mOway Smart City

Manual de profesor

Índice

Introducción

El propósito de este manual es proporcionar una guía para enseñar y aprender a programar la ciudad inteligente **mOway Smart City**.

Está organizado de modo que se comienza con ejemplos sencillos para aprender los conceptos básicos de programación. Una vez aprendidos estos conceptos, se proporcionan programas más completos, para poder descubrir todas las posibilidades de aprendizaje que aporta la ciudad inteligente mOway Smart City.

En los diferentes ejercicios de este manual se plantea un objetivo a conseguir, se proporciona la información necesaria de los elementos a utilizar y se explica la estrategia para conseguir dicho objetivo. También se incluye el programa completo de cada ejercicio, junto con el diagrama de flujo para que sea más fácil su comprensión.

Ciudad inteligente

Una **ciudad inteligente** es aquella que, gracias a las tecnologías de la información y la comunicación, funciona de manera más eficiente, consume menos energía e incluso es capaz de gestionarse de forma autónoma. Ejemplos de funciones de una ciudad inteligente pueden ser los siguientes:

- Controlar los semáforos de una calle en función de la cantidad de vehículos, para que la circulación sea más eficiente.
- Aumentar la iluminación de una calle si hay tráfico y disminuirla si no circula ningún vehículo, para consumir menos energía.
- Informar del estado de la ciudad (temperatura, calidad del aire, etc.) por medio de un servidor web que puede ser consultado a través de cualquier dispositivo (ordenador, teléfono móvil, etc.).

La ciudad inteligente **mOway Smart City** es un recurso educativo que, junto con el **robot mOway**, favorece la comprensión del concepto de “ciudad inteligente”. Es una forma estimulante de aprender nociones de electrónica y programación, ya que los resultados pueden verse de inmediato. Además, trabajar con ejemplos de la vida real (activación automática de luces y de barreras, sistemas de seguridad en los vehículos, etc.) facilita la comprensión y retención de estos conceptos.

mOway Smart City cuenta con elementos que permiten el desarrollo de prácticas que representan aplicaciones en la vida real. Por ejemplo:

- Un circuito formado por una línea negra que el robot mOway sigue de forma autónoma.
- Farolas que se encienden cuando la luz ambiente disminuye.
- Una barrera que controla la circulación por un cruce.
- Dispositivos para monitorizar la información de los sensores.
- Etc.

El centro de control de mOway Smart City es una **tarjeta controladora** basada en la arquitectura **Arduino**®. Esta tarjeta puede ser programada por el usuario por medio del entorno de Arduino. Con el fin de facilitar la programación, el paquete de instalación de mOway Smart City incluye las librerías necesarias para manejar los diferentes elementos de la ciudad (barrera, farolas y sensores).

Además de todo esto, la tarjeta controladora de mOway Smart City es compatible con los módulos de mOway (**módulo RF**, **módulo WiFi** y **módulo Cámara**). Estos elementos amplían notablemente las posibilidades de la ciudad inteligente.

Para completar la información respecto al montaje y a la programación, se recomienda leer el manual de mOway Smart City, incluido en la carpeta de instalación. También se recomienda visitar la página oficial de Arduino: http://arduino.cc/es.

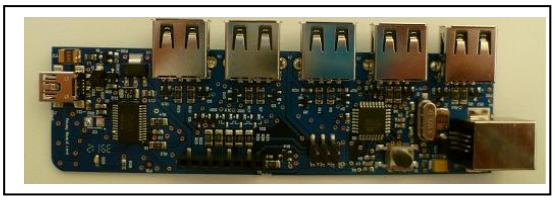
Ejercicio 1: Introducción a la programación

En este ejercicio de introducción se explica qué es un programa y cómo empezar a programar la ciudad inteligente mOway Smart City, por medio del entorno de programación de Arduino.

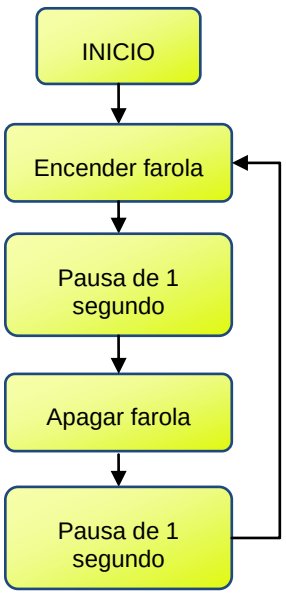
El funcionamiento de mOway Smart City está gestionado por una la tarjeta controladora, a la que se conectan los diferentes elementos de la ciudad: los sensores, las farolas y la barrera. La tarjeta controladora puede leer los sensores y en función de la información que reciba de ellos, encender las farolas, activar la barrera, etc. El funcionamiento se determina por medio de un programa definido por el usuario.

Estos son los elementos de mOway Smart City que se emplearán en esta práctica:

- Tarjeta controladora



Un **programa** consiste en una serie de instrucciones que se ejecutan de forma secuencial. Por ejemplo, un programa que haga parpadear una de las farolas podría tener las siguientes instrucciones:



Para desarrollar un programa se empieza definiendo cómo queremos que funcionen los diferentes elementos de la ciudad, por ejemplo, encender las farolas cuando oscurece, subir la barrera al detectar el paso de mOway, etc. Una vez definido el funcionamiento, hay que escribir el programa para realizar las instrucciones necesarias. Cuando se ha terminado el programa, se graba en la tarjeta controladora y se comprueba que su funcionamiento sea correcto. Si no es así, se corrige el código del programa y se vuelve a grabar la tarjeta controladora.

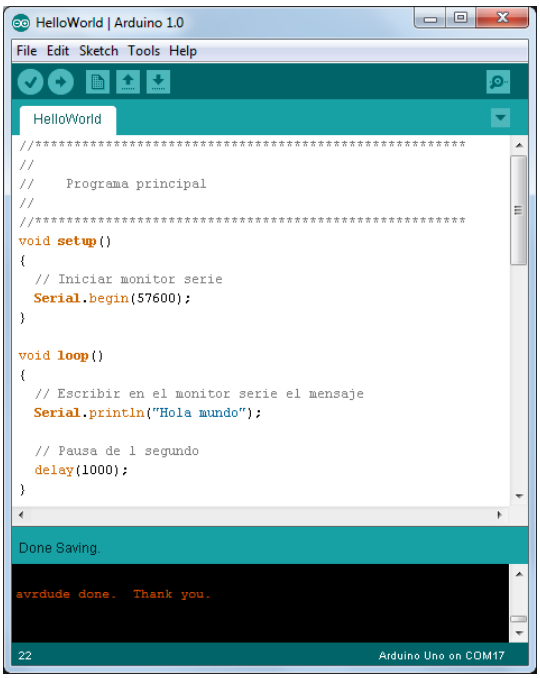
La tarjeta controladora de mOway Smart City se programa por medio del entorno Arduino. A continuación veremos las características de este entorno de programación.

Entorno Arduino

La tarjeta controladora de mOway Smart City está basada en la arquitectura de Arduino, por lo que es compatible con el entorno de programación proporcionado por Arduino. Este entorno cuenta con un editor de texto y una serie de recursos para compilar el programa y descargarlo en la tarjeta controladora. Se puede descargar desde la siguiente página: http://arduino.cc/es/Main/Software.

El lenguaje de programación empleado para mOway Smart City es similar a C/C++. A lo largo de este manual se mostrará código de ejemplo para aprender algunas de sus características básicas. Se puede encontrar una referencia a este lenguaje de programación en la página oficial de Arduino: http://arduino.cc/es/Reference/HomePage.

Para más información respecto al entorno Arduino, consultar la siguiente página: http://arduino.cc/es/Guide/Environment.



Programa 1.1

Para comprender cómo se realiza un programa en el entrono de Arduino, vamos a empezar por un ejemplo sencillo, que consistirá en mostrar en la pantalla del ordenador un mensaje de saludo. La tarjeta controladora envía este mensaje al ordenador a través de comunicación serie, por medio del cable USB.

En el lenguaje de Arduino, la estructura del programa principal se divide en dos funciones: “setup” y “loop”:

- En la función “setup” se realizan algunas de las configuraciones necesarias de la tarjeta controladora. En este caso, configuramos la comunicación serie para poder enviar el mensaje al ordenador, con una velocidad de 57600 baudios:

```
void setup()
{
  // Configura la comunicación serie con el PC
  Serial.begin(57600);
}
```

- En la función “loop” se ejecuta el programa principal, en forma de bucle infinito. En nuestro caso, consiste en mostrar el mensaje “Hola mundo” en la pantalla del ordenador cada segundo:

```
void loop()
{
  // Escribir en el monitor serie el mensaje
  Serial.println(“Hola mundo”);

  // Pausa de 1 segundo
  delay(1000);
}
```

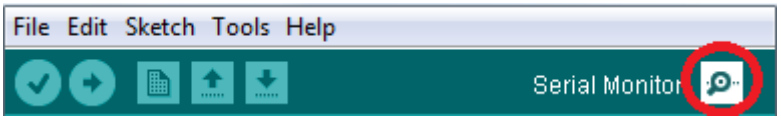
El código del programa completo es el siguiente:

```
void setup()
{
  // Configura la comunicación serie con el PC
  Serial.begin(57600);
}

void loop()
{
  // Escribir en el monitor serie el mensaje
  Serial.println(“Hola mundo”);

  // Pausa de 1 segundo
  delay(1000);
}
```

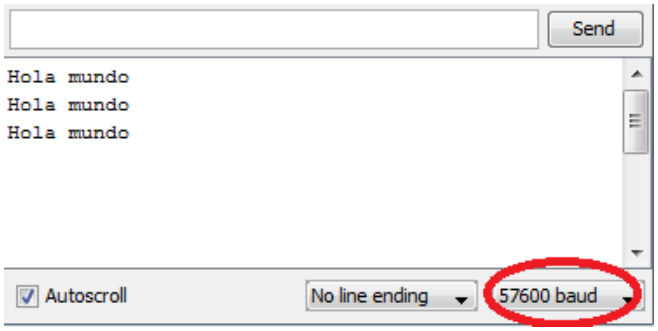
Este programa se puede probar copiándolo en el editor de texto del entorno Arduino. Después se programa la tarjeta controladora conectándola al ordenador y pulsando el icono "Upload".

IMPORTANTE: Es necesario seleccionar la tarjeta correspondiente ("Arduino Uno") en la pestaña "*Tools -> Board -> Arduino Uno*". Para más información, consultar la "Guía rápida".

Después de unos segundos, el programa quedará grabado en la tarjeta controladora. Para ver el resultado hay que abrir el monitor serie del entorno Arduino, pulsando el siguiente icono:



En la ventana del monitor serie aparecerá el mensaje de saludo. Es necesario configurar la velocidad de la comunicación serie de esta ventana con la misma que hemos indicado en el programa. En este caso es una velocidad de 57600 baudios (ver imagen):



Resumen

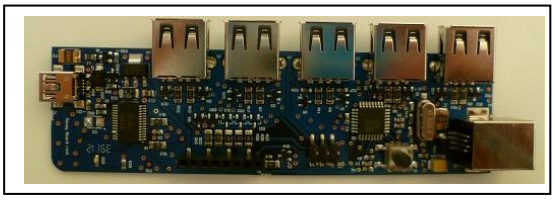
Con este primer ejercicio hemos aprendido a programar la tarjeta controladora de mOway Smart City. También hemos visto cómo podemos enviar mensajes desde la tarjeta controladora al ordenador. En ejercicios posteriores utilizaremos la comunicación serie con el ordenador para mostrar en la pantalla el valor de los sensores y para controlar los elementos de la ciudad desde el teclado.

Ejercicio 2: Variables y librerías

En este ejercicio vamos a aprender los conceptos de “variable” y “librería”. El uso de variables y librerías nos permitirá leer los sensores de la ciudad inteligente y controlar la barrera y las farolas. Por tanto, lo aprendido en este ejemplo nos va a facilitar el desarrollo de los próximos ejercicios.

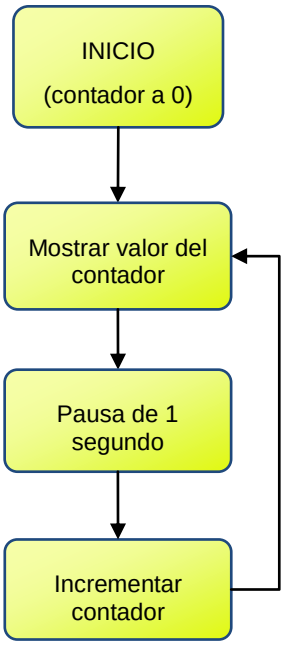
Estos son los elementos de mOway Smart City que se emplearán en esta práctica:

- Tarjeta controladora



Programa 2.1: variables

En programación, una **variable** almacena un valor que normalmente varía a lo largo del programa. Para entenderlo, vamos a ver un ejemplo en el que se utiliza una variable para mostrar en la pantalla los segundos transcurridos. A continuación se muestra el diagrama de flujo del programa:



El programa comienza definiendo la variable en la que iremos guardando los segundos transcurridos, llamada “segundos”. Es una variable de tipo entero (*int*) y comienza con un valor de 0. Para más información sobre los tipos de las variables, consultar la referencia en la página de Arduino, http://arduino.cc/es/Reference/HomePage.

```
//*****************************************************
//   Variables
//*****************************************************
int segundos = 0;                    // Variable para contar los segundos
```

En la función “setup” se configura la comunicación serie con el ordenador. Esto es necesario para que los datos se envíen correctamente al ordenador y se muestren en el monitor serie del entorno de Arduino. En este caso, se configura la velocidad serie a 57600 baudios.

```
void setup()
{
   // Iniciar monitor serie
  Serial.begin(57600);
}
```

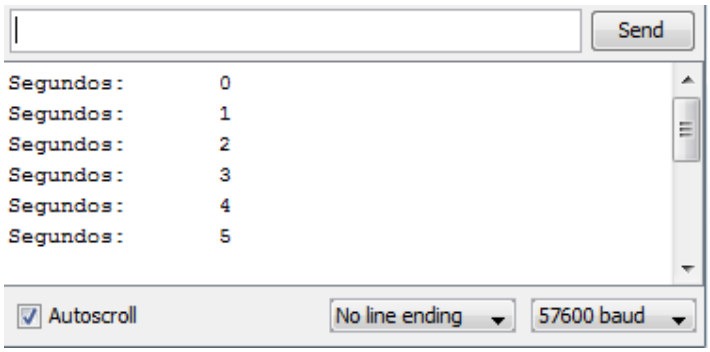
En la función “loop” se escribe en el monitor serie el valor de la variable “segundos”, se espera un segundo y se incrementa dicha variable. La función “loop” es un bucle infinito, por lo que este código se repite indefinidamente. Cada vez que se repite, la variable “segundos” se incrementa después de esperar 1 segundo.

```
void loop()
{
  // Escribir en el monitor serie el mensaje
  Serial.print(“Segundos: “);
  Serial.println(segundos);

  // Pausa de 1 segundo
  delay(1000);

  // Incrementar el contador de segundos
  segundos = segundos + 1;
}
```

El código del programa completo es el siguiente:

```
//*********************************************************
//   Variables
//*********************************************************
int segundos = 0;                    // Variable para contar los segundos

//*********************************************************
//
//   Programa principal
//
//*********************************************************
void setup()
{
  // Iniciar monitor serie
  Serial.begin(57600);
}

void loop()
{
  // Escribir en el monitor serie el mensaje
  Serial.print("Segundos: ");
  Serial.println(segundos);

  // Pausa de 1 segundo
  delay(1000);

  // Incrementar el contador de segundos
  segundos = segundos + 1;
}
```

El resultado que aparece en el monitor serie del entorno Arduino es el siguiente:



Más adelante veremos cómo las variables resultan útiles para trabajar con mOway Smart City. Por ejemplo, los valores leídos de los sensores se almacenan en variables, que son usadas para controlar los diferentes elementos de la ciudad en función de su valor.

Programa 2.2: librerías

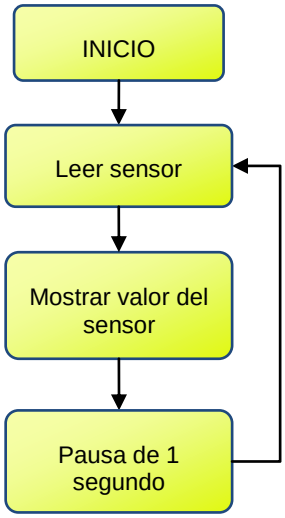
Una **librería** es un archivo que contiene código ya preparado para realizar ciertas funciones. La librería nos permite usar dicho código sin necesidad de escribirlo en nuestro programa. Por ejemplo, para leer el sensor de luz de la ciudad inteligente, habría que realizar las siguientes acciones:

1. Leer la señal analógica del sensor.
2. Transformar el valor de la señal a la escala correcta.
3. Asignar dicho valor a una variable.

Si no hubiese una librería, el usuario tendría que escribir el código de todas estas acciones en su programa, lo cual implicaría tener ciertas nociones de electrónica y además complicaría el programa.

En el caso de mOway Smart City, el uso de la librería del sensor de luz hace que no sea necesario conocer el funcionamiento del mismo, pues con sólo una línea de código la librería realiza todas las acciones necesarias (leer el puerto al que está conectado el sensor, escalar el valor recibido, etc.).

A continuación vemos un ejemplo del uso de las librerías para mostrar el valor del sensor de luz en la pantalla del ordenador. Este ejemplo es similar al anterior, pero en vez de incrementar el valor una variable, le asignamos a esta variable el valor de un sensor.



Se incluye tanto la librería del sensor de luz (“light_sensor.h”) como la librería encargada de acceder a los puertos de entrada y salida de la tarjeta controladora (“lib_io.h”). También se

incluye la librería *Wire*, necesaria para las otras librerías de mOway Smart City. Las librerías se añaden con la directiva "#include":

```
//*******************************************************
//   Librerías
//*******************************************************
#include <Wire.h>                 // Librería de comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <light_sensor.h>         // Librería del sensor de luz
```

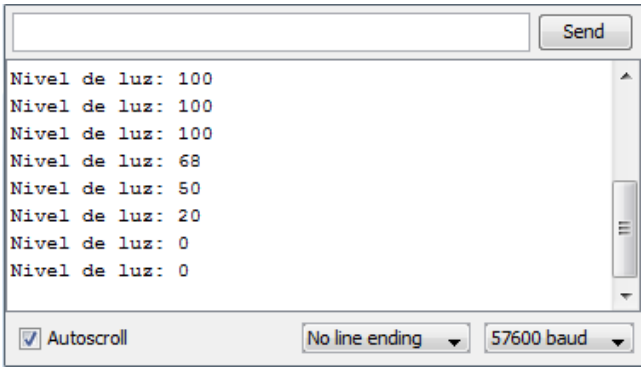
Al definir el sensor de luz, es necesario indicar el puerto de la placa en el que se conecta dicho sensor, en este caso es el conector 4 (**CON4**):

```
//*******************************************************
//   Dispositivos
//*******************************************************
LightSensor sensorLuz(CON4);  // El sensor se conecta al conector 4
```

El programa completo queda de la siguiente forma:

```
//*******************************************************
//   Librerías
//*******************************************************
#include <Wire.h>                 // Librería de comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <light_sensor.h>         // Librería del sensor de luz

//*******************************************************
//   Dispositivos
//*******************************************************
LightSensor sensorLuz(CON4);  // El sensor se conecta al conector 4

//*******************************************************
//   Variables
//*******************************************************
int nivelLuz;     // Variable para guardar el valor del sensor

//*******************************************************
//
//   Programa principal
//
//*******************************************************
void setup()
{
  // Iniciar monitor serie
  Serial.begin(57600);
}

void loop()
{
  Serial.print("Nivel de luz: ");

  nivelLuz = sensorLuz.Luz();       // Leer el sensor de luz
  Serial.println(nivelLuz);         // Mostrar valor del sensor

  delay(1000);                      // Pausa de 1 segundo
}
```

El resultado al ir tapando el sensor de luz es que el valor del sensor disminuye (en este caso, desde el 100% hasta el 0% de luz):



Resumen

Tanto las librerías como las variables nos van a ser muy útiles en el desarrollo de los programas para controlar la ciudad inteligente.

Las variables las usaremos sobre todo para almacenar los valores de los sensores, de modo que podamos controlar los elementos en función del estado de la ciudad (luz ambiental, temperatura, circulación, etc.).

Las librerías nos facilitarán mucho la programación, porque evitan añadir código en el programa principal y además hacen que no sea necesario conocer la electrónica de los componentes de la ciudad.

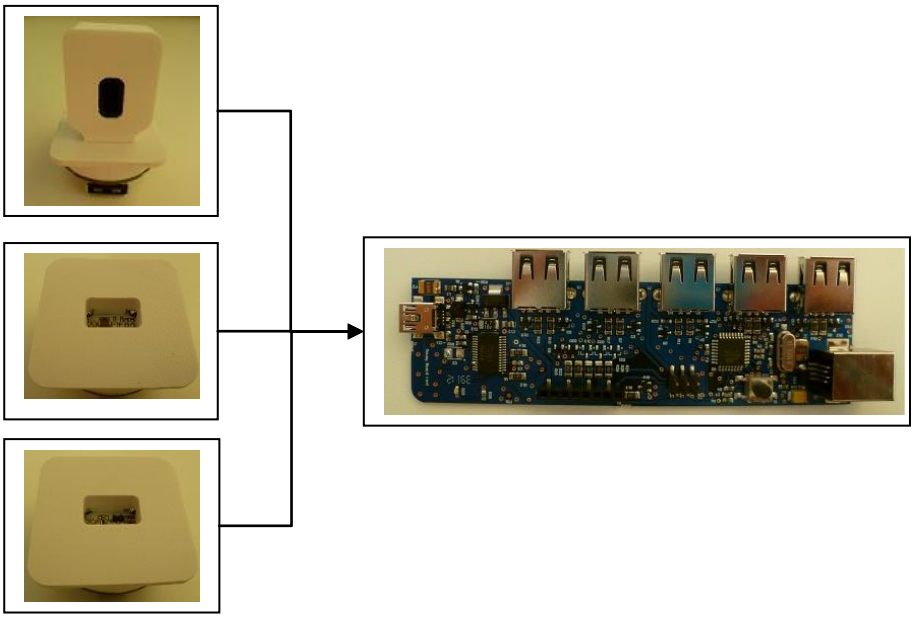
Ejercicio 3: Lectura de sensores

En este ejercicio vamos a aprender a utilizar los **sensores** de mOway Smart City. La tarjeta controladora leerá los valores de los diferentes sensores y los mostrará en la pantalla del ordenador. En ejercicios posteriores usaremos esta información proporcionada por los sensores para controlar los elementos de la ciudad inteligente de forma autónoma.

Elementos necesarios

Estos son los elementos de mOway Smart City que se emplearán en esta práctica:

- Tarjeta controladora
- Sensor de proximidad
- Sensor de luz
- Sensor de temperatura



Características de los sensores

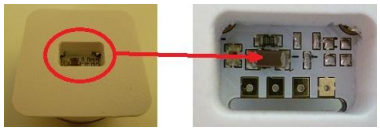
Un sensor es un dispositivo electrónico que nos permite conocer el entorno en el que se encuentra dicho sensor. La ciudad inteligente cuenta con sensores de presencia, de luz y de temperatura. A continuación se explica cada uno de ellos.

El sensor de presencia nos indica si hay un objeto cerca del propio sensor y a qué distancia se encuentra dicho objeto. Funciona por medio de un emisor de luz infrarroja y un receptor. Si hay un objeto cerca del sensor, la luz infrarroja emitida por el emisor rebota contra el objeto y es recibida por el receptor. El valor proporcionado por el sensor varía entre 0% (no se detecta objeto) y 100% (el objeto está muy cerca).

El sensor de presencia se puede utilizar para encender las farolas cuando el robot mOway pasa debajo de ellas, o para detectar la llegada de un mOway al cruce y activar la barrera.

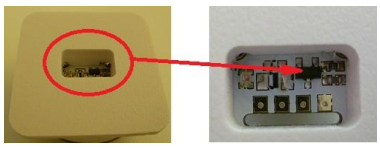
El sensor de luz nos indica la cantidad de luz ambiente que recibe la ciudad inteligente. Funciona a través de un componente electrónico llamado fotodiodo, el cual genera una corriente eléctrica en función de la cantidad de luz que incide sobre él. Proporciona un valor entre el 0% (oscuridad total) y el 100% (luz intensa).

Este sensor se puede emplear para encender las farolas cuando oscurece. El sensor de luz se distingue por el componente de color marrón indicado en la imagen:



El sensor de temperatura funciona gracias a un componente electrónico que genera un voltaje variable con la temperatura. Se puede elegir que el valor proporcionado por el sensor sea en grados Centígrados o grados Fahrenheit.

Este sensor puede utilizarse para monitorizar el estado ambiental de la ciudad inteligente. El sensor de temperatura se distingue por el componente de color negro indicado en la imagen:



Programa 3.1

Comenzamos incluyendo las librerías que contienen las funciones de lectura de los sensores: “proximity_sensor.h” para el sensor de proximidad, “light_sensor.h” para el sensor de luz y “temperature_sensor.h” para el sensor de temperatura. Junto con estas tres librerías es necesario incluir “lib_io.h” para los puertos de entrada/salida de la tarjeta controladora y “Wire.h”.

```
//*******************************************************
//  Librerías
//*******************************************************
#include <Wire.h>                       // Librería de comunicación I2C
#include <lib_io.h>                     // Librería de puertos E/S
#include <proximity_sensor.h>           // Librería de sensor de proximidad
#include <light_sensor.h>               // Librería de sensor de luz
#include <temperature_sensor.h>         // Librería de sensor de temperatura
```

A continuación definimos las variables donde vamos a guardar el valor que leamos de los sensores. Estas variables son de tipo entero (*int*).

```
//*******************************************************
//  Variables
//*******************************************************
int valorDistancia;      // Almacena el valor del sensor de proximidad
int valorLuz;            // Almacena el valor del sensor de luz
int valorTemperatura; // Almacena el valor del sensor de temperatura
```

Después definimos los dispositivos que utilizaremos, en este caso, los sensores. En esta parte del programa se define tanto el tipo de sensor como el nombre con el que lo vamos a utilizar a lo largo del programa, así como el puerto de la tarjeta controladora donde está conectado.

NOTA: El nombre de los sensores puede ser cualquiera que el usuario elija, pero conviene asignar un nombre que haga fácil identificarlo. El tipo de sensor se define en las librerías y siempre debe ser el que aparece en este código, es decir:

Sensor	Tipo
Sensor de proximidad	ProximitySensor
Sensor de luz	LightSensor
Sensor de temperatura	TemperatureSensor

En este ejercicio se ha conectado el sensor de proximidad en el conector 2 (**CON2**), el sensor de luz en el conector 4 (**CON4**) y el sensor de temperatura en el conector 5 (**CON5**).

```
//*****************************************************
//  Dispositivos
//*****************************************************
ProximitySensor sensorProximidad(CON2);
LightSensor sensorLuz(CON4);
TemperatureSensor sensorTemperatura(CON5);
```

En el programa principal se realiza la lectura del sensor de proximidad, guardando el valor del sensor en la variable “valorDistancia”. Después se muestra en el monitor serie del entorno Arduino su valor. Se realiza lo mismo para los sensores de luz y temperatura. El bucle finaliza con una pausa de 1 segundo, para dar tiempo a leer los valores en la pantalla.

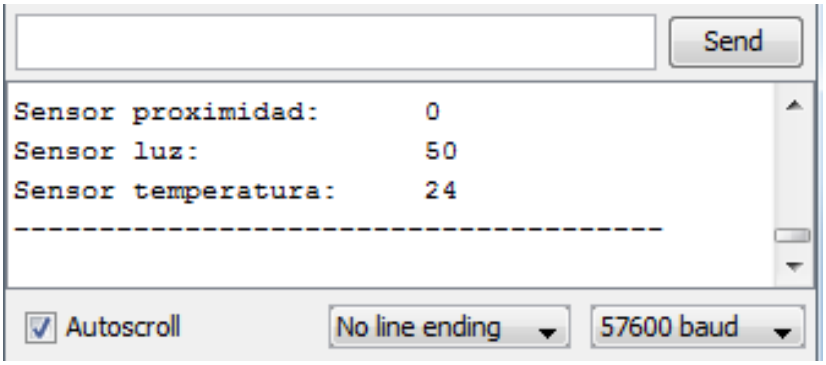
El programa completo queda de la siguiente forma:

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                       // Librería de comunicación I2C
#include <lib_io.h>                     // Librería de puertos E/S
#include <proximity_sensor.h>           // Librería de sensor de proximidad
#include <light_sensor.h>               // Librería de sensor de luz
#include <temperature_sensor.h>         // Librería de sensor de temperatura



//*****************************************************
//  Variables
//*****************************************************
int valorDistancia;      // Almacena el valor del sensor de proximidad
int valorLuz;            // Almacena el valor del sensor de luz
int valorTemperatura; // Almacena el valor del sensor de temperatura



//*****************************************************
//  Dispositivos
//*****************************************************
ProximitySensor sensorProximidad(CON2);
LightSensor sensorLuz(CON4);
TemperatureSensor sensorTemperatura(CON5);
```

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
  // Iniciar la comunicación serie
  Serial.begin(57600);
}

void loop()
{
  // Leer sensor de proximidad y mostrar valor
  valorDistancia = sensorProximidad.Distancia();
  Serial.print("Sensor proximidad:\t");
  Serial.println(valorDistancia);

  // Leer sensor de luz y mostrar valor
  valorLuz = sensorLuz.Luz();
  Serial.print("Sensor luz:\t\t");
  Serial.println(valorLuz);

  // Leer sensor de temperatura y mostrar valor
  valorTemperatura = sensorTemperatura.Temperatura(C);
  Serial.print("Sensor temperatura:\t");
  Serial.println(valorTemperatura);
  Serial.println("---------------------------------------");

  delay(1000);   // Pausa de 1 segundo
}
```

NOTA: Los caracteres "\t" sirven para tabular el texto. Se han añadido para que el resultado se muestre de forma ordenada, aunque no son imprescindibles.

El resultado se muestra a continuación. El sensor de proximidad da un valor del 0%, por lo que no está detectando ningún objeto. El sensor de luz da un valor del 50%, por lo que está detectando un nivel de luz medio. Por último, el valor del sensor de temperatura es de 24ºC.

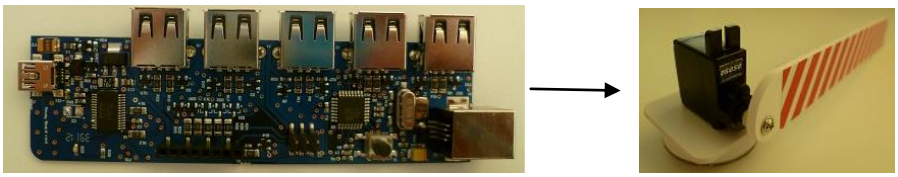


Ejercicio 4: Control de barrera

En este ejercicio vamos a aprender cómo manejar la barrera de mOway Smart City. Este elemento se puede emplear para controlar la circulación por el cruce del circuito y practicar con los sensores de obstáculos del robot mOway.

Para realizar este ejercicio se van a emplear los siguientes elementos:

- Tarjeta controladora
- Barrera



Características de la barrera

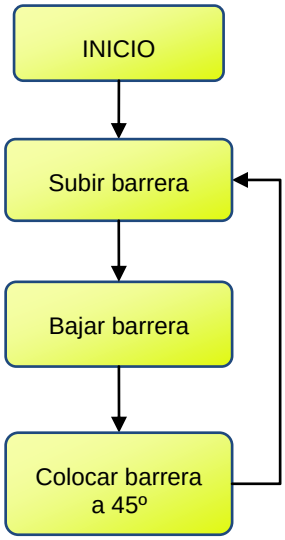
La barrera se mueve por medio de un servomotor, el cual puede subir la barrera, bajarla o colocarla en un ángulo determinado. El servomotor de la barrera se controla a través de una señal modulada por ancho de pulso (Pulse Width Modulation, PWM). La librería para la barrera se encarga de generar esta señal.

La barrera puede colocarse en cualquiera de los dos orificios del cruce del circuito. Además, puede orientarse en función del sentido en el que mOway recorra el circuito.

IMPORTANTE: La barrera sólo puede ir conectada al puerto CON1 de la tarjeta controladora, debido a que es el encargado de generar la señal modulada mencionada anteriormente.

Programa 4.1

En este programa se muestra todas las acciones que pueden realizarse con la barrera. Consiste en subir la barrera, bajarla y colocarla en un ángulo de 45º, repitiendo estos movimientos de forma indefinida.



El programa comienza definiendo las librerías. Se incluyen las librerías habituales “Wire.h” y “lib_io.h”, como hemos visto en ejercicios anteriores. También se incluye la librería de la barrera, “barrier.h”. Esta librería de la barrera utiliza a su vez la librería “Servo.h”, que es una librería propia de Arduino encargada de controlar servomotores.

```
//*************************************************
//  Librerías
//*************************************************
#include <Wire.h>                 // Librería de comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <Servo.h>                // Librería de servomotores
#include <barrier.h>              // Librería de barrera
```

A continuación se define el nombre de la barrera. A diferencia de otros elementos de mOway Smart City, en el caso de la barrera no se indica el puerto donde va conectada. Esto es debido a que siempre se conecta en el puerto **CON1** de la tarjeta controladora.

```
//*************************************************
//  Dispositivos
//*************************************************
Barrier barrera;
```

El programa principal consiste en subir la barrera, bajar la barrera y colocar la barrera en un ángulo de 45º. Entre cada movimiento se espera un segundo. Para mover la barrera hay que seguir 4 pasos:

1. Primero se inicia el control del servomotor con la función “Iniciar”. Es necesario especificar el puerto de la barrera, que siempre es “CON1_DIG”.
2. Luego se indica la acción (subir, bajar o colocar en un ángulo)
3. Se espera durante unos 300 milisegundos para que la barrera llegue a la posición indicada.
4. Por último, se detiene el control del servomotor con la función “Parar”.

```
//******************************************************
//
// Programa principal
//
//******************************************************
void setup()
{
}

void loop()
{
  // Subir barrera
  barrera.Iniciar(CON1_DIG);
  barrera.Subir();
  delay(300);
  barrera.Parar();

  delay(1000);   // Pausa de 1 segundo

  // Bajar barrera
  barrera.Iniciar(CON1_DIG);
  barrera.Bajar();
  delay(300);
  barrera.Parar();

  delay(1000);   // Pausa de 1 segundo

  // Colocar la barrera a 45º
  barrera.Iniciar(CON1_DIG);
  barrera.Angulo(45);
  delay(300);
  barrera.Parar();

  delay(1000);   // Pausa de 1 segundo
}
```

NOTA: La pausa de 300 milisegundos para que la barrera llegue a la posición deseada podría reducirse o incluso eliminarse, siempre y cuando el programa tarde unos 300 milisegundos en llegar a la siguiente función de movimiento de la barrera.

El código del programa completo queda de la siguiente forma:

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                 // Librería de comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <Servo.h>                // Librería de servomotores
#include <barrier.h>              // Librería de barrera

//*****************************************************
//  Dispositivos
//*****************************************************
Barrier barrera;

//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
}

void loop()
{
  // Subir barrera
  barrera.Iniciar(CON1_DIG);
  barrera.Subir();
  delay(300);
  barrera.Parar();

  delay(1000);   // Pausa de 1 segundo

  // Bajar barrera
  barrera.Iniciar(CON1_DIG);
  barrera.Bajar();
  delay(300);
  barrera.Parar();

  delay(1000);   // Pausa de 1 segundo

  // Colocar la barrera a 45º
  barrera.Iniciar(CON1_DIG);
  barrera.Angulo(45);
  delay(300);
  barrera.Parar();

  delay(1000);   // Pausa de 1 segundo
}
```

Aplicaciones

Con este ejercicio hemos aprendido cuáles son los pasos que hay que seguir para colocar la barrera en una posición determinada. Este elemento lo usaremos para controlar la circulación en el cruce del circuito, junto con el sensor de proximidad.

Ejercicio 5: Control de farolas

Introducción

En esta práctica vamos a aprender cómo manejar las farolas de mOway Smart City. Se muestran diferentes ejemplos de código para activar la iluminación de las farolas. Esto nos servirá para facilitar el desarrollo de prácticas posteriores, tales como adaptar la iluminación de la ciudad a la luz ambiente, regular la intensidad de iluminación en función de la circulación, etc.

Elementos necesarios

Estos son los elementos de mOway Smart City que se emplearán en esta práctica:

- Tarjeta controladora
- Farolas

Características de las farolas

Las farolas de mOway Smart City son elementos que permiten iluminar la ciudad inteligente. Cuentan con cuatro diodos LED distribuidos a lo largo del poste de cada farola. La tarjeta controladora accede a estos elementos por medio de comunicación I^2C. Esto permite conectar las farolas en serie, por lo que sólo es necesario uno de los puertos de entrada/salida de la tarjeta controladora para conectar todas las farolas.

IMPORTANTE: Las farolas sólo pueden conectarse al conector 3 de la tarjeta controladora, ya que es el único puerto preparado para la comunicación I^2C (puerto CON3 / I^2C).

Cada farola está identificada con un número del 1 al 4 en la parte posterior de la base de cada una de ellas. Como veremos más adelante, es necesario indicar este identificador en el programa a la hora de activar la farola que queremos. En la siguiente tabla se muestra el identificador correspondiente a cada farola:

Identificador de farola (en la base)	Identificador en programa
1	FAROLA_1
2	FAROLA_2
3	FAROLA_3
4	FAROLA_4

En la parte superior del poste de la farola se encuentra el LED principal. Este LED es de color blanco y sirve para iluminar el circuito de la ciudad inteligente. Se puede regular la intensidad de encendido de este LED con 7 niveles diferentes.

Nivel de luz	Identificador en programa
Nivel 1 (intensidad mínima)	NIVEL_1
Nivel 2	NIVEL_2
Nivel 3	NIVEL_3
Nivel 4	NIVEL_4
Nivel 5	NIVEL_5
Nivel 6	NIVEL_6
Nivel 7 (intensidad mínima)	NIVEL_7

A lo largo del poste de la farola hay tres LEDs azules. Pueden encenderse y apagarse de forma individual, de modo que pueden emplearse como indicadores, o bien conseguir diferentes apariencias para la farola.

LED del poste	Identificador en programa
LED superior	SUPERIOR
LED medio	MEDIO
LED inferior	INFERIOR

A continuación veremos diferentes ejemplos para aprender a usar las funciones que controlan estos dispositivos.

Programa 5.1

Vamos a comenzar por el ejemplo más sencillo, que consiste en encender el LED principal de una farola.

Primero se incluyen las librerías necesarias para las farolas (“streetlight.h”), para los puertos de la tarjeta controladora (“lib_io.h”) y la librería *Wire*, que es usada por las anteriores librerías (“Wire.h”).

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                 // Librería para comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <streetlight.h>          // Librería de farolas
```

Luego se define la farola que vamos a utilizar. En este ejemplo, se ha usado una farola con identificador “1” (ver parte posterior de la base de la farola). Por tanto, la farola se define como “FAROLA_1” (deben coincidir ambos identificadores, el de la base de la farola y el del programa).

```
//*****************************************************
//  Dispositivos
//*****************************************************
// Se define una farola de tipo “1”
Streetlight farola(FAROLA_1);
```

NOTA: En este ejemplo no es necesario realizar ninguna acción de configuración, por tanto, la función “setup” está vacía.

El programa principal consiste tan sólo en encender la farola, activando el LED principal con intensidad máxima. A continuación podemos ver el código del programa completo.

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                 // Librería para comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <streetlight.h>          // Librería de farolas

//*****************************************************
//  Dispositivos
//*****************************************************
// Se define una farola con identificador "1"
Streetlight farola(FAROLA_1);

//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{

}

void loop()
{
  // Encender farola al nivel máximo
  farola.Encender(NIVEL_7);
}
```

Ejemplo 5.2

Partiendo del código anterior, vamos a completarlo haciendo que el LED principal parpadee cada segundo. En el programa principal hay que añadir una pausa de 500 milisegundos, apagar la farola y añadir otra pausa de 500 milisegundos.

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                 // Librería para comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <streetlight.h>          // Librería de farolas


//*****************************************************
//  Dispositivos
//*****************************************************
// Se define una farola con identificador "1"
Streetlight farola(FAROLA_1);
```

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
}

void loop()
{
  // Encender farola al nivel máximo
  farola.Encender(NIVEL_7);
  delay(500);             // Pausa de 500ms

  // Apagar farola
  farola.Apagar();
  delay(500);             // Pausa de 500ms
}
```

Ejemplo 5.3

En este ejemplo vamos a variar la intensidad de luz del LED principal. Para ello indicaremos diferentes niveles de intensidad a la hora de encender la farola. Se ha añadido una pausa entre los cambios de intensidad de iluminación.

El tiempo de pausa se indica con la variable “tiempo”, para poder ajustar de forma sencilla el tiempo entre los cambios de intensidad. Con un valor de 300 (milisegundos) se aprecia los diferentes niveles de luz. En cambio, con un valor menor (de 100 milisegundos, por ejemplo), el encendido se apreciará menos “escalonado” a simple vista.

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                 // Librería para comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <streetlight.h>          // Librería de farolas


//*****************************************************
//  Variables
//*****************************************************
int tiempo = 300;       // Tiempo de espera entre cambios de intensidad

//*****************************************************
//  Dispositivos
//*****************************************************
// Se define una farola con identificador “1”
Streetlight farola(FAROLA_1);
```

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
}

void loop()
{
 farola.Encender(NIVEL_1);      // Nivel mínimo
 delay(tiempo);
 farola.Encender(NIVEL_2);
 delay(tiempo);
 farola.Encender(NIVEL_3);
 delay(tiempo);
 farola.Encender(NIVEL_4);
 delay(tiempo);
 farola.Encender(NIVEL_5);
 delay(tiempo);
 farola.Encender(NIVEL_6);
 delay(tiempo);
 farola.Encender(LEVEL_7);      // Nivel máximo
 delay(tiempo);
 farola.Apagar();                // Apagar farola
 delay(tiempo);
}
```

Ejemplo 5.4

En este ejemplo vamos a activar los LEDs del poste de forma secuencial. Estos LEDs se irán encendiendo de uno en uno, desde el más bajo hasta el más alto, para luego apagarse en el orden inverso. El LED principal permanecerá apagado.

Se parte del código anterior, de modo que, como antes, se puede variar el valor de la variable “tiempo” para que la secuencia sea más lenta o más rápida.

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                 // Librería para comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <streetlight.h>          // Librería de farolas

//*****************************************************
//  Variables
//*****************************************************
int tiempo = 300;        // Tiempo de espera entre cambios de intensidad

//*****************************************************
//  Dispositivos
//*****************************************************
// Se define una farola con identificador “1”
Streetlight farola(FAROLA_1);
```

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
}

void loop()
{
  farola.Poste(INFERIOR, ON);          // Encender LED inferior
  delay(tiempo);
  farola.Poste(MEDIO, ON);             // Encender LED del medio
  delay(tiempo);
  farola.Poste(SUPERIOR, ON);          // Encender LED superior
  delay(tiempo);
  farola.Poste(SUPERIOR, OFF);         // Apagar LED superior
  delay(tiempo);
  farola.Poste(MEDIO, OFF);            // Apagar LED del medio
  delay(tiempo);
  farola.Poste(INFERIOR, OFF);         // Encender LED inferior
  delay(tiempo);
}
```

Ejemplo 5.5

Por último, veremos un ejemplo en el que se usan tres farolas, las identificadas con “1”, “2” y “3”. Para poder usar las tres farolas en el programa, es necesario definir una instancia para cada una de ellas, indicando además el identificador de cada una. El nombre puede ser cualquiera, pero se ha elegido llamar a cada farola del mismo modo que su identificador en letras minúsculas. En este caso:

Identificador de farola	Nombre	Identificador en programa
1	farola_1	FAROLA_1
2	farola_2	FAROLA_2
3	farola_3	FAROLA_3

Cuando se tienen varias farolas hay que empezar conectando la primera de ellas al conector **CON3 / I^{2}C** de la tarjeta controladora. Después se van conectando el resto de las farolas en serie, conectando el conector USB de la primera farola con el conector MiniUSB de la segunda farola y así sucesivamente. Para más información sobre el montaje, consultar la “Guía rápida”.

A continuación se muestra el código de este ejemplo. Cada farola se enciende con una intensidad diferente y una iluminación del poste distinta.

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                 // Librería para comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <streetlight.h>          // Librería de farolas

//*****************************************************
//  Dispositivos
//*****************************************************
Streetlight farola_1(FAROLA_1);   // Farola con identificador "1"
Streetlight farola_2(FAROLA_2);   // Farola con identificador "2"
Streetlight farola_3(FAROLA_3);   // Farola con identificador "3"

//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
}

void loop()
{
  // La farola 1 se enciende al nivel mínimo
  // Se enciende un LED del poste
  farola_1.Encender(NIVEL_1);
  farola_1.Poste(INFERIOR, ON);

  // La farola 2 se enciende un nivel medio
  // Se encienden dos LEDs del poste
  farola_2.Encender(NIVEL_4);
  farola_2.Poste(INFERIOR, ON);
  farola_2.Poste(MEDIO, ON);

  // La farola 3 se enciende al nivel máximo
  // Se encienden tres LEDs del poste
  farola_3.Encender(NIVEL_7);
  farola_3.Poste(INFERIOR, ON);
  farola_3.Poste(MEDIO, ON);
  farola_3.Poste(SUPERIOR, ON);
}
```

Aplicaciones

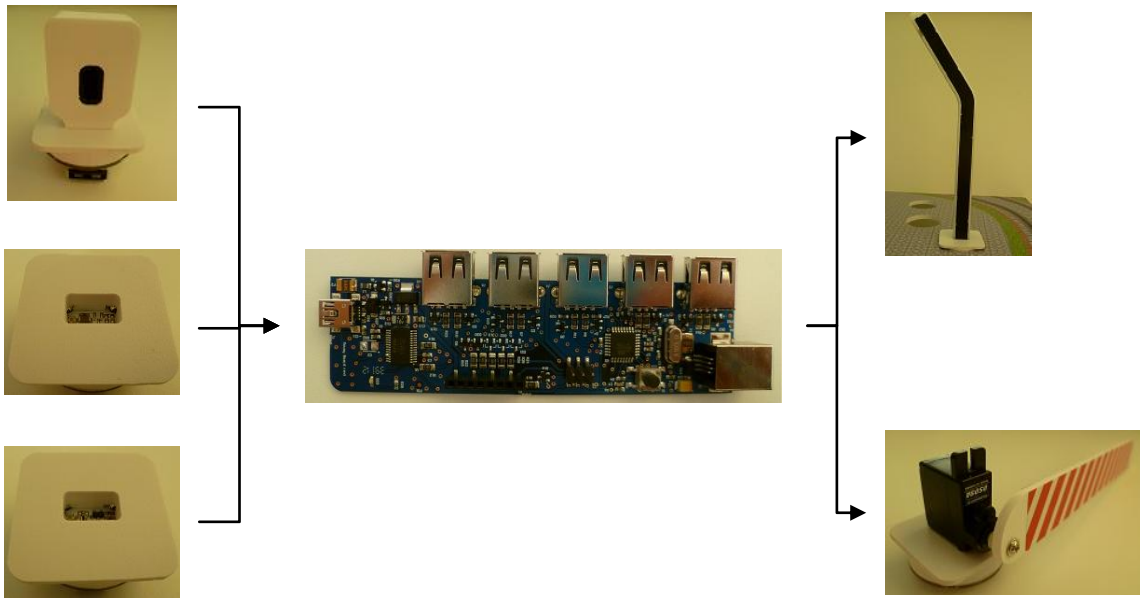
Con estos ejemplos hemos visto cómo podemos controlar los diferentes LEDs de la farola. Esto nos va a ser útil para desarrollar prácticas posteriores, en las que activaremos las farolas en función de la luz, la circulación, etc. Esto se consigue usando los sensores para conocer el entorno y, dependiendo de su estado, controlar las farolas.

Ejercicio 6: Control por teclado

Una vez que hemos aprendido cuál es el código necesario para activar las farolas, mover la barrera y leer los sensores, vamos a controlar estos elementos a través del teclado del ordenador. En este ejercicio aprenderemos a detectar una tecla pulsada en el ordenador y, en función de dicha tecla, activar el elemento correspondiente.

Para realizar este ejercicio se van a emplear los siguientes elementos:

- Tarjeta controladora
- Farola tipo “1”
- Barrera
- Sensor de presencia
- Sensor de luz
- Sensor de temperatura



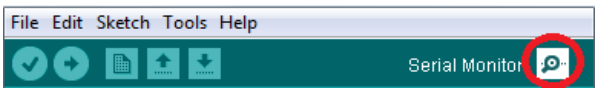
Los elementos de la ciudad se conectan a los siguientes puertos de la tarjeta controladora:

Dispositivo	Puerto de la tarjeta controladora
Barrera	CON1
Sensor de proximidad	CON2
Farola	CON3 / I^2C
Sensor de luz	CON4
Sensor de temperatura	CON5

Comunicación serie

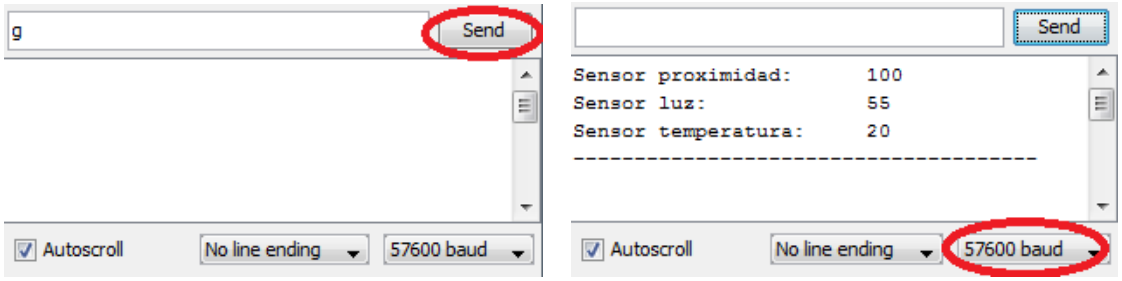
La transmisión de datos entre el ordenador y la tarjeta controladora se realiza por medio de comunicación serie. Estos datos se transmiten a través del cable USB – MiniUSB, el mismo que el empleado para programar la tarjeta. En este caso, los datos transmitidos son las teclas pulsadas.

Los datos desde el ordenador se envían a través del monitor serie del entorno Arduino (*Serial Monitor*).



Para enviar una tecla a la tarjeta controladora, se introduce esta tecla (en este ejemplo, la tecla “g”) y se pulsa el botón de enviar (“Send”). La tarjeta controladora detectará la tecla “g” y enviará al monitor serie el valor de los sensores conectados.

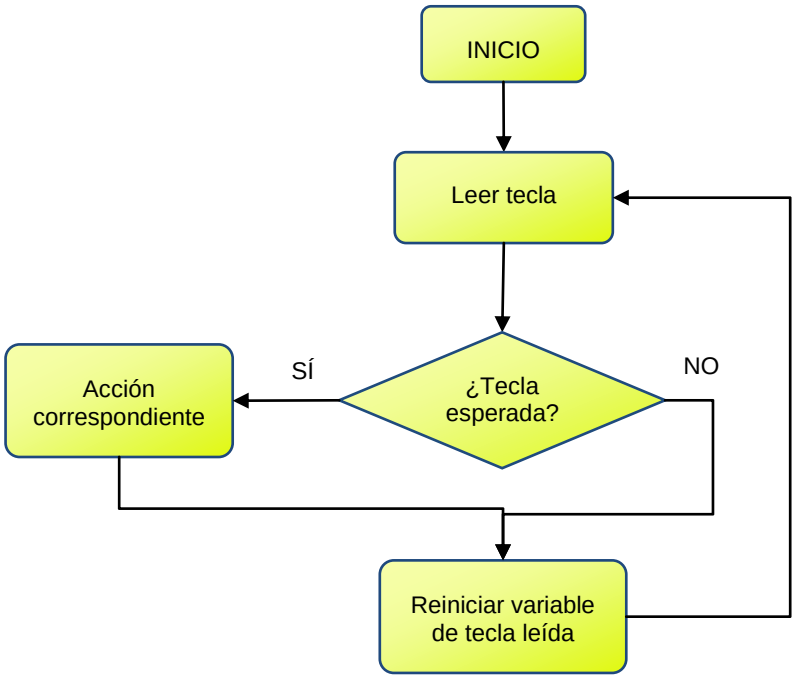
NOTA: Es necesario configurar la velocidad de comunicación serie del monitor, para que coincida con al seleccionada en el programa de la tarjeta controladora, que veremos más adelante (en este caso, 57600 baudios).



Programa 6.1

El programa de la tarjeta controladora está comprobando continuamente si ha llegado algún dato por el puerto serie, es decir, si se ha enviado alguna tecla a través del monitor serie de Arduino.

Cada vez que se recibe un dato por el puerto serie, se guarda dicho dato en una variable. Una vez que la variable contiene el dato, se comprueba si este dato corresponde con una tecla asociada con alguna acción. En ese caso, se ejecuta la acción correspondiente. Finalmente, es necesario reiniciar el valor de la variable para que la última acción no se repita de nuevo.



Las teclas empleadas para controlar los dispositivos son las siguientes:

Dispositivo	Tecla	Acción
Barrera	B	Subir/Bajar
Sensores	G	Muestra al valor de los sensores
Farola	Q / A	Encender / apagar farola
	W / S	Encender / apagar LED superior del poste
	E / D	Encender / apagar LED medio del poste
	R / F	Encender / apagar LED inferior del poste

Debido a que se usan todos los elementos de la ciudad inteligente, es necesario incluir todas las librerías usadas en los ejercicios anteriores:

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                       // Librería de comunicación I2C
#include <lib_io.h>                     // Librería de puertos E/S
#include <Servo.h>                      // Librería de servomotores
#include <barrier.h>                    // Librería de barrera
#include <proximity_sensor.h>           // Librería de sensor de proximidad
#include <light_sensor.h>               // Librería de sensor de luz
#include <temperature_sensor.h>         // Librería de sensor de temperatura
#include <streetlight.h>                // Librería de farolas
```

Necesitamos una variable para almacenar la tecla recibida desde el ordenador, a la que hemos llamado “tecla”. También usaremos otra variable para guardar el valor leído de los sensores. En este caso sólo usamos una variable para todos los sensores, ya que siempre actualizaremos su valor con el del sensor a mostrar en pantalla. Por otra parte, la variable “barreraArriba” la usaremos para saber en qué posición se encuentra la barrera.

```
//*****************************************************
//  Variables
//*****************************************************
char tecla;                             // Almacena la tecla recibida
int sensorVal;
boolean barreraArriba = false;
```

La definición de los dispositivos se realiza de la siguiente forma:

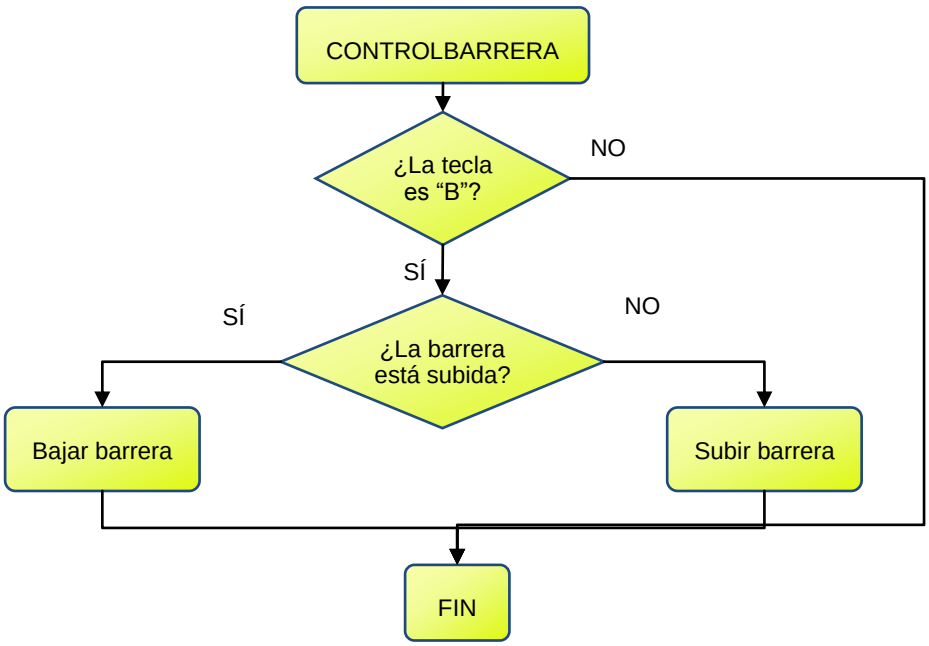
```
//*****************************************************
//  Dispositivos
//*****************************************************
Barrier barrera;
ProximitySensor sensorProximidad(CON2);
LightSensor sensorLuz(CON4);
TemperatureSensor sensorTemperatura(CON5);
Streetlight farola_1(FAROLA_1);
Streetlight farola_2(FAROLA_2);
Streetlight farola_3(FAROLA_3);
```

El programa principal configura la comunicación serie (en la función “setup”) y comprueba continuamente si hay algún dato disponible en el puerto serie (“Serial.available”). Si es así, almacena el dato en la variable “tecla” y ejecuta funciones definidas para realizar la acción correspondiente a la tecla recibida (“ControlBarrera”, “ControlFarola” y “ControlSensores”).

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
  // Iniciar la comunicación serie
  Serial.begin(57600);
}

void loop()
{
  // Si se ha pulsado alguna tecla, hay datos en el puerto serie
  if(Serial.available() > 0)
  {
    // Leer qué tecla se ha pulsado
    tecla = Serial.read();

    ControlBarrera();
    ControlFarola();
    ControlSensores();

    // Resetar variable
    tecla = 0;
  }
}
```

Para facilitar la comprensión del programa, se han definido tres funciones para controlar los elementos de la ciudad, como hemos visto antes. A continuación se describe cada una de estas funciones.

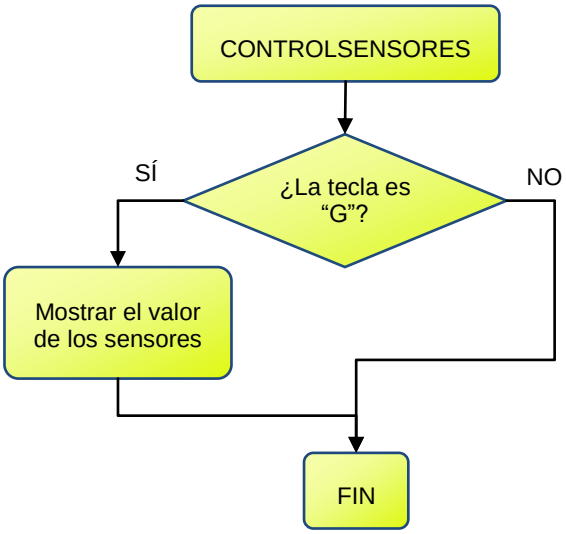
La barrera se activa con la tecla “B”, tanto para subir como para bajar. Para saber qué acción tiene que realizar el programa, se usa la variable “barreraArriba”. Esta variable es “true” cuando la barrera está subida y “false” cuando está bajada. Cada vez que se envía la tecla “B”, se comprueba el estado de la barrera: si está bajada, la acción a realizar es subirla, mientras que si está subida, la acción a realizar será bajarla.



El código es el siguiente:

```
void ControlBarrera()
{
 if(tecla == 'b' || tecla == 'B')
 {
  if(barreraArriba == true)
  {
   // Bajar barrera
   barreraArriba = false;
   barrera.Iniciar(CON1_DIG);
   barrera.Bajar();
   delay(300);
   barrera.Parar();
  }
  else
  {
   // Subir barrera
   barreraArriba = true;
   barrera.Iniciar(CON1_DIG);
   barrera.Subir();
   delay(300);
   barrera.Parar();
  }
 }
}
```

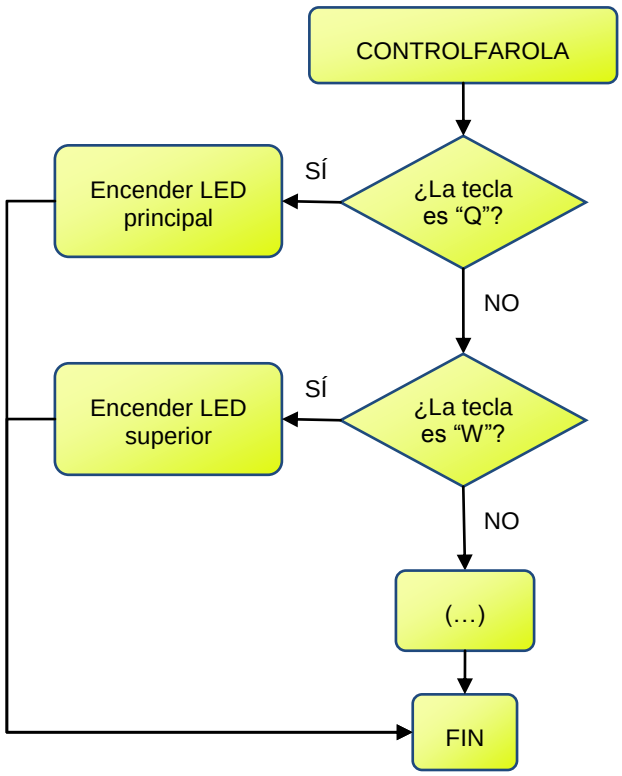
En el caso de la lectura de los sensores, la función es más sencilla. Los sensores se leen al enviar la tecla “G”. Esta función comprueba si la tecla recibida es “G” y, en ese caso, envía los valores de los sensores al monitor serie.



El código es el siguiente:

```
void ControlSensores()
{
  if(tecla == 'g' || tecla == 'G')
  {
    sensorVal = sensorProximidad.Distancia();
    Serial.print("Sensor proximidad:\t");
    Serial.println(sensorVal);
    sensorVal = sensorLuz.Luz();
    Serial.print("Sensor luz:\t\t");
    Serial.println(sensorVal);
    sensorVal = sensorTemperatura.Temperatura(C);
    Serial.print("Sensor temperatura:\t");
    Serial.println(sensorVal);
    Serial.println("---------------------------------------");
  }
}
```

En el caso de la farola, con las teclas “Q”, “W”, “E” y “R” se activan los diferentes LEDs, mientras que con las teclas “A”, “S”, “D” y “F” se apagan. Por simplicidad, el diagrama siguiente indica sólo las dos primeras acciones. El resto se realizarían de forma análoga:



El código de la función para controlar la farola es el siguiente:

```
void ControlFarola()
{
 if(tecla == 'q' || tecla == 'Q')
   farola_1.Encender(NIVEL_7);

 else if(tecla == 'w' || tecla == 'W')
   farola_1.Poste(SUPERIOR, ON);

 else if(tecla == 'e' || tecla == 'E')
   farola_1.Poste(MEDIO, ON);

 else if(tecla == 'r' || tecla == 'R')
   farola_1.Poste(INFERIOR, ON);

 else if(tecla == 'a' || tecla == 'A')
   farola_1.Apagar();

 else if(tecla == 's' || tecla == 'S')
   farola_1.Poste(SUPERIOR, OFF);

 else if(tecla == 'd' || tecla == 'D')
   farola_1.Poste(MEDIO, OFF);

 else if(tecla == 'f' || tecla == 'F')
   farola_1.Poste(INFERIOR, OFF);
}
```

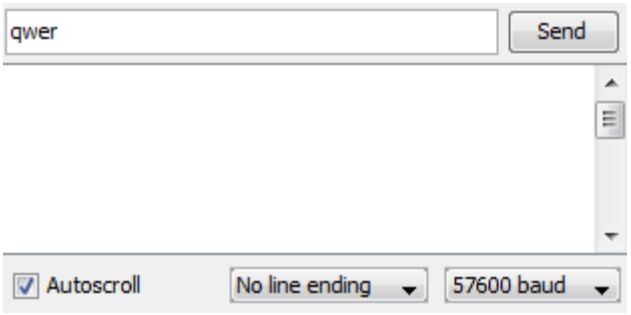
El programa completo queda de la siguiente forma:

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                       // Librería de comunicación I2C
#include <lib_io.h>                     // Librería de puertos E/S
#include <Servo.h>                      // Librería de servomotores
#include <barrier.h>                    // Librería de barrera
#include <proximity_sensor.h>           // Librería de sensor de proximidad
#include <light_sensor.h>               // Librería de sensor de luz
#include <temperature_sensor.h>         // Librería de sensor de temperatura
#include <streetlight.h>                // Librería de farolas

//*****************************************************
//  Variables
//*****************************************************
char tecla;                             // Almacena la tecla recibida
boolean barreraArriba = false;
int sensorVal;

//*****************************************************
//  Dispositivos
//*****************************************************
Barrier barrera;
ProximitySensor sensorProximidad(CON2);
LightSensor sensorLuz(CON4);
TemperatureSensor sensorTemperatura(CON5);
Streetlight farola_1(FAROLA_1);
Streetlight farola_2(FAROLA_2);
Streetlight farola_3(FAROLA_3);

//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
  // Iniciar la comunicación serie
  Serial.begin(57600);
}

void loop()
{
  // Si se ha pulsado alguna tecla, hay datos en el puerto serie
  if(Serial.available() > 0)
  {
    // Leer qué tecla se ha pulsado
    tecla = Serial.read();

    ControlBarrera();
    ControlFarola();
    ControlSensores();

    // Resetar variable
    tecla = 0;
  }
}
```

```
//*****************************************************
//
// Funciones
//
//*****************************************************


//*****************************************************
//  Control de barrera
//*****************************************************
void ControlBarrera()
{
 if(tecla == 'b' || tecla == 'B')
 {
  if(barreraArriba == true)
  {
   // Bajar barrera
   barreraArriba = false;
   barrera.Iniciar(CON1_DIG);
   barrera.Bajar();
   delay(300);
   barrera.Parar();
  }
  else
  {
   // Subir barrera
   barreraArriba = true;
   barrera.Iniciar(CON1_DIG);
   barrera.Subir();
   delay(300);
   barrera.Parar();
  }
 }
}



//*****************************************************
//  Control de sensores
//*****************************************************
void ControlSensores()
{
 if(tecla == 'g' || tecla == 'G')
 {
  sensorVal = sensorProximidad.Distancia();
  Serial.print("Sensor proximidad:\t");
  Serial.println(sensorVal);
  sensorVal = sensorLuz.Luz();
  Serial.print("Sensor luz:\t\t");
  Serial.println(sensorVal);
  sensorVal = sensorTemperatura.Temperatura(C);
  Serial.print("Sensor temperatura:\t");
  Serial.println(sensorVal);
  Serial.println("---------------------------------------");
 }
}
```

```
//*****************************************************
//  Control de farola
//*****************************************************
void ControlFarola()
{
 if(tecla == 'q' || tecla == 'Q')
  farola_1.Encender(NIVEL_7);

 else if(tecla == 'w' || tecla == 'W')
  farola_1.Poste(SUPERIOR, ON);

 else if(tecla == 'e' || tecla == 'E')
  farola_1.Poste(MEDIO, ON);

 else if(tecla == 'r' || tecla == 'R')
  farola_1.Poste(INFERIOR, ON);

 else if(tecla == 'a' || tecla == 'A')
  farola_1.Apagar();

 else if(tecla == 's' || tecla == 'S')
  farola_1.Poste(SUPERIOR, OFF);

 else if(tecla == 'd' || tecla == 'D')
  farola_1.Poste(MEDIO, OFF);

 else if(tecla == 'f' || tecla == 'F')
  farola_1.Poste(INFERIOR, OFF);
}
```

NOTA: Es posible enviar un conjunto de teclas a la vez. Por ejemplo, si se quisiera encender todos los LEDs de la farola, se podría enviar el siguiente conjunto de teclas:



Aplicaciones

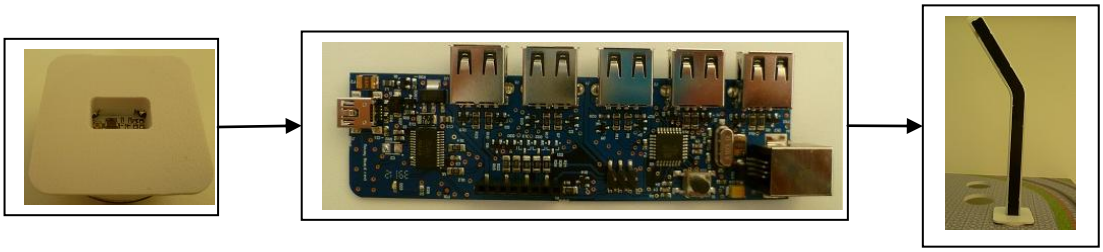
Este programa es un ejemplo de cómo podría manejarse la ciudad desde un centro de control remoto.

Ejercicio 7: Iluminación nocturna

En este ejercicio las farolas se activarán de forma autónoma cuando la luz ambiente sea baja. Para saber si está oscureciendo, usaremos el sensor de luz.

Estos son los elementos de mOway Smart City que se emplearán en esta práctica:

- Tarjeta controladora
- Sensor de luz
- Farolas



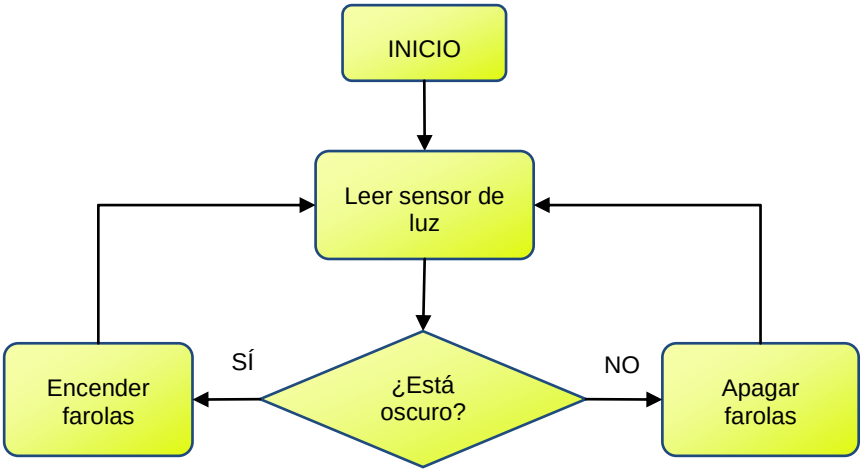
Estrategia

El objetivo de la práctica es encender las farolas de mOway Smart City cuando oscurezca, es decir, cuando la luz ambiental descienda por debajo de un nivel determinado. En cambio, si está amaneciendo, las farolas deben ser apagadas.

Para conocer cuál es el nivel de luz ambiental, es necesario leer el sensor de luz. Este sensor proporciona un valor que depende de la cantidad de luz que está recibiendo. Este valor varía entre el 0% y el 100%, siendo del 0% cuando hay oscuridad y del 100% cuando la luz es muy intensa.

Una vez que conocemos el valor del sensor hay que compararlo con un valor de referencia: si el valor del sensor es más bajo que esta referencia, significará que la cantidad de luz ambiental es baja y por tanto habrá que encender las farolas. En cambio, si el valor del sensor es más alto que la referencia, significará que la luz ambiental es suficiente y que habrá que apagar las farolas.

A continuación se muestra en un diagrama de flujo la estrategia del programa para esta aplicación:



Programa 7.1

Empezaremos incluyendo la librería del sensor de luz (“light_sensor.h”) y la librería de las farolas (“streetlight.h”), junto con las librerías habituales para mOway Smart City (“Wire.h” y “lib_io.h”).

```
//*****************************************
// Librerías
//*****************************************
#include <Wire.h>                 // Librería de comunicación I2C
#include <lib_io.h>               // Librería de puertos E/S
#include <light_sensor.h>         // Librería del sensor de luz
#include <streetlight.h>          // Librería de las farolas
```

También se utilizará una variable para guardar el valor del sensor de luz:

```
//*****************************************
// Variables
//*****************************************
int sensorVal;          // Variable para el valor del sensor de luz
```

Se define la farola que vamos a utilizar. En el ejemplo sólo se usa una farola para que el código sea lo más sencillo posible, en este caso una farola con identificador “1”. Las farolas siempre se conectan en el puerto 3. Por otra parte, como se ve en el código, el sensor de luz se conecta en el puerto 4 (CON4):

```
//**************************************
// Dispositivos
//**************************************
LightSensor sensorLuz(CON4);     // Sensor de luz en conector 4
Streetlight farola_1(FAROLA_1);  // Farola 1 en conector 3
```

En el programa principal se lee el sensor de luz. Si el valor leído es menor que la referencia, en este caso el 25%, entonces consideramos que hay poca luz ambiental y es necesario encender la farola. En cambio si el nivel de luz es mayor del 25%, consideramos que hay luz suficiente y apagamos a farola.

NOTA: El valor de referencia puede ser ajustado en función del entorno donde se encuentre el sensor.

La farola puede encenderse con 7 niveles de intensidad: desde “NIVEL_1” para la intensidad mínima hasta “NIVEL_7” para la intensidad máxima. En este caso, la farola se enciende con la intensidad máxima.

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{

}

void loop()
{
 // Leer sensor de luz
 sensorVal = sensorLuz.Luz();

 // Comparar valor con referencia del 25%
 if(sensorVal < 25)
 {
  farola_1.Encender(NIVEL_7);
 }
 else
 {
  farola_1.Apagar();
 }
}
```

A continuación se muestra el código del programa completo:

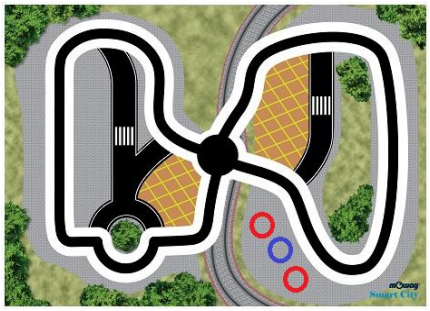
```
//*****************************************
// Librerías
//*****************************************
#include <Wire.h>                  // Librería de comunicación I2C
#include <lib_io.h>                // Librería de puertos E/S
#include <light_sensor.h>          // Librería del sensor de luz
#include <streetlight.h>      // Librería de las farolas

//*************************************
// Variables
//*************************************
int sensorVal;          // Variable para el valor del sensor de luz

//*************************************
// Dispositivos
//*************************************
LightSensor sensorLuz(CON4);     // Sensor de luz en conector 4
Streetlight farola_1(FAROLA_1);  // Farola 1 en conector 3

//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{

}

void loop()
{
  // Leer sensor de luz
  sensorVal = sensorLuz.Luz();

  // Comparar valor con referencia del 25%
  i f(sensorVal < 25)
  {
    farola_1.Encender(NIVEL_7);
  }
  else
  {
    farola_1.Apagar();
  }
}
```

Ejercicio 8: Iluminación por presencia

Una de las ventajas de las ciudades inteligentes es que pueden adaptarse a las condiciones ambientales para ahorrar energía. Por ejemplo, al anochecer en una ciudad normal, la iluminación de las carreteras se enciende y permanece encendida toda la noche. Si por esta carretera no hubiese circulación, se estaría desperdiciando energía.

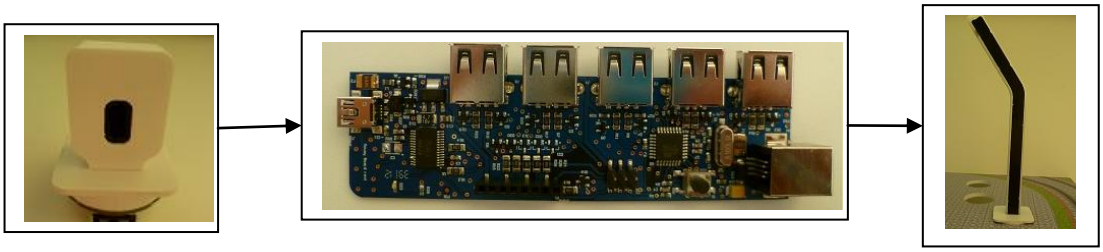
En una ciudad inteligente se puede detectar si hay circulación por la carretera. En caso de haber vehículos circulando, las farolas se encenderían con la máxima intensidad. Si, por el contrario, no hubiese vehículos, las farolas podrían reducir la intensidad de su luz, de modo que se consumiría menos energía.

En este ejercicio cambiaremos el estado de las farolas de la ciudad inteligente en función de la circulación por el circuito, para reducir el consumo energético. Para saber si hay circulación en la zona iluminada por las farolas, utilizaremos el sensor de presencia. La posición de las farolas se indica con los círculos rojos, mientras que la posición del sensor se indica con el círculo azul.



Estos son los elementos de mOway Smart City que se emplearán en esta práctica:

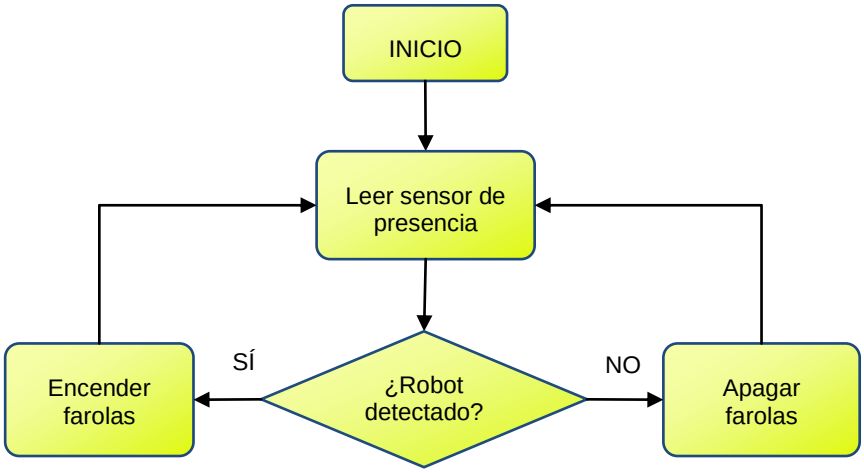
- Tarjeta controladora
- Sensor de luz
- Farolas



Programa 8.1

Para saber si hay algún robot circulando por la zona de las farolas, es necesario leer el sensor de presencia. Este sensor proporciona un valor que depende de la distancia a la que se encuentre un objeto de dicho sensor. Este valor varía entre el 0% y el 100%, siendo del 0% cuando no detecta ningún objeto y del 100% cuando el objeto está muy próximo al sensor.

Una vez que conocemos el valor del sensor hay que compararlo con un valor de referencia, que corresponderá al valor que obtenemos del sensor cuando éste detecta un objeto pasando enfrente. A continuación se muestra en un diagrama de flujo la estrategia del programa para esta aplicación:



Comenzamos incluyendo las librerías para el sensor de presencia y para las farolas:

```
//*****************************************
// Librerías
//*****************************************
#include <Wire.h>                    // Librería de comunicación I2C
#include <lib_io.h>                  // Librería de puertos E/S
#include <proximity_sensor.h>        // Librería del sensor de proximidad
#include <streetlight.h>             // Librería de las farolas
```

Definimos una variable para guardar el valor del sensor de proximidad, que será la distancia en tanto por ciento:

```
//*****************************************
// Variables
//*****************************************
int valorSensor;                     // Variable para el valor del sensor de presencia
```

Definimos los elementos de la ciudad que utilizaremos. El sensor de presencia se conecta en el puerto 2 (**CON2**) y las farolas en el puerto 3 (**CON3 / I²C**). Utilizamos las farolas identificadas con los números “1” y “2” en sus bases:

```
//**************************************
// Dispositivos
//**************************************
ProximitySensor sensorProximidad(CON2);        // Sensor de proximidad en conector 2
Streetlight farola_1(FAROLA_1);                // Farolas en conector 3
Streetlight farola_2(FAROLA_2);
```

En el programa principal se lee el sensor de presencia. Cuando el sensor de presencia no detecta ningún objeto en frente, tiene un valor de “0”. Cuando un objeto comienza a acercarse al sensor, su valor aumenta. En este caso, se ha elegido un valor de “5” para considerar que el robot está pasando en frente del sensor (corresponde al 5% del rango del sensor).

NOTA: Este valor se puede ajustar. Cuanto menor sea, más fácilmente detectará al robot, pero también es posible que se active en situaciones no deseadas.

En caso de que el robot sea detectado, las farolas se encenderán. Una vez que el robot ha pasado por debajo de las farolas, el sensor dejará de detectarlo y entonces las farolas se apagarán.

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void loop()
{
 // Leer sensor de proximidad
 valorSensor = sensorProximidad.Distancia();

 // Comparar valor con referencia del 5%
 if(valorSensor > 5)
 {
  // Encender farolas
  farola_1.Encender(NIVEL_7);
  farola_2.Encender(NIVEL_7);
 }
 else
 {
  // Apagar farolas
  farola_1.Apagar();
  farola_2.Apagar();
 }
}
```

El programa completo queda de la siguiente forma:

```
//*****************************************
// Librerías
//*****************************************
#include <Wire.h>                          // Librería de comunicación I2C
#include <lib_io.h>                        // Librería de puertos E/S
#include <proximity_sensor.h>              // Librería del sensor de proximidad
#include <streetlight.h>           // Librería de las farolas

//*****************************************
// Variables
//*****************************************
int valorSensor;                   // Variable para el valor del sensor de presencia

//*****************************************
// Dispositivos
//*****************************************
ProximitySensor sensorProximidad(CON2);          // Sensor de proximidad en conector 2
Streetlight farola_1(FAROLA_1);                  // Farolas en conector 3
Streetlight farola_2(FAROLA_2);

//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{ }

void loop()
{
  // Leer sensor de proximidad
  valorSensor = sensorProximidad.Distancia();

  // Comparar valor con referencia del 5%
  if(valorSensor > 5)
  {
    farola_1.Encender(NIVEL_7);
    farola_2.Encender(NIVEL_7);
  }
  else
  {
    farola_1.Apagar();
    farola_2.Apagar();
  }
}
```

Programa 8.2

Por motivos de seguridad, en un caso real no se apagarían totalmente las farolas, sino que se reduciría la intensidad de iluminación. En el programa siguiente vemos cómo se puede conseguir esto, simplemente sustituyendo la función de apagar las farolas por otra que encendido a un nivel menor de intensidad. Las definiciones de las librerías, las variables y los dispositivos es igual al programa anterior.

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{}

void loop()
{
  // Leer sensor de proximidad
  valorSensor = sensorProximidad.Distancia();

  // Comparar valor con referencia del 5%
  if(valorSensor > 5)
  {
    // Aumentar la intensidad de las farolas
    farola_1.Encender(NIVEL_7);
    farola_2.Encender(NIVEL_7);
  }
  else
  {
    // Reducir la intensidad de las farolas
    farola_1.Encender(NIVEL_1);
    farola_2.Encender(NIVEL_1);
  }
}
```

Aplicaciones

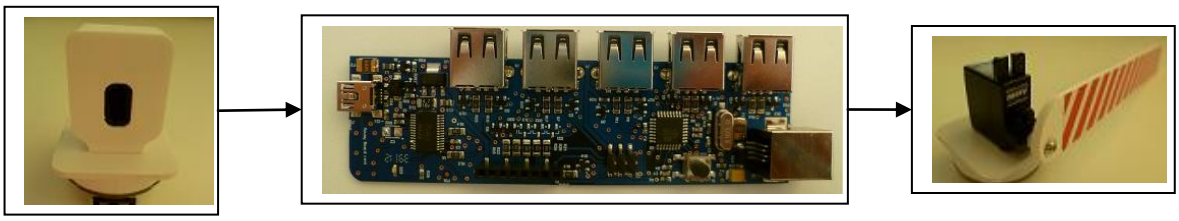
Como hemos visto antes, estos programas son un ejemplo de cómo podría ahorrarse energía en carreteras poco transitadas.

Ejercicio 9: Control de cruce

El circuito de la ciudad inteligente cuenta con un cruce en la parte central. Si hubiese dos robots circulando por el circuito, podría darse el caso de que chocasen en el cruce. Para evitarlo, se puede emplear la barrera junto con el sensor de presencia. Cuando uno de los robots se acerque al cruce, será detectado por el sensor y la barrera bajará, impidiendo que el otro robot pueda acceder al cruce. Una vez que el primer robot haya abandonado el cruce, la barrera subirá y permitirá el paso del otro robot.

Estos son los elementos que se emplearán en esta práctica:

- Tarjeta controladora
- Barrera
- Sensor de presencia
- 2 robots mOway



Estrategia

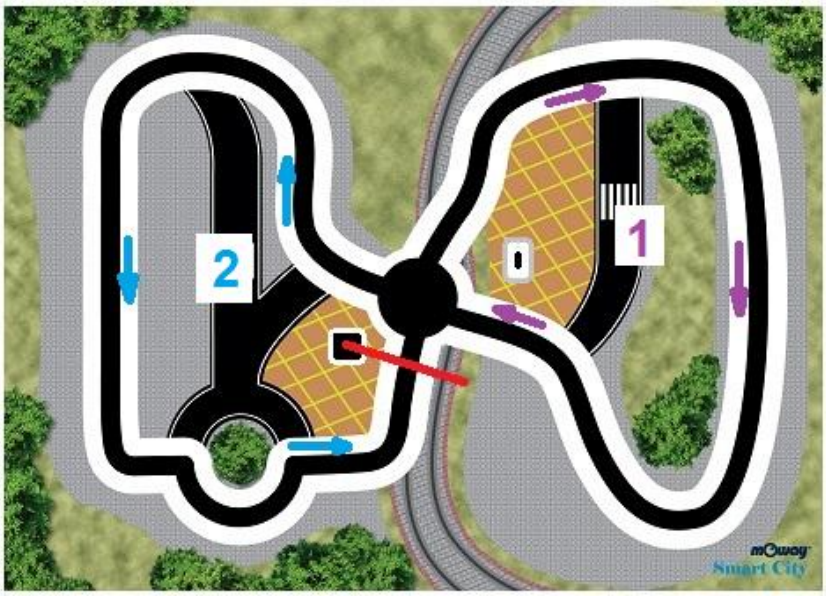
En este ejercicio circularán 2 robots mOway, cada uno de ellos por uno de los circuitos interiores. Por tanto, ambos robots tienen que estar programados para seguir la línea negra. Esto se consigue de la siguiente forma:

- El robot tiene que avanzar cuando uno de los sensores de línea detecta el color blanco y el otro detecta el color negro.
- En otro caso, el robot tiene que corregir su trayectoria para volver al estado anterior.

El **robot 1** circulará según el sentido indicado por las flechas moradas (en la imagen siguiente). Cuando este robot se acerque al cruce, será detectado por el sensor de proximidad. A su vez, la tarjeta controladora recibirá que el sensor de proximidad ha detectado al robot 1. Entonces, bajará la barrera para evitar que el robot 2 entre también en el cruce.

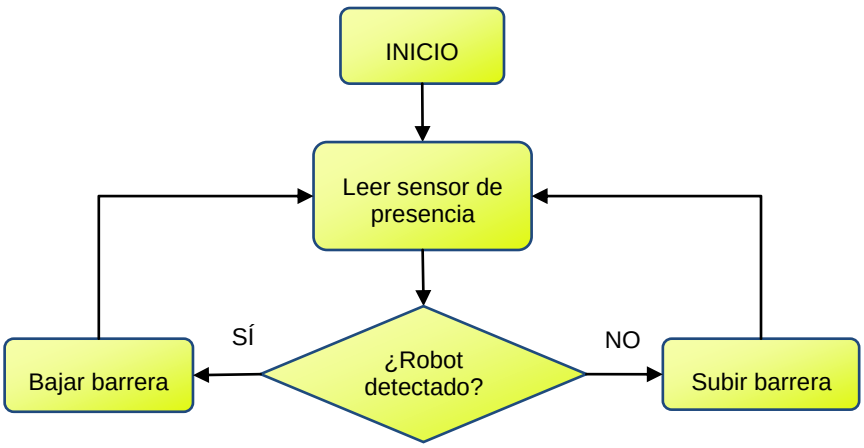
El **robot 2** circulará siguiendo las flechas azules. Además de seguir la línea negra, estará continuamente comprobando los sensores de obstáculos, de modo que si la barrera está bajada, este robot deberá pararse.

NOTA: La barrera y el sensor de presencia pueden colocarse como se indica en la imagen. También se puede cambiar sus posiciones y cambiar el sentido de circulación de los robots.



Estrategia

El programa consiste en leer continuamente el sensor de proximidad del cruce. Si el valor del sensor es mayor que una referencia concreta, significará que está detectando que el robot 1 se está acercando al cruce. En este caso, la barrera debe bajar para evitar que el robot 2 entre también al cruce. Cuando no se detecta al robot, la barrera sube para permitir el paso del robot 2.



Programa 9.1

Comenzamos incluyendo las librerías necesarias:

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Wire.h>                       // Librería de comunicación I2C
#include <lib_io.h>                     // Librería de puertos E/S
#include <Servo.h>                      // Librería de servomotores
#include <barrier.h>                    // Librería de barrera
#include <proximity_sensor.h>           // Librería de sensor de proximidad
```

Se define una variable para guardar el valor de la distancia detectada por el sensor y los elementos de la ciudad. En este caso, el sensor de proximidad se conecta en el puerto 2 (CON2). La barrera siempre se conecta en el puerto CON1:

```
//*****************************************************
//  Variables
//*****************************************************
int distancia;

//*****************************************************
//  Dispositivos
//*****************************************************
Barrier barrera;
ProximitySensor sensorProximidad(CON2);
```

En el programa se compara el valor del sensor de proximidad con una referencia del 5%. Si es mayor que ese valor, significa que está detectando el paso del robot 1 y baja la barrera. En caso de que este valor sea menor a 5, se considera que el robot 1 ya ha pasado y la barrera se levanta, dejando pasar al robot 2.

```
//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{}

void loop()
{
 distancia = sensorProximidad.Distancia();
 if(distancia > 5)
 {
  // Bajar barrera
  barrera.Iniciar(CON1_DIG);
  barrera.Bajar();
  delay(300);
  barrera.Parar();

  // Esperar a que el robot 1 pase por el cruce
  delay(1000);
 }
```

```
 else
  {
    // Subir barrera
    barrera.Iniciar(CON1_DIG);
    barrera.Subir();
    delay(300);
    barrera.Parar();
  }
}
```

El código del programa completo es el siguiente:

```
//*************************************************
//  Librerías
//*************************************************
#include <Wire.h>                       // Librería de comunicación I2C
#include <lib_io.h>                     // Librería de puertos E/S
#include <Servo.h>                      // Librería de servomotores
#include <barrier.h>                    // Librería de barrera
#include <proximity_sensor.h>           // Librería de sensor de proximidad

//*************************************************
//  Dispositivos
//*************************************************
int distancia;

//*************************************************
//  Dispositivos
//*************************************************
Barrier barrera;
ProximitySensor sensorProximidad(CON2);

//*************************************************
//
// Programa principal
//
//*************************************************
void setup()
{ }

void loop()
{
  distancia = sensorProximidad.Distancia();
  if(distancia > 5)
  {
    // Bajar barrera
    barrera.Iniciar(CON1_DIG);
    barrera.Bajar();
    delay(300);
    barrera.Parar();

    // Esperar a que el robot 1 pase por el cruce
    delay(1000);
  }
  else
  {
    // Subir barrera
    barrera.Iniciar(CON1_DIG);
    barrera.Subir();
    delay(300);
    barrera.Parar();
  }
}
```

Aplicaciones

Este programa es un ejemplo de cómo regular el tráfico en un cruce de carreteras, por medio de un sensor de presencia y una barrera.

En las ciudades convencionales, el tráfico se regula por medio de semáforos. En ciertas carreteras se podría colocar un dispositivo que detectase la presencia de vehículos, de modo que el semáforo se ponga en rojo sólo cuando haya circulación de otros vehículos, sin que haya que esperar innecesariamente en caso de que el cruce esté despejado.

Otro ejemplo puede ser el cruce de pasos a nivel, en los que, cuando el tren está pasando, se bajan unas barreras para evitar que los vehículos crucen las vías.

Ejercicio 10: Ciudad inteligente

En este ejercicio vamos a aplicar todo lo aprendido en los ejercicios anteriores para conseguir que la ciudad inteligente se gestione de forma autónoma. Utilizaremos todos los elementos de la ciudad. Va a incluir las siguientes funciones:

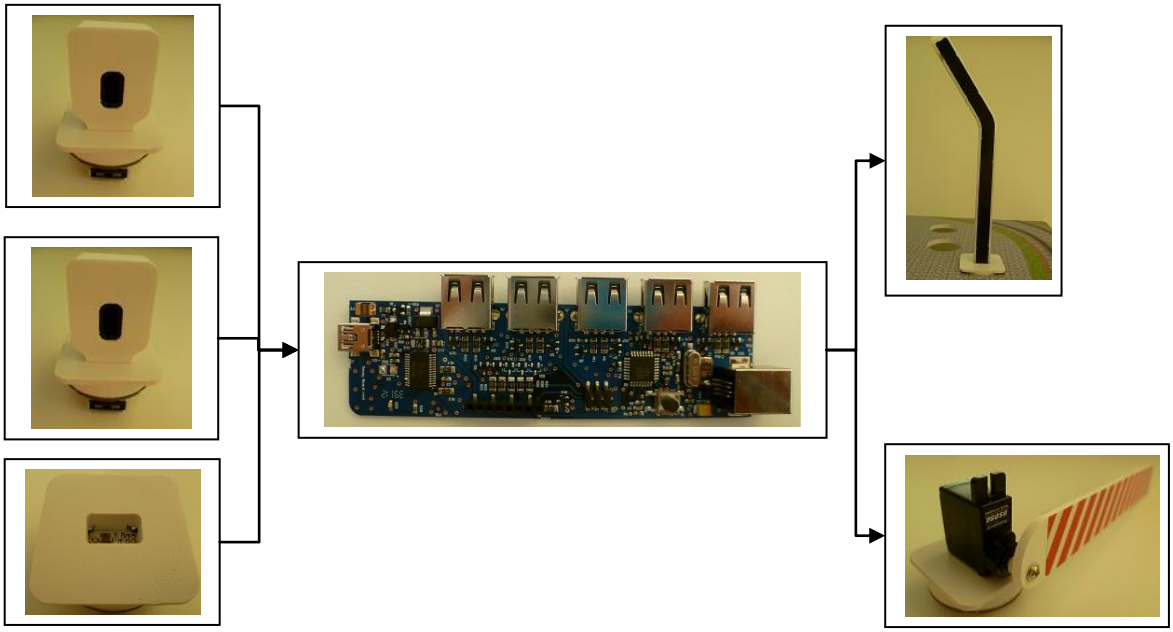
- Control del cruce
- Iluminación nocturna
- Iluminación por presencia

Como en el ejercicio de control del cruce, circularán dos mOways, cada uno de ellos por la parte interior de cada circuito. Para ello, se deberán programar para seguir la línea, así como para detenerse al encontrar un obstáculo.

NOTA: La programación de los robots también puede complementarse con el uso de otros sensores de mOway: el sensor de luz para encender el LED frontal bajo el túnel, el acelerómetro, etc.

Para realizar este ejercicio se van a emplear los siguientes elementos:

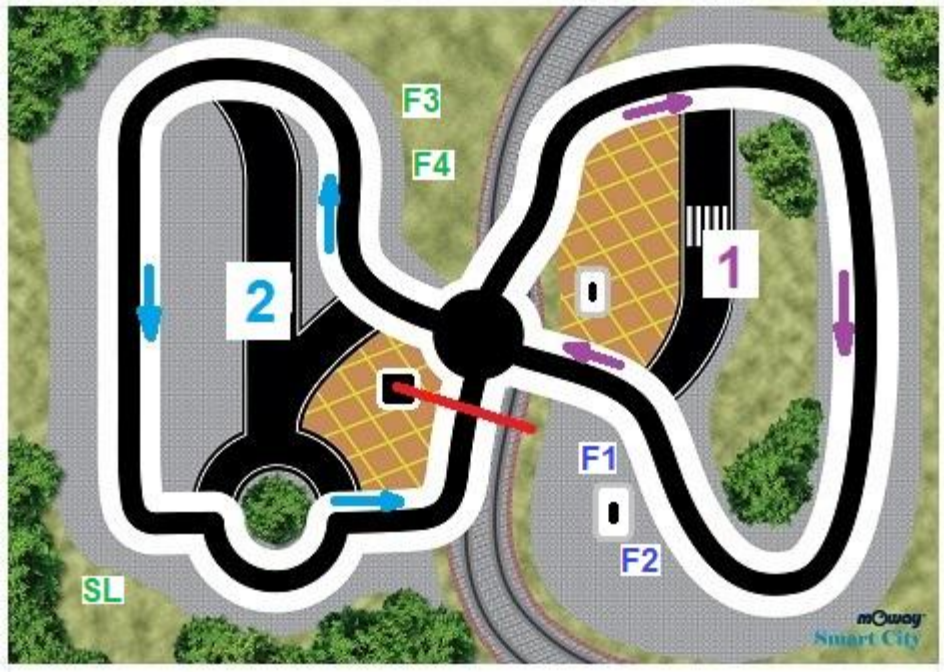
- Tarjeta controladora
- Farolas
- Barrera
- Sensor de presencia
- Sensor de luz
- Sensor de temperatura
- 2 robots mOway



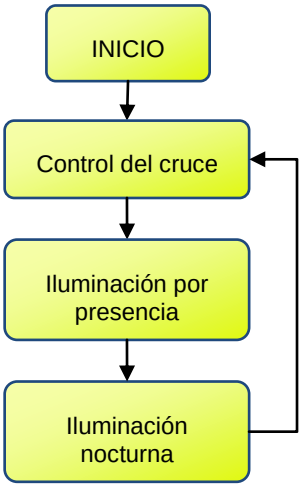
A continuación se indica las posiciones en las que se colocan los elementos para esta práctica, así como el puerto de la tarjeta controladora al que se conectan. Como en otras prácticas, las farolas se conectan en serie, desde el puerto **CON3 / I²C**.

NOTA: La farola 4 se adquiere por separado al paquete básico de mOway Smart City.

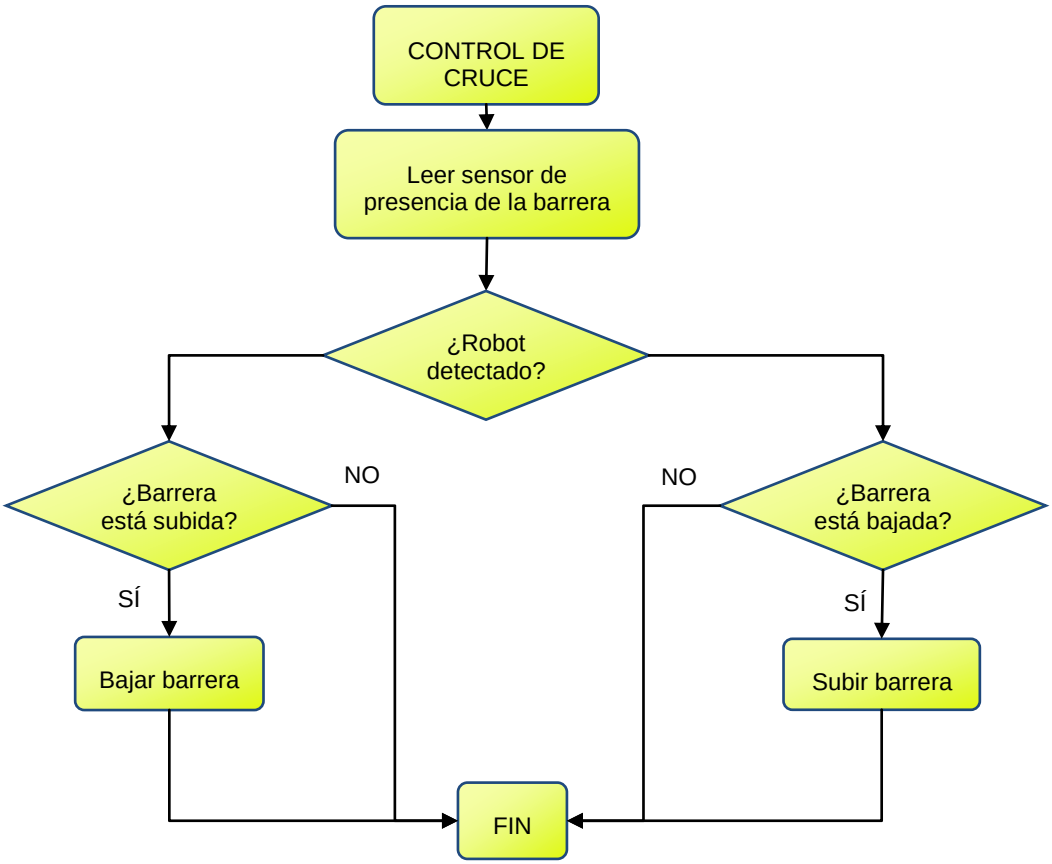
Función	Dispositivo	Símbolo	Puerto de la tarjeta controladora
Control de cruce	Barrera		CON1
	Sensor de proximidad		CON2
Iluminación por presencia	Sensor de proximidad		CON5
	Farola 1	F1	CON3 / I²C
	Farola 2	F2	CON3 / I²C
Iluminación nocturna	Sensor de luz	SL	CON4
	Farola 3	F3	CON3 / I²C
	Farola 4	F4	CON3 / I²C



Vamos a leer todos los sensores y en función del valor de cada uno, realizaremos la acción correspondiente. Para hacer más claro el programa, dividiremos el código en funciones, cada una de ellas encargada de una tarea.



La función de control del cruce se encarga de detectar cuándo el robot 1 se acerca al cruce. Cuando se acerque, esta función bajará la barrera para evitar que el robot 2 entre también en el cruce. Una vez que el robot 1 ha salido del cruce, la barrera volverá a subir para permitir el paso del robot 2.



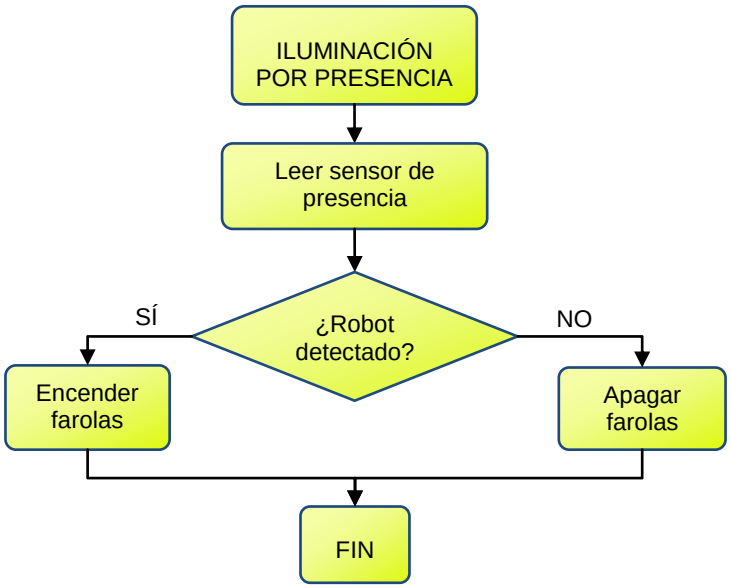
A continuación se muestra el código de la función de control de cruce.

```
//*****************************************************
// Control del cruce
//*****************************************************
void ControlCruce()
{
 // Detectar robot
 valorSensor = sensorCruce.Distancia();
 if(valorSensor > 5)
 {
  // El robot ha sido detectado
  if(subida == true)
  {
   // Si la barrera estaba subida, entonces bajarla
   barrera.Iniciar(CON1_DIG);
   barrera.Bajar();
   delay(300);
   barrera.Parar();
   delay(1000);
   subida = false;
  }
 }
```

```
  else
  {
    // El robot ha sido detectado
    if(subida == false)
    {
      // Si la barrera estaba bajada, entonces subirla
      barrera.Iniciar(CON1_DIG);
      barrera.Subir();
      delay(300);
      barrera.Parar();
      delay(1000);
      subida = true;
    }
  }
}
```

Para que el servomotor de la barrera no se active cuando no es necesario, se utiliza la variable “subida”, que indica el estado de la barrera. Si el robot es detectado y la barrera ya está subida, entonces no es necesario volver a subirla. De forma análoga, una vez que el robot ha pasado el cruce y la barrera ha bajado, no es necesario volver a bajarla.

La función de iluminación por presencia lee el sensor de presencia colocado debajo de las farolas. Cuando el robot pase por esta zona, será detectado por el sensor y las farolas se encenderán. Una vez que el robot ha pasado, el sensor deja de detectarlo y las farolas se apagarán.

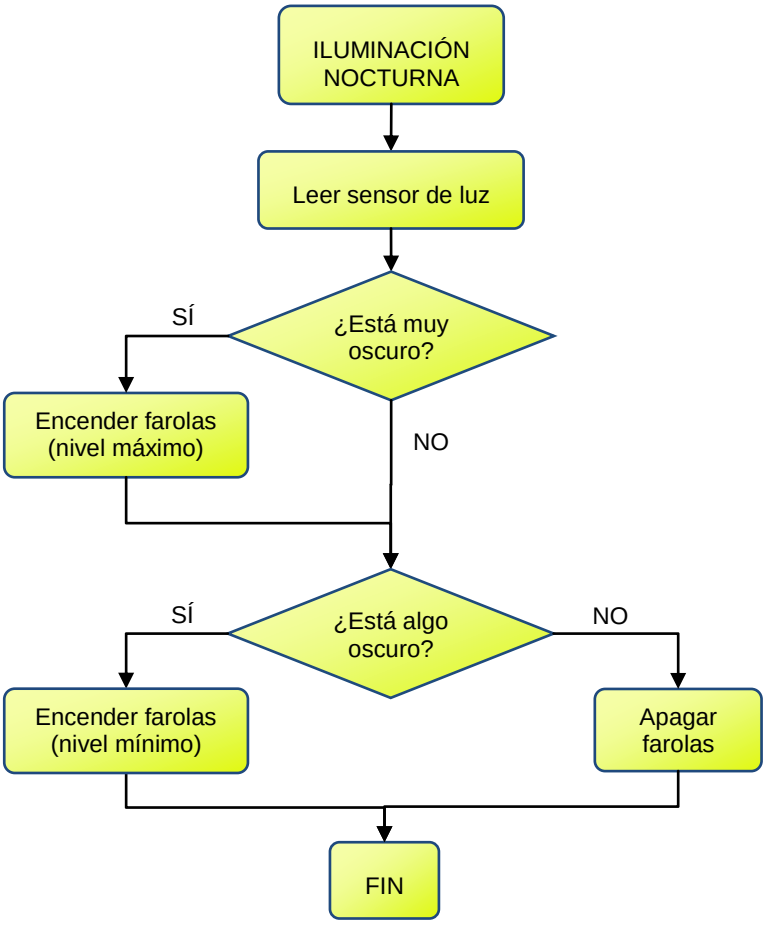


A continuación se muestra el código de la función de iluminación por presencia:

```
//*******************************************************
// Iluminación por presencia
//*******************************************************
void IluminacionPresencia()
{
 // Detectar robot
 valorSensor = sensorFarolas.Distancia();
 if(valorSensor > 5)
 {
  // Encender farolas
  farola_1.Encender(NIVEL_7);
  farola_1.Poste(INFERIOR,ON);
  farola_1.Poste(MEDIO,OFF);
  farola_1.Poste(SUPERIOR,ON);
  farola_2.Encender(NIVEL_7);
  farola_2.Poste(INFERIOR,ON);
  farola_2.Poste(MEDIO,OFF);
  farola_2.Poste(SUPERIOR,ON);
 }
 else
 {
  // Apagar farolas
  farola_1.Apagar();
  farola_1.Poste(INFERIOR,OFF);
  farola_1.Poste(MEDIO,ON);
  farola_1.Poste(SUPERIOR,OFF);
  farola_2.Apagar();
  farola_2.Poste(INFERIOR,OFF);
  farola_2.Poste(MEDIO,ON);
```

```
  farola_2.Poste(SUPERIOR,OFF);
 }
}
```

La función de iluminación nocturna adapta la luz de las farolas al nivel de iluminación ambiental. Si es muy baja, las farolas se encenderán con el nivel máximo de intensidad. Si hay algo de luz ambiental pero no es suficiente, las farolas se encenderán con el nivel mínimo. Si la luz ambiental es suficiente, las farolas se apagarán.



A continuación se muestra el código completo de la función de iluminación nocturna:

```
//*********************************************************
// Iluminación nocturna
//*********************************************************
void IluminacionNocturna()
{
 // Detectar luz ambiental
 valorSensor = sensorLuz.Luz();
 if(valorSensor < 30)
 {
  // Si está muy oscuro, encender al nivel máximo
  farola_3.Encender(NIVEL_7);
  farola_3.Poste(INFERIOR,ON);
  farola_3.Poste(MEDIO,ON);
  farola_3.Poste(SUPERIOR,ON);
```

```
    farola_4.Encender(NIVEL_7);
    farola_4.Poste(INFERIOR,ON);
    farola_4.Poste(MEDIO,ON);
    farola_4.Poste(SUPERIOR,ON);
   }
   else if(valorSensor < 60)
   {
    // Si está algo oscuro, encender al nivel mínimo
    farola_3.Encender(NIVEL_1);
    farola_3.Poste(INFERIOR,ON);
    farola_3.Poste(MEDIO,ON);
    farola_3.Poste(SUPERIOR,OFF);

    farola_4.Encender(NIVEL_1);
    farola_4.Poste(INFERIOR,ON);
    farola_4.Poste(MEDIO,ON);
    farola_4.Poste(SUPERIOR,OFF);
   }
   else
   {
    // Si no está oscuro, apagar farolas
    farola_3.Apagar();
    farola_3.Poste(INFERIOR,ON);
    farola_3.Poste(MEDIO,OFF);
    farola_3.Poste(SUPERIOR,OFF);
    farola_4.Apagar();
    farola_4.Poste(INFERIOR,ON);
    farola_4.Poste(MEDIO,OFF);
    farola_4.Poste(SUPERIOR,OFF);
   }
}
```

Finalmente, se muestra el código completo de este programa.

```
//*****************************************************
//  Librerías
//*****************************************************
#include <Servo.h>                  // Librería de servomotores
#include <Wire.h>                   // Librería de comunicación I2C
#include <lib_io.h>                 // Librería de puertos E/S
#include <barrier.h>                // Librería de barrera
#include <streetlight.h>            // Librería de farolas
#include <proximity_sensor.h>       // Librería de sensor de proximidad
#include <light_sensor.h>           // Librería de sensor de luz

//*****************************************************
//  Variables
//*****************************************************
int valorSensor;
boolean subida;

//*****************************************************
//  Dispositivos
//*****************************************************
```

```
Barrier barrera;                          // Barrera
ProximitySensor sensorCruce(CON2);        // Sensor de proximidad del cruce
LightSensor sensorLuz(CON4);              // Sensor de luz
ProximitySensor sensorFarolas(CON5);      // Sensor de proximidad de farolas
Streetlight farola_1(FAROLA_1);           // Farola activada por presencia
Streetlight farola_2(FAROLA_2);           // Farola activada por presencia
Streetlight farola_3(FAROLA_3);           // Farola activada por luz
Streetlight farola_4(FAROLA_4);           // Farola activada por luz

//*****************************************************
//
// Programa principal
//
//*****************************************************
void setup()
{
}

void loop()
{
 ControlCruce();
 IluminacionPresencia();
 IluminacionNocturna();
}

//*****************************************************
//
// Programa principal
//
//*****************************************************
//*****************************************************
// Control del cruce
//*****************************************************
void ControlCruce()
{
 // Detectar robot
 valorSensor = sensorCruce.Distancia();
 if(valorSensor > 5)
 {
  // El robot ha sido detectado
  if(subida == true)
  {
   // Si la barrera estaba subida, entonces bajarla
   barrera.Iniciar(CON1_DIG);
   barrera.Bajar();
   delay(300);
   barrera.Parar();
   delay(1000);
   subida = false;
  }
 }
 else
 {
  // El robot ha sido detectado
  if(subida == false)
  {
   // Si la barrera estaba bajada, entonces subirla
   barrera.Iniciar(CON1_DIG);
   barrera.Subir();
   delay(300);
   barrera.Parar();
   delay(1000);
   subida = true;
  }
```

```
  }
}



//*****************************************************
// Iluminación por presencia
//*****************************************************
void IluminacionPresencia()
{
 // Detectar robot
 valorSensor = sensorFarolas.Distancia();
 if(valorSensor > 5)
 {
  // Encender farolas
  farola_1.Encender(NIVEL_7);
  farola_1.Poste(INFERIOR,ON);
  farola_1.Poste(MEDIO,OFF);
  farola_1.Poste(SUPERIOR,ON);
  farola_2.Encender(NIVEL_7);
  farola_2.Poste(INFERIOR,ON);
  farola_2.Poste(MEDIO,OFF);
  farola_2.Poste(SUPERIOR,ON);
 }
 else
 {
  // Apagar farolas
  farola_1.Apagar();
  farola_1.Poste(INFERIOR,OFF);
  farola_1.Poste(MEDIO,ON);
  farola_1.Poste(SUPERIOR,OFF);
  farola_2.Apagar();
  farola_2.Poste(INFERIOR,OFF);
  farola_2.Poste(MEDIO,ON);
  farola_2.Poste(SUPERIOR,OFF);
 }
}

//*****************************************************
// Iluminación nocturna
//*****************************************************
void IluminacionNocturna()
{
 // Detectar luz ambiental
 valorSensor = sensorLuz.Luz();
 if(valorSensor < 30)
 {
  // Si está muy oscuro, encender al nivel máximo
  farola_3.Encender(NIVEL_7);
  farola_3.Poste(INFERIOR,ON);
  farola_3.Poste(MEDIO,ON);
  farola_3.Poste(SUPERIOR,ON);

  farola_4.Encender(NIVEL_7);
  farola_4.Poste(INFERIOR,ON);
  farola_4.Poste(MEDIO,ON);
  farola_4.Poste(SUPERIOR,ON);
 }
 else if(valorSensor < 60)
 {
  // Si está algo oscuro, encender al nivel mínimo
  farola_3.Encender(NIVEL_1);
  farola_3.Poste(INFERIOR,ON);
  farola_3.Poste(MEDIO,ON);
```

```
    farola_3.Poste(SUPERIOR,OFF);

    farola_4.Encender(NIVEL_1);
    farola_4.Poste(INFERIOR,ON);
    farola_4.Poste(MEDIO,ON);
    farola_4.Poste(SUPERIOR,OFF);
   }
   else
   {
    // Si no está oscuro, apagar farolas
    farola_3.Apagar();
    farola_3.Poste(INFERIOR,ON);
    farola_3.Poste(MEDIO,OFF);
    farola_3.Poste(SUPERIOR,OFF);
    farola_4.Apagar();
    farola_4.Poste(INFERIOR,ON);
    farola_4.Poste(MEDIO,OFF);
    farola_4.Poste(SUPERIOR,OFF);
   }
  }
```