

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

Fabiá 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 CALEFACCIÓN AUTOMÁTICO

Ficha técnica	4
Presentación	5
Desarrollo	6
Nivel inicial	6
Paso 1: Utilizar un módulo relé	6
Paso 2: Programar el encendido y apagado del radiador eléctrico	7
Paso 3: Subir el código a la placa Arduino	10
Paso 4: Programar el tiempo de encendido y apagado	11
Nivel intermedio	12
Paso 1: Conectar el sensor de temperatura y humedad	¡Error! Marcador no definido.
Paso 2: Instalar la extensión del sensor de temperatura y humedad	13
Paso 3: Obtener la temperatura	14
Paso 4: Activar el envío de datos a la consola	15
Paso 5: Encender el radiador según la temperatura	16
Paso 6: Conectar el módulo del display LCD	17
Paso 7: Instalar la extensión del display LCD	18
Paso 8: Hacer una prueba del display LCD con un programa de saludo	19
Paso 9: Escribir la temperatura en el display.	20
Paso 10: Incorporar un texto en el display	22
Paso 11: Conectar un potenciómetro	23
Paso 12: Crear variables	25
Paso 13: Adaptar los valores	25
Paso 14: Visualizar la temperatura ideal	26
Nivel avanzado	29
Paso 1 - Introducción a Internet de las Cosas (IoT)	29
Paso 2 - Crear un Panel de Control	30
Paso 3 - Conectar módulo O BLOQ	37
Paso 4 - Arduino IDE	37
Paso 5 - Programar sin código bloqueante	39
Paso 6 - Programación IoT.	43
Cierre	46
Glosario	

SISTEMA DE CALEFACCIÓN AUTOMÁTICO

Ficha técnica

Nivel educativo	Secundario. Ciclo Básico.
-----------------	---------------------------

Descripción general	Diseño y construcción de una maqueta/prototipo de un sistema de calefacción.
Niveles de complejidad	<p>Nivel inicial: Programar y montar, en una placa Arduino, el sistema de encendido y apagado automático de un radiador eléctrico, utilizando un temporizador.</p> <p>Nivel intermedio: Agregar un sensor de temperatura para configurar, por medio de un potenciómetro, la temperatura a la que se prende y se apaga el radiador. Incorporar, también, un display LCD para visualizar la información.</p> <p>Nivel avanzado: Monitorear la temperatura y la humedad del ambiente a través de IoT.</p>

Insumos	<p>1 x Arduino UNO R3 1 x Protoboard 1 x Cable usb tipo B 1 x Fuente de 9v 1 A (plug centro positivo, 5.5x2.1mm) 1x Módulo Relé 1 x Shield LCD DFRobots 1 x Sensor DHT11 (Temperatura y humedad) 1 x OBLOQ IOT MODULE 20 cables dupont macho hembra 20 cables dupont macho macho</p>
Equipamiento	<p>Computadora Soldador Estañ Alicate</p>

Desarrollo

Nivel inicial

Martina se mudó a una nueva casa y necesita instalar un sistema de calefacción eléctrico para su habitación. Ella sabe que un uso eficiente de la energía para calefaccionar o enfriar su hogar es fundamental para el cuidado del ambiente.

Por lo tanto, quiere lograr que se mantenga una temperatura confortable utilizando la menor cantidad de energía posible.

Se le ocurrió que la mejor manera de conseguirlo es instalando un sistema que le permita regular la temperatura de la habitación controlando el tiempo de encendido y apagado del calefactor.

En esta actividad se realizará el prototipo de un sistema de calefacción automático (bombilla de 100W o radiador eléctrico) para una habitación. Se programará un tiempo de encendido y un tiempo de apagado.

Paso 1 - Utilizar un módulo relé

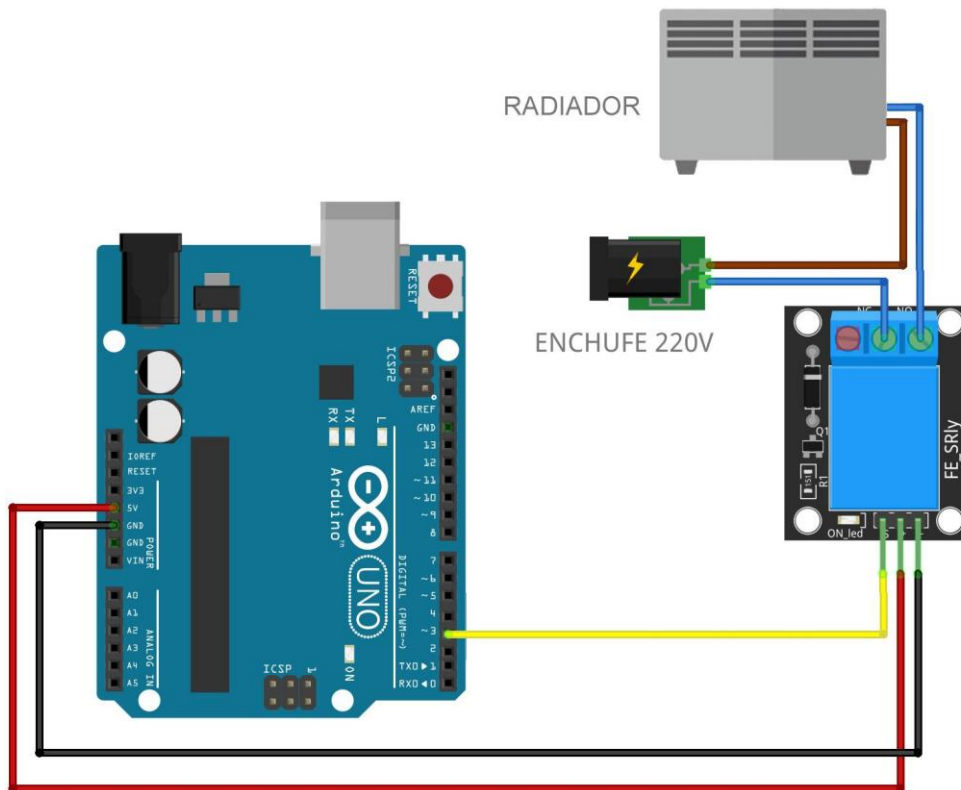
Para optimizar el consumo energético del radiador eléctrico, necesitamos controlar el encendido y apagado del mismo a través de una placa Arduino y un módulo relé

Conectamos el *pin* 3 de la placa Arduino al módulo relé con su respectiva alimentación (5V y GND).

El relé se comportará como un interruptor, prendiendo o apagando nuestro radiador.

Podemos conectarlo de la siguiente forma:

1. Desarmamos el enchufe del radiador.
2. Conectamos un cable a uno de los terminales del enchufe y el otro cable al terminal **NO** de la placa Arduino.
3. Conectamos el cable restante del enchufe al terminal **COM** de la placa Arduino.



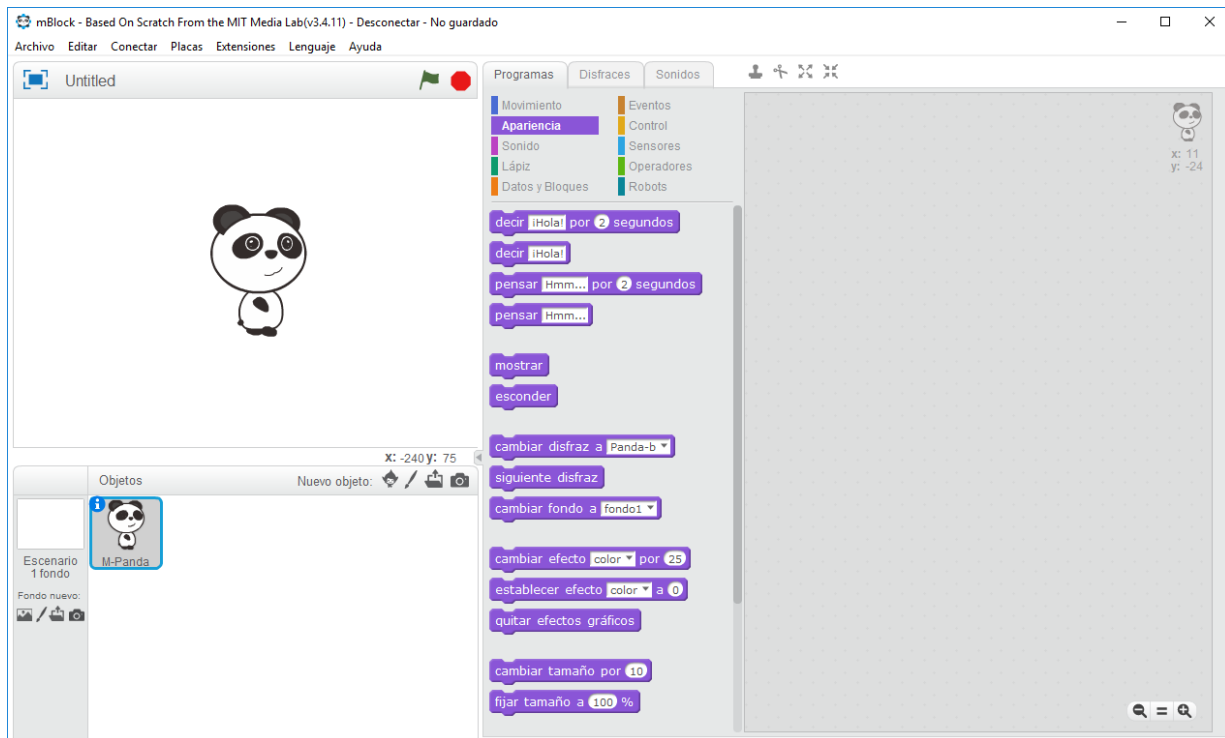
¡Atención! Para construir este dispositivo trabajaremos con un voltaje de 220V. En caso de utilizar protoboard, se recomienda no incluir en el mismo las conexiones a relé y 220V.

El código que subamos a la placa Arduino controlará la apertura y el cierre del relé, en consecuencia, el encendido y apagado del radiador eléctrico.

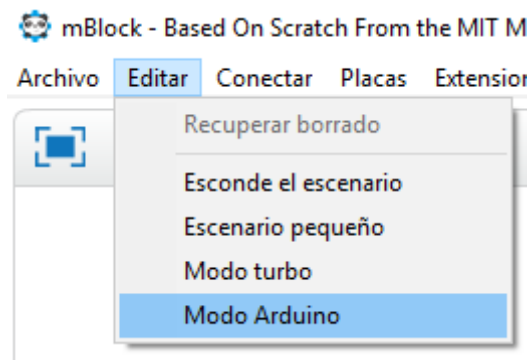
Paso 2 - Programar el encendido y apagado del radiador eléctrico

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

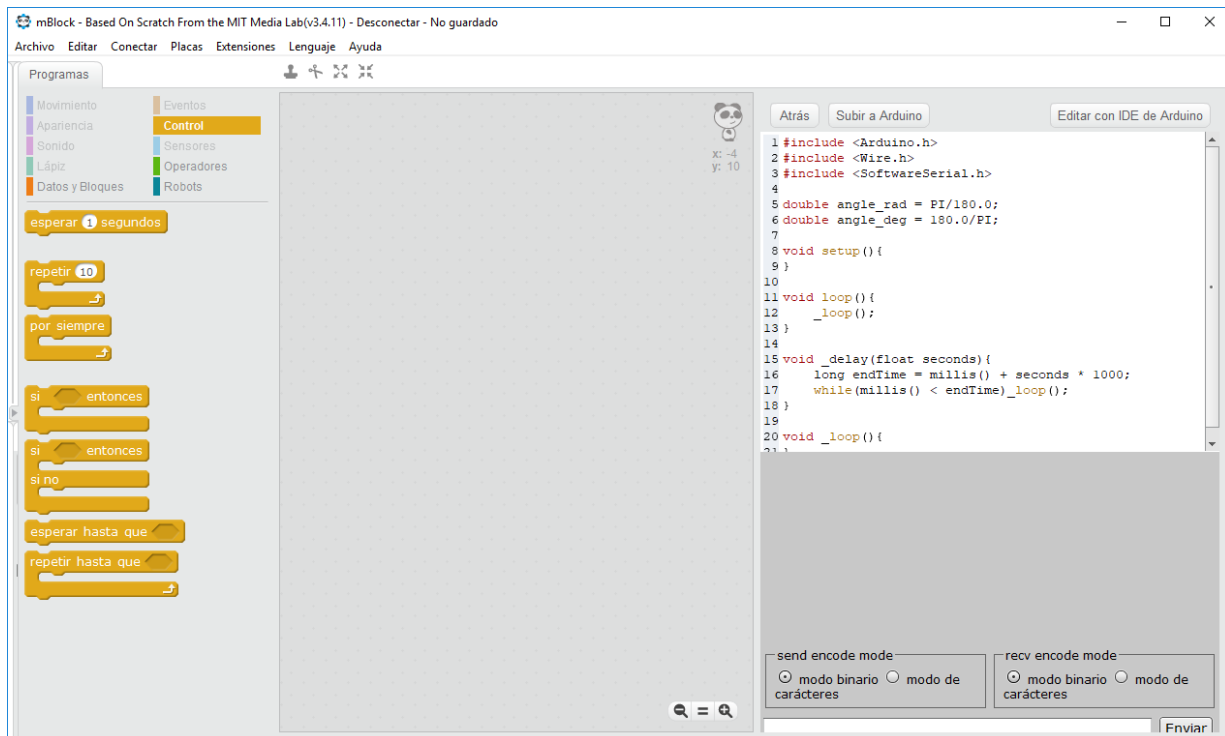
Cuando abrimos mBlock3, veremos una pantalla como la siguiente:



Ú æ! æÁ] ! [* ! æ{ æÁ ~ } Á] ! [^ ^ & c [Á å ^ Á Œ! å ~ ã } [Á & [} Á { Ó | [& \ H A 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 está 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 usará bloques de las categorías de Control. Cuando seleccionamos una de estas categorías, se pueden visualizar todos los bloques que pertenecen a ese grupo.



Para poder encender el radiador eléctrico debemos accionar el módulo relé. Realizaremos una prueba de encendido y apagado intermitente cada 5 segundos para verificar que todo funcione de forma correcta.



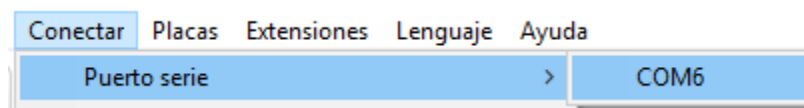
una corriente de 5V para el relé permita el paso de corriente para que nuestro radiador eléctrico se encienda.

que no permita el paso de corriente y el radiador eléctrico se apague.

Paso 3 - Subir el código a la placa Arduino

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

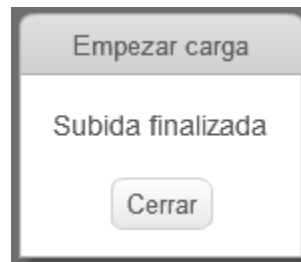
1. Conectar la placa Arduino a la entrada USB.
2. Seleccionar el puerto serie al que está conectada la placa.
3. Seleccionar el puerto serie al que está conectada la placa.



4. Clickear el botón

Subir a Arduino

Al terminar de subir nuestro código, veremos este mensaje



Paso 4 - Programar el tiempo de encendido y apagado

Para poder ahorrar energía, es necesario calcular el tiempo que el radiador permanecerá encendido. Esto va a depender de muchos factores, como el tamaño de la habitación y la temperatura exterior.

A modo de ejemplo, dejaremos el radiador encendido durante 10 minutos (600 segundos) y lo apagaremos durante 15 minutos (900 segundos). Los bloques nos quedará de la siguiente manera.



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(3,OUTPUT);
```

```

}

void loop(){
    digitalWrite( 3, 1);
    _delay( 600);
    digitalWrite( 3, 0);
    _delay( 900);
    _loop();
}

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

void _loop(){
}

```

Nivel intermedio

Martina descubrió que existe un modo para hacer aún más eficiente el uso del calefactor en términos energéticos y mantener la habitación dentro de un rango de temperaturas confortable: tiene que desarrollar un sistema para que el funcionamiento dispositivo se regule automáticamente en función de la temperatura ambiente.

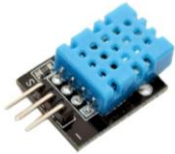
Necesita, también, agregarle un *display* donde se muestre la temperatura actual, así ella puede visualizar y configurar de forma sencilla la temperatura deseada y contar con que el sistema la mantendrá o menos constante.

En este nivel se propone agregar un sensor de temperatura y, en función de sus mediciones, se prenderá y se apagará el relé. Las temperaturas que determinarán si el sistema debe encenderse o apagarse serán controladas mediante un potenciómetro y visualizadas en un *display* LCD.

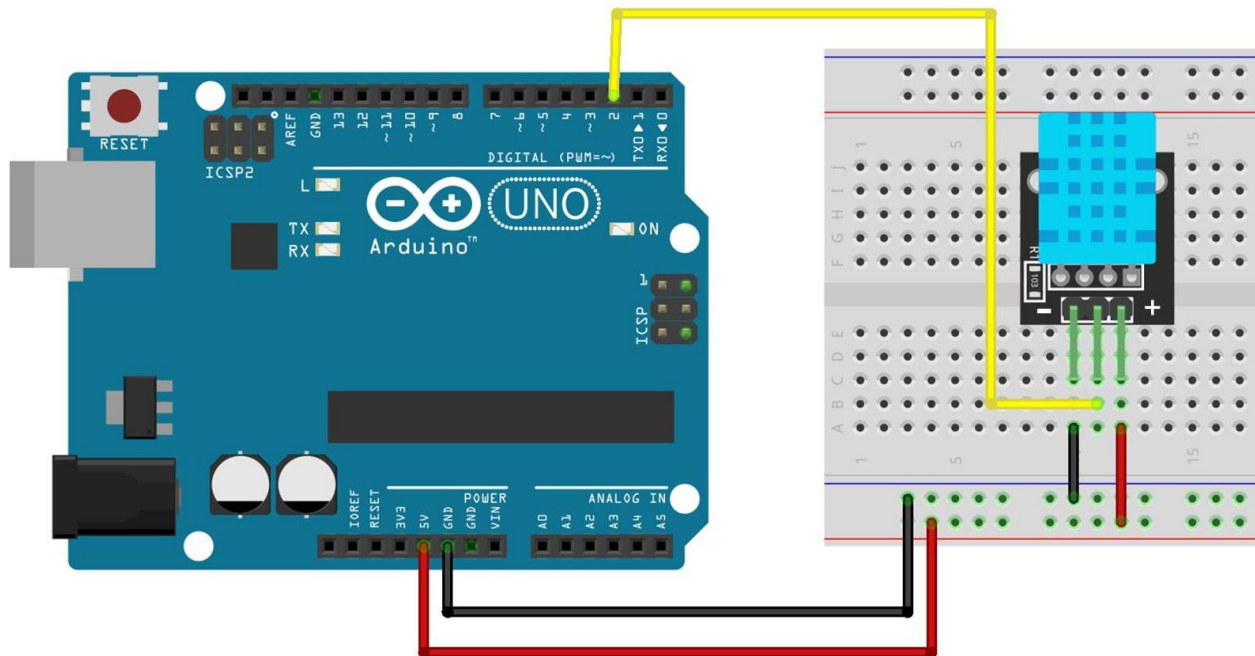
Paso 1 - Conectar el sensor de temperatura y humedad

Conectamos el sensor DHT11 como indica el siguiente esquema.

La señal de nuestro sensor de temperatura estará conectada al *pin 2* de la placa Arduino y también su respectiva alimentación (GND y 5V).

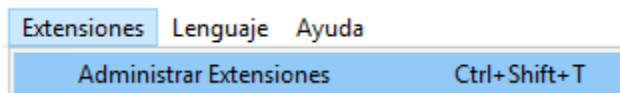


El DHT11 es un sensor que mide humedad y temperatura del aire. Este modelo de sensor posee electrónica interna que digitaliza los datos registrados y los reporta a el Arduino mediante una comunicación digital, por eso es que se conecta a un pin digital de la placa.



Paso 2 - Instalar la extensión del sensor de temperatura y humedad

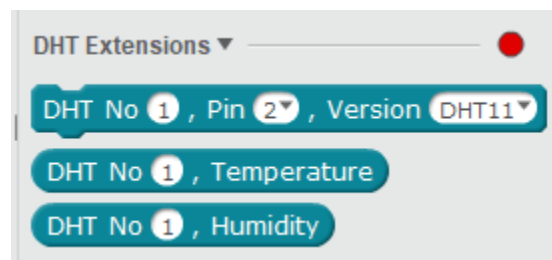
Para poder utilizar el sensor de temperatura y humedad, necesitamos instalar una extensión de mBlock3. Esto lo hacemos desde el menú



El sensor de temperatura y humedad que usaremos es el DHT11. Si en el buscador tipeamos

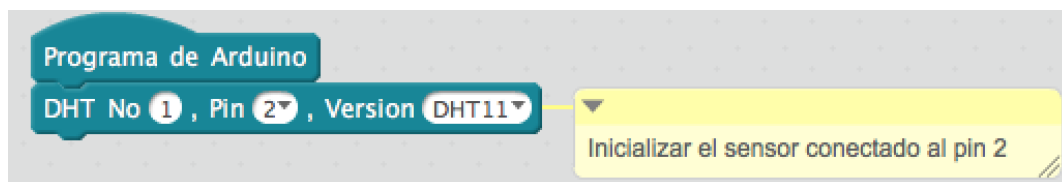


Bajamos la extensión, y tendremos nuevos bloques disponibles dentro de la categoría



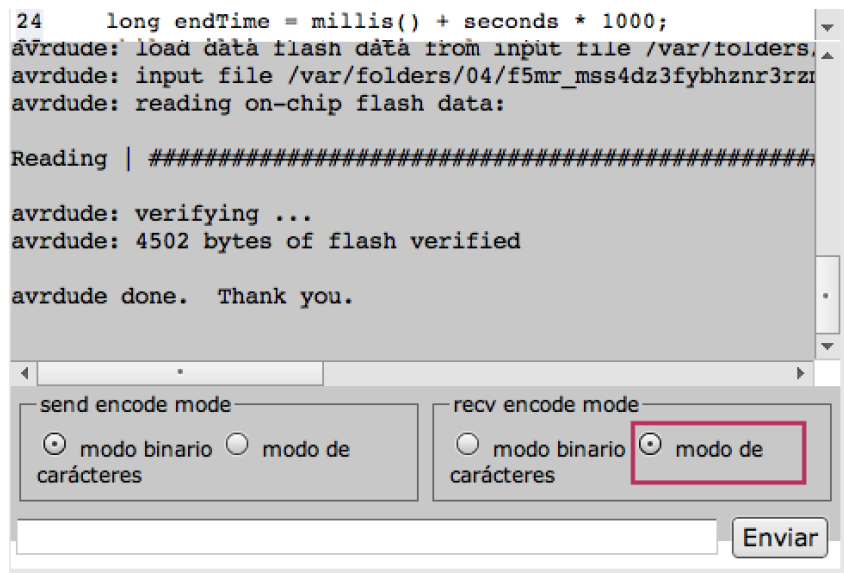
Paso 3 - Obtener la temperatura

Para utilizar el sensor, es necesario inicializarlo. En el menú de bloques, seleccionamos 'Inicializar el sensor conectado al pin 2'.



En función del enunciado del problema, solamente nos interesa el valor de la **temperatura** (aunque el sensor también mide la humedad).

Utilizaremos la consola para visualizar los datos que mide el sensor. La consola se encuentra en la parte inferior de la interfaz. Allí podremos ver la recepción de los datos, como se muestra en la imagen.

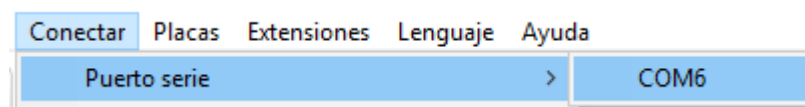


Para enviar los datos a la consola se `Serial.println(temperature);` agregamos el bloque que obtiene la temperatura. Es importante que este bloque se encuentre dentro del bucle `while(1)` de lectura.



Paso 4 - Activar el envío de datos a la consola

Una vez que está cargado nuestro programa debemos volver a conectar nuestra placa para que se envíen los datos a la consola.



Finalmente veremos la temperatura que mide el sensor.

```

24.00
24.00
24.00
24.
23:54:44.024 < 00
24.00
24.00
24.00
24.00
24.00
24.00
24.

```

send encode mode
☒ modo binario
☐ modo de caracteres

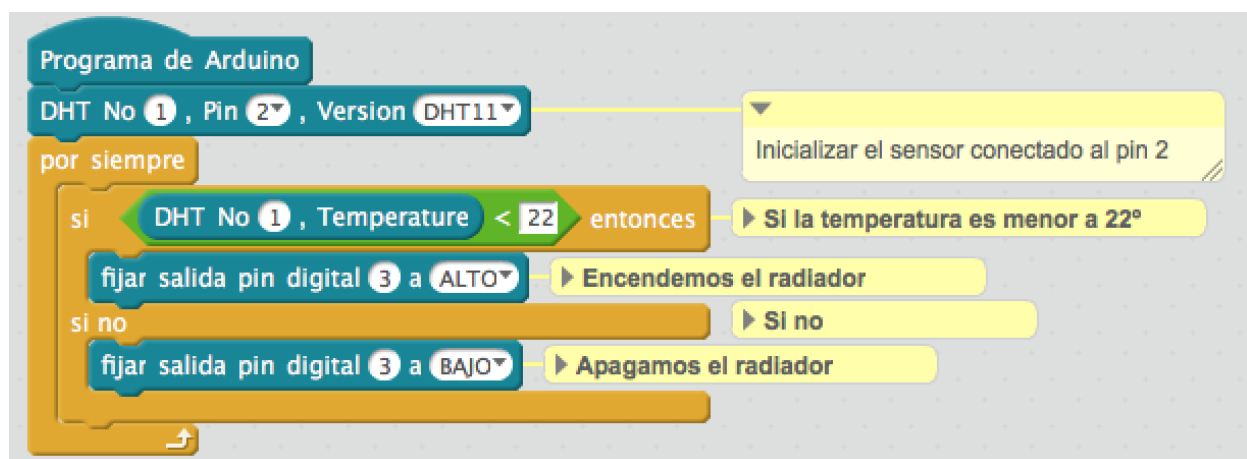
recv encode mode
☐ modo binario
☒ modo de caracteres

Enviar

Paso 5 - Encender el radiador según la temperatura

Ahora modificaremos nuestro programa para que el relé active solamente si la temperatura es menor a 22°C, si no, desactivaremos el relé A modo de ejemplo, se ha elegido esta temperatura, pero podrá ser otro valor.

Nuestro programa quedará como el siguiente.



Veremos que a la derecha se muestra el código escrito que corresponde a este programa.

```
#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(3,OUTPUT);
}

void loop(){
    if ((dht_1.readTemperature()) < ( 22)){
        digitalWrite( 3, 1);
    }else {
        digitalWrite( 3, 0);
    }
    _loop();
}

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

void _loop(){
}
```

Paso 6 - Conectar el módulo del *display* LCD

Para poder visualizar la temperatura, conectamos el módulo del *display* LCD (*shield* LCD) sobre la placa Arduino. Es importante manejar con cuidado las piezas para no forzarlas y asegurarse de que todos los *pines* estén bien conectados en el lugar que le corresponde a cada uno.



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 comunicaci3n, pantallas, sensores, interconexi3n, etc.

Paso 7 - Instalar la extensi3n del *display* LCD

Para programar el *display*, necesitamos instalar otra extensi3n como se realiz3ntes. En este

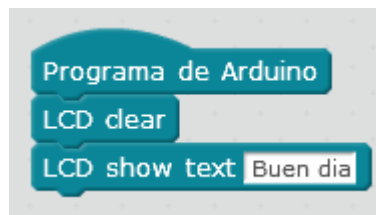


3n este momento, la interfaz de usuario de la IDE de Arduino se ve de la siguiente manera:



Paso 8 - 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 deberá ser similar al siguiente:



Nota: no utilizar tildes ni la letra Ñ con el display LCD

Veremos que a la derecha se muestra el código escrito que corresponde a este programa.

```
#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(){
}

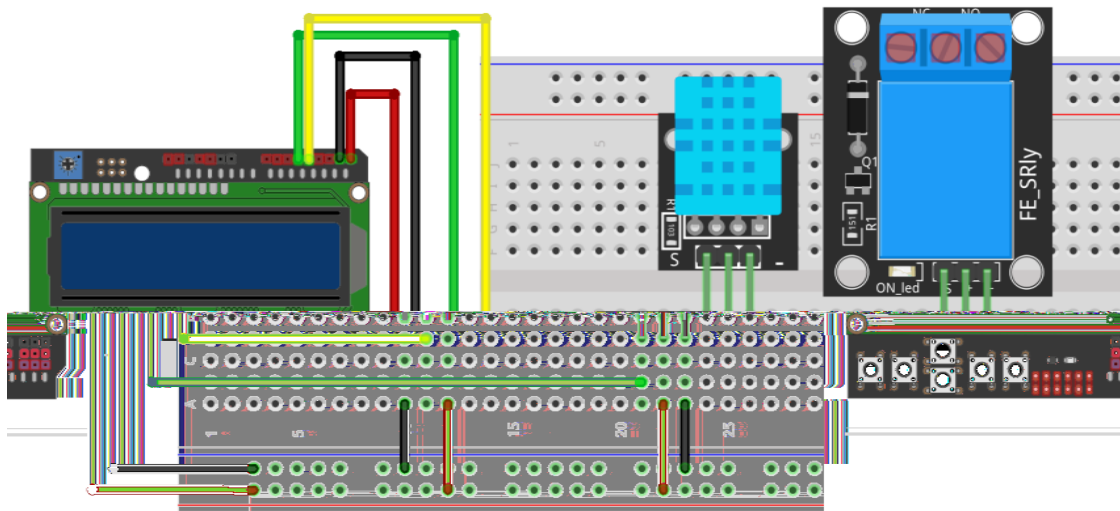
```

Si no se llega 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.

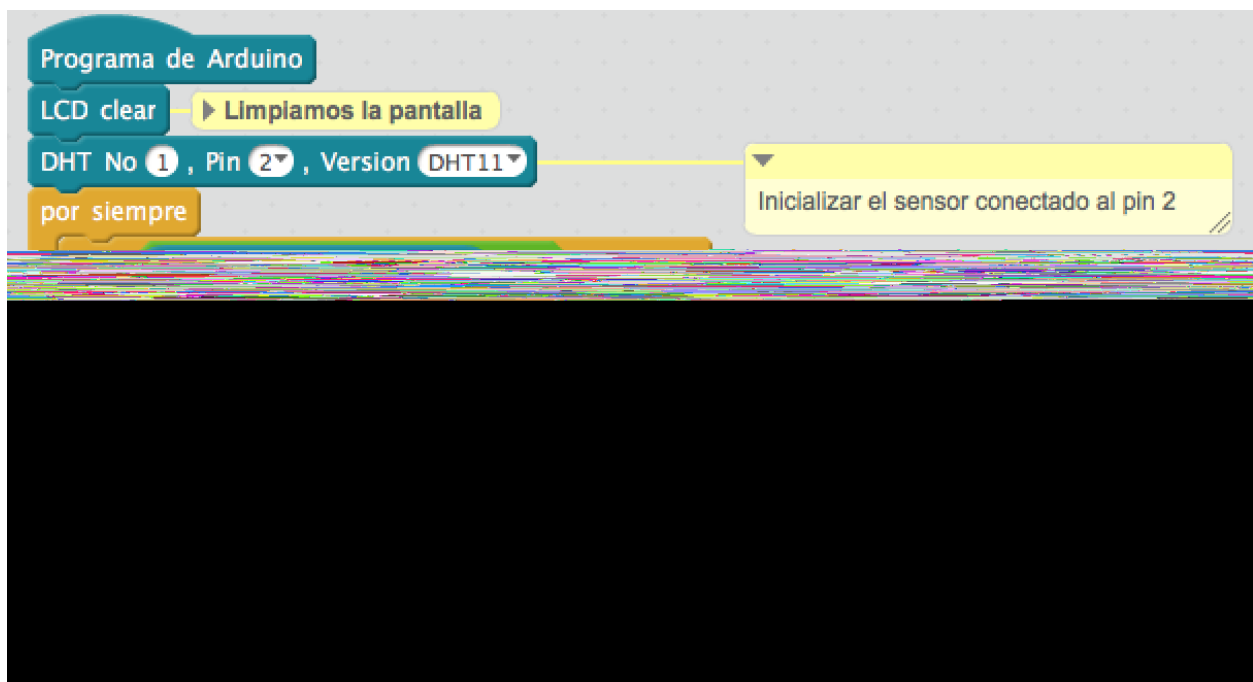
Paso 9 - Escribir la temperatura en el *display*.

Ahora, volveremos al programa anterior, conectamos nuevamente el sensor de temperatura y humedad al *pin* digital 2 y el relé al *pin* digital 3.

Deberá quedarnos como indica el siguiente es quema.



Agregaremos al código anterior (que encendía y apagaba el radiador) el bloque que permite visualizar la temperatura en el display. Quedará de la siguiente manera.



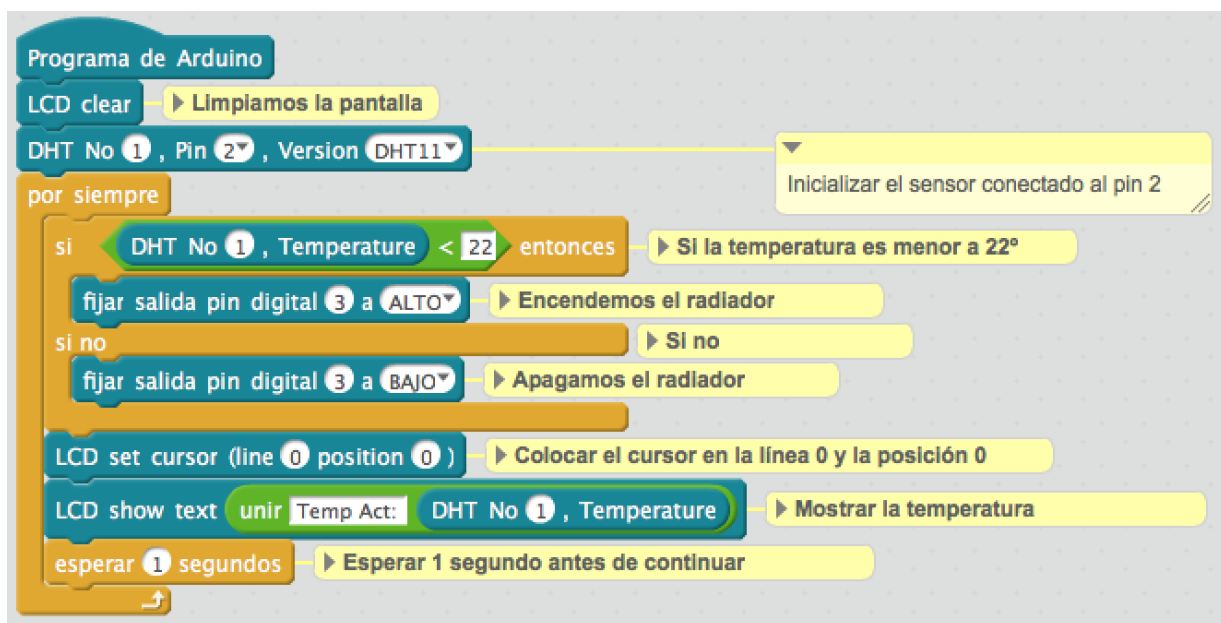
Paso 10 - Incorporar un texto en el *display*

En el *display*, mostraremos un texto antes de la temperatura actual que nos permite identificar a

bloque de la temperatura.

unir Temp Act: DHT No 1 , Temperature

Los bloques nos quedará de la siguiente manera.



Veremos que a la derecha se muestra el código escrito que corresponde a este programa.

```
#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(3,OUTPUT);
}

void loop(){
    if ((dht_1.readTemperature()) < ( 22)){
        digitalWrite( 3, 1);
    }else {
        digitalWrite( 3, 0);
    }
    lcd.setCursor( 0, 0);
    lcd.print(String( "Temp Act: " )+dht_1.readTemperature());
    _delay( 1);
    _loop();
}

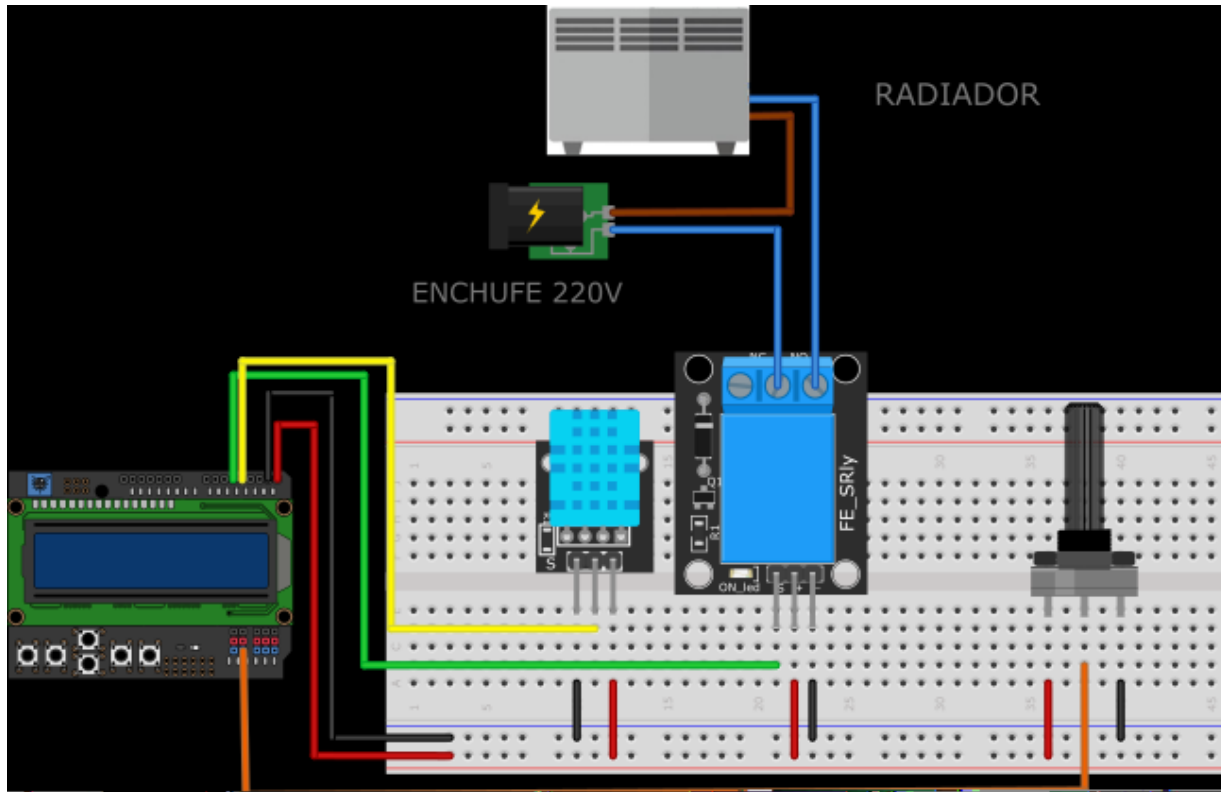
void _delay( float seconds){
    long endTime = millis() + seconds * 1000;
    while (millis() < endTime)_loop();
}

void _loop(){
}

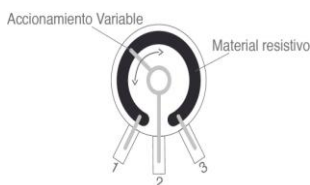
```

Paso 11 - Conectar un potenciómetro

Para cambiar la temperatura a la que se prenderá el radiador, incorporaremos un potenciómetro al *pin* analógico 1. Con el potenciómetro estableceremos de manera manual la temperatura que deseamos para la habitación.



Las entradas analógicas de la placa Arduino que van desde el *pin* A0 al A5 nos permiten saber el voltaje de la entrada con una resolución que va de **0** cuando hay **0V** a **1023** cuando hay **5V**. Si, por ejemplo, el voltaje en la entrada es de **1.25V**, estos *pines* leen un valor intermedio de **255V**.



Un potenciómetro es un resistor cuyo valor de resistencia se puede modificar y controlar de forma manual. En muchos dispositivos eléctricos los potenciómetros son utilizados para regular el nivel de tensión. Por ejemplo, en un parlante el potenciómetro se puede utilizar para ajustar el volumen; así como en un monitor se puede utilizar para controlar el brillo.

Paso 12 - Crear variables

En primer lugar, debemos obtener el valor de lectura del *pin* analógico en el cual conectamos el potenciómetro. Para obtener y guardar este dato, creamos una nueva variable.



␣ [{ à ! æ { [• Á ^ • c æ Á ç æ ! ã æ à | ^ Á & [{ [Á % V ^ {] ^ ! æ c ~ ! æ + È Á Ù ^ Á]
recordar que en las líneas de código las palabras no pueden llevar tilde ni usar la letra Ñ.

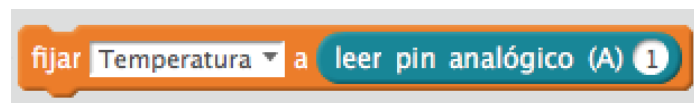
Variable nueva

Nombre de la variable:

☒ Para todos los objetos ☐ Sólo para éste objeto

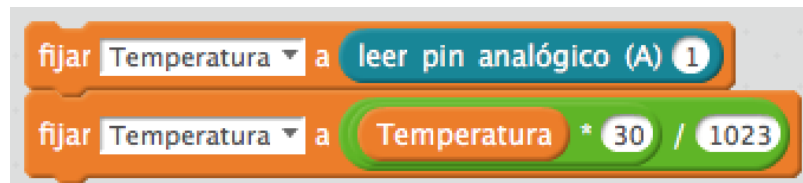
Una variable es un espacio en la memoria que nos permite guardar un dato para luego ser leído en otra instancia del programa. Es importante tener en cuenta, al momento de crear una variable, que su nombre no puede comenzar con un número ni contener espacios.

A esta variable le asignaremos el valor del *pin* analógico que corresponda.

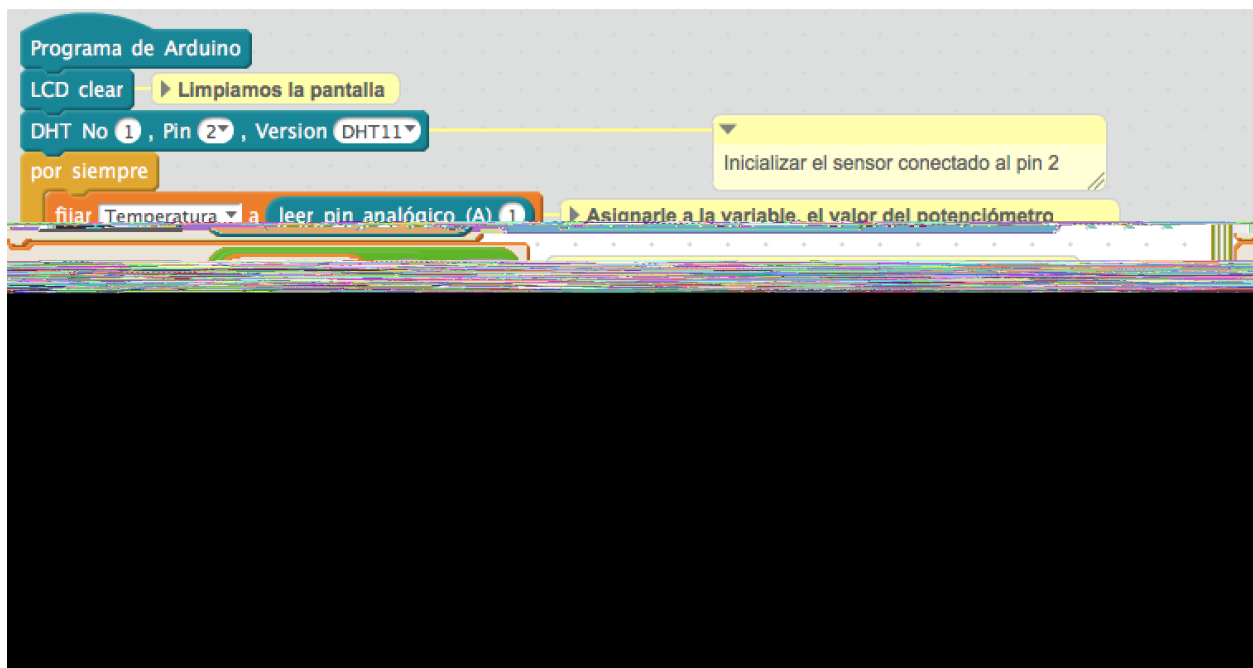


Paso 13 - Adaptar los valores

Hay que tener en cuenta que la lectura del *pin* arroja valores entre 0 y 1023, pero la temperatura debe estar, por ejemplo, a 0°C cuando el potenciómetro valga 0 y a 30°C cuando el potenciómetro valga 1023. Por lo tanto, necesitamos adaptar el valor leído a una escala que varíe entre 0 y 30. Esto puede programarse. Para esto, debemos programar nuestro sistema para que multiplique el valor que arroja el *pin* por 30 y luego lo divida por 1023.

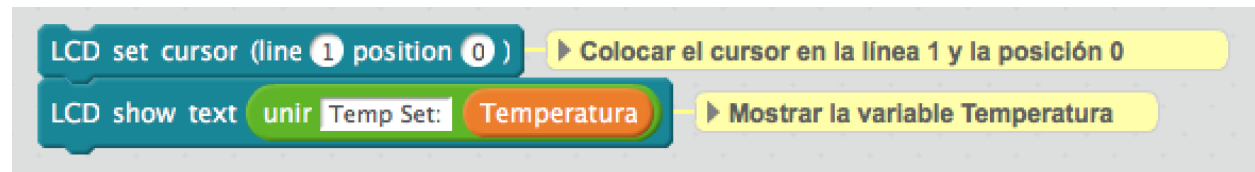


A continuación, reemplazamos el número 22 (que correspondía a la temperatura de encendido) por el valor de la temperatura de encendido. Los bloques nos quedarán de la siguiente manera:

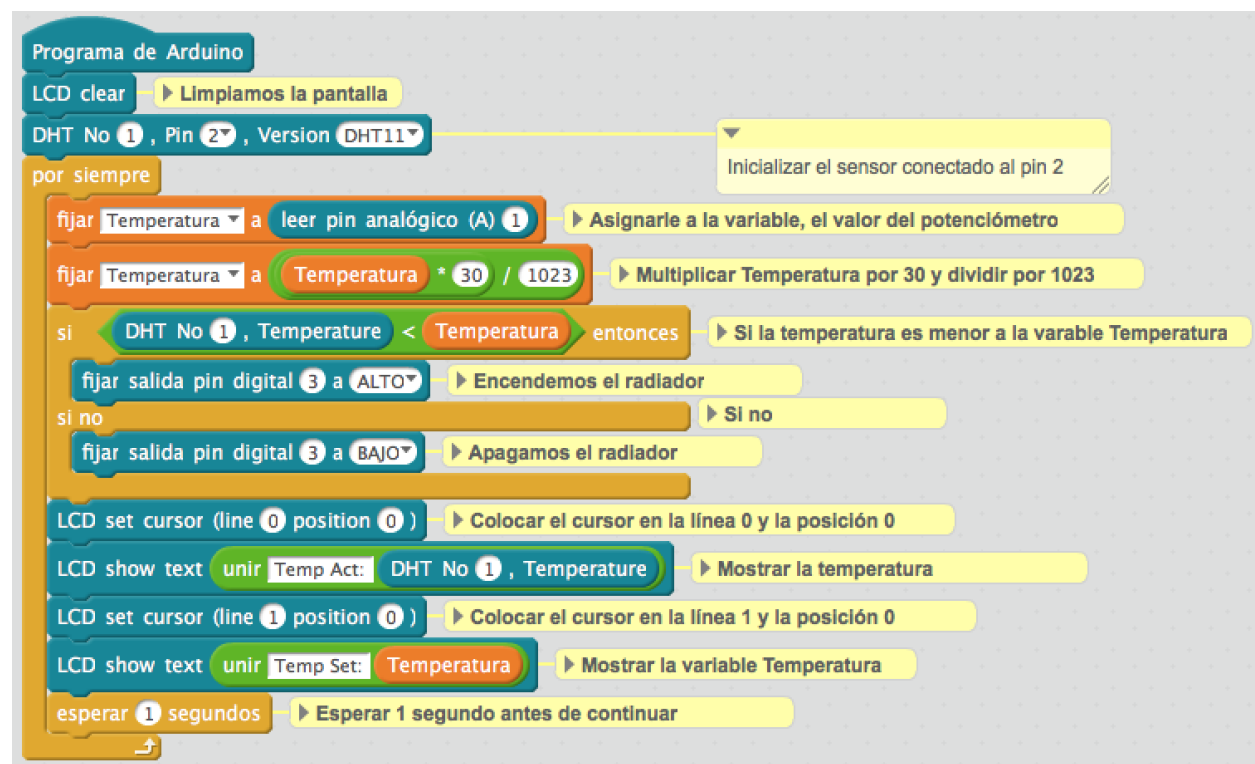


Paso 14 - Visualizar la temperatura ideal

Es necesario que el *display* sepa a qué temperatura estamos configurando nuestra calefacción. Entonces, agregamos los siguientes bloques.



Finalmente, el código completo nos quedará de la siguiente manera.



Veremos que el código escrito que corresponde a este programa es el 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;
double Temperatura;
LiquidCrystal lcd (8, 9, 4, 5, 6, 7);
DHT dht_1(2, 11);

void setup(){
    lcd.begin( 16, 2);
    lcd.clear();
    pinMode(A0+1,INPUT);
    pinMode(3,OUTPUT);
}

void loop(){
    Temperatura = analogRead(A0+ 1);
    Temperatura = ((Temperatura) * (30) / ( 1023);
    if ((dht_1.readTemperature()) < (Temperatura)){
        digitalWrite( 3, 1);
    }else {
        digitalWrite( 3, 0);
    }
    lcd.setCursor( 0, 0);
    lcd.print(String( "Temp Act: " )+dht_1.readTemperature());
    lcd.setCursor( 0, 1);
    lcd.print(String( "Temp Set: " )+Temperatura);
    _delay( 1);
    _loop();
}

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

void _loop(){
}

```

Nivel avanzado

La ciudad donde vive Martina tiene temperaturas muy frías en invierno. Por esta razón, en algunas ocasiones, debe dejar el calefactor encendido incluso cuando no se encuentra en su casa. Ella desea monitorear el funcionamiento del calefactor en estos momentos para asegurarse de que no esté consumiendo más energía de la necesaria. Para poder hacerlo, necesita instalar un sistema que le informe, utilizando internet, la temperatura del ambiente en todo momento.

Se propone agregar IoT para relevar los datos del estado actual de la temperatura. La información recopilada nos permite, por ejemplo, monitorear el funcionamiento del sistema de calefacción y el ahorro de energía, y elaborar estadísticas. Aprovechando que el sensor utilizado también mide la humedad, agregaremos este dato a nuestro registro.

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

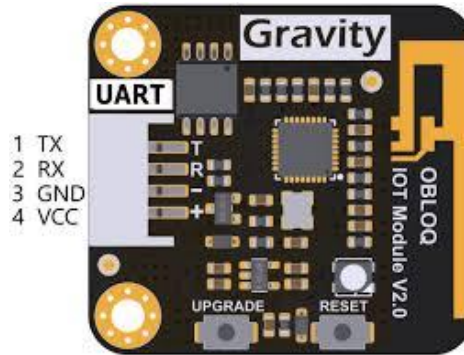
Internet de las Cosas (en inglés *Internet of Things*, abreviado IoT) es un concepto que refiere a la interconexión digital de objetos cotidianos con internet. Esta interconexión puede tener diversas funciones. Por ejemplo, puede utilizarse para monitorear la temperatura de un ambiente, enviando los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

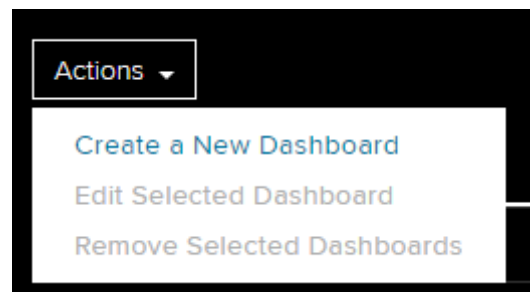
El módulo UART OBLOQ es un dispositivo WiFi a serie pensado para desarrolladores no profesionales. Permite enviar y recibir datos mediante los protocolos HTTP y MQTT.



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



Creamos un nombre y una descripción.

Create a new Dashboard

Name


Sistema de calefacción

Description

Diseño de un menu para control de temperatura y humedad del ambiente

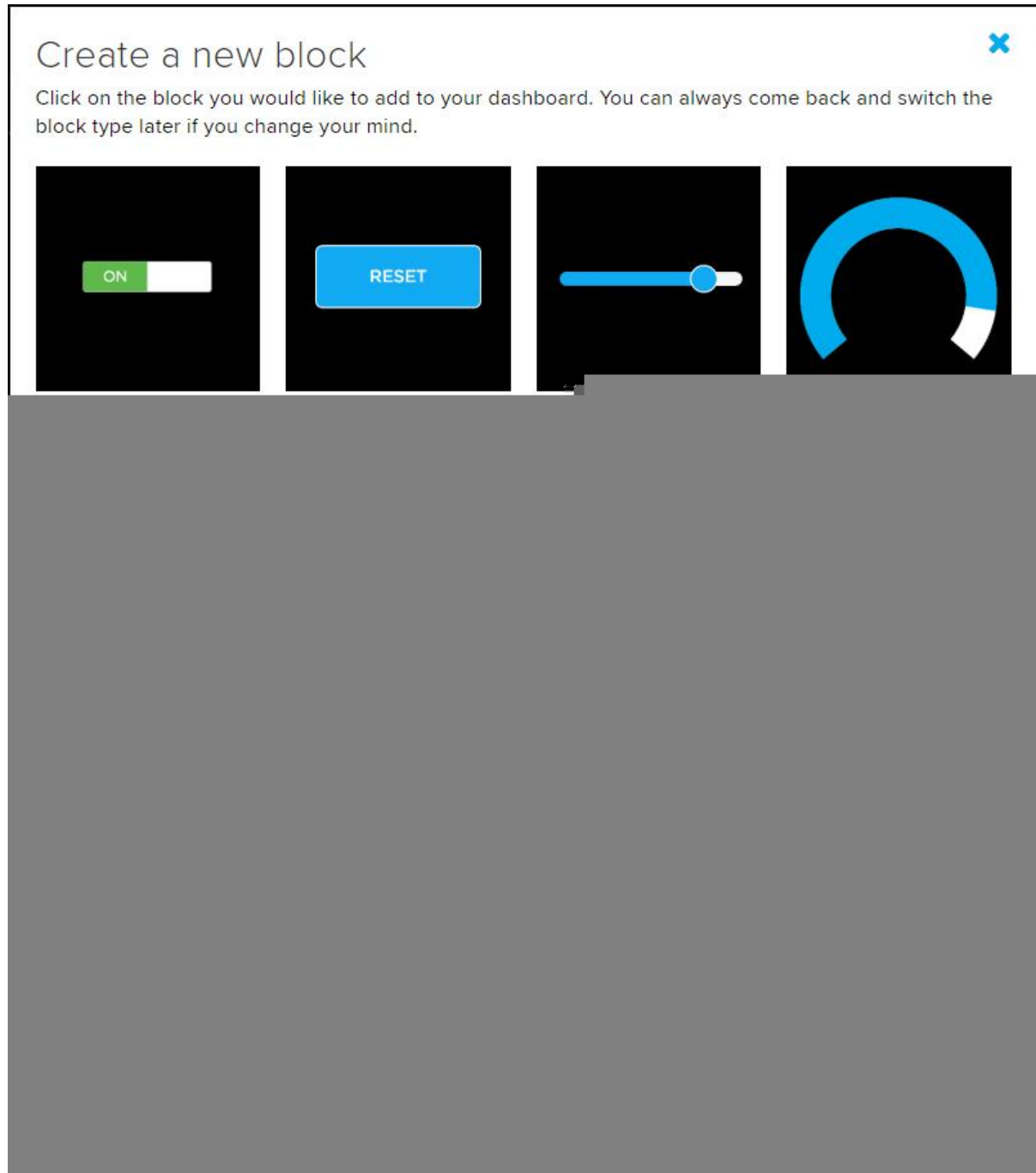
Cancel

Create

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



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



Para nuestro sistema de calefacción, podríamos ubicar dos *Line Chart* (gráficos de línea) aprovechando que nuestro sensor DHT11 también mide la humedad. De esta forma podremos visualizar el historial de cambios de temperatura y humedad de nuestro ambiente.



Ô ~ æ } á [Á æ * ! ^ * æ { [• Á ~ } Á & [] c ! [| Á æ æ Á É Á ^ | Á á ^ à ^ { [• Á æ • [&

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` y publicaremos, desde nuestro dispositivo, la temperatura que lea el sensor DHT11.

Choose up to 5 feeds

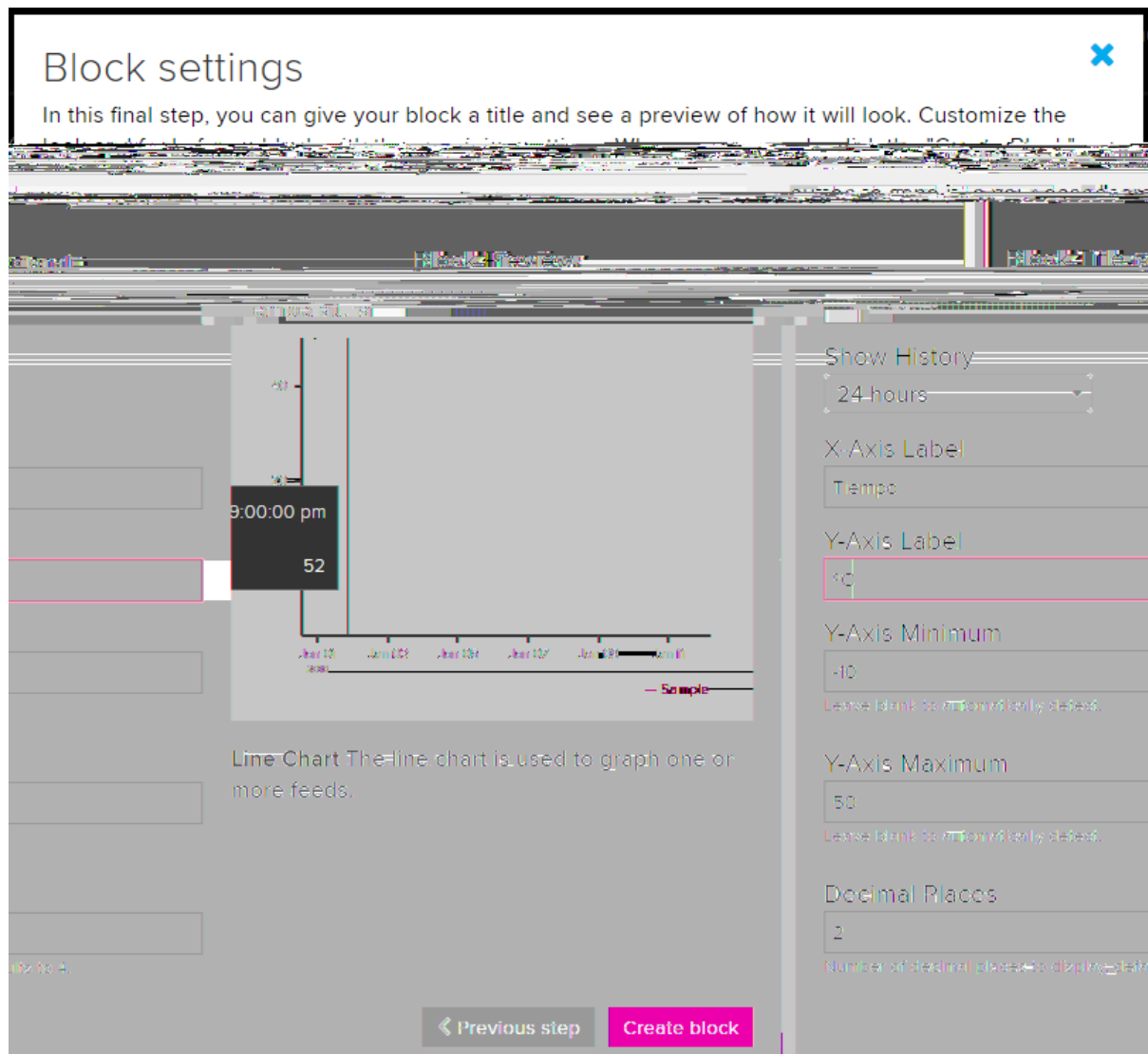
Line Chart: The line chart is used to graph one or more feeds.

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

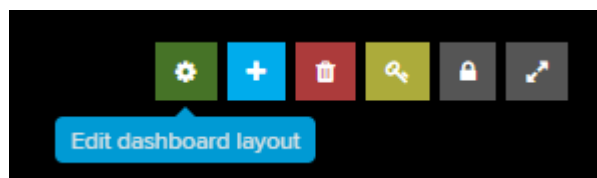
Group / Feed	Last value	Recorded
<input checked="" type="checkbox"/> temperatura		a few seconds ago 1 of 5

Š ~ ^ * [Á á ^ Á & ! ^ æ ! [É Á æ æ Á Ç Á æ [á & \ á ^ } á Á % c ^ D Á] æ ! æ Á & [] control y completamos los campos como se ve en la imagen a continuación.

P æ & ^ { [• Á & | ã & \ Á ^ } Á % Ô ! ^ æ c ^ Á à [& \ + Á Ç & ! ^ æ ! Á à [~ ~ ^ D Á] æ



Ú[â ^ { [• Á { [â ã ~ ã & æ! Á ^ | Á c æ{ æfi [Á ^ Á | æÁ ~ à ã & æ& ã 5 } Á â ^ Á | [• & [} ~ ã * ~ | æ& ã 5 } + È



Repetimos este procedimiento para el *feed*%@~ { ^ â æâ + È Á Ö^ à ^ | ð æ{ [• Á ç ã • ~ æ| ã : lo siguiente ya que aún no hay datos publicados.

/ Dashboards / Sistema de calefacción

Temperatura

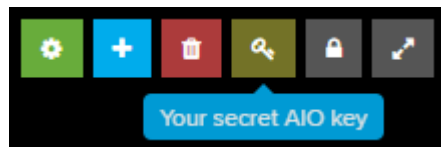


Humedad



Una vez realizado el Panel, publicaremos los datos obtenidos con los sensores para poder monitorearlos de manera remota.

Antes de salir, debemos copiar las credenciales de acceso para poder publicar en nuestros feeds



YOUR AIO KEY

our existing programs and
ew key.

727c117

REGENERATE AIO KEY

```
'
244b6b049abb07727c117"
```

care as your Adafruit username and password. P
AIOkey can view all of your data, create now for
manipulate your active feeds.

If you need to regenerate a new AIO key, all of yc
scripts will need to be manually changed to the n

Username

usuario_aio

Active Key

1234cfdd29a244b6b049abb07

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 deberá 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.

Reemplazamos el relé por un LED conectado al pin 13 para probar que nuestro programa funciona correctamente y evitarnos el conexionado a 220v. La programación del LED es exactamente la misma que la del relé por lo que una vez que corroboramos que funciona de forma correcta podremos conectar el relé como se mostraba en el circuito anterior.



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 ^ } c [! } [Á } æ c ã ç [Á á ^ Á Œ! á ~ ã } [Á ~ ~ ^ Á | | æ { æ { [• Á %œ! á ~ ã } [Á desarrollo integrado).

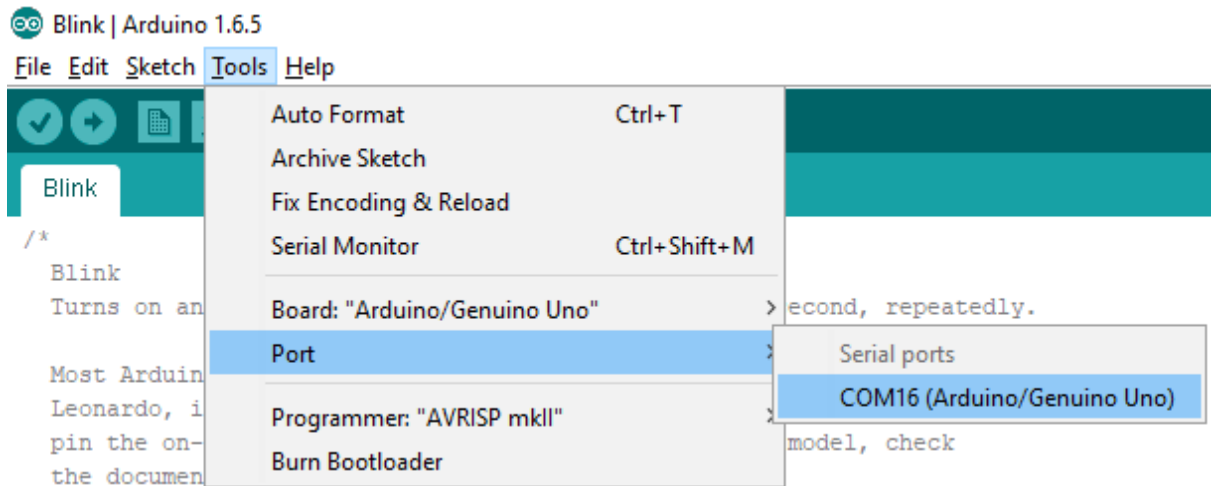
Para ello descarga el Arduino IDE desde el siguiente enlace y luego procede con la instalación del mismo: www.enfoco.net.ar/sd

1. Un bloque de código que se ejecuta por única vez al inicializarse el dispositivo. Este se define en un archivo de configuración y se llama `setup()`.

2. El bloque de código está contenido dentro de la función `loop()` y `delay()`.

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. Que esté conectada la placa.



3. Subir el código a la placa.

Sabremos que nuestro código subió correctamente si en la barra de estado se escribe `COM16 (Arduino/Genuino Uno)`.

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, 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()` se detiene todo el procesamiento hasta que se cumpla el tiempo indicado. En otras palabras, cuando el programa entra al `delay()` se detiene todo el procesamiento hasta que se cumpla el tiempo indicado.

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 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, transcurrido desde el inicio del sistema, se calcula la diferencia entre el valor de *millis* actual y el valor de *millis* anterior. Si esta diferencia es mayor o igual a 1000, 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

se le da la vuelta a (transcurre otro segundo).

En pocas palabras, el LED se mantendrá encendido cuando la temperatura supere los 22°C en lugar de parpadear. Nuestro programa queda como sigue.

```

// Incluimos las librerías 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 22°C,
        if (dht.readTemperature() > 22){
            // encendemos el LED.
            digitalWrite( 13, 1);
        }else {
            // Si no, apagamos el LED.
            digitalWrite( 13, 0);
        }

        // Imprimimos la temperatura.
        lcd.setCursor( 0, 0);
        lcd.print( "Temp Act: " );
        lcd.print(dht.readTemperature());

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

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

}

```

æ@[! æÁ ~ ~ ^ Á c ^ } ^ { [• Á ~ } Á] ! [* ! æ{ æÁ & [} Á ~ } Á & 5 å ã * [Á %} [Á à
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.

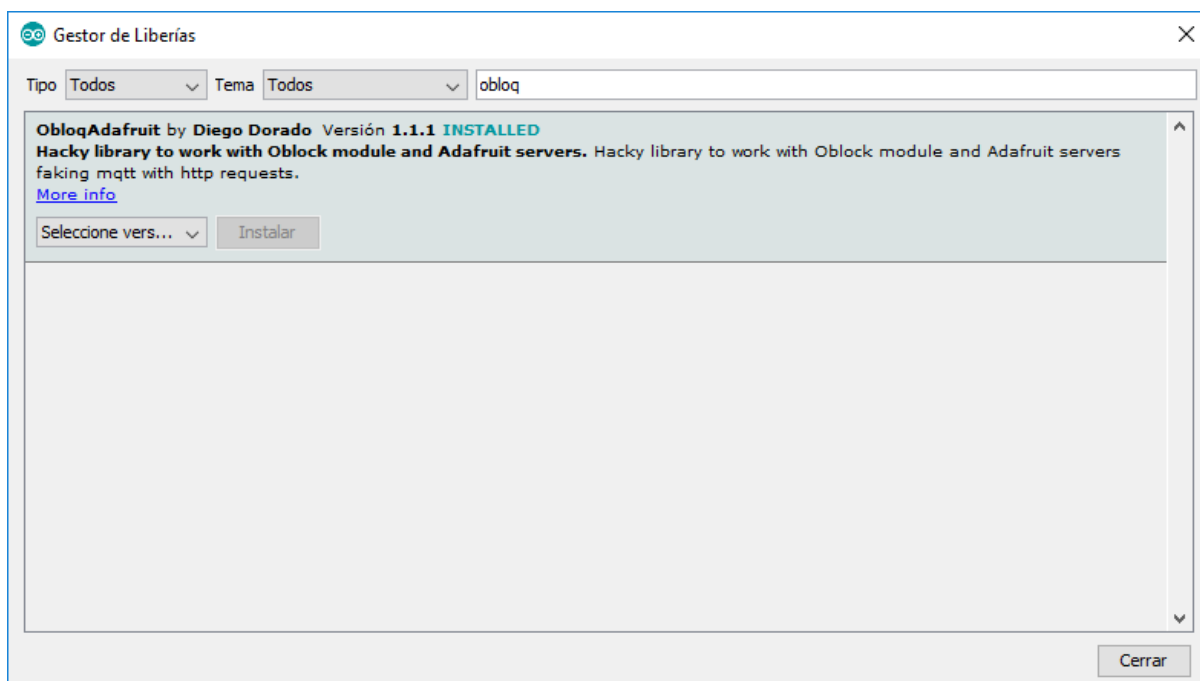


Fig. 27

En general las librerías traen códigos de ejemplo como referencia. Abrimos el ejemplo

Debemos reemplazar el SSID de la WiFi, su password, el IO_USERNAME e IO_KEY por los que copiamos de Adafruit. También modificaremos `softSerial(10,11)` por `softSerial(11,12)` ya que así es como lo conectamos en nuestra placa.

```
#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);
}
```

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" , 22); // Informar que la temperatura es de 22°C
```

Vamos a modificar nuestro programa para incorporar el envío de datos de temperatura y humedad a Adafruit. Nuestro programa con IoT queda como sigue.

```
// 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_adaf ruit"
#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 lo siguiente cada un segundo.
if (millisActual - millisAnterior >= 1000) {

    // Guardamos en la variable temperatura
    // lo que lee el sensor DHT11.
    int temperatura = dht.readTemperature();

    // Si la temperatura es mayor a 22 °C,
    if (temperatura > 22){
        // encendemos el LED,
        digitalWrite( 13, 1);
    }else {
        // si no, apagamos el LED.
        digitalWrite( 13, 0);
    }

    // Imprimimos temperatura y humedad
    lcd.setCursor( 0, 0);
    lcd.print( "Temp Act: " );
    lcd.print(temperatura);

    // Publicar en Adafruit 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, es posible extenderlo si se quiere continuar. Estas son algunas opciones sugeridas:

Controlar el encendido y apagado del radiador a través de IoT.

Programar un sistema de humidificación o deshumidificación del ambiente según la variación de la humedad.

Los proyectos proponen la construcción de dispositivos para el uso eficiente de la energía.

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, al 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.).

Glosario Técnico

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 de programación 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.

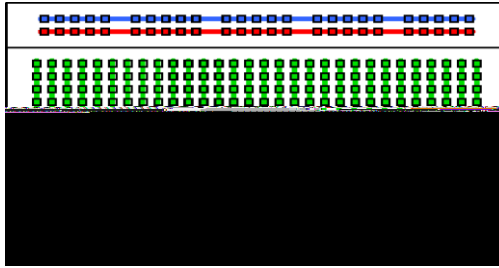
Shield: Placas de circuitos que se monta encima de la placa Arduino para expandir sus funcionalidades. Existen shields para otros tipos de microcontroladores y computadoras embebidas (Arduino Micro, RaspberryPi, etc). En general un shield sirve para ser utilizado con un único modelo de placa, en este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

Relé: Dispositivo electromagnético que funciona como un interruptor controlado por un circuito eléctrico. Por medio de un electroimán se acciona uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Se utiliza comúnmente para aislar eléctricamente dos circuitos así como también permitir controlar con bajo voltaje a dispositivos que utilizan mayor voltaje. Arduino trabaja con señales de 5 V que mediante un relé permiten controlar la activación o desactivación de dispositivos de 220 V.

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.

Potenciómetro: Un potenciómetro es un resistor cuyo valor de resistencia variable que se controla de forma manual o mecánica. Sirven para trabajar con bajos niveles de potencia, a modo de señal de control.

Los potenciómetros suelen tener una resistencia fija del valor especificado y un cursor que permite pararse en algún punto intermedio de esta resistencia. Para leer la posición del potenciómetro generalmente conectaremos las dos patas de la resistencia fija (los extremos) a VCC y GND. De esta forma tendremos en la pata central (cursor) un valor de tensión que representa la posición actual de la perilla.

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 Barragá
Omar Leandro Bobrow
Alejandro Cesar Cáeres
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
Nicolá 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 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 acompaña 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!