2 y lo guarda en “estado_sensor2”.

Los valores posibles que devuelva la función “leer” del Sensor de Calle 2 son:

Valor del Sensor de Calle 2	Descripción
ACTIVADO	Sensor detecta presencia de un objeto.
DESACTIVADO	Sensor detecta ausencia de un objeto.

```
estado_sensor2 = sensor_calle2.leer();

if(estado_sensor2 == ACTIVADO)
{
  //Instrucciones a ejecutar si se activa el sensor de calle 2.
}
else
{
  //Instrucciones a ejecutar si no se activa el sensor de calle 2.
}
```

Figura 7: Lectura del sensor de calle 1 y uso del condicional.

- **Sensor de Luz**

Instrucción	Descripción
estado_sensor_luz = sensor_luz.leer();	Lee el valor del sensor de luz.

Los valores posibles del Sensor de Luz son:

Valor del Sensor de Luz	Descripción
DIA	Sensor detecta que es de día.
NOCHE	Sensor detecta que es de noche.

```
estado_sensor_luz = sensor_luz.leer();

if(estado_sensor_luz == DIA)
{
  //Instrucciones a ejecutar si el sensor detecta que es de dia.
}
else //Si el sensor lee que es de noche
{
  //Instrucciones a ejecutar si el sensor detecta que es de noche.
}
```

Figura 8: Lectura del sensor de luz y uso del condicional.

7.1. Funciones de “espera” de Sensores y EID.

- Instrucciones de espera del pulsador 1

Instrucción	Descripción
pulsador1.esperar(PRESIONADO);	El programa se queda leyendo el pulsador hasta que este se encuentre en el estado “PRESIONADO”.
pulsador1.esperar(NO_PRESIONADO);	El programa se queda leyendo el pulsador hasta que este se encuentre en el estado “NO_PRESIONADO”.

- Instrucciones de espera del pulsador 2

Instrucción	Descripción
pulsador2.esperar(PRESIONADO);	El programa se queda leyendo el pulsador hasta que este se encuentre en el estado “PRESIONADO”.
pulsador2.esperar(NO_PRESIONADO);	El programa se queda leyendo el pulsador hasta que este se encuentre en el estado “NO_PRESIONADO”.

- Instrucciones de espera del sensor de calle 1

Instrucción	Descripción
sensor_calle1.esperar(ACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “ACTIVADO”.
sensor_calle1.esperar(DESACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “DESACTIVADO”.

- Instrucciones de espera del sensor de calle 2

Instrucción	Descripción
sensor_calle2.esperar(ACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “ACTIVADO”.
sensor_calle2.esperar(DESACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “DESACTIVADO”.

- Instrucciones de espera de sensor de luz

Instrucción	Descripción
sensor_luz.esperar(DIA);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “DIA”.
sensor_luz.esperar(NOCHE);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “NOCHE”.

- Lectura analógica del Sensor de Luz (avanzado)

En el caso del sensor de luz, también existe una función para leer el valor analógico que devuelve el sensor, según la luminosidad ambiente. Si hay oscuridad total, el valor del sensor será 0 y con una gran cantidad de luz, el valor será 1023.

Instrucción	Descripción
valor_sensor_luz = sensor_luz.leer_analogico();	Lee el valor del sensor de luz. Esta lectura es analógica. Devuelve un valor entre 0 y 1023 dependiendo de la luminosidad del ambiente.

8. Ingreso y egreso de datos por Monitor Serie

El monitor serie es una ventana que nos permite comunicarnos con el Arduino a través de la computadora. Para abrirlo, seleccionamos Herramientas ► Monitor Serie.

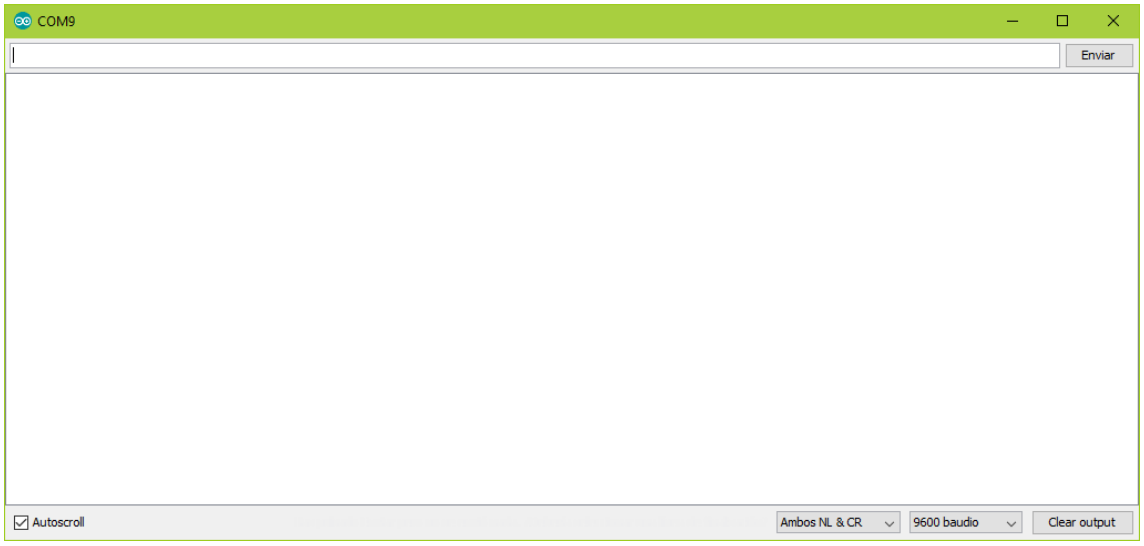

Figura 9: Monitor Serie.

Mostrar un cartel en pantalla

Instrucción	Descripción
mostrar_cartel(STRING);	Esta función permite mostrar un cartel por el monitor serie con texto STRING. Dicho texto deberá escribirse entre comillas dobles (“”).

Ejemplo:

```
#include <LibSemaforo.h>

void setup() {
  // put your setup code here, to run once:
  inicializar_sistema();

  mostrar_cartel("Hola Mundo");
  mostrar_cartel("Esto es un texto en pantalla.");
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figura 10: Ejemplo de mostrar cartel.

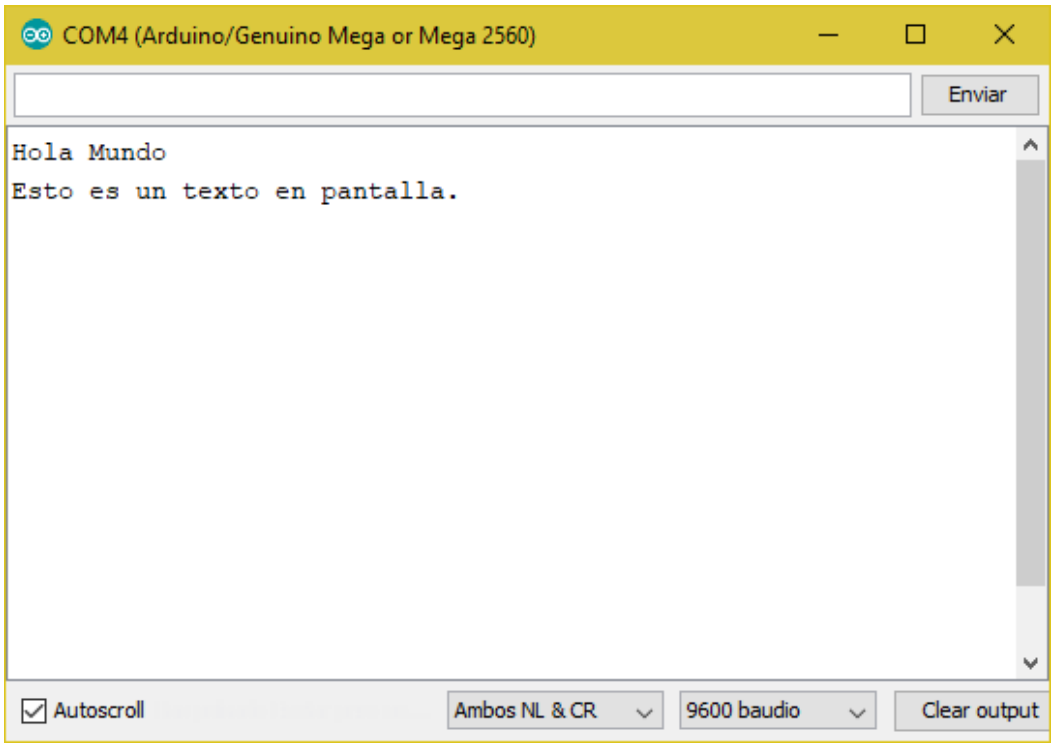

Figura 11: Ejemplo de mostrar cartel en monitor serie.

- **Mostrar un numero en pantalla**

Instrucción	Descripción
mostrar_numero(NUMERO);	Esta función permite mostrar un numero por el monitor serie. Recordar que dicho número debe ser entero.

Ejemplos:

```
#include <LibSemaforo.h>

int num;                //declarar una variable.

void setup() {
  // put your setup code here, to run once:
  inicializar_sistema();

  mostrar_numero(10);   //mostrar "10".

  num=123;              //num vale 123.
  mostrar_numero(num);  //mostrar variable num.
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figura 12: Ejemplo de mostrar número.

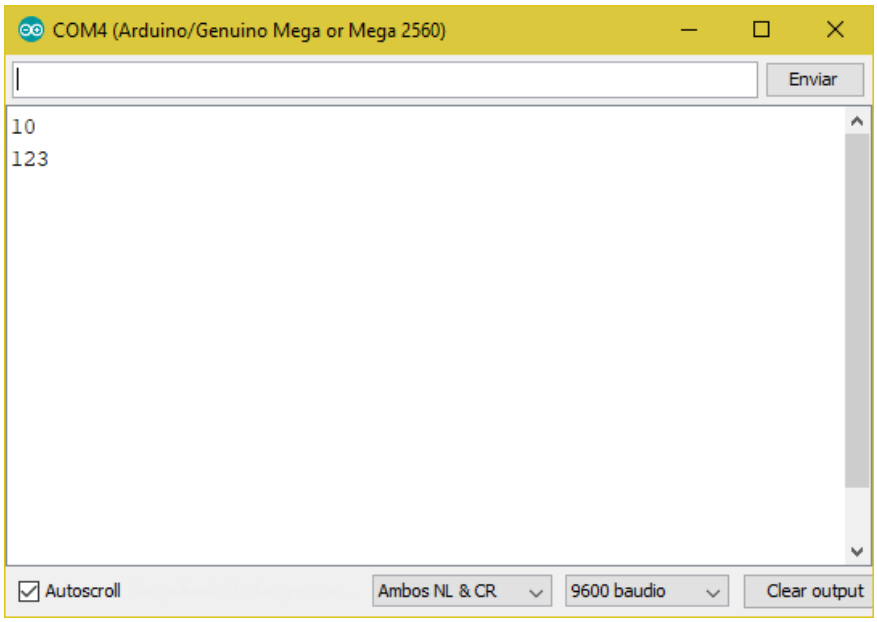

Figura 13: Ejemplo de mostrar número en monitor serie.

- **Ingreso de números**

Instrucción	Descripción
numero_ingresado = ingresar_numero();	Esta función permite leer un número que se ha ingresado por el monitor serie. Ese número se guarda en la variable "numero_ingresado".

Observación: El número ingresado deberá estar entre 0 y 32767. Si se ingresa un número mayor, habrá error.

- **Ejemplo**

En el ejemplo "Ingreso de Números", podemos ver un programa que realiza un eco del número ingresado. Esto significa que se ingresa un número, y el controlador lo muestra por pantalla.

```
#include <LibSemaforo.h>

void setup() {
  //Función de inicialización del sistema.
  inicializar_sistema();
}

void loop() {
  //Imprimir un mensaje por pantalla.
  mostrar_cartel("Ingresar un numero.");

  //Ingresar un numero y almacenarlo en la variable numero.
  numero_ingresado = ingresar_numero();

  //Imprimir mensaje por pantalla.
  mostrar_cartel("Numero ingresado: ");

  //Imprimir el numero ingresado por pantalla.
  mostrar_numero(numero_ingresado);

}
```

Figura 14: Ejemplo de librería, Ingreso de Números.

9. Ciclo de Repetición

Instrucción	Descripción

REPETIR(CANTIDAD)
{
//Instrucciones a repetirse dentro del ciclo.
}
	Las instrucciones que se encuentren entre llaves, se repetirán la cantidad de veces que sea el número entero "CANTIDAD".

Ejemplo:

El siguiente programa enciende y apaga la luz de calle (1 segundo de encendido y 1 segundo de apagado) 5 veces.

```
#include <LibSemaforo.h>

void setup() {
  // put your setup code here, to run once:
  inicializar_sistema();

  REPETIR(5){
    luz_calle.encender();
    delay(1000);
    luz_calle.apagar();
    delay(1000);
  }
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figura 15: Ejemplo de Ciclo de Repetición.

10. Para los más curiosos

Existen otras formas de realizar la repetición y de mostrar datos en el monitor serie.

- **Ciclo de Repetición (versión alternativa)**

Otra forma de realizar un ciclo repetitivo en Arduino, es utilizando el ciclo "for". A continuación, se explica brevemente cómo usarlo. Para más información, buscar en la referencia de Arduino.

Instrucción	Descripción

for(int i=1; i<=CANTIDAD; i++)
{
//Instrucciones a repetirse dentro del ciclo.
}
	Las instrucciones que se encuentren entre llaves, se repetirán la cantidad de veces que sea el número entero "CANTIDAD".

Observación:

El controlador ejecuta las instrucciones a repetir, mientras la variable entera "i" sea menor o igual a "CANTIDAD". La primera vez, "i" vale 1. Cuando llega a la última instrucción dentro del ciclo, se incrementa en 1 y compara su valor contra "CANTIDAD". Esto lo hace tantas veces como diga "CANTIDAD".

- **Mostrar cartel (versión alternativa)**

Instrucción	Descripción
Serial.println("");	Función que permite imprimir un mensaje en pantalla. Este mensaje deberá escribirse entre comillas.

Ejemplo:

Serial.println("Hola Mundo");

Esta instrucción muestra las palabras Hola Mundo por el monitor serie y salta al siguiente renglón.

Instrucción	Descripción
Serial.print("");	Función que permite imprimir un mensaje en pantalla. Este mensaje deberá escribirse entre comillas.

Ejemplo:

Serial.print("Hola Mundo");

Esta instrucción muestra las palabras Hola Mundo por el monitor serie. A diferencia de la instrucción anterior, al finalizar las palabras Hola Mundo, no saltará al siguiente renglón.

- **Mostrar número (versión alternativa)**

Las funciones Serial.print() y Serial.println() también pueden usarse para mostrar números por el monitor serie.

Ejemplo:

```
int numero;

numero = 1234;

Serial.print(“El número es: ”);

Serial.println(numero);
```

Escribiendo este código, se verá lo siguiente por el monitor serie:

- El número es: 1234