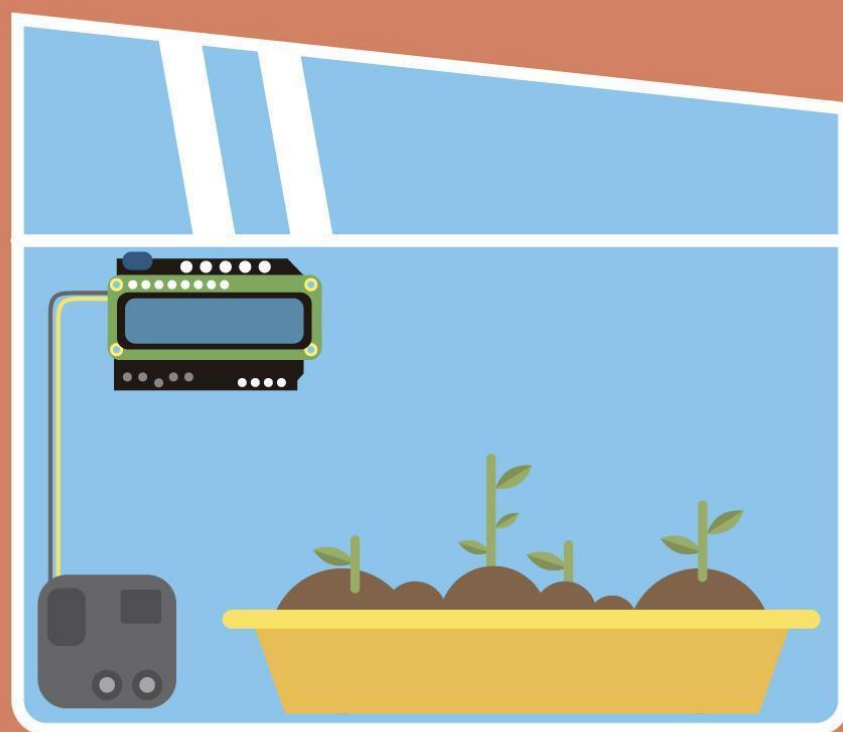


SABERES DIGITALES



INVERNADERO INTELIGENTE

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

INVERNADERO INTELIGENTE

Ficha técnica	3
Presentación	4
Desarrollo	5
Nivel Inicial:	5
Paso 1 - Fabricar la estructura del invernadero	5
Paso 2 - Realizar el montaje sencillo de un circuito en una Protoboard	5
Paso 3 - Programar los tiempos de parpadeo del LED	6
Paso 4 - Subir el código a la placa Arduino	10
Paso 5 - Conectar el sensor de temperatura y humedad	10
Paso 6 - Instalar la extensión del sensor de temperatura y humedad	11
Nivel Intermedio	14
Paso 1 - Conectar el módulo del display LCD	14
Paso 2 - Instalar la extensión del display LCD	15
Paso 3 - Hacer una prueba del display LCD con un programa de saludo	15
Paso 4 - Escribir la temperatura en el display	17
Paso 5 - Agregar la lectura de la humedad	19
Nivel Superior	21
Paso 1 - Introducción a Internet de las Cosas (IoT)	21
Paso 2 - Crear un Panel de Control	22
Paso 3 - Conectar módulo OBLOQ	29
Paso 4 - Arduino IDE	29
Paso 5 - Programar sin código bloqueante	31
Paso 6 - Programación IoT.	35
Cierre	41
Glosario	42
Reconocimientos	45

INVERNADERO INTELIGENTE

Ficha técnica

Nivel educativo	Secundario. Ciclo básico
Descripción general	Diseño, construcción y programación de un invernadero inteligente.
Niveles de complejidad	<p>Nivel inicial: realizar, utilizando una placa Arduino, el circuito de funcionamiento de un sistema de control de temperatura para un invernadero. Este contará con un sensor que medirá la temperatura del invernadero y accionará un led cada vez que esta alcance valores muy altos. De esta manera, indicará cuándo el invernadero debe ser ventilado.</p> <p>Nivel intermedio: agregar un display LCD para ver la temperatura y la humedad relativa del ambiente.</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">• 1 x Sensor DHT11 (humedad y temperatura)• 20 cables dupont hembra hembra• 20 cables dupont macho hembra• 1 x led verde 5mm• 1 x resistencias de 220 ohm.• 1 x Arduino UNO R3• 1 x Protoboard• 1 x Cable USB tipo B• 1 x Display LCD Keypad Shield• 1 x OBLOQ IOT Module• 1 x Fuente de 9V 1 A (plug centro positivo, 5.5 x 2.1 mm)
Equipamiento	<ul style="list-style-type: none">• Computadora• Soldador• Estaño• Alicates• Pinza de punta• Brusela

Otros requisitos

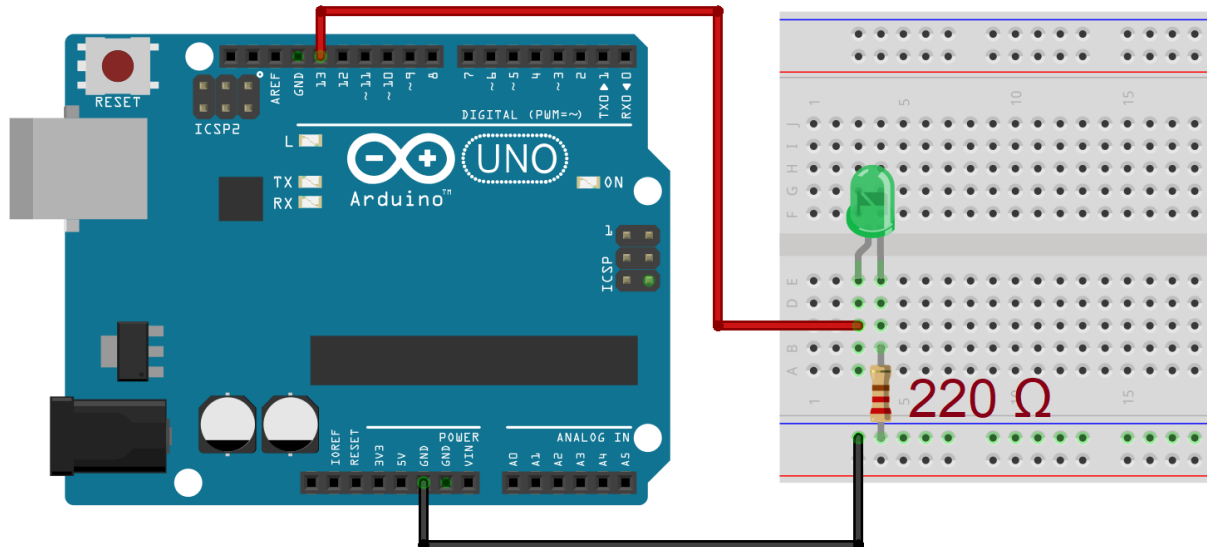
- Conexión a internet
- Descargar el programa "mblock3"

Desarrollo

Nivel Inicial

Los invernaderos son construcciones que permiten el cultivo de algunas plantas en lugares que no cuentan con las condiciones climáticas adecuadas para su crecimiento. Estos se componen de una estructura que se encuentra cubierta (tanto en las paredes laterales como en el techo) por vidrio o polietileno (un tipo de plástico) transparente. La cobertura transparente del invernadero permite el paso de la luz (radiación) y retiene el calor en su

Colocamos un led verde en la Protoboard, como indica la figura. El polo positivo del led se conecta con un cable al *pin* 13 de la placa Arduino. Luego, conectamos una resistencia entre el polo negativo del led y el de la Protoboard. Finalmente, conectamos con un cable el terminal negativo de la Protoboard con el terminal negativo (llamado GND) de la placa Arduino.



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

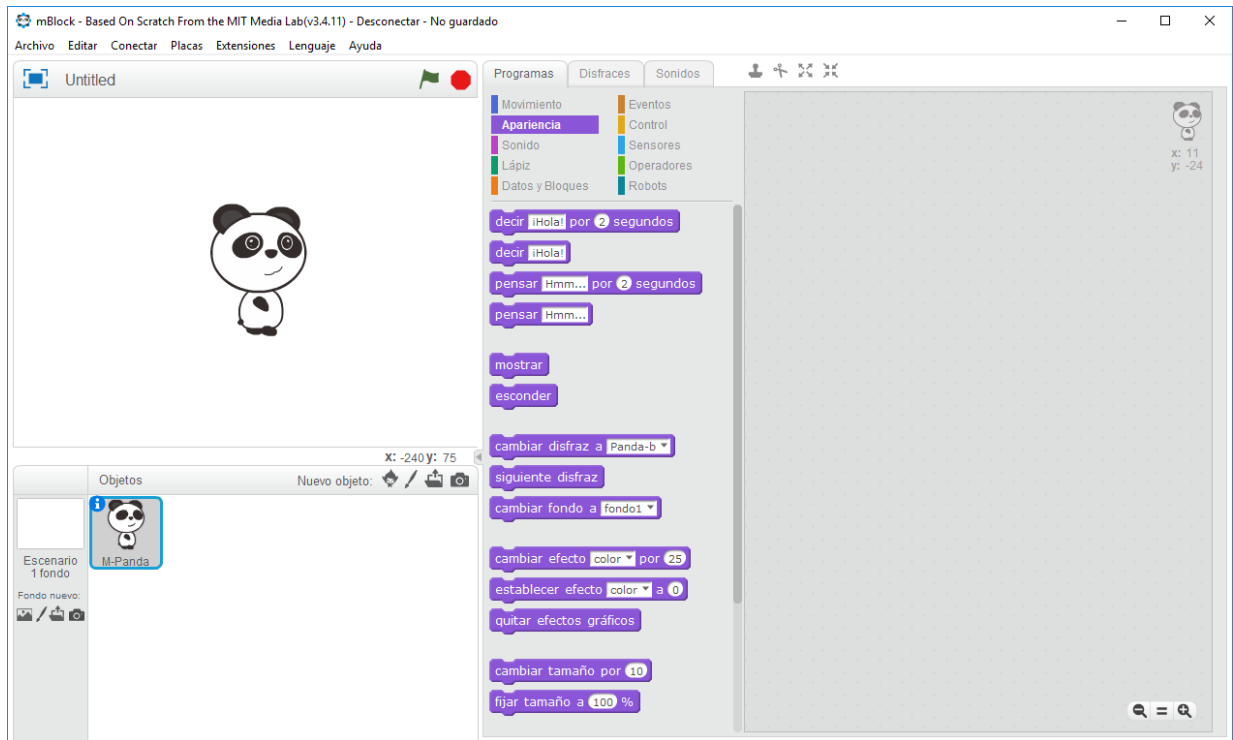
- Pata larga positiva
- Pata corta negativa



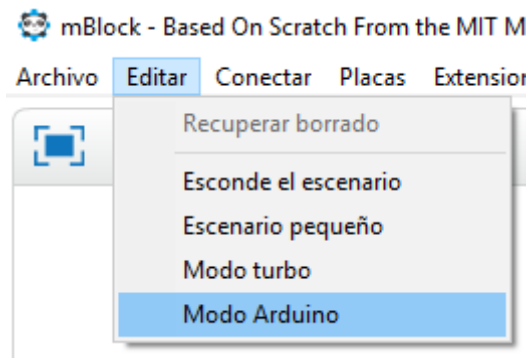
Paso 3 - Programar los tiempos de parpadeo del LED

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/>

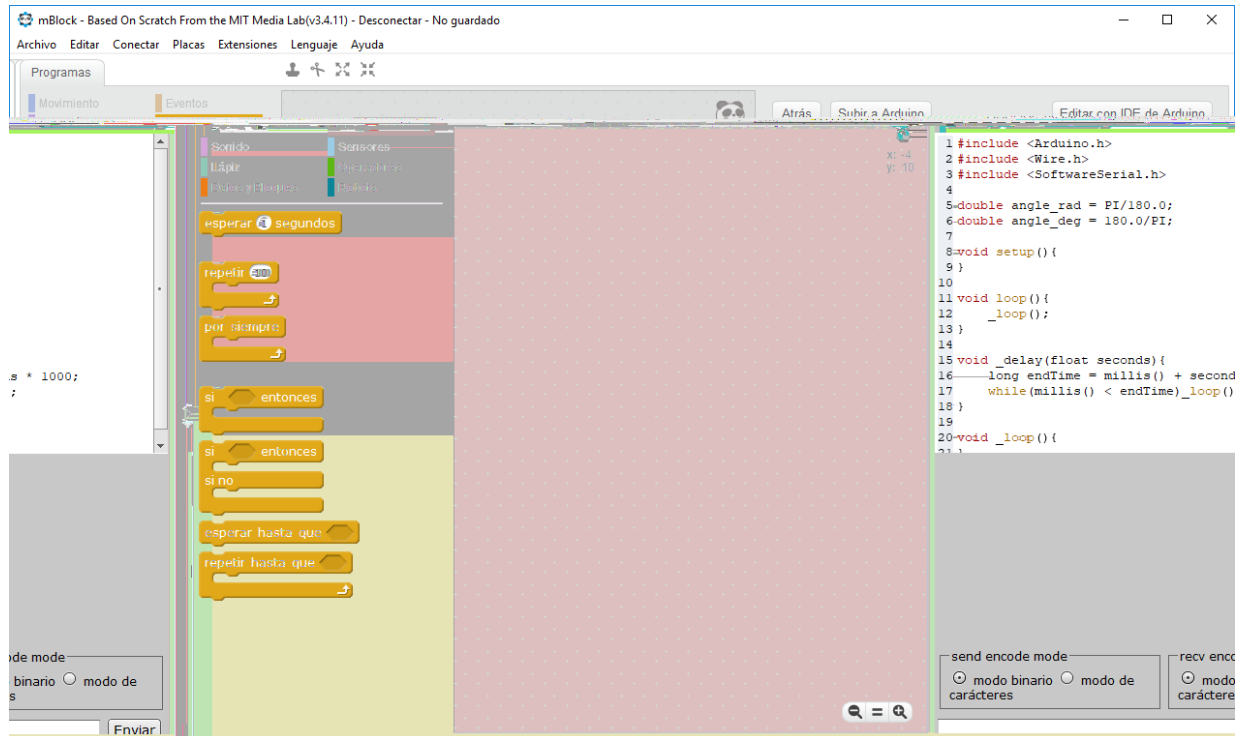
Cuando abrimos mBlock3, vemos una pantalla como la siguiente:



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



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.

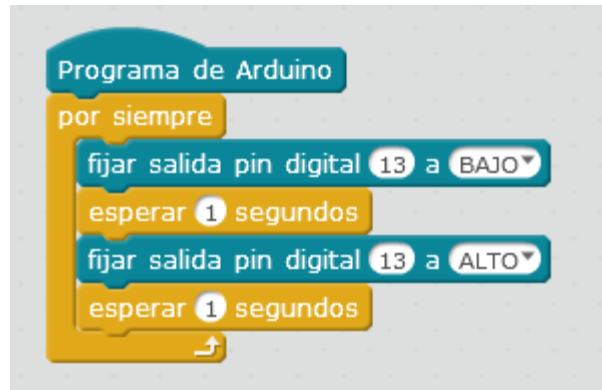


Los bloques están agrupados por categorías. En este caso, se usarán las siguientes: “**Robots**”, “**Control**”, “**Operadores**” y “**Datos y Bloques**”. Cuando seleccionamos una de estas categorías, se pueden visualizar todos los bloques que pertenecen a ese grupo.



Después de familiarizarnos con el sistema, vamos a escribir un programa que haga parpadear el led que acabamos de conectar. Utilizaremos el estado “BAJO” para indicar que el led está apagado y el estado “ALTO” para indicar que está encendido.

El código en bloque debería verse de manera similar al siguiente modelo:



Veremos que a la derecha se muestra el código escrito 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();
}

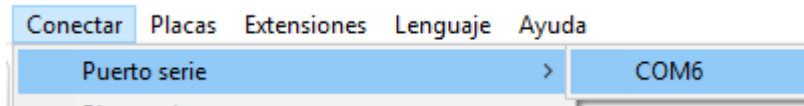
void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

Paso 4 - Subir el código a la placa Arduino

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

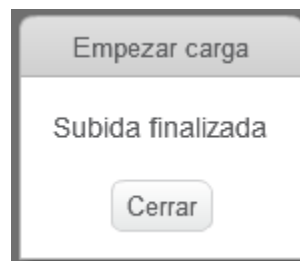
1. Conectar la placa Arduino a la entrada USB.
2. Chequear que en el menú "Placas" esté seleccionado "Arduino Uno".
3. Seleccionar el puerto serie al que está conectada la placa.



4. Clickear el botón

Subir a Arduino

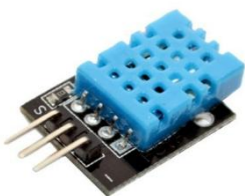
Al terminar de subir nuestro código, veremos este mensaje:



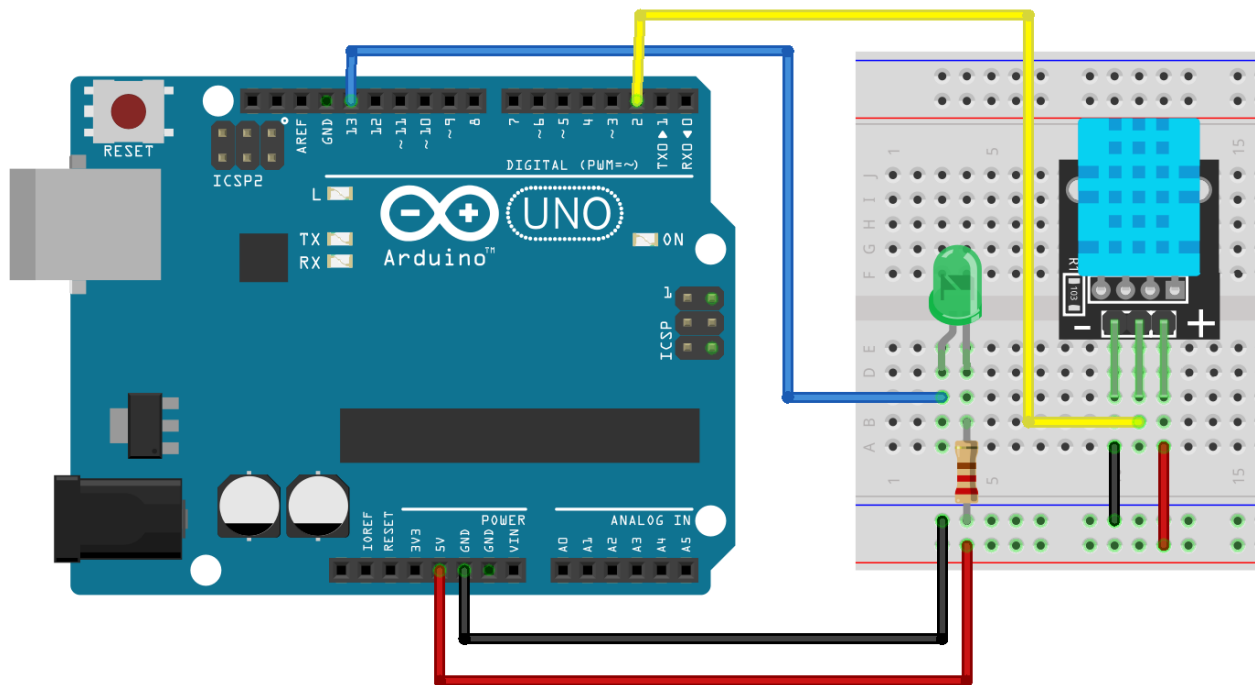
Con la conexión previamente realizada, el led debería encenderse y apagarse siguiendo intervalos de un segundo.

Paso 5 - Conectar el sensor de temperatura y humedad

Conectamos el sensor DHT11 como indica el siguiente esquema:

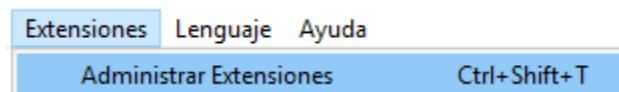


El DHT11 es un sensor que mide humedad y temperatura del aire. Este modelo de sensor posee electrónica interna que digitaliza los datos registrados y los reporta a el Arduino mediante una comunicación digital, por eso es que se conecta a un pin digital de la placa.



Paso 6 - Instalar la extensión del sensor de temperatura y humedad

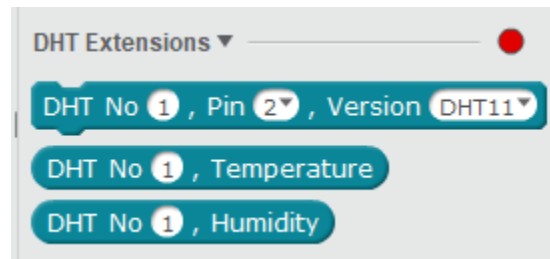
Para poder utilizar el sensor de temperatura y humedad, necesitamos instalar una extensión de mBlock3. Esto lo hacemos desde el menú:



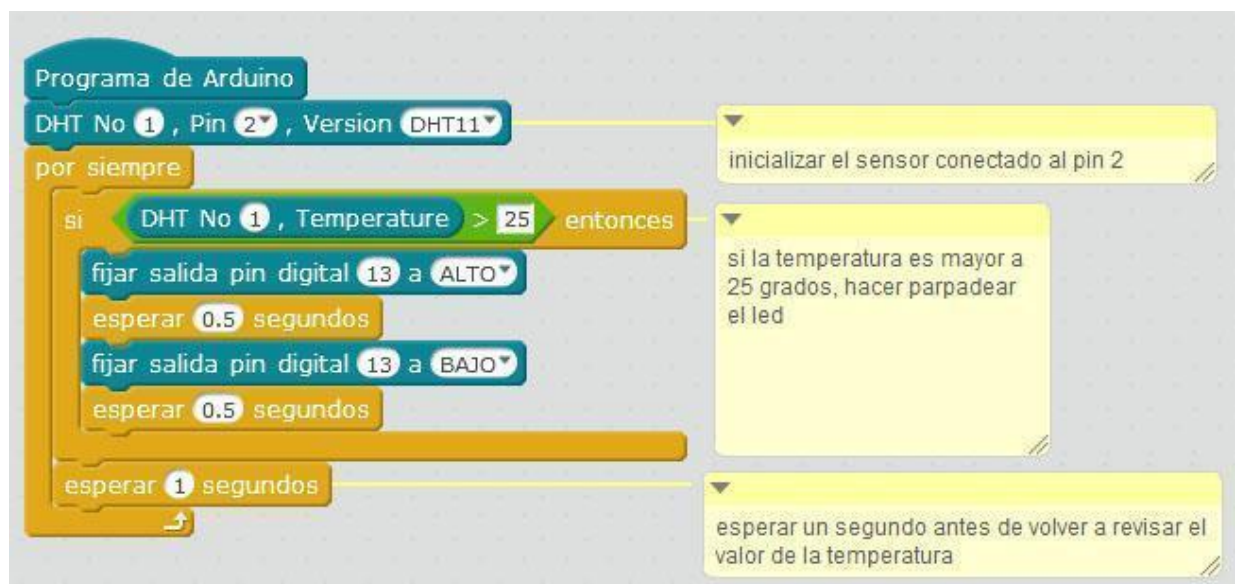
El sensor de temperatura y humedad que usaremos es el DHT11. Si en el buscador tipeamos "dht", encontraremos el que precisamos entre los resultados, de esta manera: "DHT Extensions".



Bajamos la librería y veremos que contamos con nuevos bloques disponibles dentro de la categoría “**Robots**”.



Ahora podemos modificar nuestro programa para que el led parpadee sólo cuando la temperatura sea superior a, por ejemplo, 25 °C. Nuestro programa quedaría, en ese caso, como se ve a continuación:



Y el código escrito que se puede ver a la derecha, debería ser similar al siguiente:

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

#include "DHT.h"

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
DHT dht_1(2,11);
```

```

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

void loop(){
    if((dht_1.readTemperature()) > (25)){
        digitalWrite(13,1);
        _delay(0.5);
        digitalWrite(13,0);
        _delay(0.5);
    }
    _delay(1);
    _loop();
}

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

void _loop(){
}

```

Si acercamos una fuente de calor que se encuentre a más de 25 °C al sensor (a modo de prueba), deberíamos ver que el led se enciende.

Nivel Intermedio

Ahora, además de contar con un led que indique cuándo es necesario ventilar el invernadero, se desea poder visualizar en una pantalla cuál es la temperatura a la que se encuentra su interior en cada momento. En esta pantalla se puede incluir, también, el nivel de humedad que posee el ambiente, que es medido en simultáneo por el sensor.

En esta segunda instancia del proyecto se propone agregar un *display* LCD que muestre la temperatura y la humedad ambiente del invernadero que mide el sensor DHT11.

Paso 1 - Conectar el módulo del *display* LCD

Dado que el display LCD con que trabajamos tiene formato tipo shield, se puede conectar al Arduino apilando y encastrando ambas placas. Es importante tener cuidado y no forzar las piezas, deben coincidir todos los pines perfectamente.



Los *shields* son placas de circuitos modulares que se montan unas encima de otras para agregar nuevas funcionalidades a la placa Arduino. Existen los que agregan funciones tales como comunicación, pantallas, sensores, interconexión, etc.

Paso 2 - Instalar la extensión del *display* LCD

Para programar el *display*, necesitamos instalar otra extensión. Como antes, buscamos en el administrador de extensiones ingresando ahora la palabra clave “lcd”. Encontraremos la que nos interesa con el nombre “LCD”.

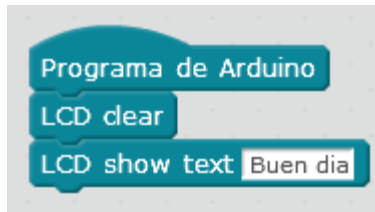


Bajamos la librería. Veremos que contamos con nuevos bloques disponibles dentro de la categoría “**Robots**”.



Paso 3 - Hacer una prueba del *display* LCD con un programa de saludo

Para aproximarnos al funcionamiento del *display* LCD, escribiremos un programa que nos muestre un saludo en el mismo. Este programa debería ser similar al siguiente modelo:



No utilizar tildes ni la letra Ñ al trabajar con el display LCD.

Y el código escrito que se puede ver a la derecha, debería ser similar al siguiente:

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

#include <LiquidCrystal.h>
double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup(){
    lcd.begin(16, 2);
    lcd.clear();
    lcd.print("Buen dia");
}

void loop(){
    _loop();
}

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

void _loop(){
}
```

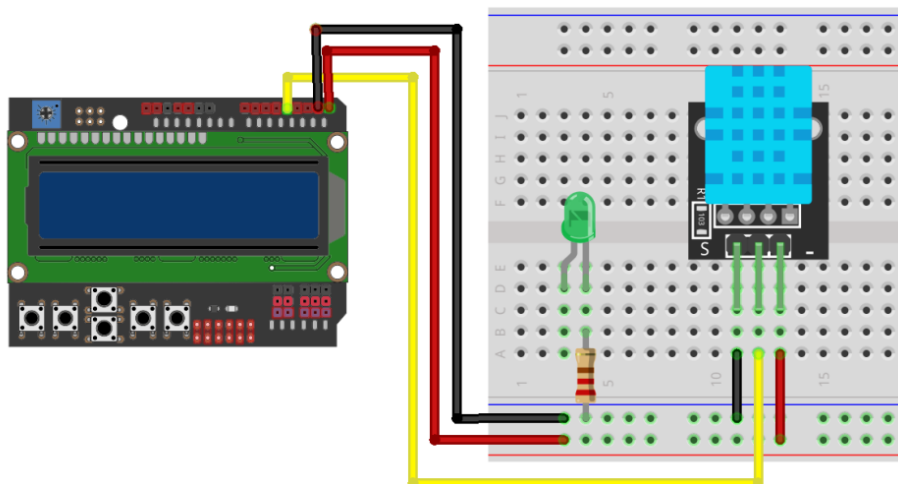
En este momento deberíamos ver el saludo “Buen dia” en nuestro *display*. Si no se llegara a leer el texto, es necesario ajustar el contraste del mismo con el potenciómetro azul (*preset*) que se encuentra en la esquina superior izquierda del *shield* y tiene la referencia RP1. Para ajustarlo podemos utilizar un destornillador.



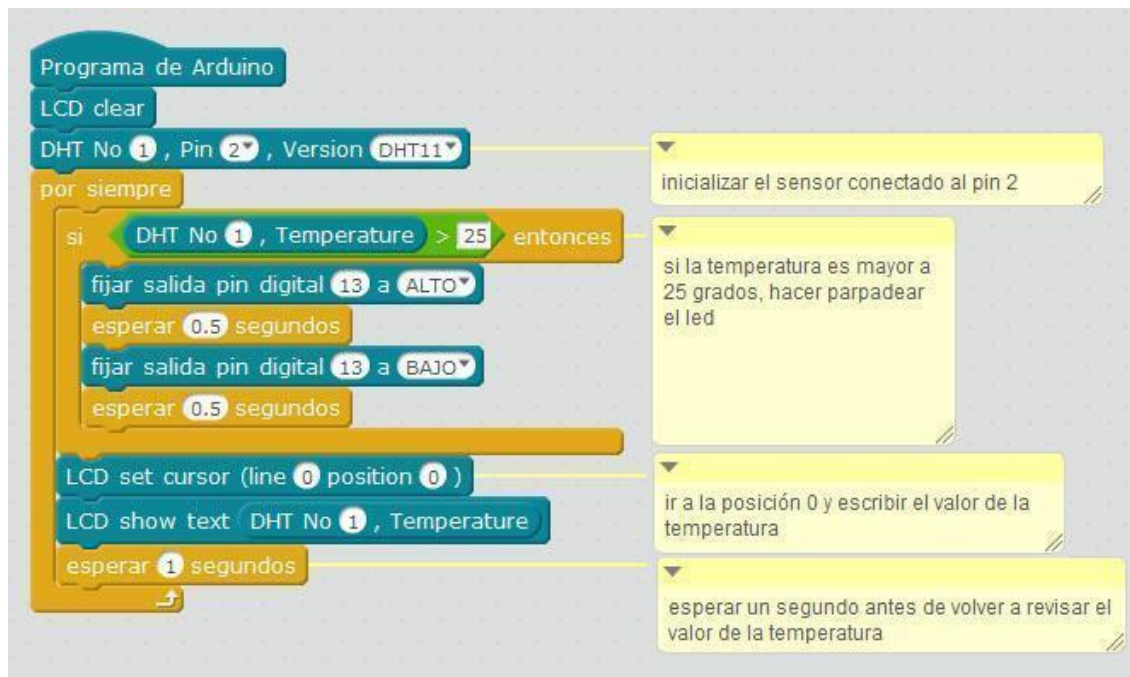
Paso 4 - Escribir la temperatura en el *display*

Ahora volvemos a nuestro sistema y conectamos nuevamente el sensor de temperatura y humedad y el LED.

Debería quedarnos como indica el siguiente esquema:



Modificamos el programa que habíamos hecho en el nivel inicial para que ahora también muestre en el *display* el valor de temperatura leído por el sensor. Con esta nueva funcionalidad, el programa debería quedar como se ve a continuación:



Y el código escrito que se puede ver a la derecha, debería ser similar al siguiente:

```

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

#include "DHT.h"
#include <LiquidCrystal.h>
double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
DHT dht_1(2,11);

void setup(){
  lcd.begin(16, 2);
  lcd.clear();
  pinMode(13,OUTPUT);
}

void loop(){
  if((dht_1.readTemperature()) > (25)){

```

```

        digitalWrite(13,1);
        _delay(0.5);
        digitalWrite(13,0);
        _delay(0.5);
    }
    lcd.setCursor(0,0);
    lcd.print(dht_1.readTemperature());
    _delay(1);
    _loop();
}

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

void _loop(){
}

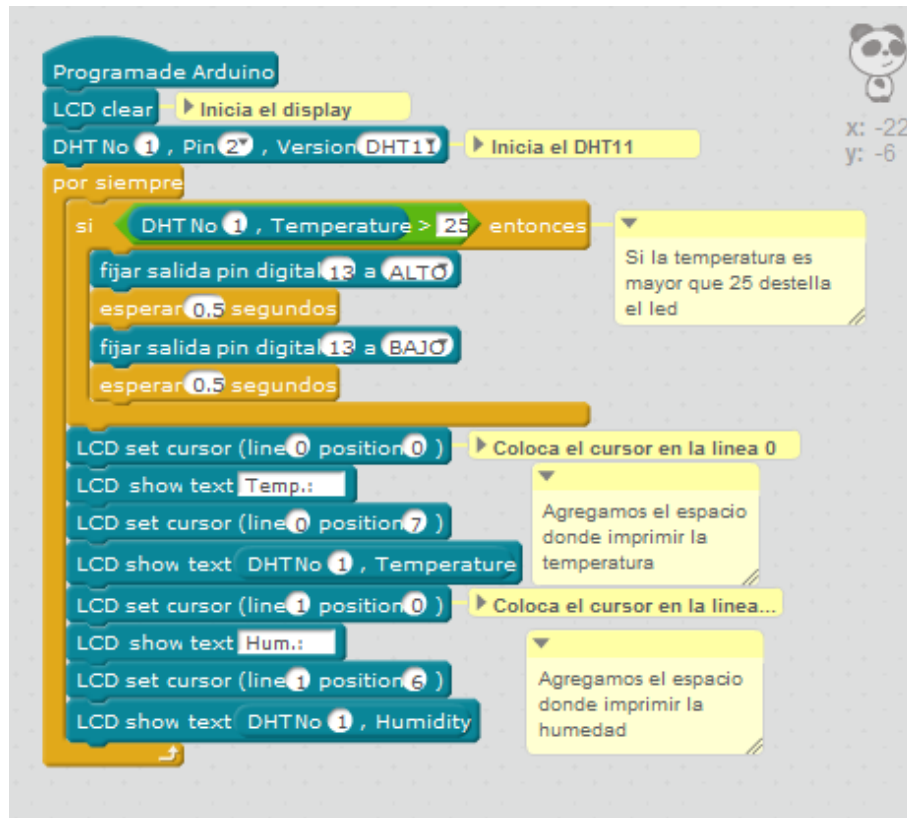
```

Paso 5 - Agregar la lectura de la humedad

Ya que este mismo sensor registra también la humedad, modificaremos el programa para que muestre su valor en el *display* junto con el de la temperatura. Aprovecharemos que el *display* tiene dos filas para escribir el valor de la temperatura en la primera fila y el de la humedad en la segunda.

Además, haremos que el sistema escriba los textos “Temp.:” y “Hum.:” en el *display* inmediatamente antes de los valores para que su lectura resulte más clara.

El programa ahora debería quedar como se ve a continuación:



Y el código escrito que se puede ver a la derecha debería ser similar al siguiente:

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

#include "DHT.h"
#include <LiquidCrystal.h>
double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
DHT dht_1(2,11);

void setup(){
  lcd.begin(16, 2);
  lcd.clear();
  pinMode(13,OUTPUT);
}

void loop(){
  if((dht_1.readTemperature()) > (25)){
    digitalWrite(13,1);
    _delay(0.5);
    digitalWrite(13,0);
  }
}
```

```

        _delay(0.5);
    }
    lcd.setCursor(0,0);
    lcd.print("Temp.:    ");
    lcd.setCursor(7,0);
    lcd.print(dht_1.readTemperature());
    lcd.setCursor(0,1);
    lcd.print("Hum.:    ");
    lcd.setCursor(6,1);
    lcd.print(dht_1.readHumidity());
    _loop();
}

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

void _loop(){
}

```

Nivel Superior

Se desea monitorear, desde una central, las condiciones internas de humedad y temperatura del invernadero. Para ello se decidió instalar un sistema de IoT, que permite enviar la información que recoge nuestro sensor a la central a través de internet.

En esta actividad se programará el envío de los 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.

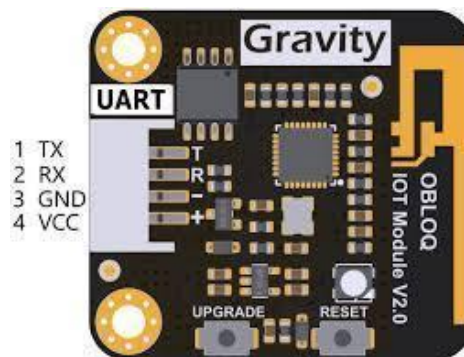
Para poder incorporar IoT a nuestro proyecto es necesario:

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.

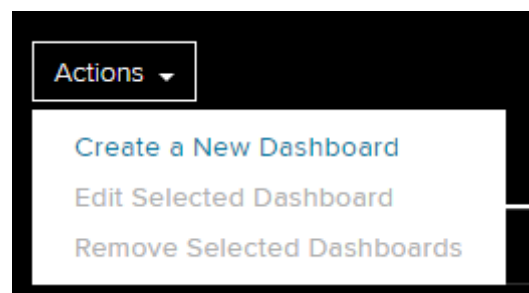
El módulo UART OBLOQ es un dispositivo WiFi a serie pensado para desarrolladores no profesionales. Permite enviar y recibir datos mediante los protocolos HTTP y MQTT.



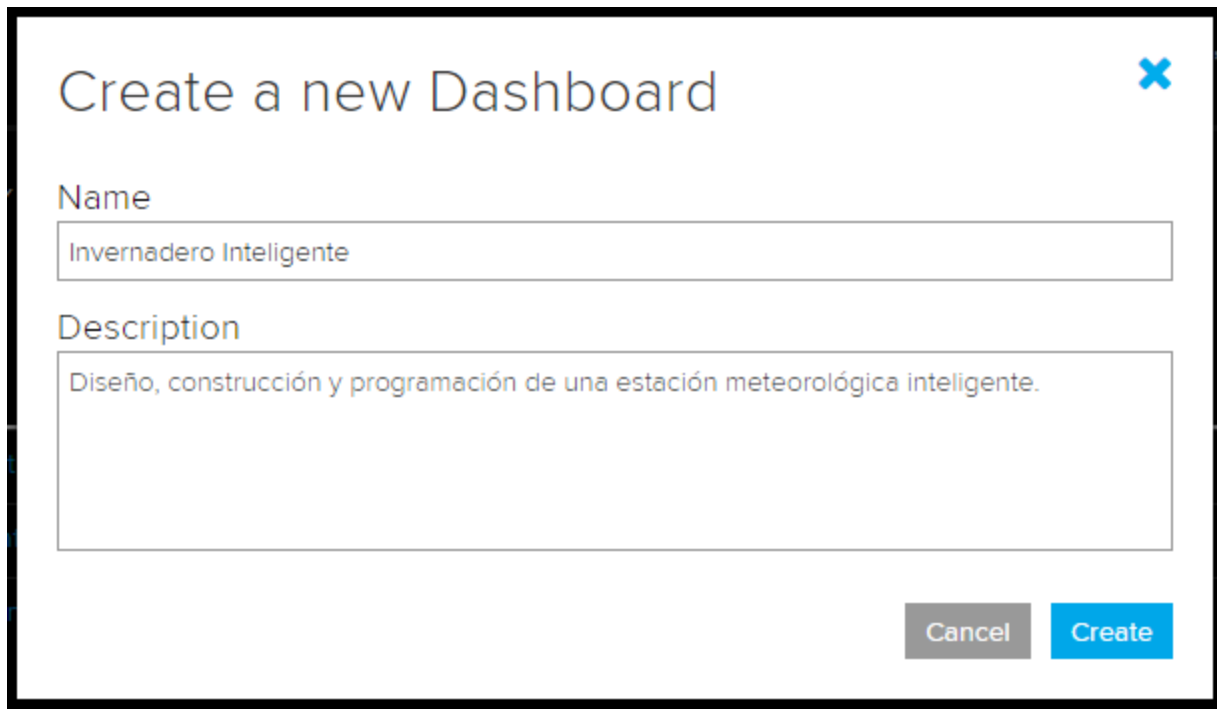
Paso 2 - Crear un Panel de Control

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.

Debemos crear una cuenta de usuario en io.adafruit.com. Una vez que ingresamos con nuestro usuario, creamos un nuevo panel haciendo click en “Create a New Dashboard”.



Creamos un nombre y una descripción.



Create a new Dashboard


Name

Invernadero Inteligente

Description

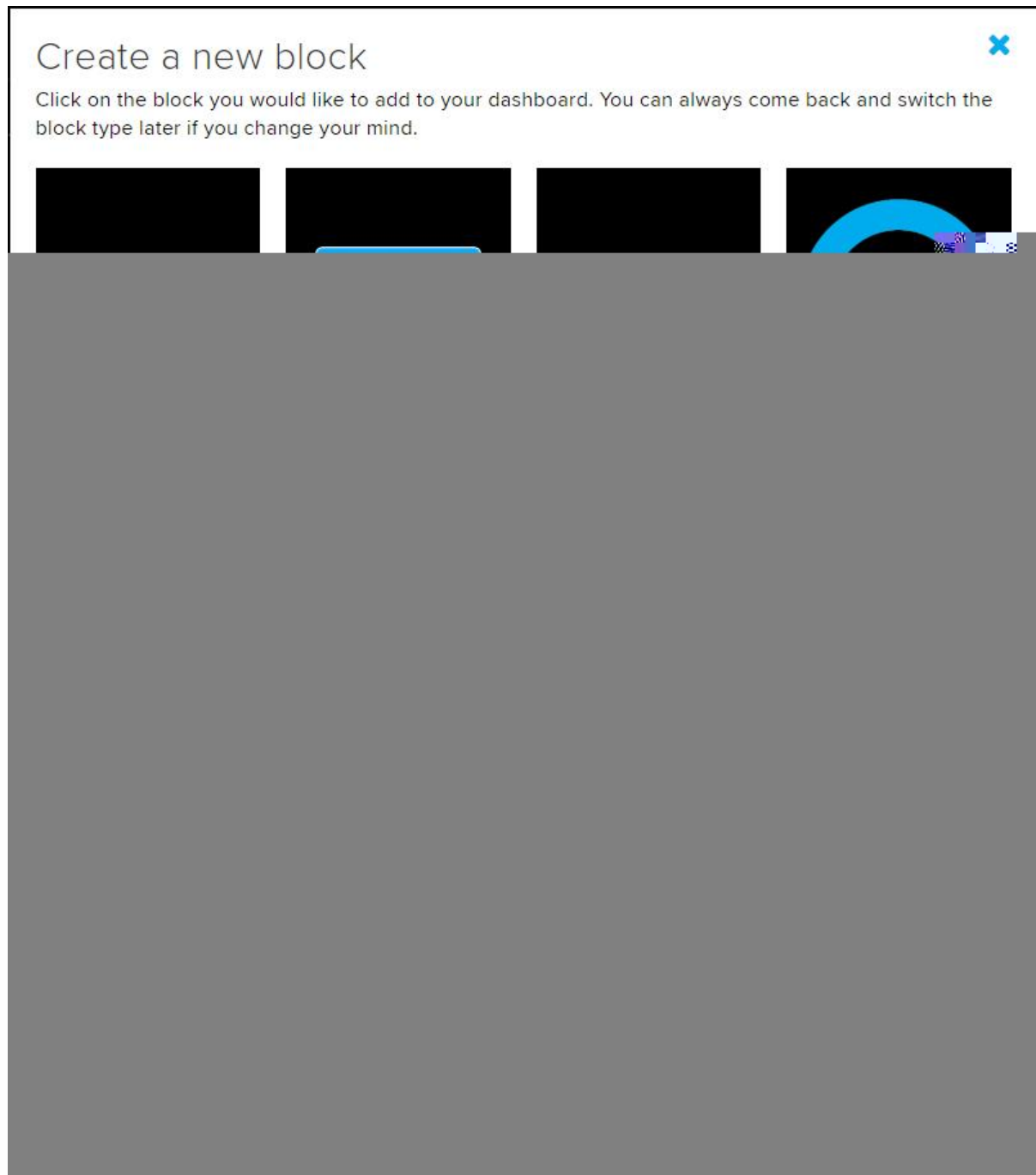
Diseño, construcción y programación de una estación meteorológica inteligente.

Cancel Create

Hacemos click en el nuevo panel creado y veremos una pantalla vacía. Podemos comenzar a agregar bloques haciendo click en .



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



Para nuestro invernadero, podríamos ubicar dos *Line Chart* (gráficos de línea) que nos permitan ver el historial de los cambios que surjan en la temperatura y la humedad.



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.

En las líneas de código las palabras no pueden llevar tilde ni usar la letra Ñ.

Llamamos al primer *feed* “temperatura”. En él publicaremos, desde nuestro dispositivo, el valor de la temperatura que lea el sensor DHT11.

Choose up to 5 feeds

Line Chart: The line chart is used to graph one or more feeds.

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

Group / Feed	Last value	Recorded
<input checked="" type="checkbox"/> temperatura		a few seconds ago 1 of 5

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

Show History

24 hours

X-Axis Label

Y-Axis Label

Y-Axis Minimum

Leave blank to automatically detect.

Y-Axis Maximum

Leave blank to automatically detect.

Decimal Places

Number of decimal places to display, defaults to 4.

Block Preview

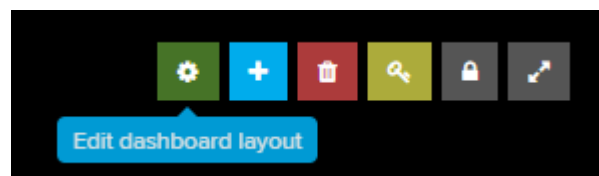
Line Chart The line chart is used to graph one or more feeds.

Previous step

Create block

Hacemos click en "Create block" (crear bloque) para completar la operación.

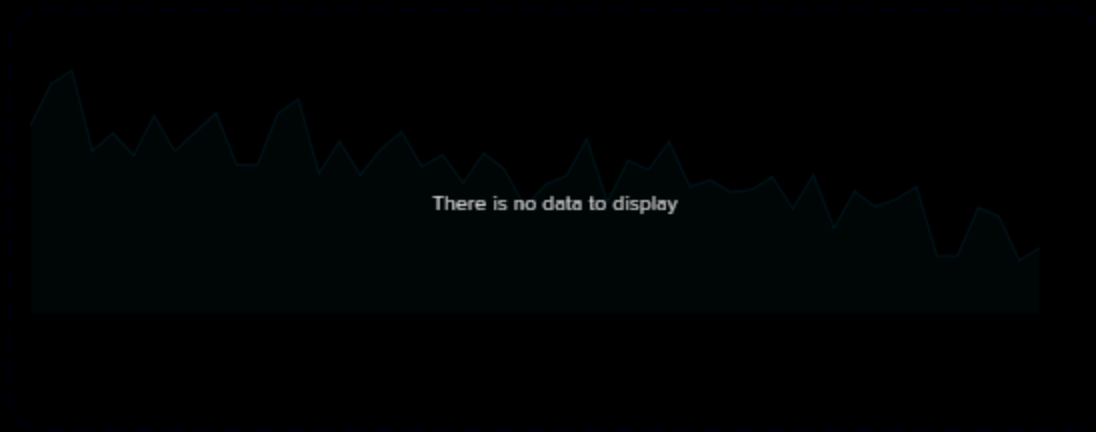
Podemos modificar el tamaño y la ubicación de los bloques haciendo click en la "rueda de configuración".



Repetimos este procedimiento para el *feed* "humedad". Deberíamos, dado que aún no contamos con datos publicados, visualizar algo similar a lo siguiente:

/Dashboards/Invernadero Inteligente

Temperatura

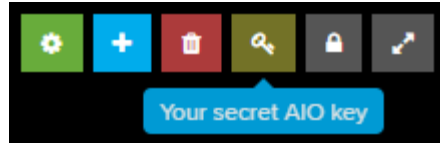


Humedad



Una vez realizado el Panel, programaremos nuestro sistema para que publique en él los datos que obtenga por medio de los sensores, lo que nos permitirá monitorearlos de manera remota.

Antes de salir, debemos copiar las credenciales de acceso para poder publicar en nuestros *feeds* “temperatura” y “humedad”. Para ver nuestras credenciales, hacemos click en el ícono de la “llave”.



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

usuario_aio

Active Key

1234cfdd29a244b6b049abb07727c117

REGENERATE AIO KEY

[Hide Code Samples](#)

Arduino

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

Copiamos el código que nos ofrece para Arduino, con nuestro usuario y key. Más adelante se verá que estos datos aparecen en el código de la siguiente manera:

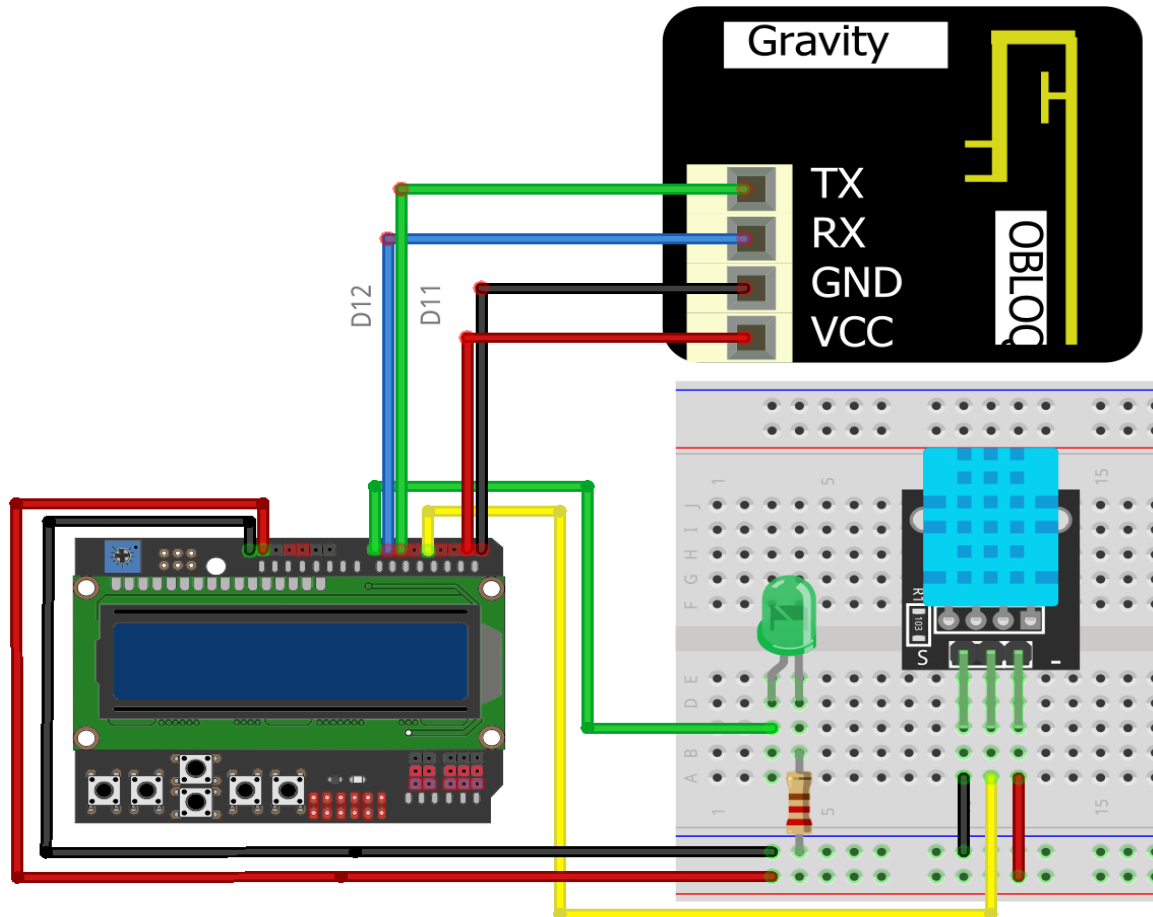
```
#define IO_USERNAME "usuario_adafruit"
#define IO_KEY "key_adafruit"
```

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"
```

Paso 3 - Conectar módulo OBLOQ

A continuación se muestra el diagrama de conexión de Arduino UNO y OBLOQ.



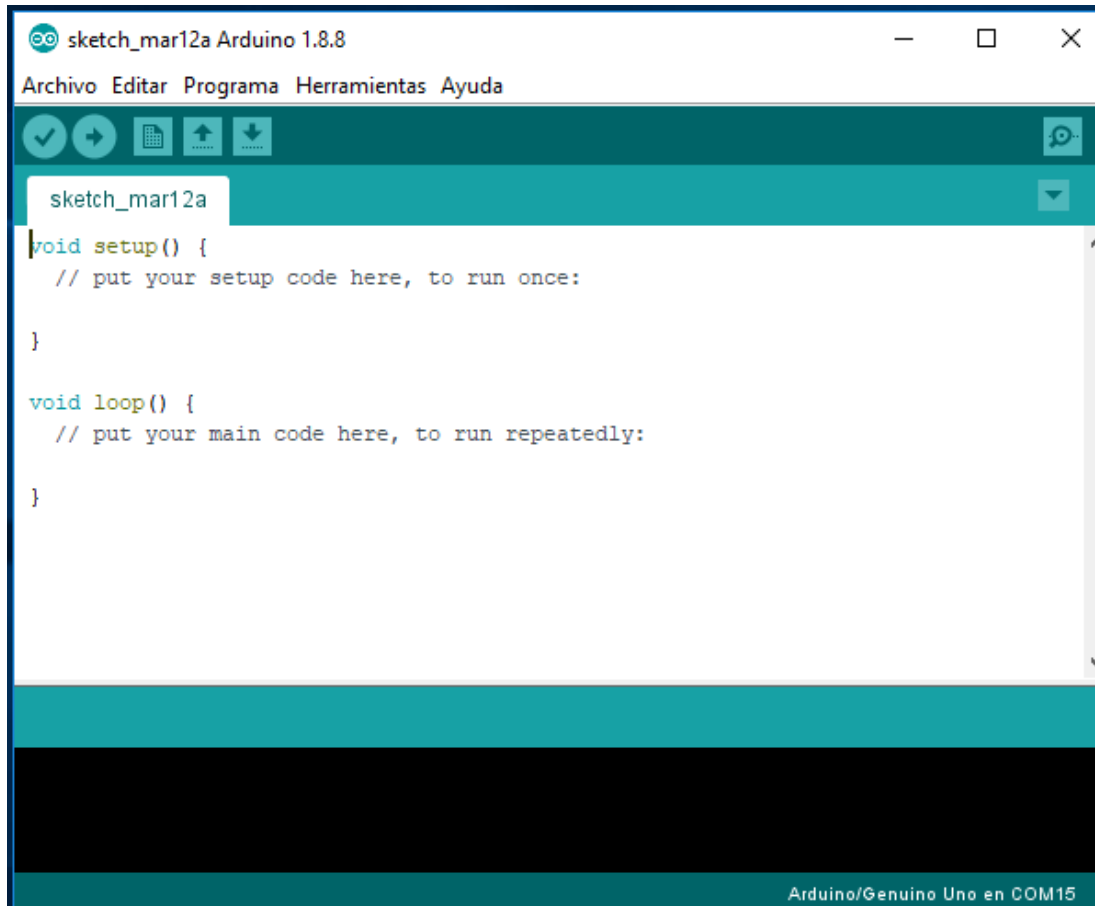
Paso 4 - Arduino IDE

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:



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.  
}
```

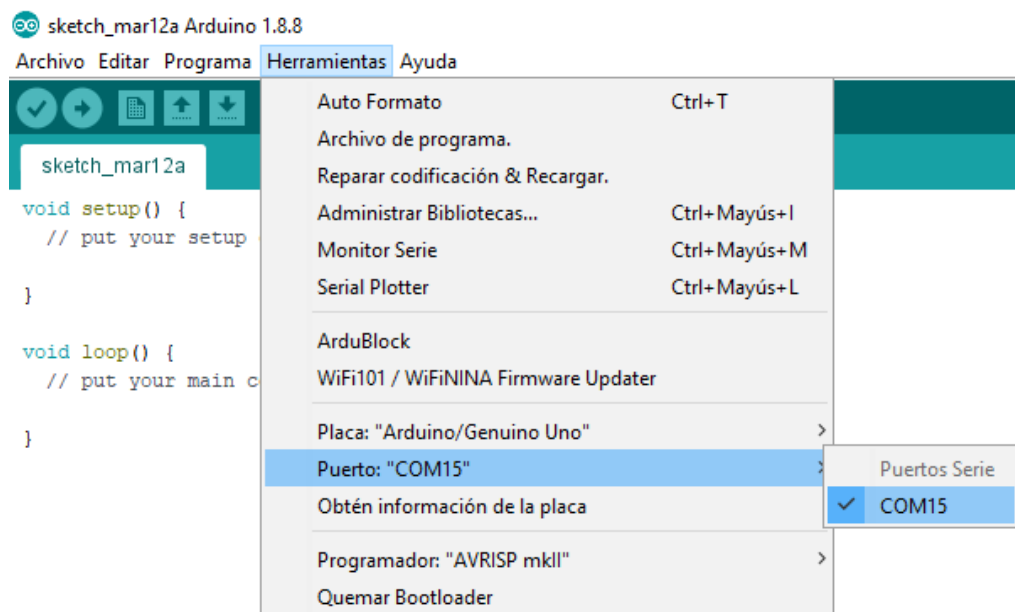
En líneas generales, un programa de Arduino es:

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.



3. Clickear el botón de “Subir” .

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 emplear 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.
}

```

En el ejemplo 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.

Ahora vamos a modificar nuestro programa para dejar de utilizar código bloqueante. Con el objetivo de simplificar un poco la programación, determinaremos que el led se mantenga encendido cuando la temperatura supere los 25°C en lugar de hacerlo parpadear.

Nuestro programa actualizado debería quedar como se ve a continuación:

```

// Incluimos las librerias necesarias.
#include "DHT.h"
#include <LiquidCrystal.h>

// Declaramos nuestros objetos de LCD y DHT11.
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
DHT dht(2,11);

// 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);
    // Inicializamos el display LCD de 16 columnas y 2 filas.
    lcd.begin(16, 2);
    // Borramos el display LCD.
    lcd.clear();
}

void loop() {

    long millisActual = millis();

    // Realizamos lo siguiente cada un segundo.
    if (millisActual - millisAnterior >= 1000) {

        // Si la temperatura es mayor a 25 grados
        if(dht.readTemperature() > 25){
            // encendemos el led.
            digitalWrite(13,1);
        }else{
            // Si no, apagamos el led.
            digitalWrite(13,0);
        }

        // Imprimimos los valores de temperatura y humedad en el lcd.
        lcd.setCursor(0,0);
        lcd.print("Temp.: ");
        lcd.print(dht.readTemperature());
        lcd.setCursor(0,1);
    }
}

```

```

lcd.print("Hum.: ");
lcd.print(dht.readHumidity());

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

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

}

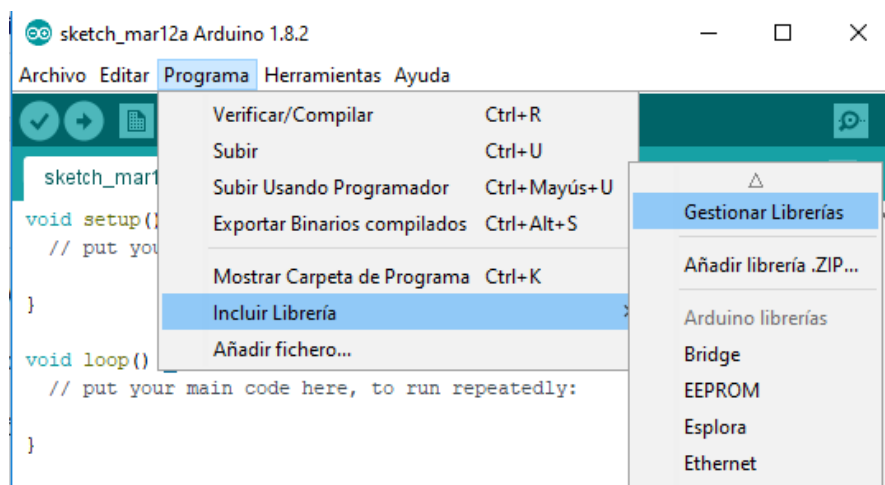
```

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

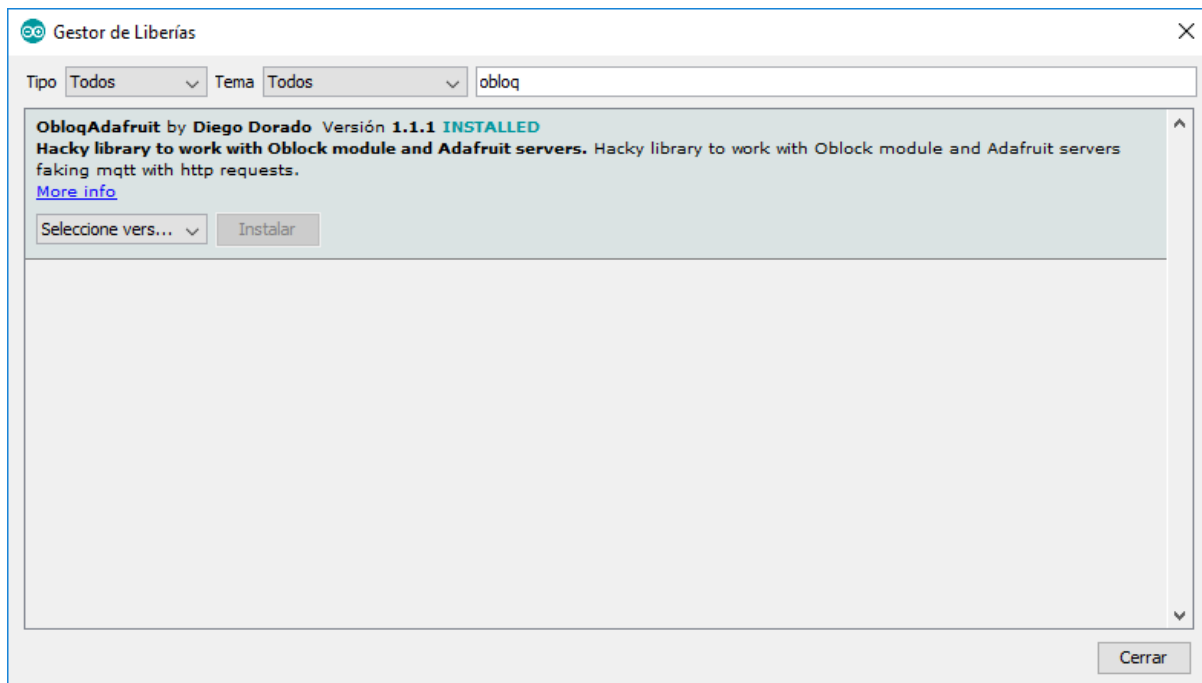
Paso 6 - Programación IoT.

Utilizaremos la librería ObloqAdafruit para informar a Adafruit los datos de temperatura y humedad del invernadero. Podremos monitorear este estado desde el Panel de Control que hemos creado previamente.

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

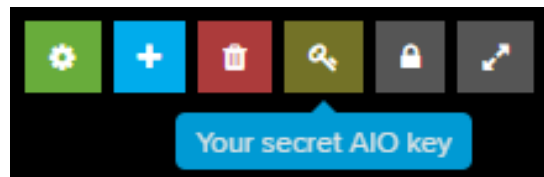


Se abrirá una ventana con un buscador en margen superior. Debemos escribir Obloq, seleccionar la librería ObloqAdafruit y apretar el botón Instalar.



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. Para ello maximizamos la pantalla de Adafruit y hacemos click en el ícono de la “llave”.



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"
```

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(11,12);
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("temperatura", 25); // informar que la temperatura es de 25°C
```

Vamos a modificar nuestro programa para incorporar el envío de datos de temperatura y humedad a Adafruit. Nuestro programa de control y monitoreo de las condiciones del invernadero con IoT debería quedar como se ve a continuación:

```
// Incluimos las librerías necesarias.
#include "DHT.h"
#include <LiquidCrystal.h>
#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"

// Declaramos nuestros objetos de LCD, DHT11 y OBloq.
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
DHT dht(2,11);
SoftwareSerial softSerial(11,12);
ObloqAdafruit olq(&softSerial,WIFI_SSID,WIFI_PASSWORD,IO_USERNAME,IO_KEY);

// 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);
    // Inicializamos el display LCD de 16 columnas y 2 filas.
    lcd.begin(16, 2);
    // Borramos el display LCD.
    lcd.clear();

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

void loop() {

    long millisActual = millis();

    // Realizamos la siguiente tarea cada un segundo.
    if (millisActual - millisAnterior >= 1000) {

        // Guardamos en las variables temperatura
        // y humedad los valores que lee el sensor DHT11.
        int temperatura = dht.readTemperature();
        int humedad = dht.readHumidity();

        // Si la temperatura es mayor a 25 grados,
        if(temperatura > 25){
            // encendemos el led.
            digitalWrite(13,1);
        }else{

```



```
    // Si no, apagamos el led.
    digitalWrite(13,0);
}

// Imprimimos temperatura y humedad en el display lcd.
lcd.setCursor(0,0);
lcd.print("Temp.: ");
lcd.print(temperatura);
lcd.setCursor(0,1);
lcd.print("Hum.: ");
lcd.print(humedad);

// Publicar en Adafruit los valores de temperatura y humedad.
olq.publish("temperatura", temperatura);
olq.publish("humedad", humedad);

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

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

Cierre

Una vez finalizado este proyecto, si se quiere continuar, es posible extenderlo. Una opción sugerida es:

- Complementar con el proyecto “Sistema de riego automatizado” para agregarlo al invernadero.

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.

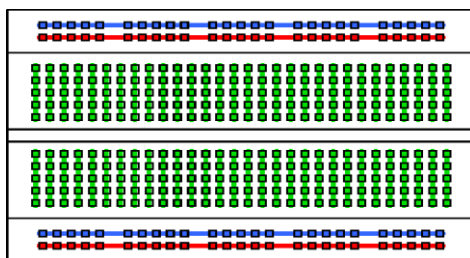
Shield: Placas de circuitos que se monta encima de la placa Arduino para expandir sus funcionalidades. Existen shields para otros tipos de microcontroladores y computadoras embebidas (Arduino Micro, RaspberryPi, etc). En general un shield sirve para ser utilizado con un único modelo de placa, en este caso para Arduino UNO.

El shield suele tener la misma forma que la placa Arduino y tienen pines de conexión que encastran perfectamente con los pines de esta. Los shields poseen diferentes usos como: comunicaciones, sensores, actuadores, interconexión con otros sistemas, sonido, protoboard y una larga lista de etcéteras.

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 via 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 utilizan colores para codificar su valor. 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.

Sensor DHT11: Se utiliza para medir humedad y temperatura. El sensor de temperatura consiste en un termistor, un dispositivo que cambia su resistencia en función de la temperatura que percibe. El sensor de humedad consta de un sustrato higroscópico (un dispositivo que atrae vapor de agua) conectado a dos electrodos que miden su resistencia. Cuanto mayor es la humedad del ambiente, mayor es también la conductividad del sustrato. El DHT11 combina el sensado de ambas variables, integra también un circuito electrónico digital encargado de digitalizar la información y transmitirla al arduino mediante un pin digital a modo de paquete de información.

Por este motivo es que para realizar un programa que utilice este sensor es necesario utilizar una librería que se encarga de gestionar la comunicación entre el Arduino y DHT11. El sensor posee 4 pines de conexión, dos de ellos son alimentación eléctrica (VCC y GND), mientras que un pin se utiliza para la comunicación. Hay un pin que no tiene uso. En algunos casos podremos encontrar el sensor montado en una pequeña placa de interconexión que solamente tiene 3 pines, descartando el pin que no tiene uso.

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 "UNIPE". En particular en el desarrollo de los capítulos 1 y 2, los cuales estuvieron a cargo de los profesores Fernando Raúl Alfredo Bordinon 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!