Área de Educación Tecnológica
Educación Tecnológica II
2 Año

Sistema Estacionamiento

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:

1. 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 ► LibEstacionamiento

   O bien, escribir esta línea al comienzo:

   #include <LibEstacionamiento.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 <LibEstacionamiento.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

Por cada espacio para estacionar, existen dos luces (LEDs): roja y verde. En total hay 8 LEDs: 4 rojos y 4 verdes. Existen instrucciones para encender y apagar cada LED.

- **Luz Auto 1**

Instrucción	**Descripción**
luz_auto1.encender(ROJO);	Enciende luz roja del espacio 1.
luz_auto1.encender(VERDE);	Enciende luz verde del espacio 1.
luz_auto1.apagar(ROJO);	Apaga luz roja del espacio 1.
luz_auto1.apagar(VERDE);	Apaga luz verde del espacio 1.

- **Luz Auto 2**

Instrucción	**Descripción**
luz_auto2.encender(ROJO);	Enciende luz roja del espacio 2.
luz_auto2.encender(VERDE);	Enciende luz verde del espacio 2.
luz_auto2.apagar(ROJO);	Apaga luz roja del espacio 2.
luz_auto2.apagar(VERDE);	Apaga luz verde del espacio 2.

- **Luz Auto 3**

Instrucción	**Descripción**
luz_auto2.encender(ROJO);	Enciende luz roja del espacio 2.
luz_auto2.encender(VERDE);	Enciende luz verde del espacio 2.
luz_auto2.apagar(ROJO);	Apaga luz roja del espacio 2.
luz_auto2.apagar(VERDE);	Apaga luz verde del espacio 2.

- **Luz Auto 4**

Instrucción	**Descripción**
luz_auto2.encender(ROJO);	Enciende luz roja del espacio 2.
luz_auto2.encender(VERDE);	Enciende luz verde del espacio 2.
luz_auto2.apagar(ROJO);	Apaga luz roja del espacio 2.
luz_auto2.apagar(VERDE);	Apaga luz verde del espacio 2.

Barreras

Tanto en la entrada como en la salida, hay motores que manejan sendas barreras para permitir el ingreso y egreso de autos al estacionamiento.

- **Barrera de Entrada**

Instrucción	**Descripción**
subir_barrera_entrada();	Sube la barrera de entrada.
bajar_barrera_entrada();	Baja la barrera de entrada.

- Barrera de Salida

Instrucción	**Descripción**
subir_barrera_salida();	Sube la barrera de salida.
bajar_barrera_salida();	Baja la barrera de salida.

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. 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 roja del espacio de auto 1 (1 segundo de encendido y 1 segundo de apagado) 5 veces.

```
#include <LibEstacionamiento.h>

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

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

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

}
```

Figura 4: Ejemplo de Ciclo de Repetición.

6. 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_entrada: Para leer el sensor de entrada.
- estado_salida: Para leer el sensor de salida.
- estado_auto1: Para leer el sensor de auto 1.
- estado_auto2: Para leer el sensor de auto 2.
- estado_auto3: Para leer el sensor de auto 3.
- estado_auto4: Para leer el sensor de auto 4.
- estado_pulsador: Para leer el pulsador.
- numero_ingresado: Para ingresar un número o una clave por teclado.
- contador: Para llevar la cuenta de algún evento.

7. Condicional

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).

8. 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 6 (“Declaración de Variables”).

- **Pulsador**

Instrucción	**Descripción**
estado_pulsador = pulsador.leer();	Lee el valor del pulsador y lo guarda en “estado_pulsador”.

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_pulsador = pulsador.leer();

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

Figura 5: Lectura del pulsador y uso del condicional.

- **Sensor de Auto 1**

Instrucción	Descripción
estado_auto1=sensor_auto1.leer();	Lee el valor del sensor de auto 1 y lo guarda en “estado_auto1”.

Los valores posibles del Sensor de Auto 1 son:

Valor del Sensor de Auto 1	Descripción
ACTIVADO	Sensor detecta presencia de un auto.
DESACTIVADO	Sensor detecta ausencia de un auto.

```
estado_auto1 = sensor_auto1.leer();

if(estado_auto1 == ACTIVADO)
{
  //Instrucciones a ejecutar si se activa el sensor 1.
}
else
{
  //Instrucciones a ejecutar si no se activa el sensor 1.
}
```

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

- **Sensor de Auto 2**

Instrucción	Descripción
estado_auto2 = sensor_auto2.leer();	Lee el valor del sensor de auto 2 y lo guarda en “estado_auto2”.

Los valores posibles del Sensor de Auto 1 son:

Valor del Sensor de Auto 2	Descripción
ACTIVADO	Sensor detecta presencia de un auto.
DESACTIVADO	Sensor detecta ausencia de un auto.

```
estado_auto2 = sensor_auto2.leer();

if(estado_auto2 == ACTIVADO)
{
  //Instrucciones a ejecutar si se activa el sensor 2.
}
else
{
  //Instrucciones a ejecutar si no Dse activa el sensor 2.
}
```

Figura 7: Lectura del sensor 2 y uso del condicional.

- **Sensor de Auto 3**

Instrucción	Descripción
estado_auto3 = sensor_auto3.leer();	Lee el valor del sensor de auto 3 y lo guarda en “estado_auto3”.

Los valores posibles del Sensor de Auto 3 son:

Valor del Sensor de Auto 3	Descripción
ACTIVADO	Sensor detecta presencia de un auto.
DESACTIVADO	Sensor detecta ausencia de un auto.

```
estado_auto3 = sensor_auto3.leer();

if(estado_auto3 == ACTIVADO)
{
  //Instrucciones a ejecutar si se activa el sensor 3.
}
else
{
  //Instrucciones a ejecutar si no se activa el sensor 3.
}
```

Figura 8: Lectura del sensor 3 y uso del condicional.

- **Sensor de Auto 4**

Instrucción	Descripción
estado_auto4 = sensor_auto4.leer();	Lee el valor del sensor de auto 4 y lo guarda en “estado_auto4”.

Los valores posibles del Sensor de Auto 4 son:

Valor del Sensor de Auto 4	Descripción
ACTIVADO	Sensor detecta presencia de un auto.
DESACTIVADO	Sensor detecta ausencia de un auto.

```
estado_auto4 = sensor_auto4.leer();

if(estado_auto4 == ACTIVADO)
{
  //Instrucciones a ejecutar si se activa el sensor 4.
}
else
{
  //Instrucciones a ejecutar si no se activa el sensor 4.
}
```

Figura 9: Lectura del sensor 4 y uso del condicional.

Sensores de presencia en las barreras

- **Sensor Barrera de Entrada**

Instrucción	Descripción
estado_entrada = sensor_barrera_entrada.leer();	Lee el valor del sensor de la barrera de entrada.

Los valores posibles del Sensor de la barrera de entrada son:

Valor del Sensor de barrera de entrada	Descripción
ACTIVADO	Sensor detecta presencia de un auto.
DESACTIVADO	Sensor detecta ausencia de un auto.

```
estado_entrada = sensor_barrera_entrada.leer();

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

Figura 10: Lectura del sensor de entrada y uso del condicional.

- **Sensor Barrera de Salida**

Instrucción	Descripción
estado_salida = sensor_barrera_salida.leer();	Lee el valor del sensor de la barrera de salida.

Los valores posibles del Sensor de la barrera de salida son:

Valor del Sensor de barrera de entrada	Descripción
ACTIVADO	Sensor detecta presencia de un auto.
DESACTIVADO	Sensor detecta ausencia de un auto.

```
estado_salida = sensor_barrera_salida.leer();

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

Figura 11: Lectura del sensor de salida y uso del condicional.

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

- Instrucciones de espera del pulsador

Instrucción	Descripción
pulsador.esperar(PRESIONADO);	El programa se queda leyendo el pulsador hasta que este se encuentre en el estado “PRESIONADO”.
pulsador.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 auto 1

Instrucción	Descripción
sensor_auto1.esperar(ACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “ACTIVADO”.
sensor_auto1.esperar(DESACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “DESACTIVADO”.

- Instrucciones de espera del sensor de auto 2

Instrucción	Descripción
sensor_auto2.esperar(ACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “ACTIVADO”.
sensor_auto2.esperar(DESACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “DESACTIVADO”.

- Instrucciones de espera del sensor de auto 3

Instrucción	Descripción
sensor_auto3.esperar(ACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “ACTIVADO”.
sensor_auto3.esperar(DESACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “DESACTIVADO”.

- Instrucciones de espera del sensor de auto 4

Instrucción	Descripción
sensor_auto4.esperar(ACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “ACTIVADO”.
sensor_auto4.esperar(DESACTIVADO);	El programa se queda leyendo el sensor hasta que este se encuentre en el estado “DESACTIVADO”.

- Instrucciones de espera del sensor de barrera de entrada

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

- Instrucciones de espera del sensor de barrera de salida

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

9. 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.

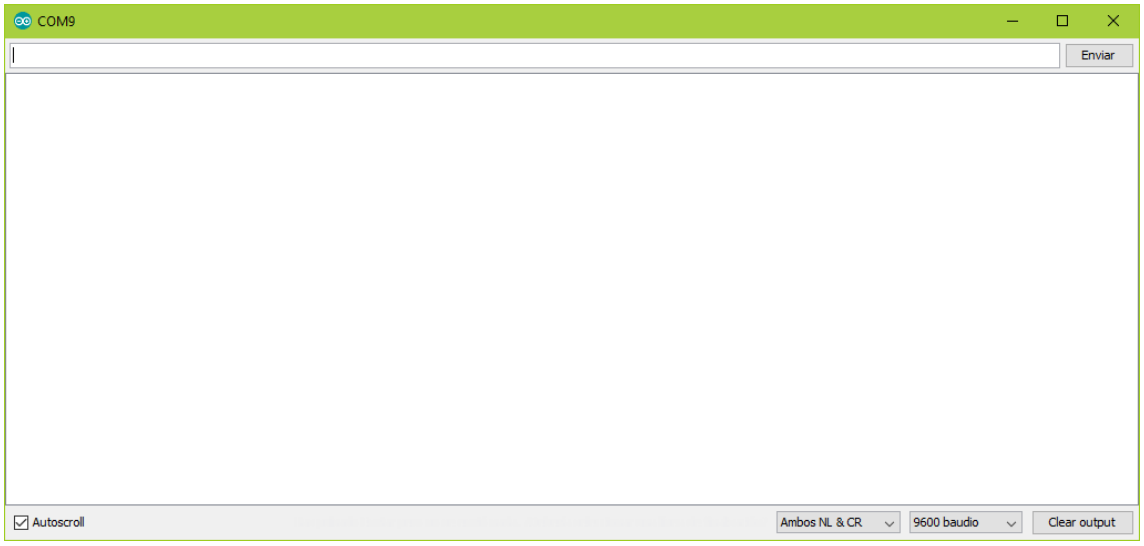

Figura 12: Monitor Serie.

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 (“”).
“\n”	Agregar al final del string para poner un salto de línea

Ejemplo:

```
#include <LibEstacionamiento.h>

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

  //Agregamos \n para poner un salto de linea.
  mostrar_cartel("Hola Mundo\n");
  mostrar_cartel("Esto es un texto en pantalla.\n");
}

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

}
```

Figura 13: Ejemplo de mostrar cartel.

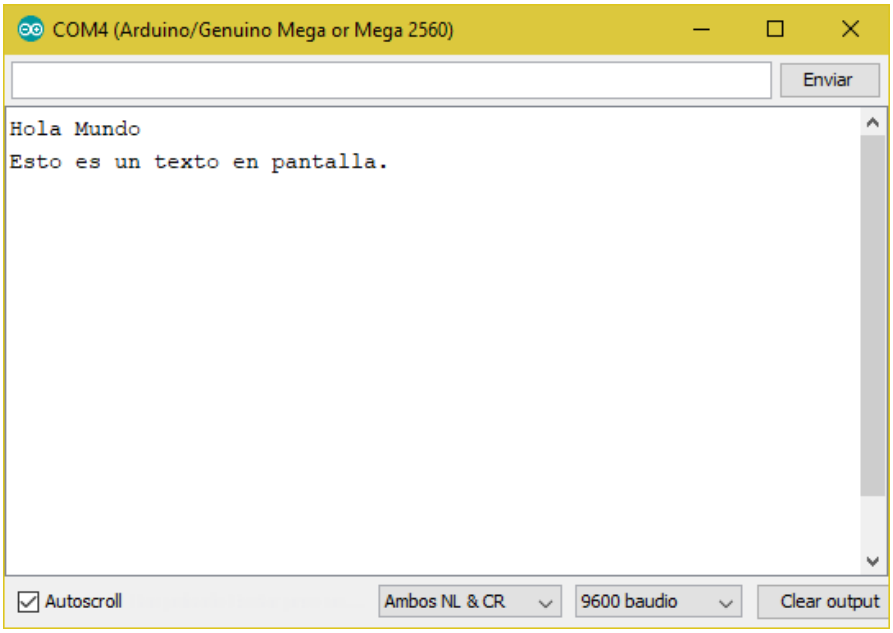

Figura 14: Ejemplo de mostrar cartel en monitor serie.

- **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 <LibEstacionamiento.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);
}

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

}
```

Figura 15: Ejemplo de mostrar número.

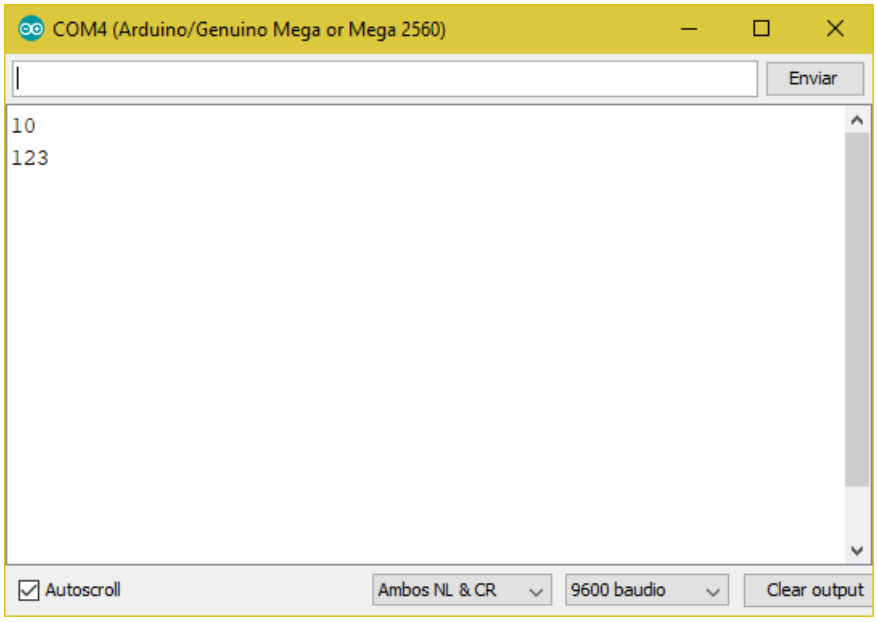

Figura 16: Ejemplo de mostrar número en monitor serie.

- **Ingreso de números**

Instrucción	Descripción
numero_ingresado = ingresar_numero();	Esta función permite leer un número que se ha ingresado por el monitor serie, y se guarda en “numero_ingresado”.

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

- **Ejemplo 1**

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 <LibEstacionamiento.h>

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

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

  //Ingresar un numero y almacenarlo como entero.
  numero_ingresado = ingresar_numero();

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

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

}
```

Figura 17: Ejemplo de librería, Ingreso de Números.

- **Ejemplo 2**

Ingreso de una clave numérica:

```
void loop() {
  // put your main code here, to run repeatedly:
  mostrar_cartel("Ingresar clave: \n");

  numero_ingresado = ingresar_numero();

  if(numero_ingresado == 123)
  {
    //Instrucciones si se ingresó 123.
  }
  else
  {
    //Instrucciones si no se ingresó 123.
  }
}
```

Figura 18: Ingreso de una clave numérica.

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