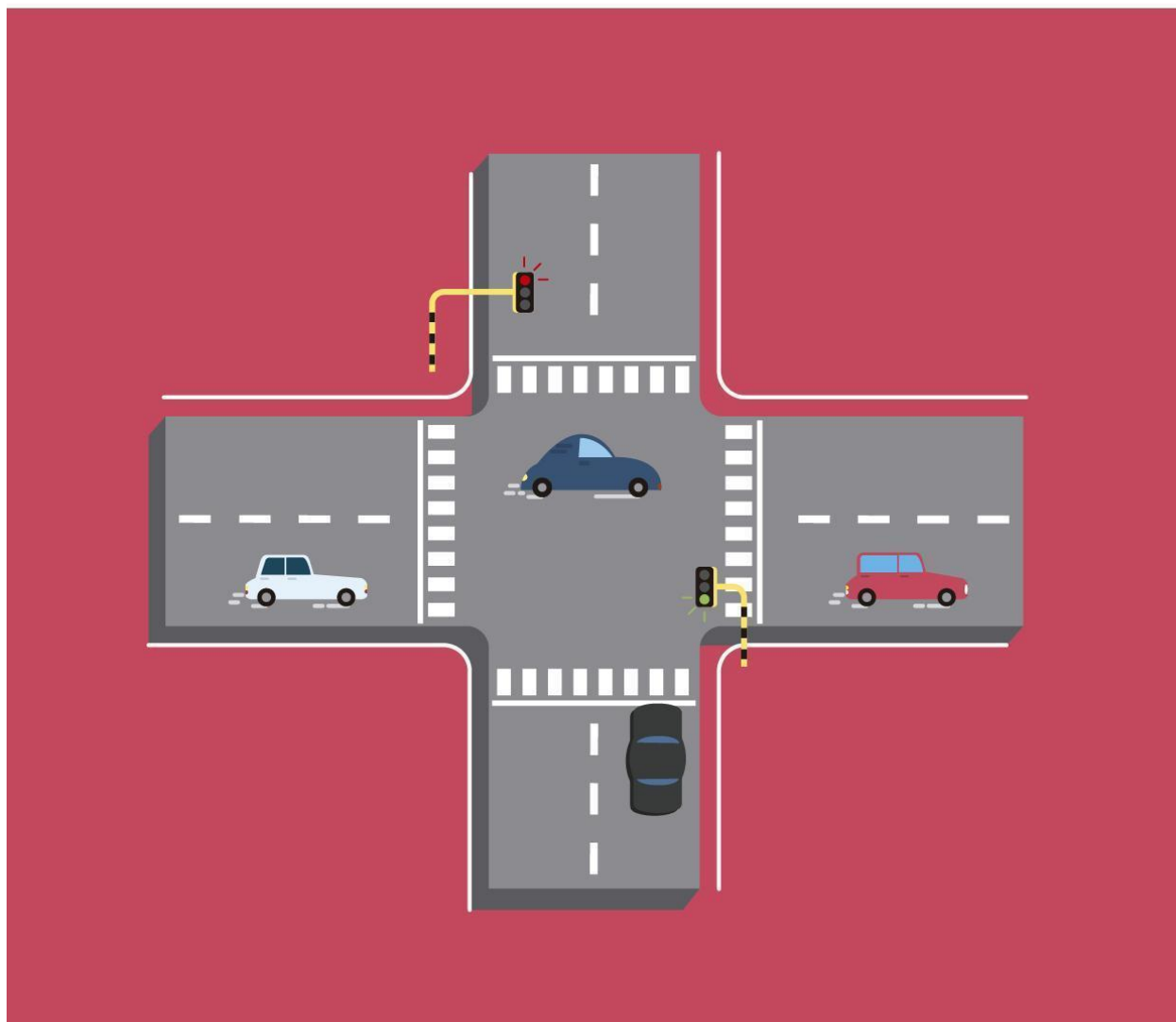


SABERES DIGITALES



AUTOMATIZACIÓN DE SISTEMA DE SEMAFORIZACIÓN

AUTORIDADES

Presidente de la Nación

Mauricio Macri

Vicepresidenta de la Nación

Marta Gabriela Michetti

Jefe de Gabinete de Ministros

Marcos Peña

Ministro de Educación, Cultura, Ciencia y Tecnología

Alejandro Finocchiaro

Titular de la Unidad de Coordinación General del Ministerio de Educación, Cultura, Ciencia y Tecnología

Manuel Vidal

Subsecretario de Coordinación Administrativa

Javier Mezzamico

Director Ejecutivo INET

Leandro Goroyesky

Gerenta General de EDUCAR Sociedad del Estado

Liliana Casaleggio

Directora Nacional de Asuntos Federales

María José Licio Rinaldi

Director Nacional de Educación Técnico - Profesional

Fabián Prieto

Coordinador de Secundaria Técnica

Alejandro Anchava

Responsable de Formación Docente Inicial y Continua INET

Judit Schneider

Coordinador General En FoCo

Pablo Trangone

SISTEMA DE SEMAFORIZACIÓN

Ficha técnica	3
Presentación	4
Desarrollo	5
Nivel Inicial	5
Paso 1 - Realizar el montaje sencillo de un circuito en una Protoboard	5
Paso 2 - Programar los tiempos de encendido de cada luz del semáforo.	6
Paso 3 - Subir el código a la placa Arduino	9
Paso 4 - Expandir el circuito a tres LED	10
Paso 5 - Soldar y ensamblar	12
Nivel Intermedio	15
Paso 1 - Armar el circuito	15
Paso 2 - Programar la secuencia para dos semáforos.	16
Nivel Superior	20
Paso 1 - Introducción a Internet de las Cosas (IoT)	20
Paso 2 - Crear un Panel de Control	21
Paso 3 - Conectar módulo OBLOQ	28
Paso 4 - Arduino IDE	28
Paso 5 - Programar sin código bloqueante	30
Paso 6 - Programación IoT.	34
Cierre	40
Glosario	42
Reconocimientos	43

SISTEMA DE SEMAFORIZACIÓN

Ficha técnica

Nivel educativo	Secundario. Ciclo Básico.
Descripción general	Diseño y construcción de una maqueta/prototipo de un sistema de semáforos para la circulación ordenada y óptima del tráfico vehicular.
Niveles de complejidad	<p>Nivel inicial: realizar, con una placa Arduino, el circuito de funcionamiento de un semáforo y construir la maqueta que lo aloje con impresión 3D.</p> <p>Nivel intermedio: generar un segundo semáforo que esté sincronizado con el primero en la misma intersección.</p> <p>Nivel avanzado: informar los datos obtenidos a un dispositivo móvil a través de Internet de las Cosas (IoT).</p>
Insumos	<ul style="list-style-type: none"> • 2 x LED rojo 5mm • 2 x LED amarillo 5mm • 2 x LED verde 5mm • 10 x Resistencias de 220 ohm • 20 x Cables dupont macho hembra • 20 x Cables dupont macho macho • Filamento para impresora 3D PLA • 1 x Arduino UNO R3 • 1 x Protoboard • 1 x Cable USB tipo B • 1x OBLOQ Modulo IoT • 1 x Fuente de 9V 1A (plug centro positivo, 5.5 x 2.1mm)
Equipamiento	<ul style="list-style-type: none"> • Computadora • Impresora 3D • Soldador • Estaño • Alicata • Pinza de punta • Brusela
Otros requisitos	<ul style="list-style-type: none"> • Conexión a internet

	<ul style="list-style-type: none"> • Descargar el programa “mBlock3” http://www.mblock.cc/software-1/mblock/mblock3/
--	---

Presentación

Descripción ampliada del proyecto

En este proyecto, se proponen inicialmente el diseño, la construcción y la programación de un prototipo de un semáforo. El mismo estará compuesto por 6 luces LED (2 de cada color del semáforo tradicional -verde, amarillo y rojo-) y estará programado para que realice una secuencia en la que cambie el encendido de las luces de cada color en un tiempo determinado.

En los niveles de complejidad intermedio y avanzado se propone programar dos semáforos para una intersección que estén sincronizados entre sí. En última instancia, los datos de operación de los semáforos se compartirán a través de Internet de las Cosas (IoT) para poder realizar un seguimiento y un control de su estado desde la central.

Al final de esta guía se puede encontrar un glosario donde se provee la información técnica necesaria para poder poner el proyecto en funcionamiento. El mismo cuenta con aclaraciones sobre los diversos elementos electrónicos involucrados así como también conceptos claves.

Objetivos

- Aproximarse al conocimiento y al manejo de distintos componentes electrónicos mediante la construcción de un prototipo de un semáforo (y de una intersección con dos semáforos en los niveles intermedio y avanzado).
- Construir circuitos eléctricos sencillos con el propósito de simular el funcionamiento de las luces de un semáforo (y dos semáforos en los niveles intermedio y avanzado).
- Conocer los componentes de la interfaz de Arduino.
- Analizar la programación de estructura secuencial de un programa que permita el funcionamiento de un prototipo de semáforo y su sincronización con otro semáforo (nivel intermedio).
- Aproximarse a la impresión 3D.
- Utilizar IoT (Internet de las Cosas) para registrar y monitorear el funcionamiento de los semáforos (nivel avanzado).
- Utilizar una placa Protoboard para realizar pruebas rápidas del funcionamiento del circuito. Soldar componentes electrónicos para el armado de la maqueta final.

Desarrollo

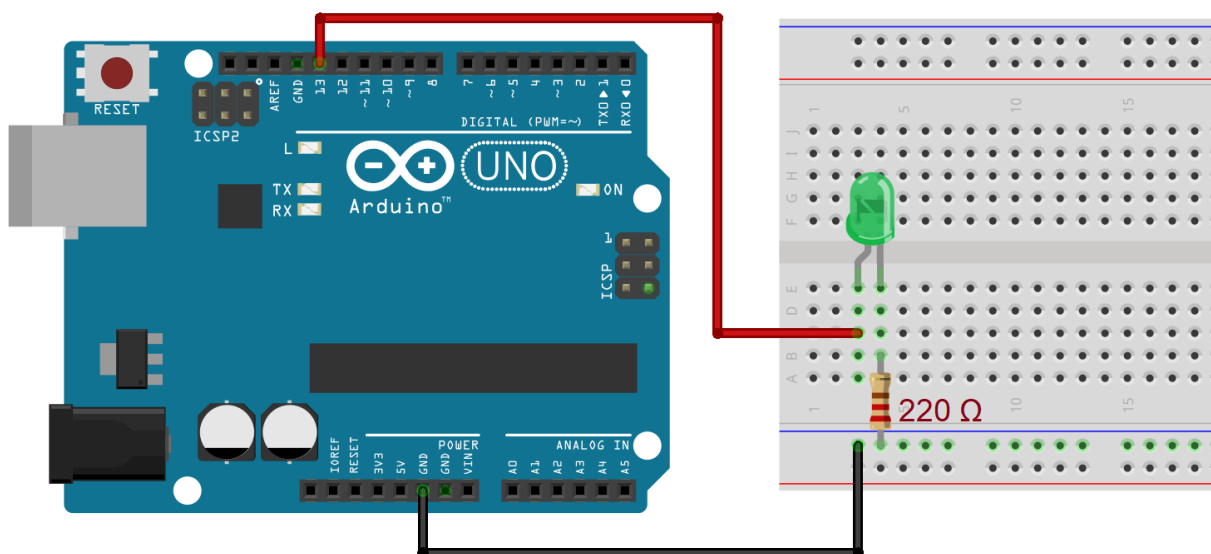
Nivel Inicial

Se necesita fabricar un semáforo para una calle muy transitada del centro. Además de construirlo, se deben programar sus luces. Para hacerlo, hay que tener en cuenta cuál es el orden en el que va a encenderse la luz de cada color, cuánto tiempo permanecerá encendida cada una, y cómo será el cambio entre una y otra (por ejemplo, si al prenderse una luz se apagará la otra automáticamente o si coincidirán encendidas juntas durante algunos segundos).

En esta actividad se propone al grupo que genere la maqueta de un semáforo con un circuito de seis luces LED (2 de cada color) y la maqueta realizada por impresión 3D.

Paso 1 - Realizar el montaje sencillo de un circuito en una Protoboard

Colocamos un LED verde en la protoboard, como indica la figura.



Esquema 1

En un LED, la pata larga siempre es la “pata positiva” y es donde se conecta a nuestro *pin*.

- Pata larga positiva
- Pata corta negativa



Paso 2 - Programar los tiempos de encendido de cada luz del semáforo.

La programación la realizaremos con mBlock3, un entorno de programación basado en Scratch2 que permite programar proyectos de Arduino utilizando bloques. Se puede descargar siguiendo este enlace: <http://www.mblock.cc/software-1/mblock/mblock3/>

Cuando abrimos mBlock3, vemos una pantalla como la siguiente:

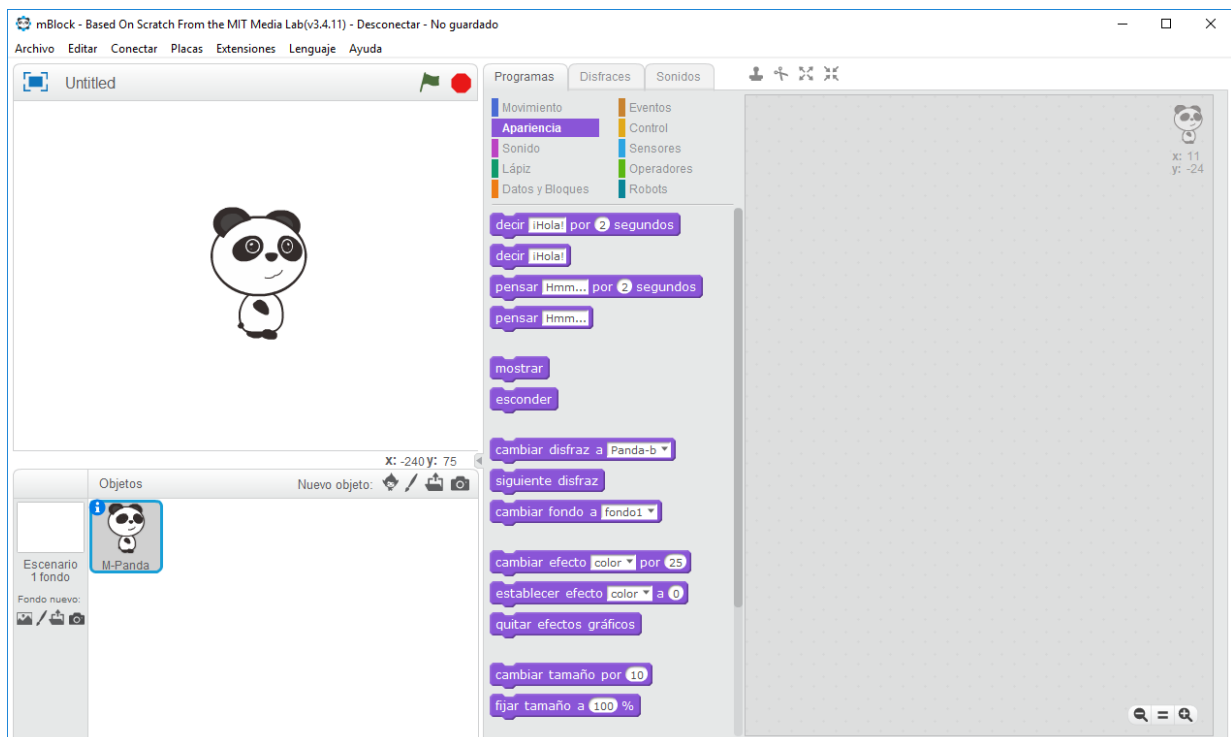


Fig. 1

Para programar un proyecto de Arduino con mBlock3 debemos seleccionar el “Modo Arduino” desde el menú.

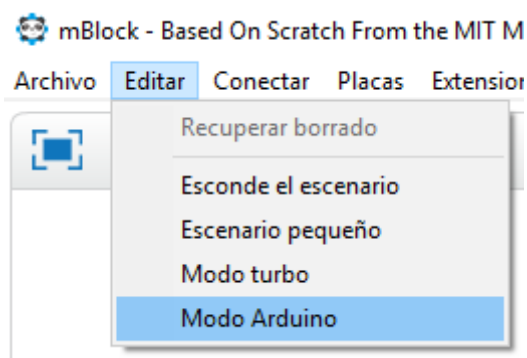


Fig. 2

Al seleccionar este modo, el programa cambiará de aspecto. Se verá un área en el centro que es la que utilizaremos para programar con bloques. A la derecha se verá un campo donde aparecerá el código escrito que le corresponde a los bloques que están en el centro. Este código se irá escribiendo automáticamente a medida que se vaya armando el programa con los bloques.

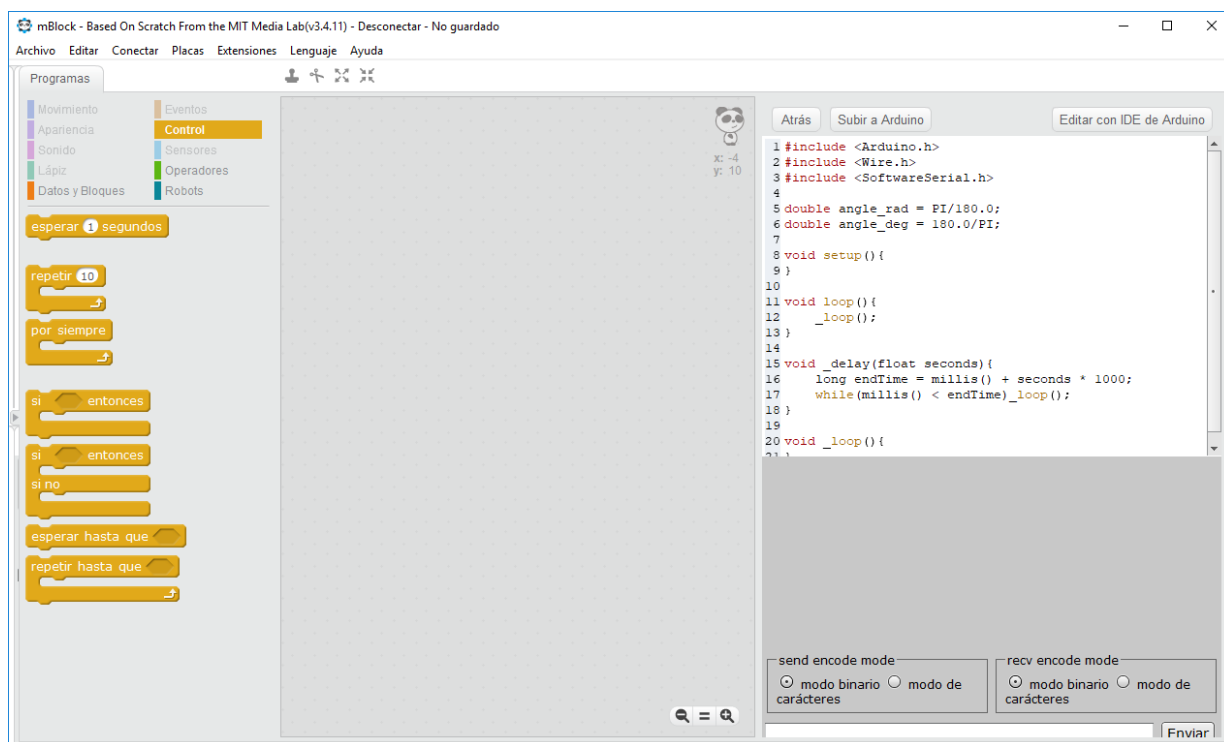


Fig. 3

Los bloques están agrupados por categorías. En este caso, se usarán las siguientes: **“Robots”**, **“Control”**, **“Operadores”** y **“Datos y Bloques”**. Cuando seleccionamos una de estas categorías, se pueden visualizar todos los bloques que pertenecen a ese grupo.



Fig. 4

Después de familiarizarnos con el sistema, vamos a empezar escribir un programa que haga parpadear el LED que acabamos de conectar. Debería verse de manera similar al siguiente modelo:

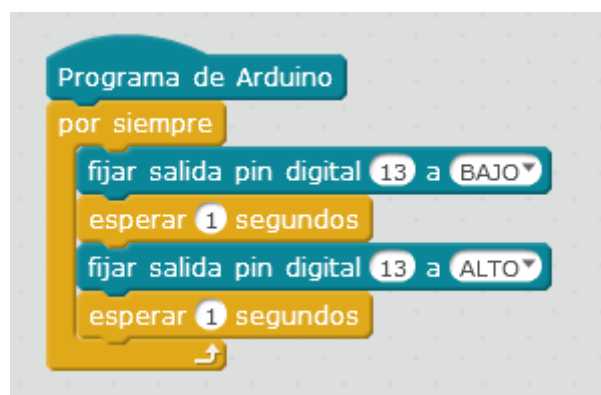


Fig. 5

Veremos que a continuación se muestra el código escrito generado por mBlock, que corresponde a este programa:

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

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;

void setup(){
    pinMode(13,OUTPUT);
}

void loop(){
    digitalWrite(13,1);
    _delay(1);
    digitalWrite(13,0);
    _delay(1);
}
```

```

    _loop();
}

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

void _loop(){
}

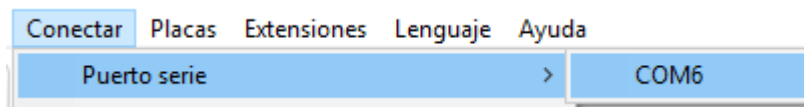
```

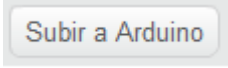
Fragmento de código 1

Paso 3 - Subir el código a la placa Arduino

Para subir el código de nuestro programa a la placa Arduino, necesitamos:

1. Conectar la placa Arduino a la entrada USB.
2. Chequear que en el menú “Placas” esté seleccionado “Arduino Uno”.
3. Seleccionar el puerto serie al que está conectada la placa.



4. Clickear el botón  que se encuentra en la parte derecha superior de la interfaz.

Al terminar de subir nuestro código, veremos este mensaje:

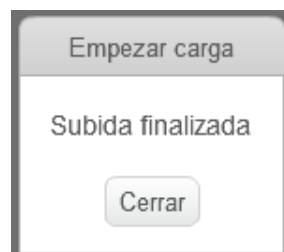
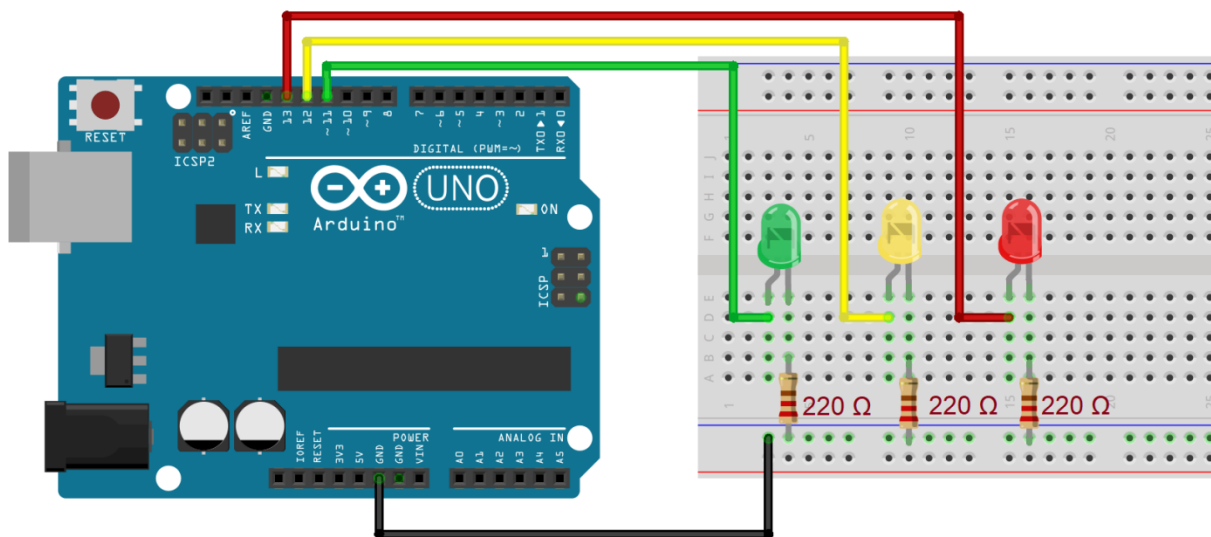


Fig. 6

Con la conexión previamente realizada, el LED debería encenderse y apagarse siguiendo intervalos de un segundo.

Paso 4 - Expandir el circuito a tres LED

Continuamos agregando tres LED, uno de cada color (verde, amarillo y rojo), con sus respectivas resistencias (de 220 ohm), como indica el circuito. Conectamos el rojo al *pin* 13, el amarillo al *pin* 12 y el verde al *pin* 11.



Esquema 2

Colocamos todas las resistencias en los pines GND de la protoboard.

En mBlock3, modificamos el programa anterior para que ahora realice ciclos de encendido y apagado para las luces de los tres colores con los intervalos de demora que nos parezcan adecuados.

El nuevo programa debería ser similar al siguiente modelo:

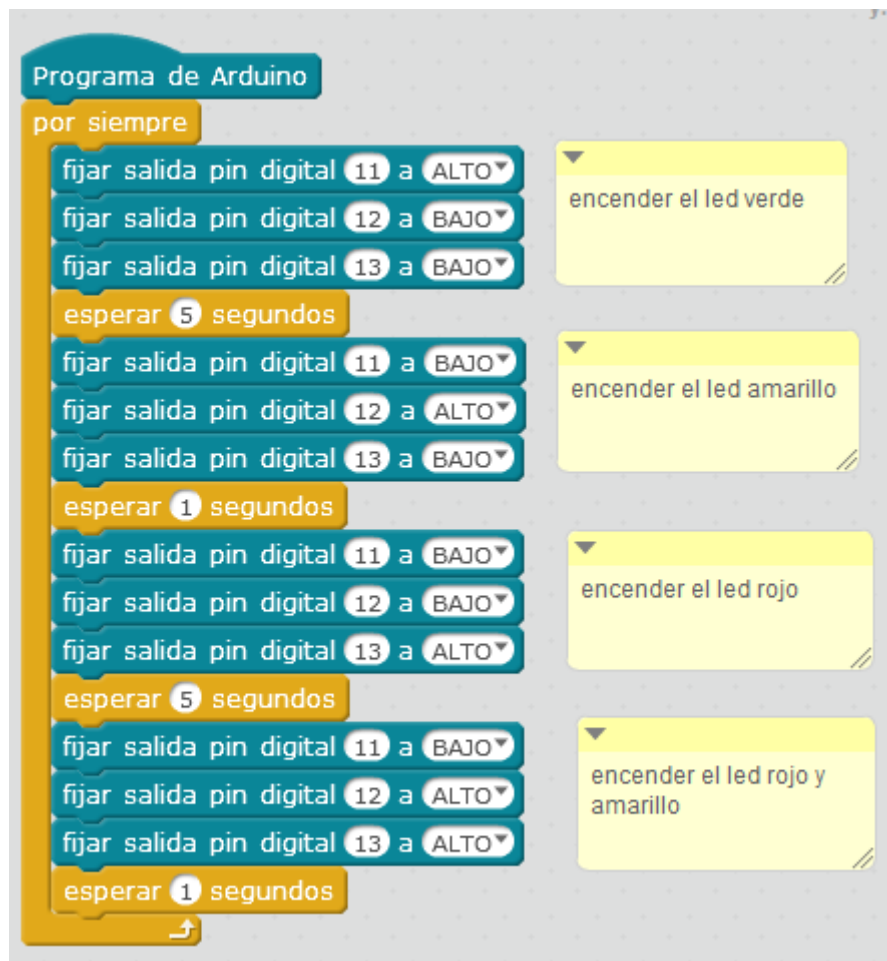


Fig. 7

El código escrito generado por mBlock de la Fig. 7, debería ser similar al siguiente:

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

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;

void setup(){
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}

void loop(){
  digitalWrite(11,1);
  digitalWrite(12,0);
```

```

    digitalWrite(13,0);
    _delay(5);
    digitalWrite(11,0);
    digitalWrite(12,1);
    digitalWrite(13,0);
    _delay(1);
    digitalWrite(11,0);
    digitalWrite(12,0);
    digitalWrite(13,1);
    _delay(5);
    digitalWrite(11,0);
    digitalWrite(12,1);
    digitalWrite(13,1);
    _delay(1);
    _loop();
}

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

void _loop(){
}

```

Fragmento de código 2

Paso 5 - Soldar y ensamblar

En este momento se procederá a construir la maqueta del semáforo. Para ello se debe soldar cada componente con la placa e imprimir el modelo de semáforo 3D que alojará el circuito de las luces. Se debe tener en cuenta que si se pretende seguir con el nivel intermedio y agregar un segundo semáforo, puede resultar más conveniente saltar este paso y continuar con los siguientes para programar la conexión de los dos semáforos con la Protoboard antes del armado final.

Para armar la maqueta del semáforo, se deben imprimir las piezas en la impresora 3D. El modelo 3D del semáforo se puede descargar de forma libre y gratuita en el siguiente enlace: www.enfoco.net.ar/sd

Una vez descargado el modelo, lo imprimimos con la impresora 3D. Cuando estén listas todas las piezas, las ensamblamos para construir el semáforo.

En caso de querer realizar una modificación en el modelo, independientemente del programa de modelado que utilicemos, debemos exportar nuestras piezas en formato .stl.

El .stl es un formato de archivo informático de diseño asistido por computadora (CAD) que define la geometría de objetos sólidos 3D. Es el formato más popular a la hora de intercambiar digitalmente modelos de objetos para ser impresos en 3D..

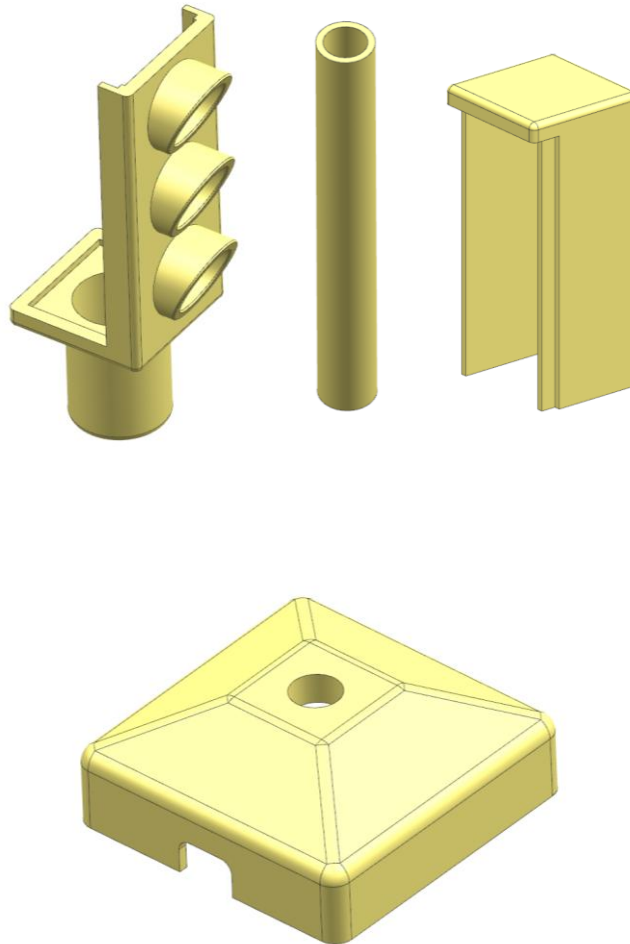
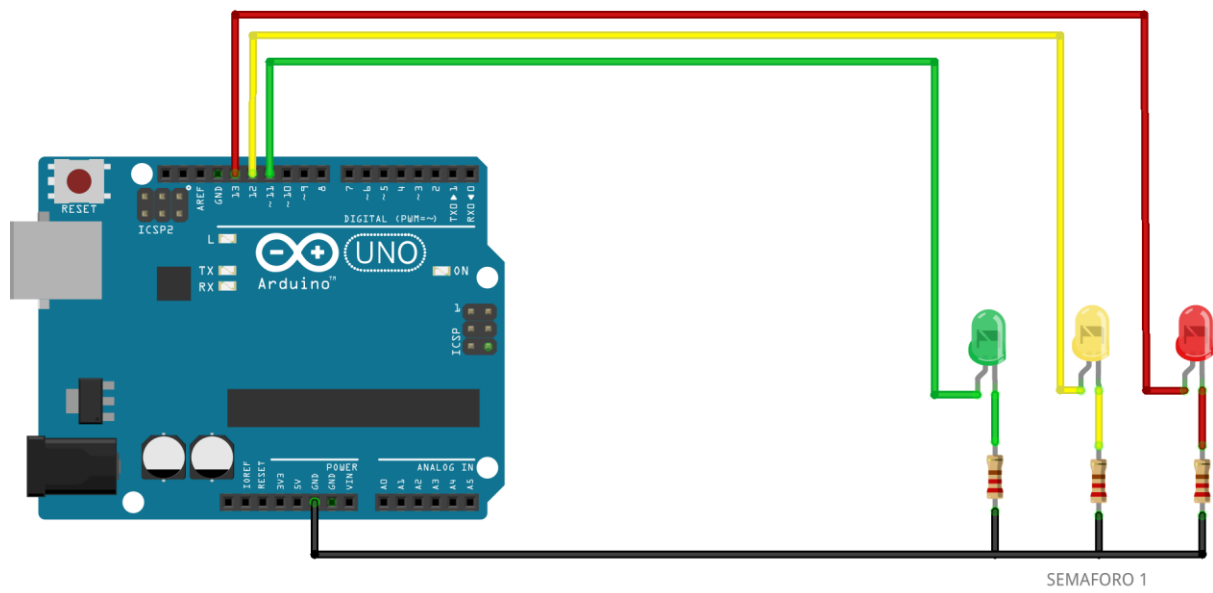


Fig. 8

Una vez que las conexiones han sido probadas y el sistema de luces funciona correctamente, podemos pasar a armar los circuitos de la maqueta prescindiendo de la protoboard. Para hacerlo, soldamos cada LED con su cable y su resistencia. Luego, lo colocamos dentro de la pieza correspondiente del semáforo.



Esquema 3

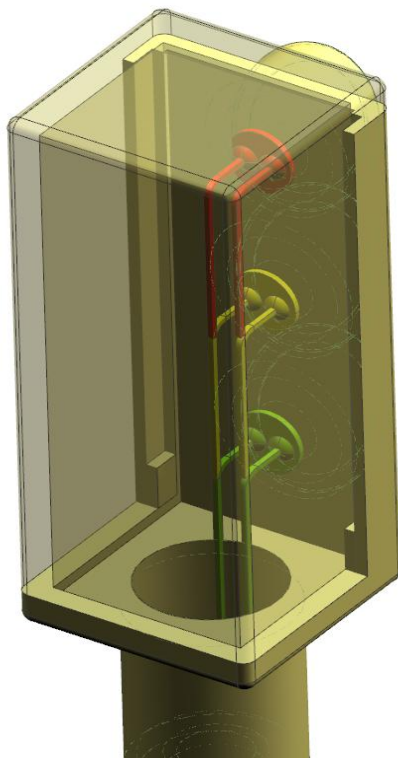


Fig. 8

Nivel Intermedio

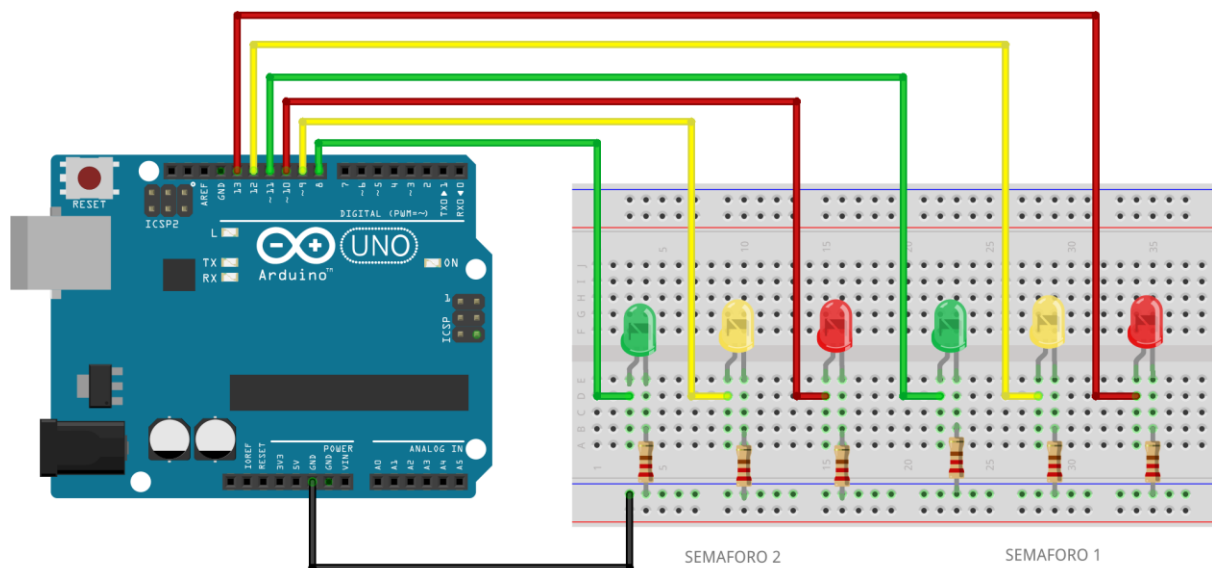
Para que el semáforo pueda cumplir su función, es necesario agregar un segundo semáforo que lo acompañe y regule el tráfico que circula por la otra calle de la intersección. Antes de instalar el segundo semáforo, debemos tener en cuenta que ambos semáforos deberán estar sincronizados entre sí para funcionar de forma adecuada y conjunta.

En este segundo momento se agregan la conexión y la programación de un segundo semáforo desde la misma placa Arduino.

Paso 1 - Armar el circuito

Para armar la intersección con dos semáforos debemos conectar a la protoboard tres LED más que en el circuito anterior (rojo, amarillo y verde), con sus respectivas resistencias.

Conectaremos el LED rojo al *pin 10*, el amarillo al *pin 9* y el verde al *pin 8*



Esquema 3

Paso 2 - Programar la secuencia para dos semáforos.

En este momento, vamos a sumar la secuencia de nuestro segundo semáforo a nuestro código de Arduino previamente armado. La planificación de la secuencia para dos semáforos es más compleja, por lo que requiere que se le dedique un poco más de tiempo a su armado y comprensión.

Para pensar cómo armar el código, es posible imaginar esta secuencia como viñetas de un cómic en donde hay una serie de acciones a realizar y una demora entre cuadro y cuadro. También puede realizarse una observación con los alumnos de algún semáforo cercano a la escuela.

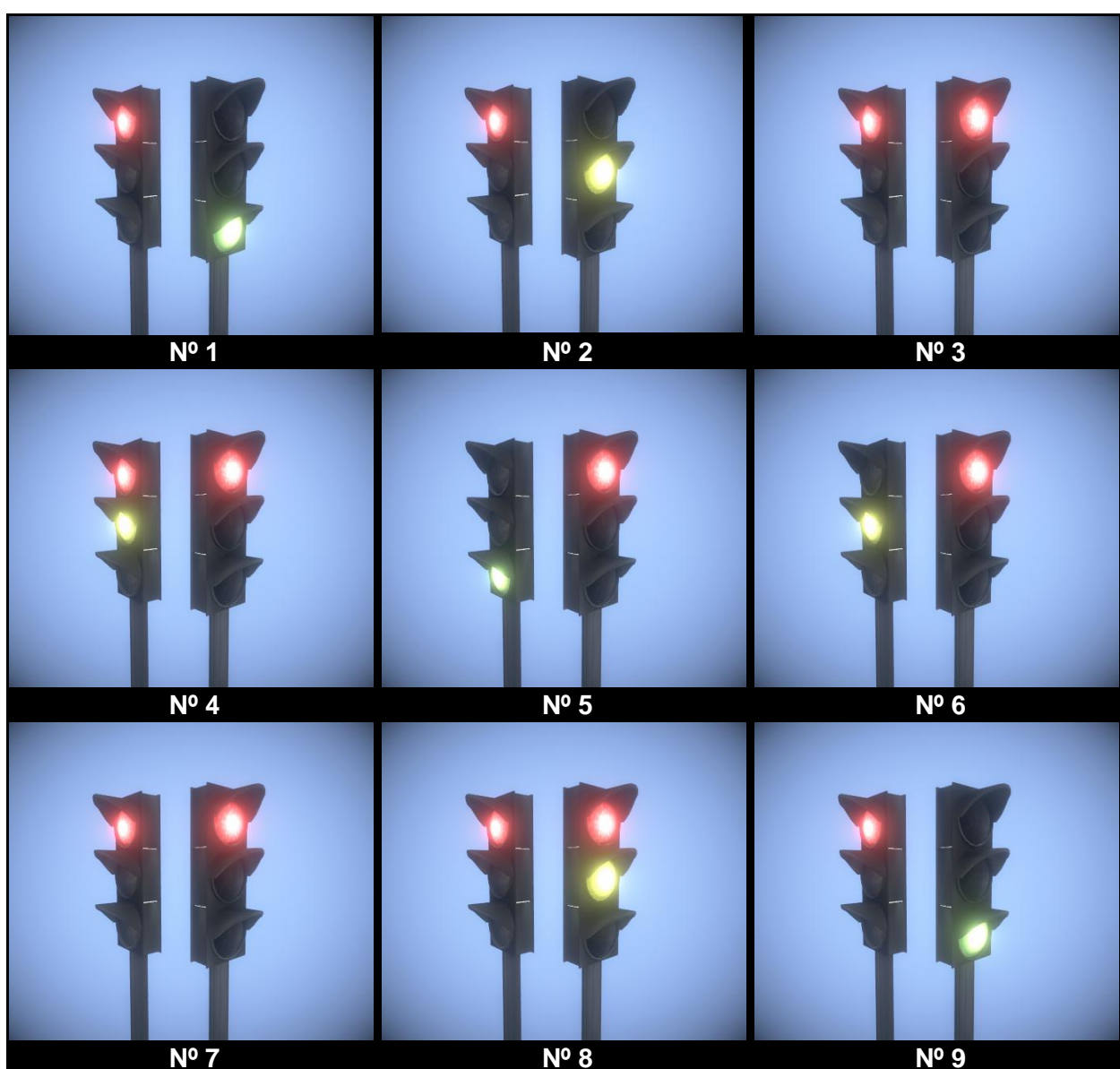


Fig. 10

Como se puede ver en la Fig. 10, el cuadro N° 9 es igual al cuadro N° 1. Es en ese punto donde comienza a repetirse la secuencia. Por lo tanto, en total tenemos ocho pasos que se repiten y una determinada demora en cada paso.

Cuando programamos esta secuencia en bloques deberíamos obtener algo similar al siguiente modelo:



Fig. 11

El código escrito generado por mBlock en la Fig. 11, debería ser similar al siguiente:

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

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;

void setup(){
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}

void loop(){
  digitalWrite(8,0);
  digitalWrite(9,0);
  digitalWrite(10,1);
  digitalWrite(11,1);
  digitalWrite(12,0);
  digitalWrite(13,0);
  _delay(5);
  digitalWrite(11,0);
  digitalWrite(12,1);
  _delay(1);
  digitalWrite(12,0);
  digitalWrite(13,1);
  _delay(1);
  digitalWrite(9,1);
  _delay(1);
  digitalWrite(9,0);
  digitalWrite(10,0);
  digitalWrite(8,1);
  _delay(5);
  digitalWrite(8,0);
  digitalWrite(9,1);
  _delay(1);
  digitalWrite(9,0);
  digitalWrite(10,1);
```

```

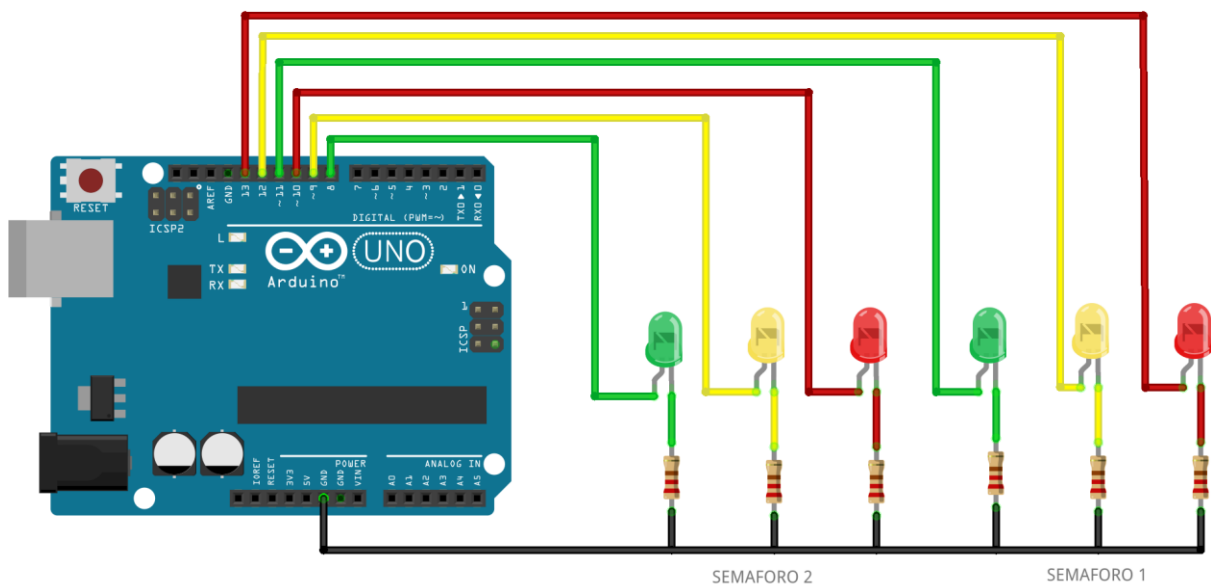
    _delay(1);
    digitalWrite(12,1);
    _delay(1);
    _loop();
}

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

void _loop(){
}

```

Fragmento de código 3



Esquema 4

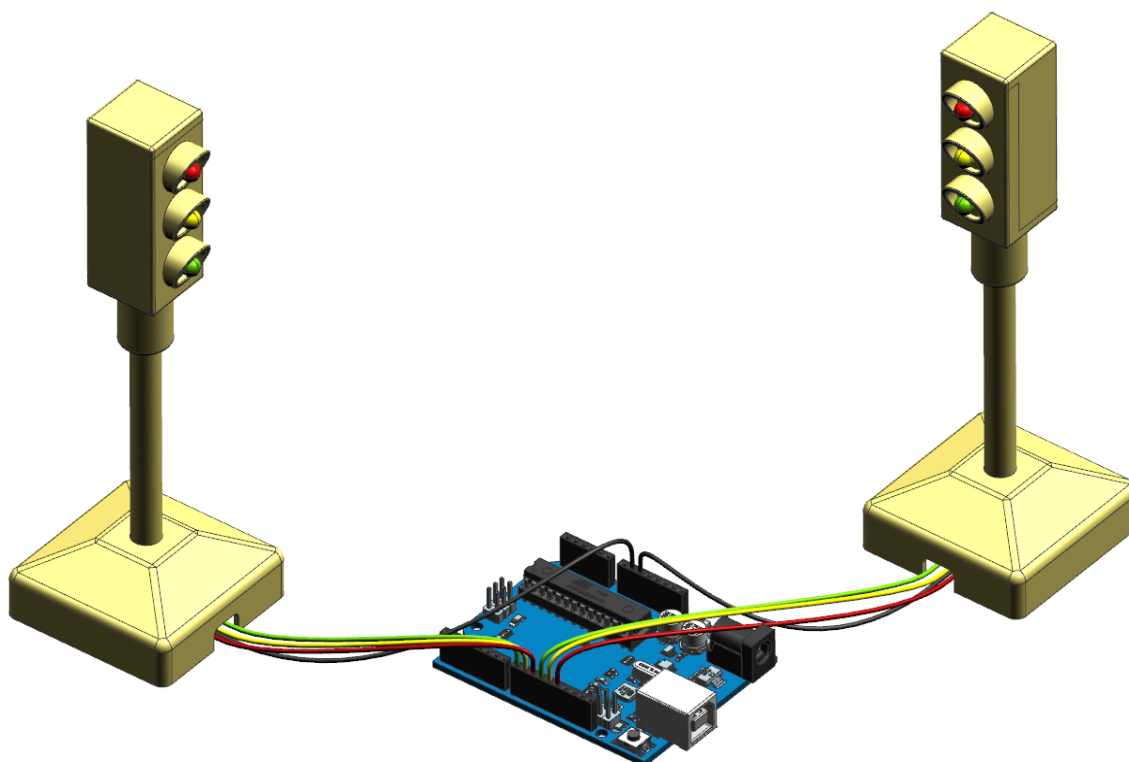


Fig.12

Nivel Superior

Se desea monitorear, desde una central, el estado y el funcionamiento de todos los semáforos que se han instalado. Para ello se decidió instalar un sistema de IoT, que permite enviar a la central vía internet esta información, que es obtenida a través de los sistemas que ya se instalaron para cada semáforo.

En esta actividad se programará el envío de datos obtenidos a un dispositivo móvil a través de Internet de las Cosas (IoT).

Paso 1 - Introducción a Internet de las Cosas (IoT)

Internet de las Cosas (en inglés *Internet of Things*, abreviado IoT) es un concepto que refiere a la interconexión digital de objetos cotidianos con internet. Esta interconexión puede tener diversas funciones. Por ejemplo, puede utilizarse para monitorear la temperatura de un ambiente, enviando los datos obtenidos por un sensor a una central donde se recopile la información. De esta manera podría visualizarse en un dispositivo móvil la temperatura de un laboratorio, de un invernadero o de una sala de un hospital.

Para poder incorporar IoT a nuestro proyecto es necesario:

1. Un dispositivo capaz de conectarse a internet.
2. Un servidor que reciba y aloje los datos.

Existen diversas formas de lograr el cometido de registrar y almacenar los datos del sistema de tanques construido. En este caso, se detallará cómo hacerlo con un módulo OBLOQ de DFRobot, y con los servidores de Adafruit.

El módulo UART OBLOQ es un dispositivo WiFi a serie pensado para desarrolladores no profesionales. Permite enviar y recibir datos mediante los protocolos HTTP y MQTT.

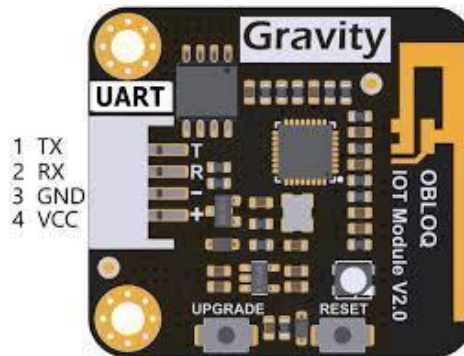


Fig. 13

Paso 2 - Crear un Panel de Control

En primer lugar, se explicará cómo crear un Panel de Control en Adafruit. Luego, se verá cómo vincular los controles del Panel con los datos que se intercambian con el dispositivo. Primero debemos crear una cuenta de usuario en io.adafruit.com.

Una vez que ingresamos con nuestro usuario, creamos un nuevo panel haciendo click en el botón “Actions” y seleccionamos “Create a New Dashboard”.

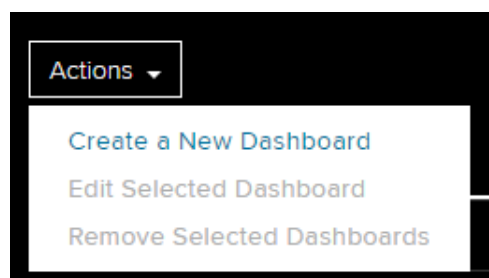


Fig. 14

Luego designamos un nombre y una descripción. Y al finalizar presionamos el botón "Create".

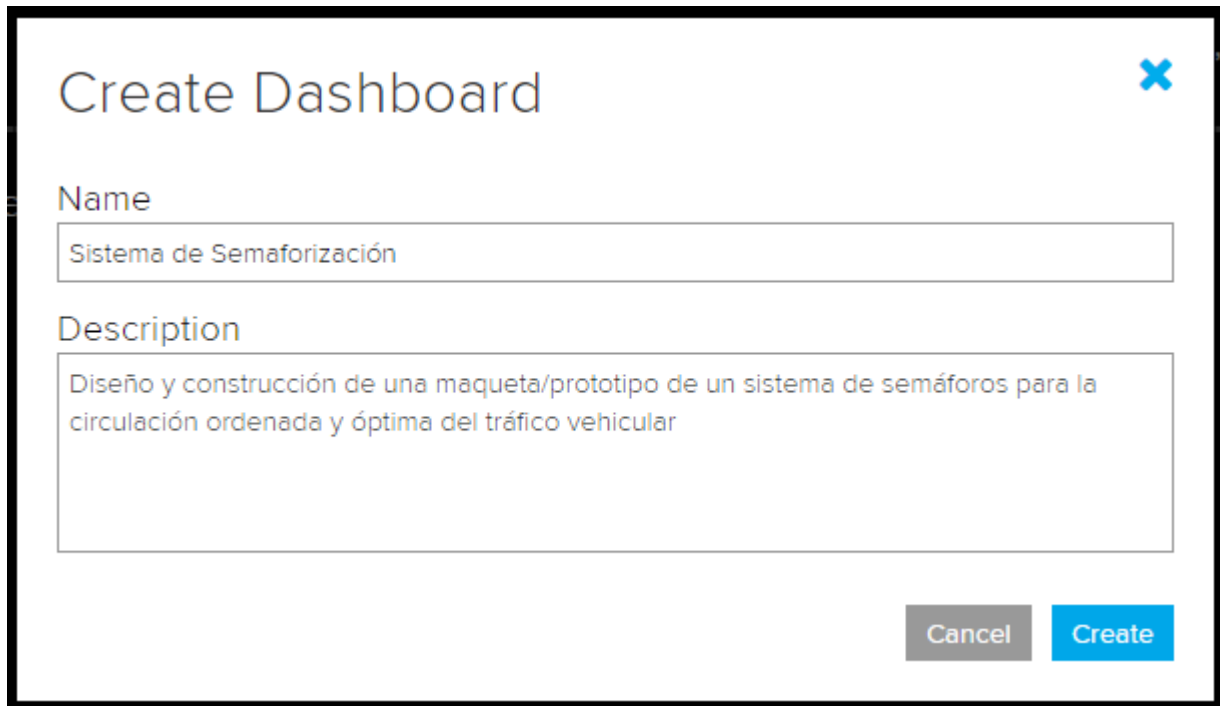
A screenshot of a 'Create Dashboard' form. The form has a title 'Create Dashboard' at the top left and a blue 'X' close button at the top right. Below the title, there is a 'Name' label followed by a text input field containing 'Sistema de Semaforización'. Below that is a 'Description' label followed by a larger text area containing 'Diseño y construcción de una maqueta/prototipo de un sistema de semáforos para la circulación ordenada y óptima del tráfico vehicular'. At the bottom right of the form are two buttons: a grey 'Cancel' button and a blue 'Create' button.

Fig. 15

Seguidamente hacemos click en el nuevo panel creado y veremos una pantalla vacía.


Podemos comenzar a agregar bloques haciendo click en .



Fig. 16

Veremos una serie de controles posibles como en la siguiente imagen:

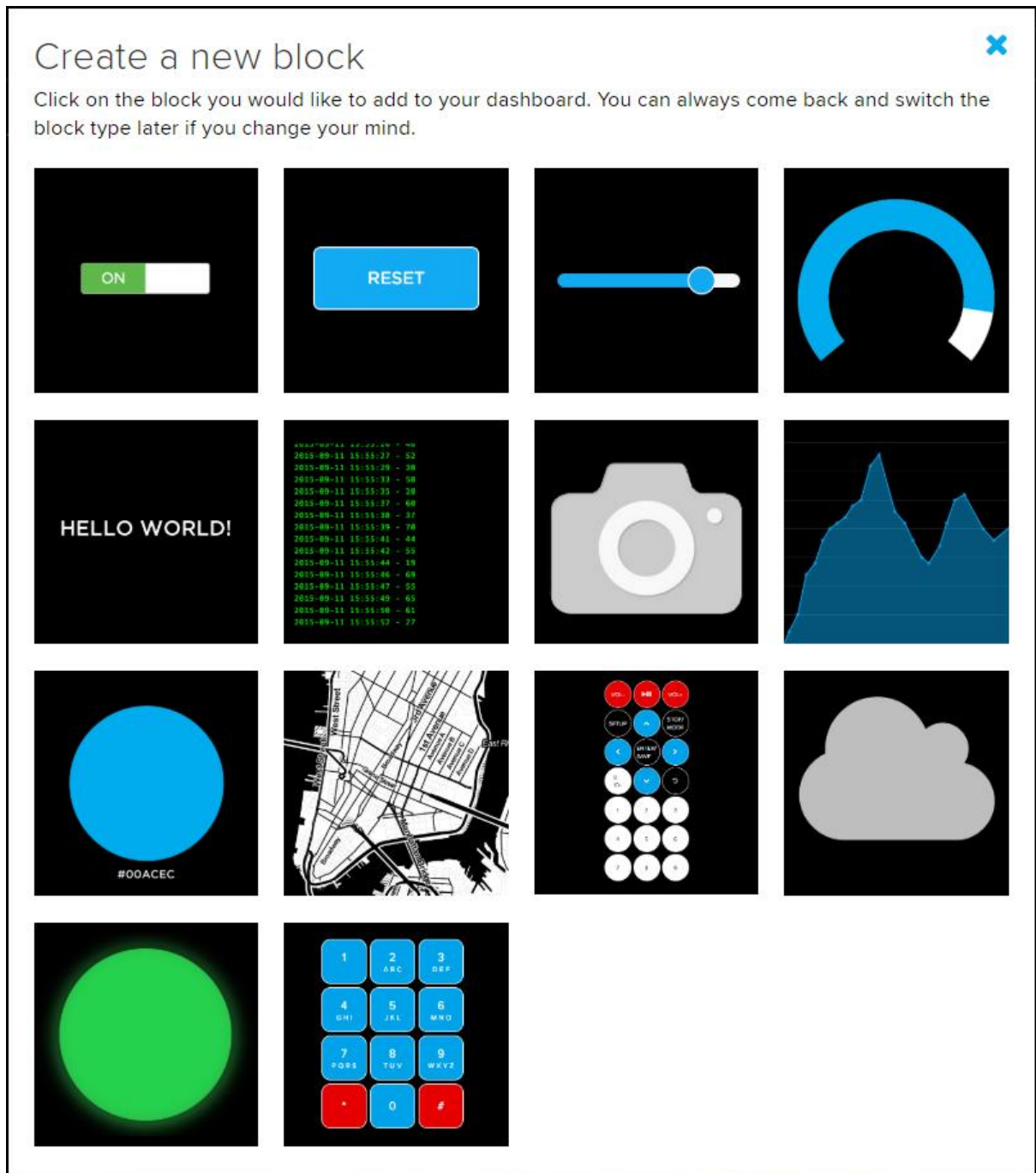


Fig. 17

Para nuestro semáforo, podríamos ubicar tres *Indicator* (indicadores) con diferentes colores, uno abajo del otro.



Fig. 18

Cuando agregamos un control al panel debemos asociarlo a un “feed”.

Un **feed** es una fuente de datos en la que uno puede publicar así como también se puede suscribir para recibir los datos de cierto feed.

Llamamos al primer *feed* “semaforo1” y le damos al botón “Create”. En él publicaremos, desde nuestro dispositivo, la información sobre el estado del primer semáforo.

Choose feed ✕

Indicator: A simple on/off indicator lamp. Feed values are compared using the given conditions. If the conditions are true, then "On Color" is used, if false, "Off Color". All values are assumed to be numeric for comparison. If the current feed value can't be converted to a number, it will be treated as a string.

If you have lot of feeds, you may want to use the search field. You can also create a feed quickly below.

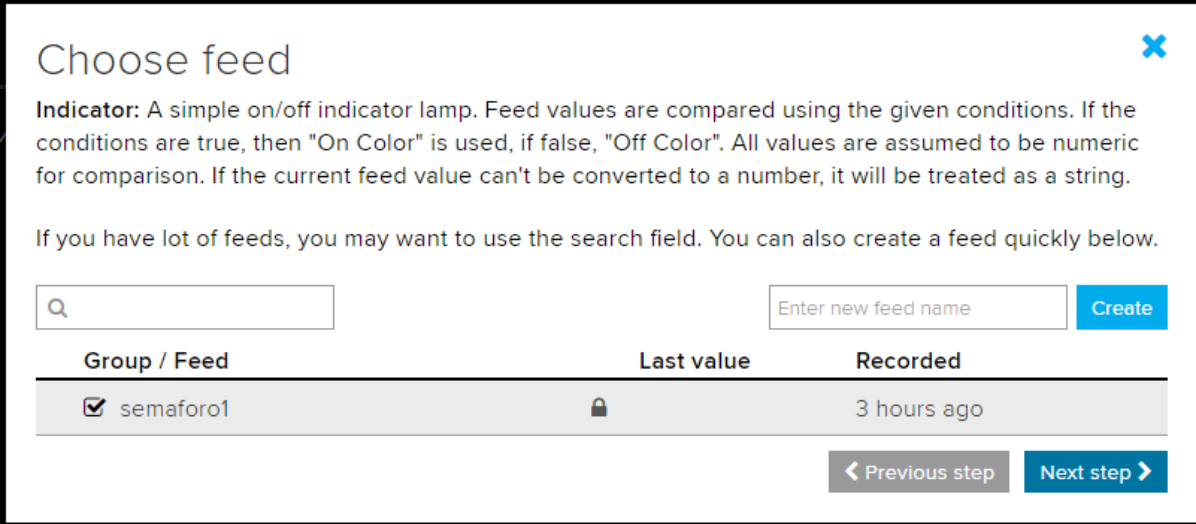
Create

Group / Feed	Last value	Recorded
<input type="checkbox"/> Welcome Feed		2 days

< Previous step Next step >

Fig. 19

Luego de crearlo, seleccionamos el feed creado y hacemos click en “*Next step*” (paso siguiente) para configurar nuestro control.



Choose feed ✕

Indicator: A simple on/off indicator lamp. Feed values are compared using the given conditions. If the conditions are true, then "On Color" is used, if false, "Off Color". All values are assumed to be numeric for comparison. If the current feed value can't be converted to a number, it will be treated as a string.

If you have lot of feeds, you may want to use the search field. You can also create a feed quickly below.

Create

Group / Feed	Last value	Recorded
<input checked="" type="checkbox"/> semaforo1		3 hours ago

◀ Previous stepNext step ▶

Fig. 20

Este *feed* tendrá el estado del semáforo expresado con la siguiente convención:

- Cuando el valor sea 1 estará en verde.
- Cuando el valor sea 2 estará en amarillo.
- Cuando el valor sea 3 estará en rojo.

Podemos incluir un título, un color que indique el estado activado y otro color que indique el estado desactivado. Este indicador estará en estado activado cuando se cumpla la condición determinada.

En este caso, queremos que el indicador esté de color rojo (activado) cuando el *feed* “semaforo1” sea igual a 3. Cuando sea distinto de 3 podemos elegir un color gris para indicar que está desactivado.

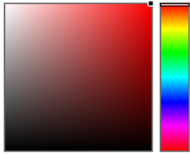
Para configurarlo, completamos los campos como se ve en la imagen a continuación.

Block settings

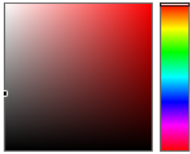
In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

On Color



Off Color



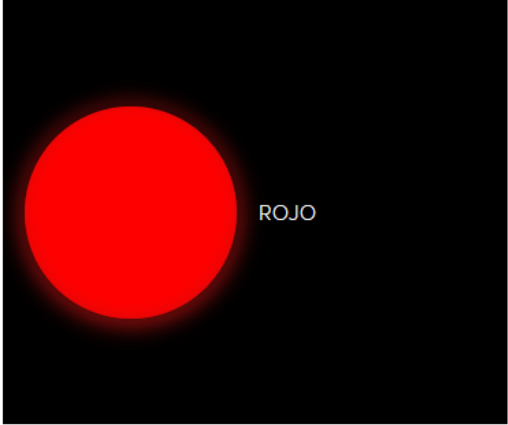
Conditions

=

▼

[Add Condition](#)

Block Preview



Indicator A simple on/off indicator lamp. Feed values are compared using the given conditions. If the conditions are true, then "On Color" is used, if false, "Off Color". All values are assumed to be numeric for comparison. If the current feed value can't be converted to a number, it will be treated as a string.

Test Value

[Previous step](#)

Create block

Fig. 21

Al finalizar la configuración, hacemos click en "Create block" (crear bloque) para completar la operación.

Si se quiere, podemos modificar el tamaño y la ubicación de los bloques haciendo click en la “rueda de configuración”.

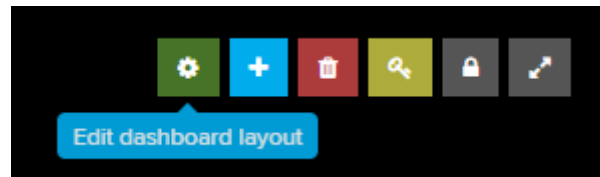


Fig. 22

Repetimos este procedimiento para los demás colores y para el *feed* “semaforo2” que indicará el estado del segundo semáforo. Al finalizar la configuración de los colores para los dos semáforos, deberíamos visualizar algo similar a lo siguiente:

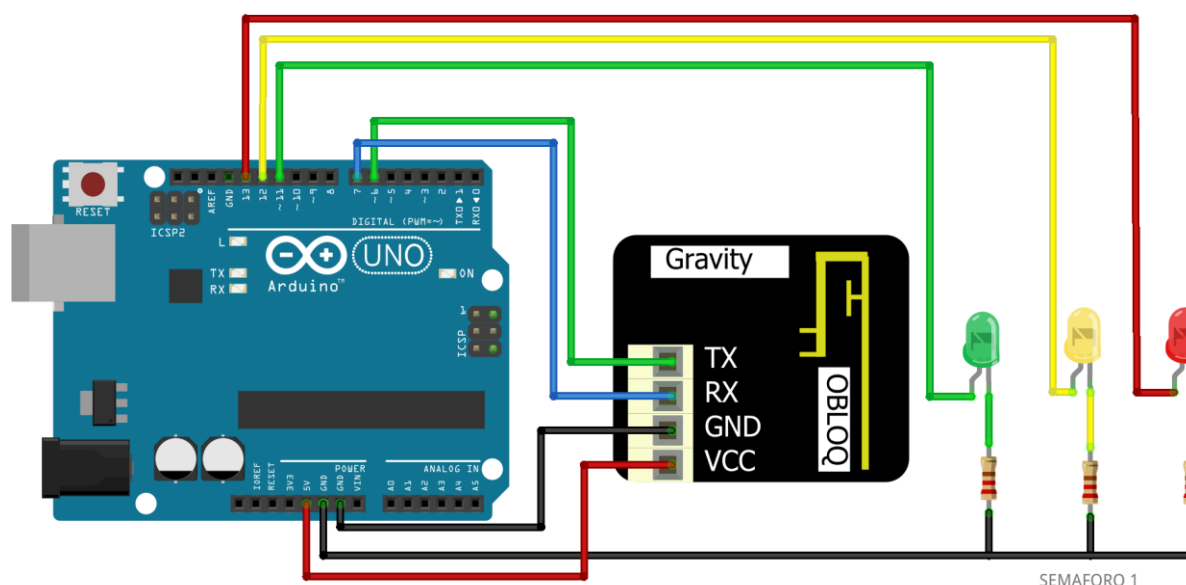


Fig. 23

Una vez realizado el Panel, publicaremos los estados de los semáforos para poder monitorearlos de manera remota. Minimice la pantalla, luego sacaremos información de ella.

Paso 3 - Conectar módulo OBLOQ

A continuación se muestra el diagrama de conexión de Arduino UNO y OBLOQ. Para simplificar el proceso y la explicación, se implementará IoT para un sólo semáforo inicialmente. Este proceso puede extenderse luego para abarcar los dos semáforos.



Esquema 4

Paso 4 - Arduino IDE

La programación por bloques tiene sus ventajas desde un punto de vista didáctico pero cuando el programa crece en complejidad puede resultar poco práctico. A menudo podemos encontrarnos con el hecho de que ciertas operaciones no pueden resolverse utilizando bloques o que hacerlo con este método resulta más engorroso y difícil de interpretar que si se utilizara el código escrito.

Hasta ahora hemos visto cómo al realizar nuestra programación en bloques se generaba simultáneamente un código escrito en el área lateral derecha. Para esta sección de la actividad se propone trabajar directamente sobre el código, para ello vamos a recurrir a el entorno nativo de Arduino que llamamos “Arduino IDE” (IDE proviene la sigla entorno de desarrollo integrado).

Para ello descarga el Arduino IDE desde el siguiente enlace y luego procede con la instalación del mismo: www.enfoco.net.ar/sd

Veremos que se nos presenta la siguiente interfaz:

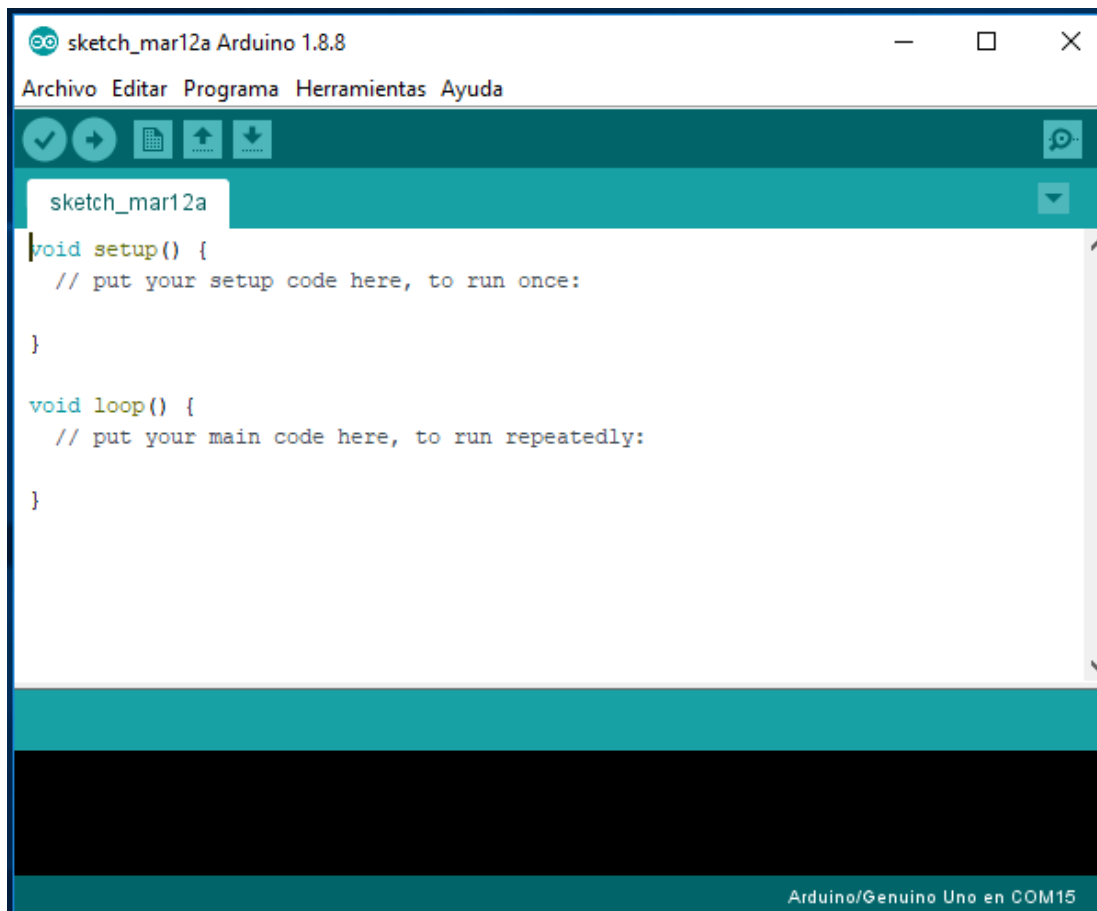


Fig. 24

A continuación, se presenta una estructura mínima de un *sketch* (un programa) de Arduino:

```
void setup() {  
  // Código de inicialización. Se ejecuta una sola vez.  
}  
  
void loop() {  
  // Código principal. Se ejecuta repetidamente.  
}
```

En líneas generales, un programa de Arduino es:

1. Un bloque de código que se ejecuta por única vez al inicializarse el dispositivo. Este bloque de código está contenido dentro de la función “setup” (se coloca dentro de `void setup() { y }`).
2. Un bloque de código que se ejecuta repetidamente luego de la función “setup”. Este bloque de código está contenido dentro de la función “loop” (se coloca dentro de `void loop() { y }`).

Después de `//` se incluyen comentarios para el lector que no tienen ningún efecto en el programa. Estos comentarios sirven para clarificar el código y que sea más fácil de interpretar para otras personas.

Los pasos para subir el código a través del Arduino IDE son similares a los que hemos visto para mBlock3:

1. Conectar la placa a la entrada USB.
2. Chequear que estén seleccionados la placa “Arduino/Genuino Uno” y el puerto serie al que está conectada la placa.

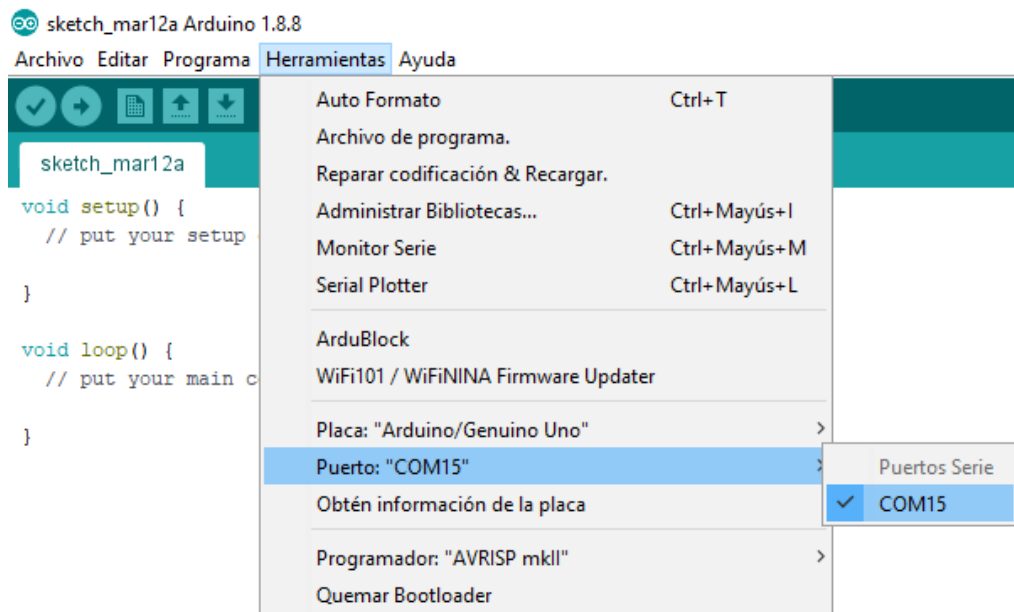


Fig. 25

3. Clickear el botón de “Subir”  para comprobar los paso anteriores.

Sabremos que nuestro código subió correctamente si en la barra de estado se escribe “Subido”.

Paso 5 - Programar sin código bloqueante

Antes de comenzar a utilizar IoT debemos hacer una aclaración con respecto a la función `delay()` que figura en el código que usamos hasta ahora. Esta función brinda un tiempo de espera al sistema que puede utilizarse con varios fines. Suele utilizarse bastante en las primeras aproximaciones a la programación, ya que su comportamiento resulta fácil de comprender y su programación no requiere más que una línea de código. Sin embargo, esta función tiene una complicación, dado que genera un “código bloqueante”. Esto significa que, cuando el programa entra en esa función, se detiene todo el procesamiento hasta que se cumpla el tiempo indicado. En otras palabras,

cuando el programa entra al *delay* queda “colgado” por el período de tiempo establecido.

Al utilizar IoT, es conflictivo utilizar código “bloqueante”, ya que al detenerse el procesamiento se impide también que el sistema realice otras operaciones que funcionan en simultáneo. Por ejemplo, las tareas de publicación y el mantenimiento constante de la conexión a internet.

Para evitar estos problemas, se puede utilizar una alternativa de código “no bloqueante”, como la función `millis()`. Esta función arroja un valor sobre un conteo de tiempo, que se realiza desde el momento en que se inicia el sistema. Es decir, funciona como un cronómetro (en milisegundos) que, cada vez que es consultada desde el código, “devuelve” el valor en el que se encuentra. De esta manera podemos pedirle al sistema que informe cuánto tiempo transcurrió desde el inicio de las operaciones para dar indicaciones temporales sobre una tarea, sin detener todas las demás.

A continuación se presenta un ejemplo de cómo se puede programar la intermitencia de un LED que se prenda y apague cada un segundo (expresado en 1000 milisegundos) sin utilizar código bloqueante. Lo haremos utilizando la función *millis* para consultar cuánto tiempo pasó.

```
int estado = LOW;
// Se declara "millisAnterior" con valor inicial igual a cero.
long millisAnterior = 0;

void setup() {
  // Inicializa el pin digital 13 como una salida.
  pinMode(13, OUTPUT);
}

void loop() {

  long millisActual = millis();

  if (millisActual - millisAnterior >= 1000) {
    // Conmuta el estado del LED.
    if (estado == LOW) {
      estado = HIGH;
    } else {
      estado = LOW;
    }
  }

  // Setea el estado del LED.
  digitalWrite(13, estado);
}
```



```

    // Guarda la última vez que conmutamos el LED.
    millisAnterior = millisActual;
}

// Y en este punto nuestro procesador queda libre
// para realizar otras tareas.

}

```

Fragmento de código 4

En el ejemplo anterior, se puede observar que para tomar el valor de *millis* se define un valor inicial, al que llamamos “millisAnterior”, que es igual a cero. Luego, para consultar el tiempo transcurrido desde el inicio del sistema, se calcula la diferencia entre el valor de “millisActual” y el de “millisAnterior”. En el caso de nuestro ejemplo, como queremos generar una intermitencia de 1 segundo, necesitamos evaluar si esta diferencia es mayor o igual a 1000. En caso de que haya transcurrido más de un segundo, el sistema modificará el estado de la luz. Si ha transcurrido menos tiempo, el estado se mantendrá estable.

En última instancia se establece que, si esta diferencia es mayor o igual a 1000, se le asigne a “millisAnterior” el valor de “millisActual”. De esta manera, la diferencia entre “millisActual” y “millisAnterior” vuelve a ser cero hasta que vuelva a transcurrir otro segundo.

En el ejemplo de la Fig. 7 había dos estados: LOW (bajo) y HIGH (alto). En el caso del semáforo podemos considerar los siguientes:

1. Poner en verde
2. Esperando en verde
3. Poner en amarillo
4. Esperando en Amarillo
5. Poner en rojo
6. Esperando en rojo
7. Poner en rojo y amarillo
8. Esperando en rojo y amarillo

Podemos visualizar en este ejemplo, como queda el fragmento de código 3 quitando el código bloqueante.

```

int estado = 1; //Poner en Verde
long millisAnterior = 0;

void setup() {
    // Seteamos los pines 11,12 y 13 como salidas.
    pinMode(11,OUTPUT);
}

```

```

pinMode(12,OUTPUT);
pinMode(13,OUTPUT);
// Apagamos los tres LED.
digitalWrite(11,LOW);
digitalWrite(12,LOW);
digitalWrite(13,LOW);
}

void loop() {

    long millisActual = millis();

    //Poner en Verde
    if (estado==1) {
        digitalWrite(11,HIGH);
        digitalWrite(12,LOW);
        digitalWrite(13,LOW);
        millisAnterior = millisActual; // Guarda millisAnterior.
        estado = 2; // Cambia al siguiente estado.
    }

    //Esperando en verde.
    if (estado==2 && millisActual - millisAnterior >= 5000) {
        estado = 3; // Cambia al siguiente estado.
    }

    //Poner en amarillo.
    if (estado==3) {
        digitalWrite(11,LOW);
        digitalWrite(12,HIGH);
        digitalWrite(13,LOW);
        millisAnterior = millisActual; // Guarda millisAnterior.
        estado = 4; // Cambia al siguiente estado.
    }

    //Esperando en Amarillo.
    if (estado==4 && millisActual - millisAnterior >= 1000) {
        estado = 5; // Cambia al siguiente estado.
    }

    //Poner en rojo.
    if (estado==5) {
        digitalWrite(11,LOW);
        digitalWrite(12,LOW);
        digitalWrite(13,HIGH);
        millisAnterior = millisActual; // Guarda millisAnterior.
    }
}

```

```

    estado = 6;                                // Cambia al siguiente estado.
}

//Esperando en rojo.
if (estado==6 && millisActual - millisAnterior >= 5000) {
    estado = 7; // Cambia al siguiente estado.
}

//Poner en rojo y amarillo.
if (estado==7) {
    digitalWrite(11,LOW);
    digitalWrite(12,HIGH);
    digitalWrite(13,HIGH);
    millisAnterior = millisActual; // Guarda millisAnterior.
    estado = 8;                    // Cambia al siguiente estado.
}

//Esperando en rojo y amarillo.
if (estado==8 && millisActual - millisAnterior >= 1000) {
    estado = 1; // vuelve al estado inicial.
}

// En este punto nuestro procesador queda libre
// para realizar otras tareas.

}

```

Fragmento de código 5

Teniendo un programa de funcionamiento de semáforos con un código “no bloqueante” (es decir, sin usar la función *delay*) estamos en condiciones de incorporar IoT a nuestro proyecto.

Paso 6 - Programación IoT.

Utilizaremos la librería ObloqAdafruit para informar a Adafruit cada vez que cambie el estado del semáforo. Podremos monitorear este estado desde el Panel de Control que hemos creado.

En primer lugar debemos instalar la librería en el Arduino IDE. Para esto debemos ingresar al menú Programa > Incluir Librería > Gestionar Librerías.

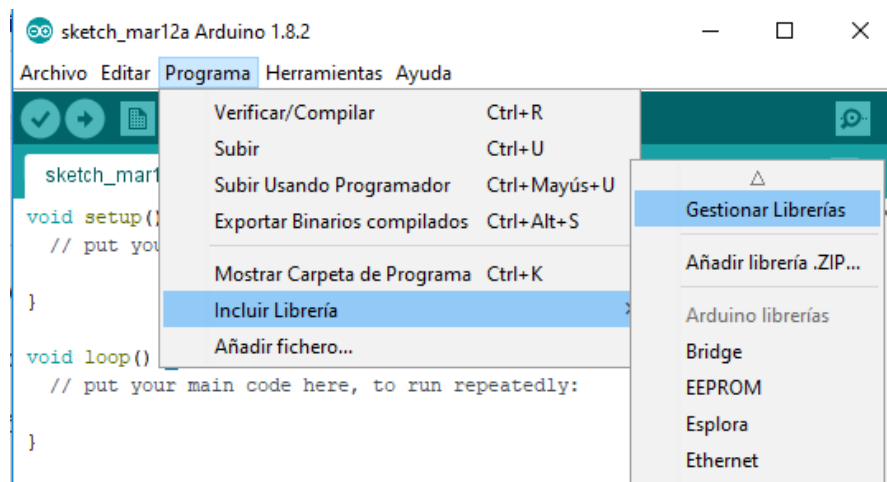


Fig. 26

Se abrirá una ventana con un buscador en margen superior. Debemos escribir Obloq, seleccionar la librería ObloqAdafruit y apretar el botón Instalar.

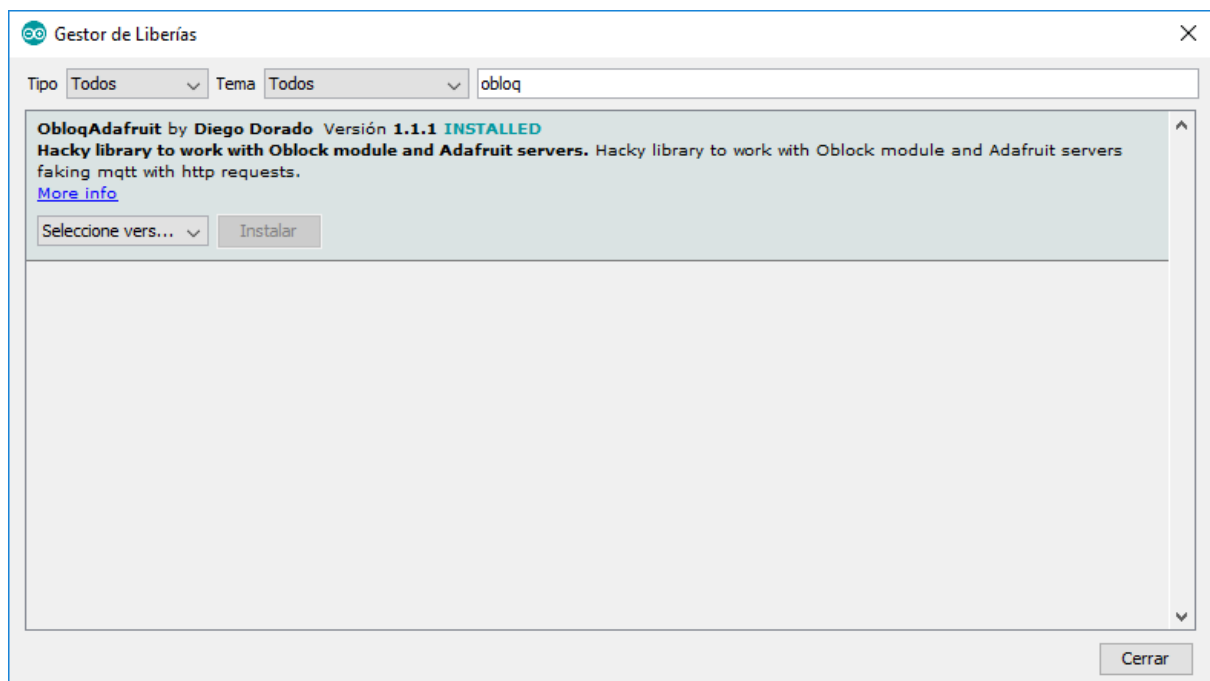


Fig. 27

En general las librerías traen códigos de ejemplo como referencia. Abrimos el ejemplo “Publicar” ubicado en Archivo > Ejemplos > ObloqAdafruit > Publicar.

Debemos reemplazar el SSID de la WiFi, su password, el IO_USERNAME e IO_KEY que son nuestras credenciales que copiaremos de Adafruit. Pare ello maximizamos la pantalla de Adafruit y hacemos click en el ícono de la “llave”.

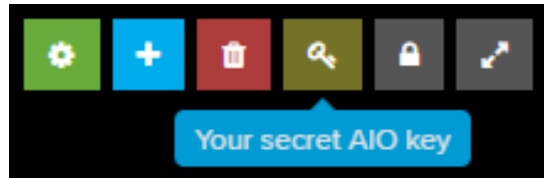


Fig. 28

Copiamos el código que nos ofrece para Arduino, con nuestro usuario y key.

YOUR AIO KEY

Your Adafruit IO key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your AIO key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new AIO key, all of your existing programs and scripts will need to be manually changed to the new key.

Username

Active Key

REGENERATE AIO KEY

[Hide Code Samples](#)

Arduino

```
#define IO_USERNAME "usuario_aio"
#define IO_KEY      "1234cfdd29a244b6b049abb07727c117"
```

Fig. 29

Estos datos aparecen en el código de la siguiente manera:

```
#define IO_USERNAME "usuario_adafruit"
#define IO_KEY      "key_adafruit"
```

Se deberán reemplazar en esas dos líneas el usuario y key por los que se hayan obtenido en Adafruit. Por ejemplo:

```
#define IO_USERNAME "usuario_aio"
#define IO_KEY      "1234cfdd29a244b6b049abb07727c117"
```

También modificaremos `softSerial(10,11)` por `softSerial(6,7)` ya que así es como lo conectamos en nuestra placa, quedándonos el siguiente código:

```

#include "SoftwareSerial.h"
#include "ObloqAdafruit.h"

// Indicamos conexión de wifi.
#define WIFI_SSID      "SSID_de_Wifi"
#define WIFI_PASSWORD  "PWD_de_WIFI"

// Copiamos las credenciales obtenidas anteriormente en Adafruit.
#define IO_USERNAME    "usuario_adafruit"
#define IO_KEY          "key_adafruit"

SoftwareSerial softSerial(6,7);
ObloqAdafruit olq(&softSerial,WIFI_SSID,WIFI_PASSWORD,IO_USERNAME,IO_KEY);

```

El *setup* debe incluir la línea de inicialización del softwareSerial:

```

void setup()
{
    softSerial.begin(9600);
}

```

Se debe agregar también la función de actualización o “update”: `olq.update()`. Por esto es importante que nuestro código no sea bloqueante.

```

void loop()
{
    olq.update();
    // ..
    // ..
}

```

Para publicar un *feed*, utilizaremos la función `publish` del objeto `olq` :

```

olq.publish("semaforo1", 2); // informar que el estado "amarillo"

```

Finalmente, nuestro programa de semáforo con IoT queda como sigue:

```

#include "SoftwareSerial.h"
#include "ObloqAdafruit.h"

// Indicamos conexión de wifi.
#define WIFI_SSID      "SSID_de_Wifi"
#define WIFI_PASSWORD  "PWD_de_WIFI"

// Copiamos las credenciales obtenidas anteriormente en Adafruit.

```

```

#define IO_USERNAME    "usuario_adafruit"
#define IO_KEY         "key_adafruit"

SoftwareSerial softSerial(6,7);
ObloqAdafruit olq(&softSerial,WIFI_SSID,WIFI_PASSWORD,IO_USERNAME,IO_KEY);

int estado = 1; //Poner en verde.
long millisAnterior = 0;

void setup() {
    // Seteamos los pines 11,12 y 13 como salidas.
    pinMode(11,OUTPUT);
    pinMode(12,OUTPUT);
    pinMode(13,OUTPUT);
    // Apagamos los tres LED.
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);

    // Inicializamos la comunicación con el módulo OBloq.
    softSerial.begin(9600);
}

void loop() {

    long millisActual = millis();
    //Poner en verde.
    if (estado==1) {
        digitalWrite(11,HIGH);
        digitalWrite(12,LOW);
        digitalWrite(13,LOW);
        millisAnterior = millisActual; // Guarda millisAnterior.
        estado = 2; // Cambia al siguiente estado.
        olq.publish("semaforo1", 1); // Publicar en adafruit.
    }

    //Esperando en verde.
    if (estado==2 && millisActual - millisAnterior >= 5000) {
        estado = 3; // Cambia al siguiente estado.
    }

    //Poner en amarillo.
    if (estado==3) {
        digitalWrite(11,LOW);
        digitalWrite(12,HIGH);
        digitalWrite(13,LOW);
    }
}

```

```

    millisAnterior = millisActual; // Guarda millisAnterior.
    estado = 4; // Cambia al siguiente estado.
    olq.publish("semaforo1", 2); // Publicar en Adafruit.
}

//Esperando en amarillo.
if (estado==4 && millisActual - millisAnterior >= 1000) {
    estado = 5; // Cambia al siguiente estado.
}

//Poner en rojo.
if (estado==5) {
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,HIGH);
    millisAnterior = millisActual; // Guarda millisAnterior.
    estado = 6; // Cambia al siguiente estado.
    olq.publish("semaforo1", 3); // Publicar en adafruit.
}

//Esperando en rojo.
if (estado==6 && millisActual - millisAnterior >= 5000) {
    estado = 7; // Cambia al siguiente estado.
}

//Poner en rojo y amarillo.
if (estado==7) {
    digitalWrite(11,LOW);
    digitalWrite(12,HIGH);
    digitalWrite(13,HIGH);
    millisAnterior = millisActual; // Guarda millisAnterior.
    estado = 8; // Cambia al siguiente estado.
    olq.publish("semaforo1", 4); // Publicar en Adafruit.
}

//Esperando en rojo y amarillo.
if (estado==8 && millisActual - millisAnterior >= 1000) {
    estado = 1; // Vuelve al estado inicial.
}

// Llamamos a que la librería actualice lo que necesite.
olq.update();
}

```

Fragmento de código 6

Cierre

Una vez finalizado este proyecto, si se quiere continuar, es posible extenderlo. Estas son algunas opciones sugeridas:

- Agregar un semáforo para peatones a la intersección.
- Agregar un sonido que acompañe cada luz para que sirva de referencia para las personas con discapacidad visual.
- Agregar un pulsador para solicitar el cruce de peatones. Esto se puede hacer sobre el circuito de Arduino o desde la programación de IoT, que puede manejarse con un celular.

El proceso de resolución de problemas como los que se han planteado aquí permite la movilización y la integración de distintos saberes, en la búsqueda de soluciones posibles a una situación dada. Si bien la información aquí fue presentada a modo de instructivo, se espera que sean los estudiantes organizados en pequeños grupos quienes vayan encontrando las mejores formas para construir los dispositivos. Esto implica preparar los materiales para que cada grupo cuente con todo lo necesario para la construcción del proyecto. Además, en el interior de cada grupo, los estudiantes deben distribuirse los roles y las tareas de acuerdo a las demandas que van teniendo en las actividades.

Es importante que los docentes acompañen las producciones de cada grupo monitoreando los avances de todos los estudiantes y presentando la información que se considere necesaria para continuar la tarea. Pero, al mismo tiempo, es necesario que habiliten espacios para que los alumnos realicen hipótesis, planteen interrogantes, indaguen, prueben y realicen ajustes de acuerdo a lo que ellos mismo van pensando sobre cómo llevar a cabo el proyecto.

En este sentido, registrar lo que se va haciendo, las preguntas de los alumnos, las pruebas, los errores y cómo se fueron construyendo los dispositivos, permite reflexionar sobre la propia práctica, reforzar los aprendizajes construidos a lo largo de este proceso y poder volver a ese material disponible para próximos proyectos que se realicen.

Una vez terminado el proyecto, se sugiere reunir y organizar con el grupo el registro que se hizo del proceso realizado. Esta instancia de sistematización también permite movilizar capacidades vinculadas a la comunicación porque implica tomar decisiones respecto a cómo se quiere mostrar el proyecto a otros (otros grupos, otras escuelas, otros docentes, a la comunidad, etc.).

Te recomendamos pasar por el repositorio de saberes Digitales para obtener más materiales y otros ejemplos: www.enfoco.net.ar/sd

Glosario

Electrónica y Arduino

Arduino: Placa electrónica que contiene un microcontrolador programable y sistema de comunicación (USB y serial) que permite al usuario cargarle diversos programas así como también comunicarse con la misma. Del lado de la computadora se utiliza un IDE (entorno de desarrollo integrado) para generar el código, compilarlo y quemarlo en la placa. Existen múltiples IDE compatibles con las placas Arduino.

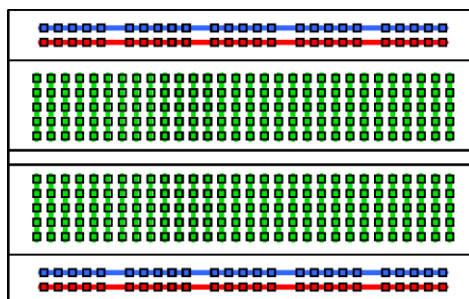
El microcontrolador posee entradas analógicas y digitales así como salidas digitales, PWM y servo. Las entradas y salidas digitales son las que permiten leer o escribir estados del tipo binarios. Pueden adoptar la forma de 0 ó 1, alto o bajo, verdadero o falso. Para prender y apagar los LED del semáforo utilizamos salidas digitales, las mismas están nombradas con números desde el 0 al 13.

Las entradas analógicas permiten leer información que puede adoptar diferentes niveles de tensión, tal como la lectura de un termómetro analógico, la posición de un potenciómetro, etc. Las mismas están identificadas en la placa como A0 a A5.

Puerto COM: Es el puerto de comunicaciones a través del cual un sistema operativo informático se comunica con un dispositivo externo tal como una placa Arduino. La asignación de los mismos suele realizarse de forma automática al conectar la placa vía USB. Dicha asignación suele ser dinámica, lo que significa que a veces cambia el número al conectar una misma placa en otro puerto USB o al conectar varias placas. En todos los IDE de programación es necesario especificar el puerto COM a través del cual nos comunicaremos con la placa Arduino.

Protoboard: Es una placa experimental que permite el prototipado rápido de circuitos electrónicos. Tiene orificios para insertar las patas de los componentes permitiendo que se conecten sin tener que recurrir a la soldadura.

El mismo posee una grilla de orificios que se encuentran conectados entre sí siguiendo el esquema de la imagen. Las líneas de conexión superior e inferior recorren la placa de punta a punta y suelen utilizarse para la alimentación del circuito, mientras que las líneas verdes se suelen utilizar para interconectar componentes. Tomar en cuenta que las líneas verdes se interrumpen en el centro de la placa. Generalmente se utilizan cables del tipo dupont para realizar conexiones en la protoboard.



LED: Componente electrónico tipo diodo que emite luz. Es necesario tomar en cuenta la polaridad del mismo para ponerlo en funcionamiento. Conectándolo con la polaridad invertida generalmente no va a traer mayores consecuencias que la imposibilidad de hacer que encienda. Existen dos formas de distinguir la polaridad del mismo: podemos identificar

la pata negativa como la pata más corta u observando el lado plano en el encapsulado del mismo.



Resistencia: La resistencia eléctrica es una característica de todo material conductor eléctrico de hacer oposición al paso de la corriente eléctrica, es uno de los componentes más utilizados en la electrónica. El valor resistivo se mide en ohm y se representa con el símbolo Ω . Existen resistencias de valores que van desde menos de 1 ohm hasta varios millones. Se suelen utilizar para determinar la cantidad de corriente de una rama de circuito, por ejemplo para evitar que se queme el LED por exceso de corriente.

Impresión 3D

Formato .stl: El .stl es un formato de archivo que contiene la forma de un objeto sólido. Este formato de archivo no contiene información tal como color, texturas o propiedades físicas. Los archivos STL son objetos ya consolidados por lo que resulta útil para el intercambio e impresión de los mismos. Este formato de archivo no resulta práctico en caso de necesitar editar la geometría del objeto. Esto se debe a que no contiene informaciones paramétrica sobre la generación de las diversas formas, curvas o capas que se utilizan a la hora de diseñar. Se lo puede considerar como la bajada o exportación final de un diseño, en ciertos aspectos equivalente a pensar en exportar un documento PDF partir de un documento de texto. Es posible generar archivos STL partiendo desde distintos tipos de entornos de diseño y modelado en 3D.

Código G: Es el nombre que recibe el conjunto de acciones que va a tener que realizar la impresora 3D, o cualquier otro tipo de máquina CNC, para completar el trabajo solicitado. Estas instrucciones se generan partiendo del análisis de un archivo STL y realizando el cálculo de todos los movimientos y trayectorias que realizará cada componente de la impresora (motores, avance de filamento, calentador de extrusor, calentador de la base, etc) para materializar el objeto en cuestión. Debido a que cada marca y modelo de impresora 3D tiene diferentes características mecánicas, el código G generado para imprimir cierto objeto solo va a servir para ejecutarse en un modelo de impresora específico.

Internet de las cosas

Panel de Control Adafruit: Los sistemas IoT trabajan apoyándose en un servidor que se encarga de centralizar y gestionar la información que reportan los diversos sensores así como responder a las consultas de los dispositivos que buscan acceder a dicha información (mostrarla en pantalla, tomar decisiones, etc). Adafruit es una plataforma online con posibilidad de uso gratuito que ofrece el servicio de gestión de esta información. La misma ofrece un alto grado de compatibilidad con diversos estándares de trabajo IoT y se encuentra principalmente orientada al uso educativo.

Feed: fuente de datos en la que uno puede publicar y a la que puede suscribirse. Es decir, permite enviar datos, para que estos sean almacenados en el tiempo así como también leerlos, recibiendo las actualizaciones de quienes estén publicando allí. Es una forma de almacenar información en una gran base de datos de forma ordenada, utilizando el concepto de etiquetas tanto al momento de escribirla como el de leerla.

Reconocimientos

Este trabajo es fruto del esfuerzo creativo de un enorme equipo de entusiastas y visionarios de la pedagogía de la innovación, la formación docente, la robótica, la programación, el diseño y la impresión 3D. Les agradecemos por el trabajo en equipo inspirador para traer a la realidad la obra que, en forma conjunta, realizamos INET y EDUCAR del Ministerio de Educación, Cultura, Ciencia y Tecnología de la Nación Argentina.

Contenidos

Equipo INET

Alejandro Anchava
Joreliz Andreyana Aguilera Barragán
Omar Leandro Bobrow
Alejandro Cesar Cáceres
Ezequiel Luberto
Gustavo Roberto Mesiti
Alejandro Palestrini
Judit Schneider
Pablo Trangone

Equipo Educar:

Pablo Aristide
Mayra Botta
Anabela Cathcarth
Eduardo Chiarella
María Laura Costilla
Diego Dorado
Facundo Dyszel
Federico Frydman
Matías Rinaldi
Uriel Rubilar
Camila Stecher
Carolina Sokolowicz
Nicolás Uccello

Para la confección de esta obra se contó con el apoyo de la Universidad Pedagógica Nacional "UNIPED". En particular en el desarrollo de los capítulos 1 y 2, los cuales estuvieron a cargo de los profesores Fernando Raúl Alfredo Bordignon y Alejandro Adrián Iglesias.

Producción y comunicación

Juliana Zugasti

Diseño y edición

Leonardo Frino
Mario Marrazzo

Corrección de estilo

María Cecilia Alegre

Agradecimientos especiales

Mariano Consalvo. Equipo ABP

Damián Olive. Equipo de ABP

María José Licio Rinaldi, Directora Nacional de Asuntos Federales INET, quien siempre acompañó a este equipo en todas las gestiones para su implementación

Estamos comprometidos en instalar la innovación en la escuela secundaria técnica: la robótica, la programación, el pensamiento computacional, los proyectos tecnológicos, el ABP, la impresión 3D, de manera más accesible para todos.

Agradecemos enormemente, docente, tu continua dedicación y compromiso con el futuro de tus estudiantes.

¡Estamos ansiosos por saber qué es lo que vamos a crear juntos!