

Proyecto de Invernadero para Cultivo de Orquídeas Basado en la Plataforma Web Thingspeak Utilizando la Plataforma Electrónica Arduino

1. Plataforma WEB Thingspeak

La Plataforma web que se utiliza se puede acceder a través de <https://www.thingspeak.com>.

ThingSpeak nos permite interactuar con los dispositivos utilizando tecnologías Web estándar. ThingSpeak funciona a través de un servicio de host gratuito, o bien, en servidores personales, esta última opción no ha sido abordada, sino que se ha estado trabajando siempre en la nube.

La API (Application Program Interface) permite a los programas de software poder ser combinados, de manera que se consiguen mejores aplicaciones, a través de diferentes tipos de programas y ventanas que muestran sus resultados.

Como Thingspeak está en colaboración con Mathworks <https://en.wikipedia.org/wiki/MathWorks> que es la empresa de Matlab y Simulink entre otros, es por ello que la página de Thingspeak acepta implementar dentro de su amplio espectro de ventanas, código de Matlab para el procesamiento de datos.

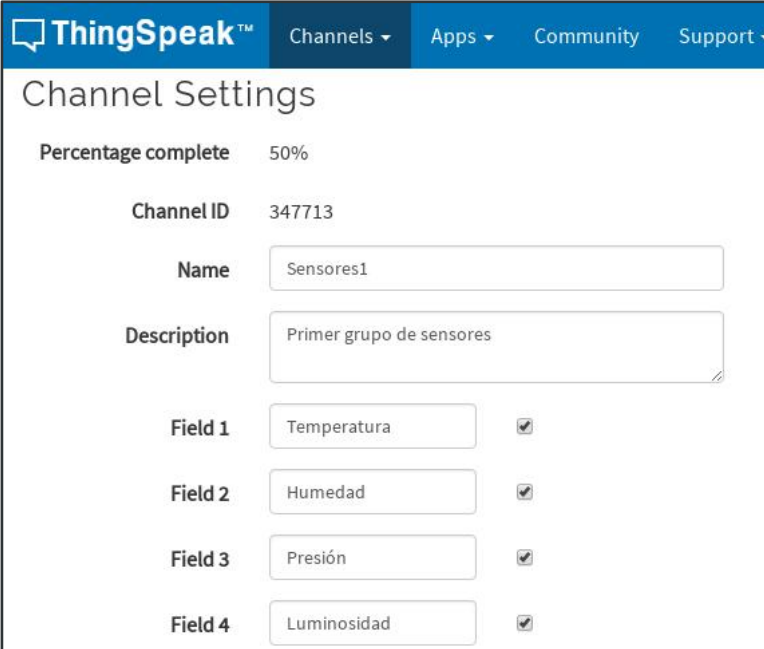
La API de ThingSpeak siempre trabaja con datos, esa es su gran especialidad. Es una API abierta para el Internet de las Cosas que permite recopilar, almacenar, analizar, visualizar y actuar sobre la información recogida en sensores y dispositivos como aplicaciones web y móviles, redes sociales como Twitter, soluciones de mensajería, VoIP y nube como Twilio, hardware de código abierto como Arduino, Raspberry Pi o BeagleBone (los reyes del Internet de las Cosas y la robótica) o con lenguajes de cálculo computacional como MATLAB... ThingSpeak es una API conocida entre los desarrolladores y dispone ya de una gran comunidad.

ThingSpeak API funciona siempre con canales, los cuales contienen los campos de datos, ubicación y estado. Para empezar a trabajar con esta interfaz es necesario crear un canal, donde se recopilará la información de dispositivos y aplicaciones, datos que posteriormente se pueden analizar y visualizar en gráficos. En nuestro caso, el canal creado es el [ubensors](https://thingspeak.com/channels/347713), que se puede ver públicamente en <https://thingspeak.com/channels/347713>. El proceso con la API siempre es el mismo.

Una vez creado el canal, se crean los distintos campos (o fields por su nombre en idioma inglés) los cuales serán los datos que se podrán procesar en Matlab y mostrarse en los gráficos. La ruta del proyecto con la API siempre será como la siguiente *thingspeak.com/channels/CHANNEL_ID/FIELD_ID* sustituyendo los campos *CHANNEL_ID* y *FIELD_ID* por los datos del canal recién abierto.

Para nuestro caso, tenemos un solo canal **Sansores1**, cuyo Channel ID es 347713. Adentro de este canal, se encuentran 4 campos de datos que muestran los datos relevados por los sensores, dichos campos son:

- Campo 1 : Temperatura
- Campo 2 : Humedad
- Campo 3 : Presión
- Campo 4 : Luminosidad



The image shows the 'Channel Settings' page for a ThingSpeak channel. The header includes the ThingSpeak logo and navigation links: Channels, Apps, Community, and Support. The main content area displays the following information:

- Percentage complete:** 50%
- Channel ID:** 347713
- Name:** Sensores1
- Description:** Primer grupo de sensores
- Field 1:** Temperatura (checked)
- Field 2:** Humedad (checked)
- Field 3:** Presión (checked)
- Field 4:** Luminosidad (checked)

Figura 1: Datos del Canal

Los 4 campos son los grupos de datos como hemos dicho antes. Ahora estos grupos de datos pueden ser mostrados en gráficos, en función del tiempo, que realiza Thingspeak. A medida que ingresa un dato, se le agrega automáticamente una etiqueta de tiempo que se utilizará para graficar.

Además de esto, los datos pueden ser tomados de los gráficos para realizar un procesamiento con la herramienta que se denomina Matlab Analysis, que como su nombre lo indica, puede procesar los datos mediante un código de Matlab.

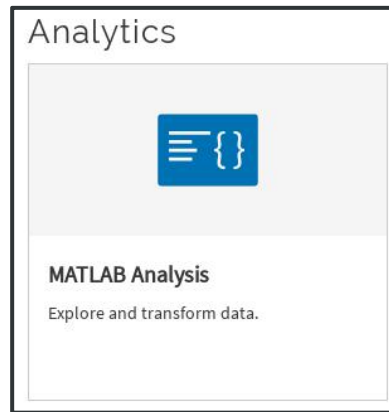


Figura 2: Icono del Matlab Analysis en Apps

Los datos cuando se ingresan al servidor web, se ingresan en un vector el cual va acompañado del Channel_ID (número de canal), Write_API_Key (clave para poder escribir el canal) y un vector con los datos, cuya posición indica a qué campo corresponde ([Field1 Field2 Field3 Field4]).

De igual manera, cuando se requieren los datos se pueden solicitar:

- *La última entrada:* que devuelve un vector parecido al anterior con los últimos datos ingresados.
- *Los datos de un campo:* devuelve un vector con todos los datos que se encuentran almacenados en el campo.

De esta forma utilizamos el código de Matlab para realizar un análisis de los últimos datos ingresados, y así ejecutar un el procesamiento de los mismos.

a) El Matlab Analysis

Dentro de la aplicación del Matlab Analysis, se toman en consideración los requerimientos seteados (temperatura baja, temperatura alta, humedad baja, humedad alta, presión alta, presión baja, luminosidad alta, luminosidad baja) y mediante este análisis, el mismo Matlab Analysis publicará un TalkBack para que pueda ser levantado por el arduino haciendo las veces de comando u orden que se manda desde el servidor web al arduino.

El código de matlab es sencillo y en la versión gratuita de Thingspeak, se limita la cantidad de procesamiento, por ello es que la prueba es reducida y no se puede colocar códigos pretenciosos o extensos.

Código del Matlab Analysis

```
Temp=1; Hum=2; Pres=3; Ilum=4;  
dato=thingspeakRead(347713,'ReadKey','COH2SVB7L2Y85LIB','Fields',[Temp Hum  
Pres Ilum]);
```

```

COMMAND1="";
if ~isnan(dato(Temp))
    if dato(Temp) < 22
        COMMAND1 = 'Tbaja';
    elseif dato(Temp) > 26
        COMMAND1 = 'Talta';
    end
    response1=webwrite('https://api.thingspeak.com/talkbacks/18565/commands.json',
'api_key','EFWWA5DCID6FBYHO','command_string',COMMAND1)
end
COMMAND2="";
if ~isnan(dato(Hum))
    if dato(Hum) > 65
        COMMAND2 = 'Halta';
    elseif dato(Hum) < 35
        COMMAND2 = 'Hbaja';
    end
    response2=webwrite('https://api.thingspeak.com/talkbacks/18565/commands.json',
'api_key','EFWWA5DCID6FBYHO','command_string',COMMAND2)
end
COMMAND3="";
if ~isnan(dato(Pres))
    if dato(Pres) < 1023
        COMMAND3 = 'Pbaja';
    elseif dato(Pres) > 1044
        COMMAND3 = 'Palta';
    end
    response3=webwrite('https://api.thingspeak.com/talkbacks/18565/commands.json',
'api_key','EFWWA5DCID6FBYHO','command_string',COMMAND3)
end
COMMAND4="";
if ~isnan(dato(Ilum))
    if dato(Ilum) < 500
        COMMAND4 = 'Ibaja';
    elseif dato(Ilum) > 1000
        COMMAND4 = 'Ialta';
    end
    response4=webwrite('https://api.thingspeak.com/talkbacks/18565/commands.json',
'api_key','EFWWA5DCID6FBYHO','command_string',COMMAND4)
end

```

El código básicamente analiza los datos y verifica si dichos datos están entre los parámetros especificados y una vez verificado esto, lanza un comando que escribe un TalkBack para que el arduino lo busque.

Dos comandos para analizar en el Matlab analysis

Hay dos comandos de Matlab que resaltan porque pertenecen a una librería propia de Thingspeak para Matlab que se puede bajar desde <https://www.mathworks.com/matlabcentral/fileexchange/52244-thingspeak-support-to-olbox>.

El primer comando es

```
dato=thingSpeakRead(347713,'ReadKey','COH2SVB7L2Y85LIB','Fields',[Temp Hum  
Pres Illum]);
```

Este comando lee los últimos datos del canal 347713, utiliza una ReadKey que es la clave de autorización para poder leer, y luego asigna los nombres de las variables en las cuales se guardarán los datos. En este caso el campo Temperatura guardará su último dato en la variable Temp, y así sucesivamente.

El segundo comando es

```
response1=webwrite('https://api.thingspeak.com/talkbacks/18565/commands.json','api  
_key','EFWWA5DCID6FBYHO','command_string',COMMAND1)
```

Este comando escribe en la web de Thingsepeak un comando talkbac para que pueda ser buscado por el arduino. La primera parte es la dirección a dónde va a escribir el comando. El número 18565 es la dirección asignada a nuestros talkbacks. Se puede ver en Apps - Talkbacks. Luego se encuentra la *api_key* que es la clave para poder escribir el comando (hay api de lectura como de escritura), y por último el comando que se desea escribir.

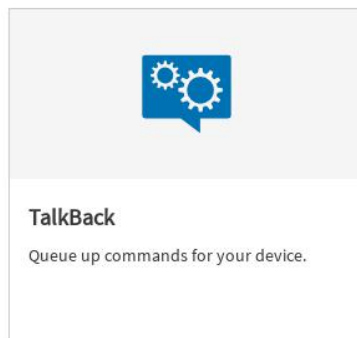
Este comando se encuentra cuatro veces en el script, porque debe postear un comando por variable del invernadero.

b) El TalkBack Command

Es un comando que se puede escribir en el servidor de Thingspeak, pero tiene ciertas particularidades. La particularidad más sobresaliente del funcionamiento de cualquier TalkBack es que una vez capturado por alguien, ese TalkBack es borrado y no puede ser recuperado nunca más. El servidor no guarda un historial de comandos TalkBack, pero puede almacenar una cantidad de ocho mil comando sin que estos sean requeridos. A partir de ese punto, comenzará a actuar en forma de cola, borrando el comando más antiguo ante el ingreso de uno nuevo.

A diferencia del canal y los campos que poseen una Write API Key y una Read API Key, el TalkBck command tiene una sola API Key que es utilizada tanto para la

lectura como para la escritura.



[Apps](#) / [TalkBack](#) / [Control_ambiental](#)

[Edit TalkBack](#)

| | |
|--------------------|------------------------------------|
| Name: | Control_ambiental |
| TalkBack ID: | 18565 |
| API Key: | EFWA5DCID6FBYH0 |
| | Regenerate API Key |
| Created: | 2017-10-16 12:45 pm |
| Logged to Channel: | Sensores1 |

Help

Example API Endpoints

Add a TalkBack Command

```
POST https://api.thingspeak.com/talkbacks/18565/commands.json
api_key=EFWA5DCID6FBYH0
```

Get a TalkBack Command

```
GET https://api.thingspeak.com/talkbacks/18565/commands/COMMAND_
```

Update a TalkBack Command

```
PUT https://api.thingspeak.com/talkbacks/18565/commands/COMMAND_
api_key=EFWA5DCID6FBYH0
```

c) El React

Para que todo lo anterior funcione, debe haber un disparador entre el ingreso del dato al canal y el Matlab Analysis. Este disparador es el React que se encuentra también en Apps.

En él solamente se configura la reacción ante un determinado evento. Como por ejemplo en el caso de los sensores, el evento es el ingreso de un dato al canal, pero como además solicita otro dato más, está forzado con la condición de que la temperatura sea mayor a -25 grados, cosa que no sucede aquí, es decir que siempre se ejecutará.

Podemos tener varios grupos de comandos talkback, por ello cada grupo tiene un nombre, una tipo de condición, que para nuestro caso es numérica, la frecuencia es cada vez que se inserta un dato, el canal sobre el que se insertan los datos, en este caso sensores 1, la condición (forzada para nuestro caso) la temperatura mayor a -25; luego la reacción, lanza el Matlab Analysis llamado ML_analisis que es el código de Matlab que se vió en a).

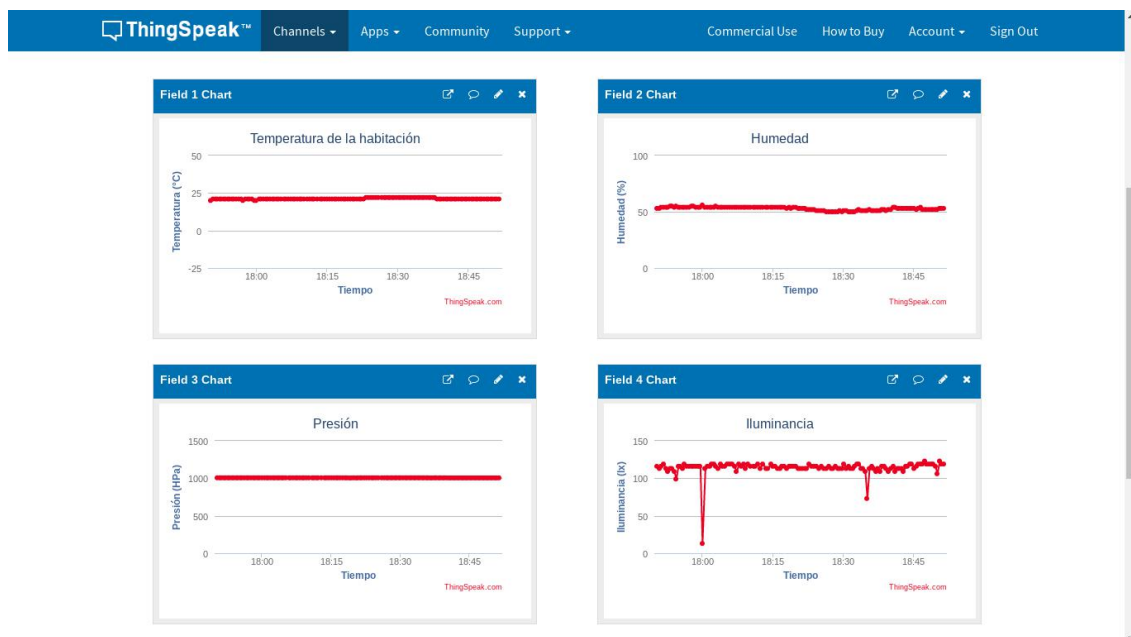
Apps / React / React_Ctrl_Amb

Edit React

| | |
|------------------|---|
| Name: | React_Ctrl_Amb |
| Condition Type: | Numeric |
| Test Frequency: | On data insertion |
| Last Ran: | 2018-06-09 21:51 |
| Channel: | Sensores1 |
| Condition: | Field 1 (Temperatura) is greater than -25 |
| MATLAB Analysis: | ML_analisis |
| Run: | Each time the condition is met |
| Created: | 2017-10-17 10:51 am |

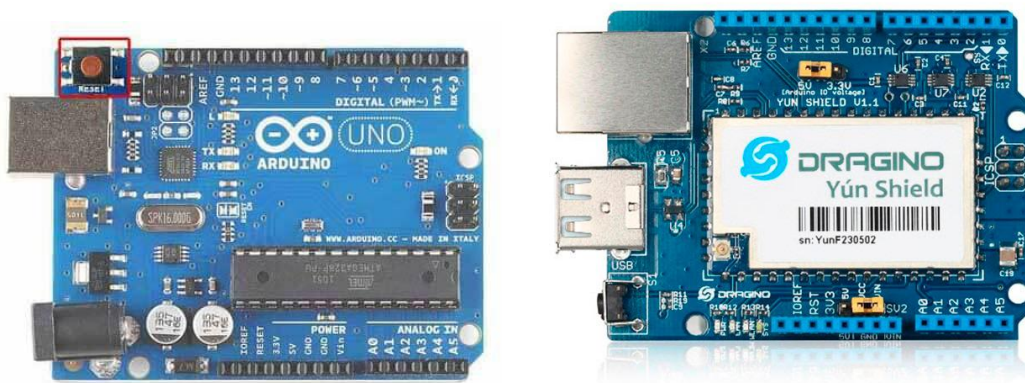
Figura 3: La configuración del React

Los canales de Thingspeak donde se muestran los datos y también se extraen para su análisis se ve de la siguiente forma.



1. Plataforma Electrónica Arduino

La plataforma electrónica sobre la que se montó el sistema de sensores para el invernadero fue Arduino. Se implementó un Arduino UNO con un shield que permitía la comunicación con el sistema de servicio web de Thingspeak. Este shield es el Dragino YUN.



El sistema de Shield se acopla directamente al Arduino UNO y se utiliza para expandir a través de internet las capacidades del Arduino.

Para poder hacer uso del Arduino, hay que instalar el IDE de arduino que se encuentra en <http://arduino.cc>

```
Arduino IDE Window: _04_yun_sensors | Arduino 1.8.6 Hourly Build 2018/08/14 10:25
File Edit Sketch Tools Help
[Icons]
_04_yun_sensors
#include "Bridge.h" // libreria conexion YUN WEB
#include "HttpClient.h" // libreria cliente http
#include "ThingSpeak.h" // libreria thingspeak (se baja de thingspeak)
#include "YunClient.h" // libreria en la p'qagina de la dragino YUN
#include <Wire.h> // librería I2C
#include <BH1750.h> // Librería sensor de intensidad luminosa
#include "DHT.h" // Librería sensor de Humedad y Temperatura DHT & AM2302
#include <SPI.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_Sensor.h>

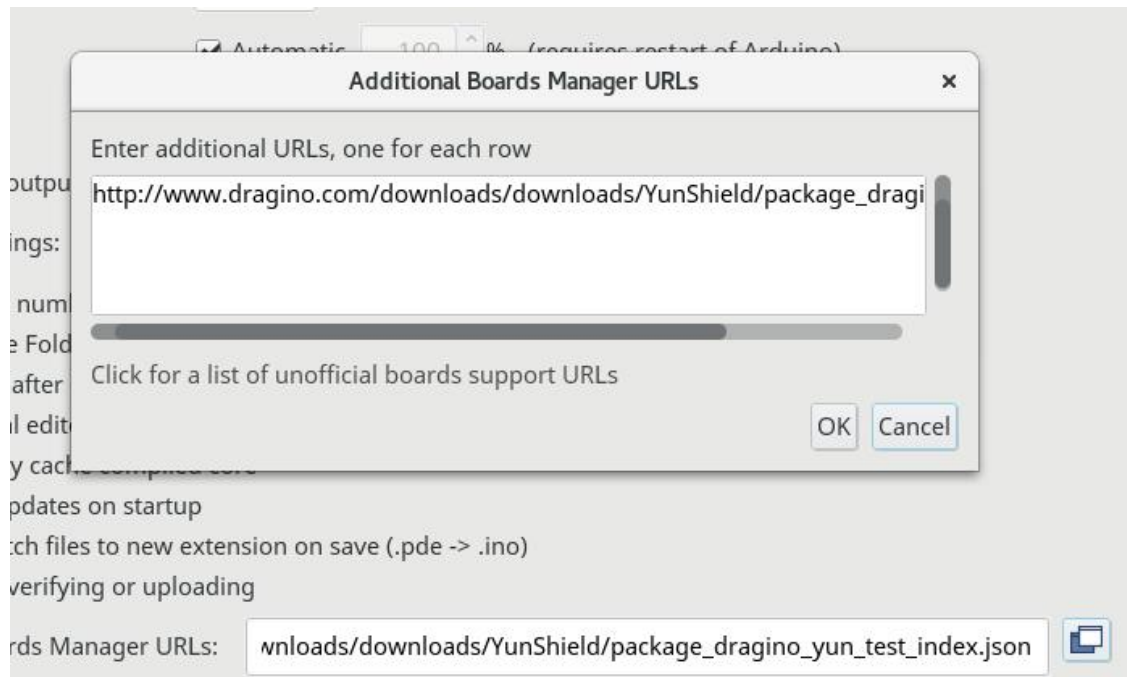
#define DHTPIN 2 // pin de conexión del SDA del sensor
#define DHTTYPE DHT22 // DHT 22 (AM2302)

YunClient client; //genera el cliente para el servicio web.

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on /dev/ttyACM0
```


Luego, una vez hecho esto, hay que instalar todas las librerías para poder utilizar el Shield Dragino YUN y los sensores. Para ello, en Preferencias, hay que agregar como se muestra en la figura, el siguiente link

http://www.dragino.com/downloads/downloads/YunShield/package_dragino_yun_test_index.json



Luego, para poder ver los sensores, y programarlos, hay que agregar las siguientes librerías:

- Adafruit Unified Sensor
- Adafruit BMP 280 Library
- <https://github.com/claws/BH1750> (descargar librería e instalarla a mano)

Se pueden agregar desde el IDE de Arduino, Sketch, incluir librería.

Una vez que se tienen todas las librerías instaladas se puede programar el arduino.

Programa Arduino YUN

El siguiente programa es el que realiza el relevo de los datos de los sensores, arma el paquete para enviar por internet a Thingspeak, luego lee el comando talkback y ejecuta la orden.

```

//*****
//*
//*          PROGRAMA PARA EL INVERNADERO DE ORQUÍDEAS
//*
//*****

#include "Bridge.h" // libreria conexion YUN WEB
#include "HttpClient.h" // libreria cliente http
#include "ThingSpeak.h" // libreria thingspeak (se baja de thingspeak)
#include "YunClient.h" // libreria en la p'qagina de la dragino YUN
#include <Wire.h> // librería I2C
#include <BH1750.h> // Librería sensor de intensidad luminosa
#include "DHT.h" // Librería sensor de Humedad y Temperatura DHT &
AM2302
#include <SPI.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_Sensor.h>

#define DHTPIN 2 // pin de conexión del SDA del sensor
#define DHTTYPE DHT22 // DHT 22 (AM2302)

YunClient client; //genera el cliente para el servicio web.

//ThingSpeak: datos para conexión, escritura y lectura
String thingSpeakAPI = "api.thingspeak.com"; // de la pagina de thingspeak
String talkBackAPIKey = "EFWWA5DCID6FBYHO"; // de la pagina de thingspeak
String talkBackID = "18565"; // de la pagina de thingspeak, talkback ID
const int checkTalkBackInterval = 15 * 1000; // Intervalo para checkear el
TalkBack (segundos * 1000 = intervalo)
unsigned long myChannelNumber = 347713; // de la pagina de thingspeak, es el
numero de canal
const char * myWriteAPIKey = "2ZR7MG3GLXA1GC0D"; // el código que te
habilita la escritura del canal, sale de thinsgpeak

// Se agregan estas 4 variables para guardar las lecturas de los sensores
long tem; // variable donde se guarda el valor de la temperatura
long pre; // variable donde se guarda el valor de la presión
long hum; // variable donde se guarda el valor de la humedad
long lux; // variable donde se guarda el valor de los lúmenes

BH1750 lightMeter;
DHT dht(DHTPIN, DHTTYPE); // genera un objeto DHT que funciona con ese
pin y es de ese tipo

```

```
Adafruit_BMP280 bme;
```

```
long lastConnectionTime = 0; // Variable para el control de la conexión (va por librería HTTPClient o YunClient)
```

```
void setup()
```

```
{  
  // parpadeo del led en el pin 13 para ver cuando pasa por este punto  
  pinMode(13, OUTPUT);  
  digitalWrite(13, LOW);  
  delay(1000);  
  digitalWrite(13, HIGH);  
  delay(1000);  
  digitalWrite(13, LOW);  
  
  // Se inicializa el puente (bridge) a la página de ThingSpeak  
  Bridge.begin(); // se inicializa sólo el puente  
  ThingSpeak.begin(client); // se linkea el puente con la página de ThingSpeak y la YUN  
  // Initialize the I2C bus (BH1750 library doesn't do this automatically)  
  Wire.begin();  
  lightMeter.begin(0x23);  
  dht.begin();           // se pone en funcionamiento el sensor  
  bme.begin(0x76);  
}
```

```
// COMIENZO DEL PROGRAMA QUE VA A ESTAR LEYENDO LOS SENSORES  
Y ESCRIBIENDO LA PAGINA THINGSPEAK
```

```
void loop()
```

```
{  
  // Acá ponemos las lecturas de los sensores  
  //do{  
    tem = dht.readTemperature();  
    hum = dht.readHumidity();  
    //}while(isnan(t) || isnan(h));  
    lux = lightMeter.readLightLevel();  
    pre = bme.readPressure()/100;  
  
    ThingSpeak.setField(1,tem);  
    ThingSpeak.setField(2,hum);  
    ThingSpeak.setField(3,pre);  
    ThingSpeak.setField(4,lux);  
    ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);  
  }  
}
```

```

    // Chekea a ThingSpeak para ver los comandos del TalkBack
    checkTalkBack(); // llamado de la función checkTalkBack que se encuentra al final
del bloque
    delay(1000);
    checkTalkBack(); // llamado de la función checkTalkBack que se encuentra al final
del bloque
    delay(1000);
    checkTalkBack(); // llamado de la función checkTalkBack que se encuentra al final
del bloque
    delay(1000);
    checkTalkBack(); // llamado de la función checkTalkBack que se encuentra al final
del bloque
    delay(checkTalkBackInterval);
}
void checkTalkBack() // En el entorno del IDE de Arduino, las funciones se definen al
finalizar el bloque de programa y no arriba como en C
{
    HttpClient client; // genera un nuevo cliente HTTP ya que el otro se destruyó con la
escritura

    String talkBackCommand;
    char charIn;
    String talkBackURL = "http://" + thingSpeakAPI + "/talkbacks/" + talkBackID +
"/commands/execute?api_key=" + talkBackAPIKey;

    // Realiza una solicitud tipo HTTP GET a la TalkBack API:
    client.get(talkBackURL);

    while (client.available()) { //Con esta función extrae letra por letra el comando
        charIn = client.read();
        talkBackCommand += charIn;
    }
    //Serial.println(talkBackCommand); //esta era una línea para que mostrara por
consola cual era el comando que recibía
    // Enciende o apaga el led según la configuración en la página de ThingSpeak
    if (talkBackCommand == "Talta") //verificación de la temperatura
    {
        digitalWrite(2, HIGH); //si temperatura es más alta que la asignada, enciende el
led en el pin 2
    }
    else if (talkBackCommand == "Tbaja")
    {
        digitalWrite(2, LOW); //si la temperatura es más baja que la asignada, apaga el
led en el pin 2
    }
}

```

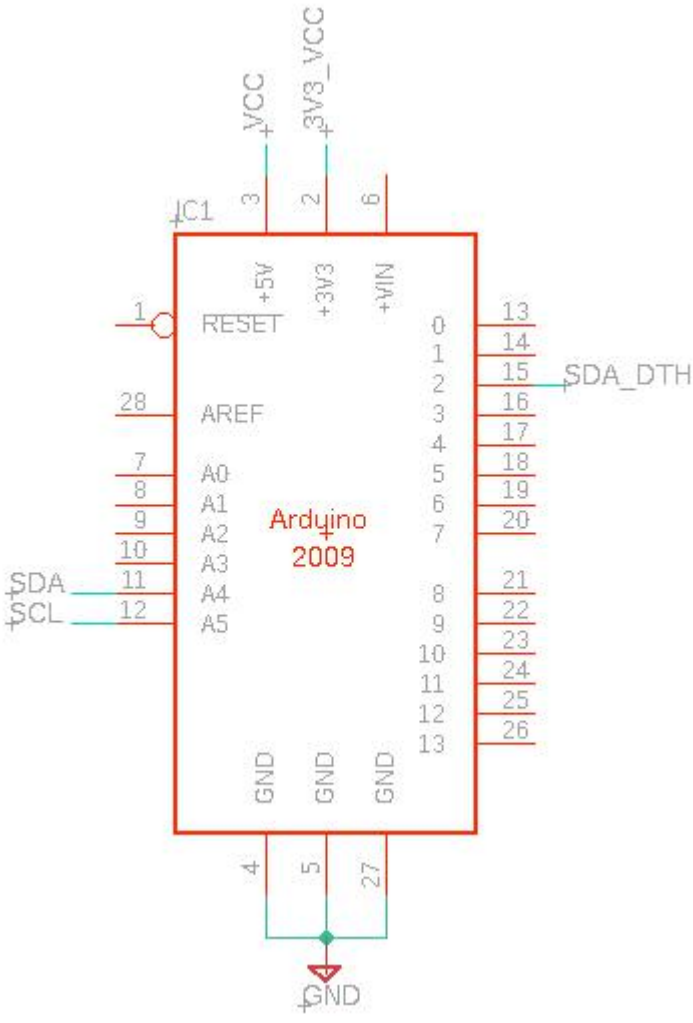
```

    }
    if (talkBackCommand == "Palta") //verificación de la
    {
        digitalWrite(3, HIGH); //si la presión es más alta que la asignada, enciende el led
        en el pin 3
    }
    else if (talkBackCommand == "Pbaja")
    {
        digitalWrite(3, LOW); //si la presión es más baja que la asignada, apaga el led en
        el pin 3
    }
    if (talkBackCommand == "Halta")
    {
        digitalWrite(4, HIGH); //si la humedad es más alta que la asignada, enciende el
        led en el pin 4
    }
    else if (talkBackCommand == "Hbaja")
    {
        digitalWrite(4, LOW); //si la humedad es más baja que la asignada, apaga el led
        en el pin 4
    }
    if (talkBackCommand == "Lalta")
    {
        digitalWrite(5, HIGH); //si los lúmenes son más altos que los asignados,
        enciende el led en el pin 5
    }
    else if (talkBackCommand == "Lbaja")
    {
        digitalWrite(5, LOW); //si los lúmenes son más bajos que los asignados, apaga el
        led en el pin 5
    }

    Serial.flush();
    delay(1000);
}

```

ANEXO 1 - CONEXIONES, ESQUEMÁTICO Y PLACA DE SENSORES



SENSOR DE PRESION BAROMETRICA



```
#include <Wire.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_Sensor.h>
uint16_t pre;
Adafruit_BMP280 bme;

void setup(){
  Serial.begin(9600);
  Wire.begin();
  bme.begin(0x76);
}

void loop(){
  do{
    pre = bme.readPressure()/100;
  }while(isnan(pre));
  Serial.print("La presion es: ");
  Serial.print(pre,4);
  Serial.println(" Hectopascals");
}
```

SENSOR DE INTENSIDAD LUMINICA



```
#include <wire.h>
#include <BH1750.h>
BH1750 lightmeter(0x23);

void setup{
  Serial.begin(9600);
  wire.begin();
  lightmeter.begin();
}

void loop(){
  uint16_t lux;
  while(1){
    do{
      lux = lightmeter.readlightlevel();
    }while(isnan(lux));
    serial.print("Luxes: ");
    serial.println((float)lux);
  }
}
```


SENSOR DE TEMPERATURA Y HUMEDAD



```
#include "DTH.h"
#define DTHPIN 2
#define DTHTYH DTH22
float h,t;
DTH dth(DTHPIN, DTHTYH);

void setup(){
  Serial.begin(9600);
  dth.begin();
}

void loop(){
  do{
    h=dth.readHumidity();
    t=dth.readTemperature();
  }while(isnan(h) || isnan(t));
  Serial.print("La temperatura es ");
  Serial.println(f,2);
  Serial.print("La humedad es ");
  Serial.print(h,2);
}
```