



Instituto Tecnológico y de Estudios Superiores  
de Monterrey

Implementación de internet de las cosas (TC1004B): Reto

Tarea 5.1: Administración del Reto

Asesor

Claudia M. Solís Garza

A00827095 – Omar Enrique González Uresti

A00827638 – Emiliano García Aguirre

A00828073 – Axel Giovanni Villanueva Cuéllar

A00828368 – Jesús Urquidez Calvo

A01720388 – Guillermo Andrés Castillo Chapa

4 de octubre de 2020

**Resumen**

Se presenta lo ocurrido mientras se realizó el pre-inicio, el desarrollo de inicio, y se muestran las evidencias de la planeación que se tiene para el proyecto.

# Índice

<b>1. Pre-Inicio</b>	<b>1</b>
1.1. Omar Enrique González Uresti . . . . .	1
1.2. Emiliano García Aguirre . . . . .	1
1.3. Axel Giovanni Villanueva Cuéllar . . . . .	1
1.4. Jesús Urquidez Calvo . . . . .	1
1.5. Guillermo Andrés Castillo Chapa . . . . .	2
<b>2. Inicio</b>	<b>2</b>
2.1. Nombre . . . . .	2
2.2. Área a la que pertenece su aplicación . . . . .	2
2.3. Descripción de la aplicación . . . . .	2
2.4. Diagrama . . . . .	3
2.5. Lista de los sensores/actuadores . . . . .	3
2.6. Visualización de datos . . . . .	3
<b>3. Planeación</b>	<b>4</b>
<b>A. Evidencias Pre-Inicio</b>	<b>5</b>
A.1. Omar Enrique González Uresti . . . . .	5
A.2. Emiliano García Aguirre . . . . .	6
A.3. Axel Giovanni Villanueva Cuéllar . . . . .	7
A.4. Jesús Urquidez Calvo . . . . .	8
A.5. Guillermo Andrés Castillo Chapa . . . . .	9
<b>B. Evidencias Planeación</b>	<b>10</b>

## 1. Pre-Inicio

Las evidencias de lo que se menciona a continuación se pueden encontrar en Apéndice [A](#).

### Omar Enrique González Uresti

Creo que la complejidad de poder realizar la conexión se debe a la cantidad de cosas técnicas que se deben hacer previamente. Además, de que tuve errores al conectar el puerto, sin embargo, supe arreglarlo y pude solucionarlo.

### Emiliano García Aguirre

Al realizar la conexión del MCU con mi computadora me topé con un par de problemas. El primero fue que no había conectado bien el cable a mi computadora y no registraba el puerto. Después, cuando intenté conectar el microprocesador al internet escribí mal mi contraseña varias veces hasta que finalmente corrió el código como debía.

### Axel Giovanni Villanueva Cuéllar

Con la experiencia previa que tenía programando en Arduino, efectuar la mayorías de las cosas requeridas lo consideré como algo muy sencillo de hacer. Como única complicación que tuve fue la de encontrar un cable que en verdad pasara información y no fuera sólo de energía, pero a parte de eso todo corrió sin problemas. Además tras leer el código e intentar obtener el API para la página usada, se reparó que lograr lo último no iba a ser posible, y que con lo que se imprimió en el Panel de Control era suficiente para probar que había funcionado.

### Jesús Urquidez Calvo

No tuve complicación a la hora de aplicar el programa del LED. Sin embargo a la hora de activar el código de Wifi se me complicó a la hora de notar el `Serial.begin(115200)` que estaba diferente y

además de que no se podía conectar a la API ya que el registro en la página no era posible.

## Guillermo Andrés Castillo Chapa

En mi caso yo aprendí que el NodeMCU no acepta redes 5G y me tardé un poco en entender el problema, para resolverlo lo que hice fue agarrar la red 2.4 de mi casa y ya con eso funcionó bien. Yo nunca había usado Arduino entonces fue un poco complicado pero ya entendí cómo hacer lo básico.

## 2. Inicio

### Nombre

IoP: Internet of Plants<sup>1</sup>.

### Área a la que pertenece su aplicación

Medio ambiente, sustentabilidad.

### Descripción de la aplicación

Se tiene la intención de diseñar una aplicación que se encargue de cuidar plantas domésticas<sup>2</sup>.

El sistema funcionará mediante un sensor de humedad el cual estará conectado a la planta, y estará brindando la información de las condiciones de la tierra; con este dato, se encenderá un LED RGB para ir indicando el estado por medio de su color, y dependiendo del caso, el sistema accionará un sistema de riego para la planta.

Asimismo se planea agregar un sensor ultrasónico para medir la cantidad de agua en el recipiente<sup>3</sup>, y de igual manera se contará con un LED RGB para indicar el nivel de agua.

---

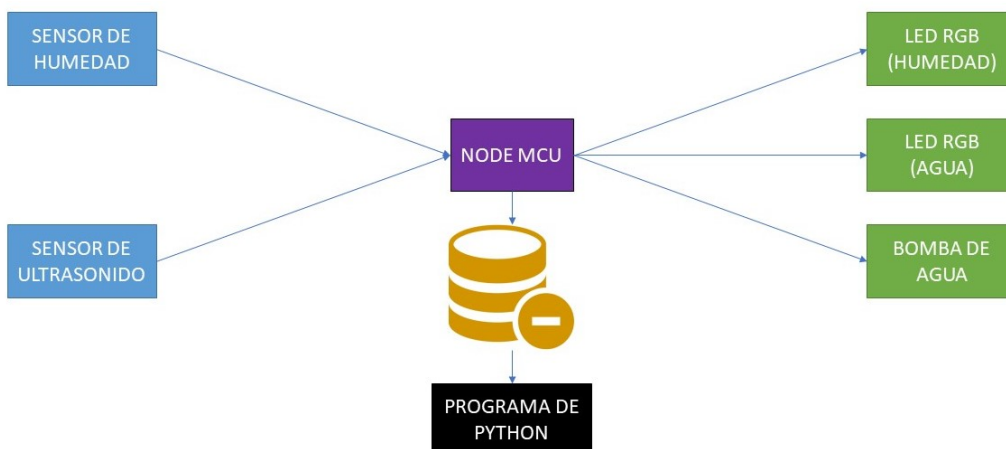
<sup>1</sup>Un guiño muy directo a la materia que se está cursando.

<sup>2</sup>Si bien se puede escalar a sistemas más grandes de manera un tanto sencilla, como “minimum valuable product” se hará con solo una planta en el hogar.

<sup>3</sup>A partir de la distancia que existe entre el sensor y el agua es como se espera obtener el resultado. Se resalta que dependiendo de la capacidad de un sensor ultrasónico de detectar el agua será lo que decida si esta parte del proyecto se implementará.

## Diagrama

Se presenta el diagrama con los componentes básicos y el cómo estarán relacionados entre ellos.



## Lista de los sensores/actuadores

- Sensor de humedad.
- 2 LED RGB.
- Bomba de agua de 3V.
- Sensor ultrasónico.
- Node MCU.

## Visualización de datos

Se plantea hacer una aplicación para la computadora elaborada en Python en donde se presenten los datos obtenidos. De manera preliminar se anticipa usar PyQT en combinación con Pandas para lograr lo anterior.

La manera de representar la información será mediante gráficas de tiempo para ilustrar el cambio que ha tenido conforme pasan los días, así como otras conclusiones que se puedan plantear a partir de la información recopilada (cantidad de agua usada, periodicidad en que se riega la planta) como datos disponibles para el usuario y así pueda estar informado de lo más relevante relacionado a su planta.

### 3. Planeación

Para facilitar la colaboración entre los miembros del equipo se empleará GitHub y se usará [este repositorio](#) en donde todos los integrantes ya se encuentran como colaboradores.

Para la parte de administración de proyecto se usarán las mismas herramientas que ofrece la plataforma, por lo que en la sección de “[Projects](#)” se encuentran el estado (por hacer, en proceso, terminado) de las diversas actividades que se tienen previstas por hacer.

Se destaca de igual manera que en “[Milestones](#)” se encuentran las fechas importantes que se tienen previstas<sup>4</sup>, mientras que en “[Issues](#)” están las actividades que permiten cumplir de manera satisfactoria las anteriores.

La manera en como está el repositorio actualmente se encuentra en el Apéndice [B](#).

---

<sup>4</sup>Son las fechas de entrega oficial por parte de la clase, pero se agregarán más en caso de sentir que se requiere realizar un mejor seguimiento

### A. Evidencias Pre-Inicio

Omar Enrique González Uresti

```
Arduino Archivo Editar Programa Herramientas Ayuda LED Arduino 1.8.13
LED
#define LED_BUILTIN 2
#define BUTTON_BUILTIN 0

void setup() {
  // El LED integrado está conectado al pin 2.
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(BUTTON_BUILTIN, INPUT);
}

void loop() {
  // Leer el estado del botón. Cuando está pulsado el pin se pone a nivel bajo
  int estado = digitalRead(BUTTON_BUILTIN);
  // Configurar el LED de acuerdo al estado del botón
  digitalWrite(LED_BUILTIN, estado);
}

Subido
connecting...
chip is ESP8266X
features: WIFI
crystal is 20MHz
MAC: 48:3f:da:0d:ee:0a
uploading stub...
running stub...
stub running...
configuring flash size...
auto-detected flash size: 4MB
compressed 261328 bytes to 193992...
wrote 261328 bytes (193992 compressed) at 0x00000000 in 17.1 seconds (effective 122.5 kbit/s)...
flash of data verified.
leaving...
hard resetting via RTS pin...

1 NodeMCU 1.0 (ESP-12E Module) en /dev/cu.usbserial-000
```

The screenshot displays the Arduino IDE interface. The left pane shows the source code for a program named 'Prueba\_Internet'. The code includes the `ESP8266WiFi` library and sets up a serial connection at 115200 baud. It defines a Wi-Fi network 'IZZI-238C' and a password 'F8883741Z38C'. The `void setup()` function connects to the Wi-Fi and prints a message. The `void loop()` function creates a `WiFiClient` object and attempts to connect to the API endpoint `api.wunderground.com` using a specific API key. The right pane shows the output window, which contains the compilation log and the serial monitor output. The serial monitor shows the device connecting to the IZZI-238C IP address and receiving a 503 Service Unavailable response from the API endpoint.

## Emiliano García Aguirre

```
LED | Arduino 1.8.13
File Edit Sketch Tools Help

LED

#define LED_PIN 2
#define BUTTON_PIN 0

void setup() {
  // El LED integrado está conectado al pin 2.
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}

void loop() {
  // Leer el estado del botón. Cuando está pulsado el pin se pone a nivel bajo
  int estado = digitalRead(BUTTON_PIN);
  // Configurar el LED de acuerdo al estado del botón
  digitalWrite(LED_PIN, estado);
}

New | Opening...
Sketch uses 257184 bytes (24%) of program storage space. Maximum is 1048576 bytes.
Global variables use 26792 bytes (32%) of dynamic memory, leaving 55128 bytes for local variables. Maximum is 81920 bytes.
Uploading...
Initial port COM5
Connecting...
Chip is ESP8266EX
Features: WDT,
Crystal is 26MHz
MAC: 68:03:cd:4d:4a:09
Compiling sketch...
Running sketch...
Sketch running...
Changing baud rate to 460800
Done.
Configuring flash size...
Auto-detected flash size: 4MB
Compressed 241344 bytes to 183092...
Wrote 241344 bytes (183092 compressed) at 0x00000000 in 4.5 seconds (effective 468.4 Kbit/s)...
Flash of data verified.
```

```
COM5

.....
WiFi connected
IP address:
192.168.0.115
connecting to api.wunderground.com
Ha fallado la conexión
connecting to api.wunderground.com
Ha fallado la conexión
connecting to api.wunderground.com
Ha fallado la conexión
connecting to api.wunderground.com
Ha fallado la conexión
connecting to api.wunderground.com
Ha fallado la conexión
connecting to api.wunderground.com
Ha fallado la conexión
connecting to api.wunderground.com
URL de la petición: http://api.wunderground.com:80/api/tu clave API de wunderground.com/conditions/lang:SP/q/autoip.json

Memoria libre en el ESP8266: 50512 Bytes

HTTP/1.1 503 Service Unavailable
Server: awselsb/2.0
Content-Type: text/plain; charset=utf-8
Content-Length: 19
Expires: Sat, 03 Oct 2020 23:25:31 GMT
Cache-Control: max-age=0, no-cache
Pragma: no-cache
Date: Sat, 03 Oct 2020 23:25:31 GMT
Connection: close
X-Origin-Hint: Default

Service Unavailable
Cerrando la conexión
```



## Axel Giovanni Villanueva Cuéllar



```
LED | Arduino 1.8.13
File Edit Sketch Tools Help

LED
#define LED_BUILTIN 2
#define BUTTON_BUILTIN 0

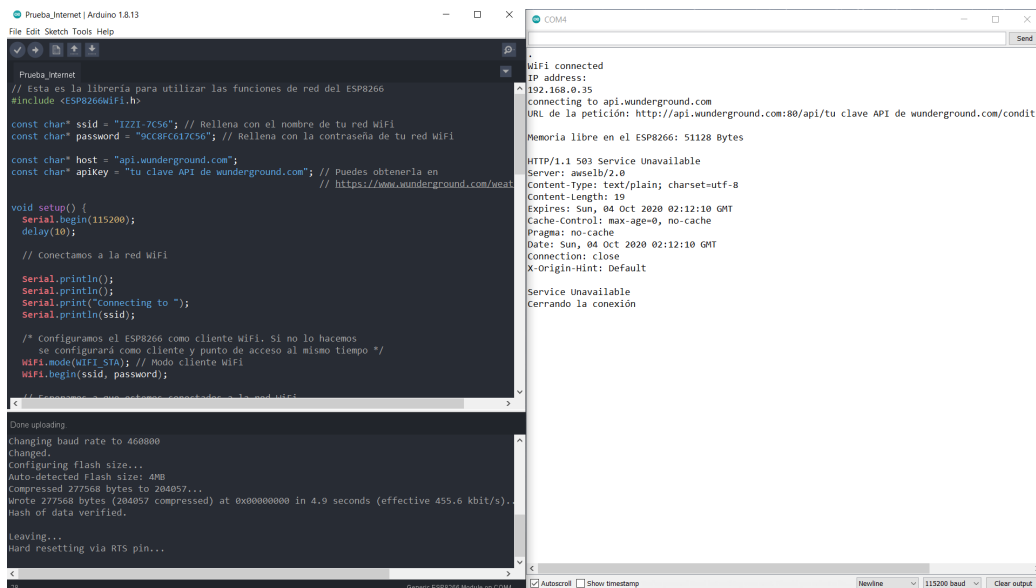
void setup() {
  // El LED integrado está conectado al pin 2.
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(BUTTON_BUILTIN, INPUT);
}

void loop() {
  // Leer el estado del botón. Cuando esté pulsado el pin se pone a nivel bajo
  int estado = digitalRead(BUTTON_BUILTIN);
  // Configurar el LED de acuerdo al estado del botón
  digitalWrite(LED_BUILTIN, estado);
}

Done uploading.
MAC: bc:dd:c2:46:9a:d2
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 261344 bytes to 193892...
Wrote 261344 bytes (193892 compressed) at 0x00000000 in 4.6 seconds (effective 455.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

Done! ESP8266 Module via COM4
```



```
Prueba_Internet | Arduino 1.8.13
File Edit Sketch Tools Help

Prueba_Internet
// Esta es la libreria para utilizar las funciones de red del ESP8266
#include <ESP8266WiFi.h>

const char* ssid = "I2ZI-7C56"; // Rellena con el nombre de tu red WiFi
const char* password = "9CCaC0i7C56"; // Rellena con la contraseña de tu red WiFi

const char* host = "api.wunderground.com";
const char* apiKey = "tu clave API de wunderground.com"; // Puedes obtenerla en https://www.wunderground.com/weather-api

void setup() {
  Serial.begin(115200);
  delay(10);

  // Conectamos a la red WiFi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  /* Configuramos el ESP8266 como cliente WiFi. Si no lo hacemos
   se configurará como cliente y punto de acceso al mismo tiempo */
  WiFi.mode(WIFI_STA); // Modo cliente WiFi
  WiFi.begin(ssid, password);
}

Done uploading.
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 277568 bytes to 204057...
Wrote 277568 bytes (204057 compressed) at 0x00000000 in 4.9 seconds (effective 455.6 kbit/s)...
Hash of data verified.

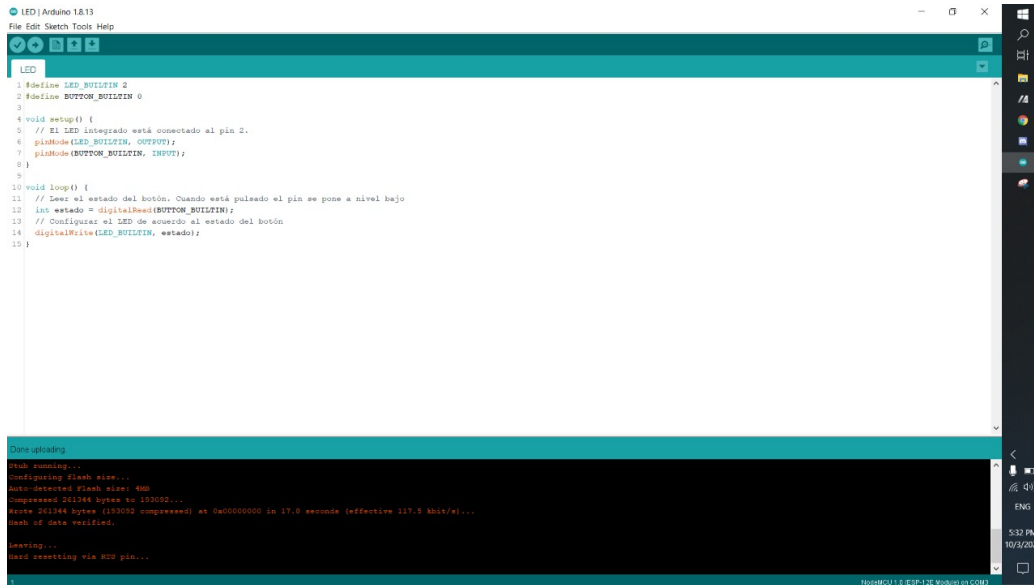
Leaving...
Hard resetting via RTS pin...

Done! ESP8266 Module via COM4

COM4
Send

WiFi connected
IP address:
192.168.0.35
connecting to api.wunderground.com
URL de la petición: http://api.wunderground.com:80/api/tu clave API de wunderground.com/conditi
Memoria libre en el ESP8266: 51128 Bytes
HTTP/1.1 503 Service Unavailable
Server: awselb/2.0
Content-Type: text/plain; charset=utf-8
Content-Length: 10
Expires: Sun, 04 Oct 2020 02:12:10 GMT
Cache-Control: max-age=0, no-cache
Pragma: no-cache
Date: Sun, 04 Oct 2020 02:12:10 GMT
Connection: close
X-Origin-Hint: Default
Service Unavailable
Cerrando la conexión
```

## Jesús Urquidez Calvo



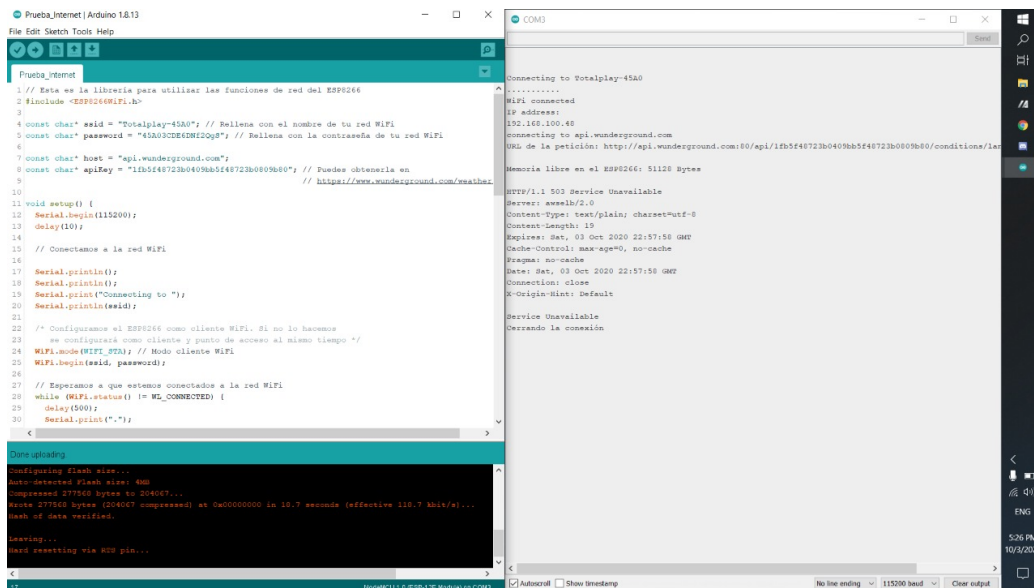
```
1 LED | Arduino 1.8.13
File Edit Sketch Tools Help

LED

1 #define LED_BUILTIN 2
2 #define BUTTON_BUILTIN 0
3
4 void setup() {
5   // El LED integrado está conectado al pin 2.
6   pinMode(LED_BUILTIN, OUTPUT);
7   pinMode(BUTTON_BUILTIN, INPUT);
8 }
9
10 void loop() {
11   // Leer el estado del botón. Cuando está pulsado el pin se pone a nivel bajo
12   int estado = digitalRead(BUTTON_BUILTIN);
13   // Configurar el LED de acuerdo al estado del botón
14   digitalWrite(LED_BUILTIN, estado);
15 }

Serial (gnding)
Sketch running...
Configuring flash size...
Auto-detected flash size: 4096
Compressed 261144 bytes to 193092...
Sketch 261144 bytes (193092 compressed) at 0x00000000 in 17.6 seconds (effective 117.5 Kbit/s)...
Flash of data verified.
Leaving...
Board resetting via RST pin...

Arduino Uno (ESP-12E Module on COM)
```



```
1 Prueba_Internet | Arduino 1.8.13
File Edit Sketch Tools Help

Prueba_Internet

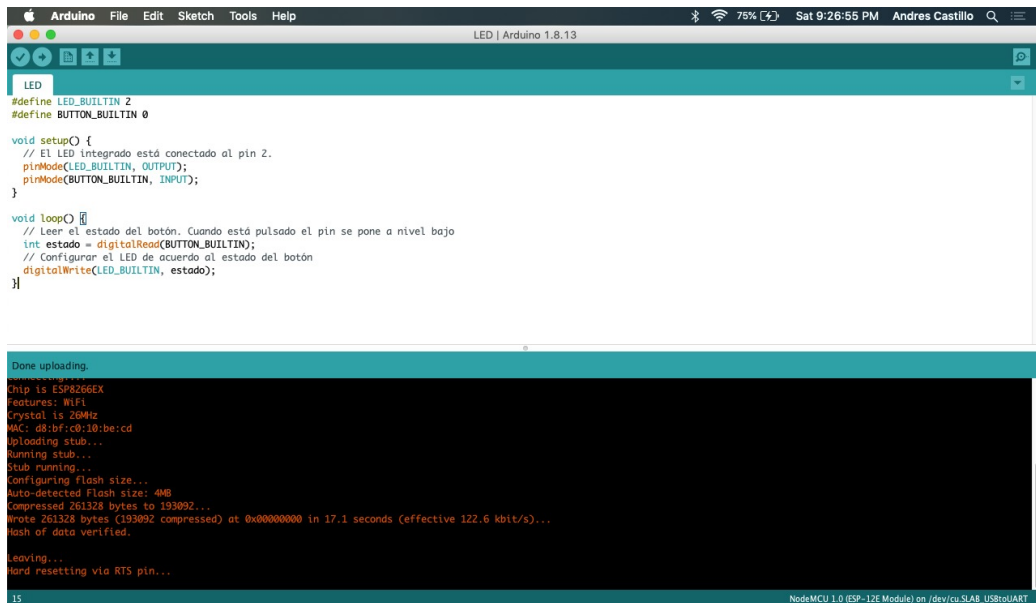
1 // Esta es la libreria para utilizar las funciones de red del ESP8266
2 #include <ESP8266WiFi.h>
3
4 const char* ssid = "Totalplay-43A0"; // Reemplaza con el nombre de tu red WiFi
5 const char* password = "43A03CDEGND0g8"; // Reemplaza con la contraseña de tu red WiFi
6
7 const char* host = "api.wunderground.com";
8 const char* apiKey = "1fb5f48723b0409bb5f48723b0809b60"; // Puedes obtenerla en
9 // https://www.wunderground.com/weather
10
11 void setup() {
12   Serial.begin(115200);
13   delay(10);
14
15   // Conectamos a la red WiFi
16
17   Serial.println();
18   Serial.println();
19   Serial.print("Connecting to ");
20   Serial.println(ssid);
21
22   /* Configuramos el ESP8266 como cliente WiFi. Si no lo hacemos
23    se configurará como cliente y punto de acceso al mismo tiempo */
24   WiFi.mode(WIFI_STA); // Modo cliente WiFi
25   WiFi.begin(ssid, password);
26
27   // Esperamos a que estemos conectados a la red WiFi
28   while (WiFi.status() != WL_CONNECTED) {
29     delay(500);
30     Serial.print(".");
31   }
32
33   Serial.println();
34
35   // Hacemos una petición GET al servidor
36   String url = "http://api.wunderground.com/0/api/1fb5f48723b0409bb5f48723b0809b60/conditions/latest?";
37   String query = "q=" + ssid + "&apiKey=" + apiKey;
38   url += query;
39   Serial.print("URL de la petición: ");
40   Serial.println(url);
41
42   // Hacemos la petición
43   WiFiClient client = WiFi.begin();
44   client.connect();
45   Serial.print("Conectado al servidor ");
46   Serial.println(url);
47
48   // Hacemos la petición
49   String response = client.get(url);
50   Serial.print("Respuesta del servidor: ");
51   Serial.println(response);
52 }

Serial (gnding)
Sketch running...
Configuring flash size...
Auto-detected flash size: 4096
Compressed 277560 bytes to 204067...
Sketch 277560 bytes (204067 compressed) at 0x00000000 in 18.7 seconds (effective 118.7 Kbit/s)...
Flash of data verified.
Leaving...
Board resetting via RST pin...

COM1
Connecting to Totalplay-43A0
.....
WiFi connected
IP address:
192.168.100.46
Connecting to api.wunderground.com
URL de la petición: http://api.wunderground.com/0/api/1fb5f48723b0409bb5f48723b0809b60/conditions/latest?
Memoria libre en el ESP8266: 51120 Bytes
HTTP/1.1 503 Service Unavailable
Server: awoah/2.0
Content-Type: text/plain; charset=utf-8
Content-Length: 19
Expires: Sat, 03 Oct 2020 22:57:58 GMT
Cache-Control: max-age=0, no-cache
Pragma: no-cache
Date: Sat, 03 Oct 2020 22:57:58 GMT
Connection: close
X-Debug-Host: Default
Service Unavailable
Verrando la conexión

Arduino Uno (ESP-12E Module on COM)
```

## Guillermo Andrés Castillo Chapa



```
Arduino File Edit Sketch Tools Help
LED | Arduino 1.8.13

LED
#define LED_BUILTIN 2
#define BUTTON_BUILTIN 0

void setup() {
  // El LED integrado está conectado al pin 2.
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(BUTTON_BUILTIN, INPUT);
}

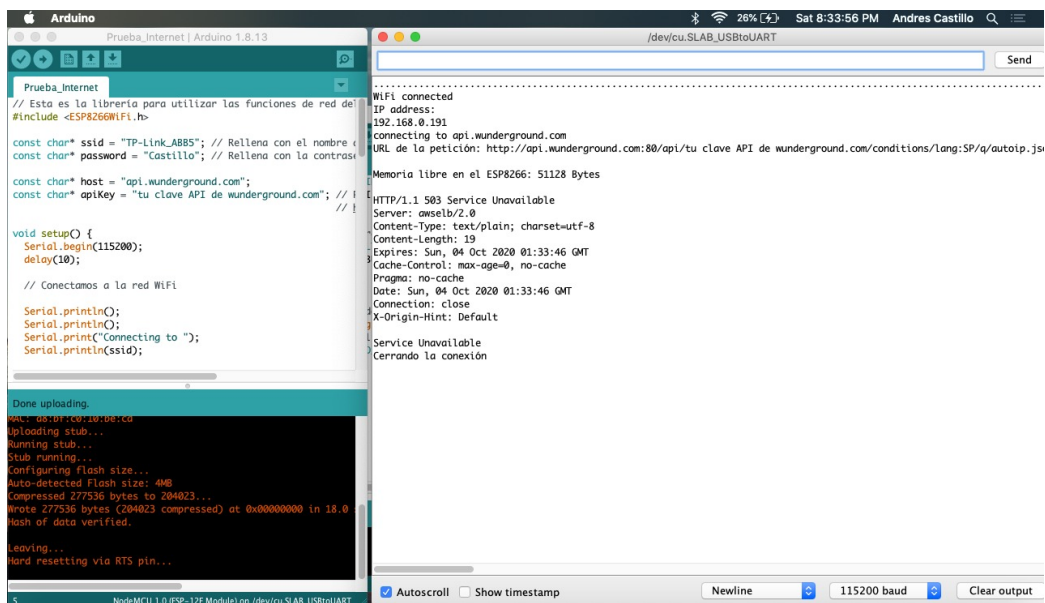
void loop() {
  // Leer el estado del botón. Cuando está pulsado el pin se pone a nivel bajo
  int estado = digitalRead(BUTTON_BUILTIN);
  // Configurar el LED de acuerdo al estado del botón
  digitalWrite(LED_BUILTIN, estado);
}
```

Done uploading.

Chip is ESP8266EX  
Features: WiFi  
Crystal is 26MHz  
MAC: 08:bf:c0:10:be:cd  
Uploading stub...  
Writing stub...  
Stub running...  
Configuring Flash size...  
Auto-detected Flash size: 4MB  
Compressed 261328 bytes to 193092...  
Wrote 261328 bytes (193092 compressed) at 0x00000000 in 17.1 seconds (effective 122.6 kbit/s)...  
Hash of data verified.

Leaving...  
Hard resetting via RTS pin...

NodeMCU 1.0 (ESP-12E Module) on /dev/cu.SLAB\_USBtoUART



```
Arduino File Edit Sketch Tools Help
Prueba Internet | Arduino 1.8.13

Prueba Internet
// Esta es la libreria para utilizar las funciones de red de
#include <ESP8266WiFi.h>

const char* ssid = "TP-Link_A8B5"; // Rellena con el nombre de
const char* password = "Castillo"; // Rellena con la contraseña
const char* host = "api.wunderground.com";
const char* apiKey = "tu clave API de wunderground.com"; //

void setup() {
  Serial.begin(115200);
  delay(10);

  // Conectamos a la red WiFi

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
}
```

Done uploading.

MAC: 08:bf:c0:10:be:cd  
Uploading stub...  
Writing stub...  
Stub running...  
Configuring Flash size...  
Auto-detected Flash size: 4MB  
Compressed 277536 bytes to 204023...  
Wrote 277536 bytes (204023 compressed) at 0x00000000 in 18.0...  
Hash of data verified.

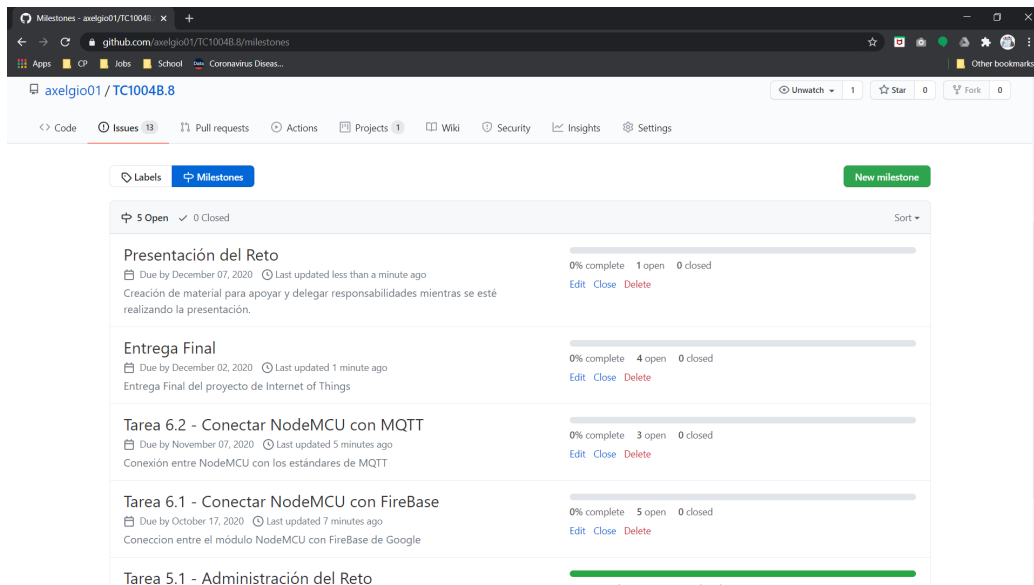
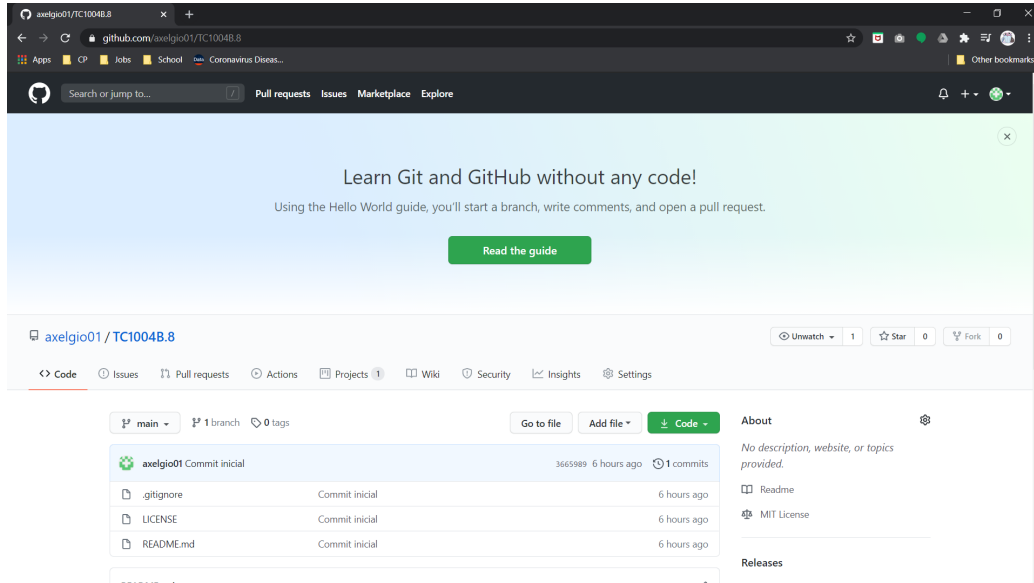
Leaving...  
Hard resetting via RTS pin...

NodeMCU 1.0 (ESP-12E Module) on /dev/cu.SLAB\_USBtoUART

WiFi connected  
IP address:  
192.168.0.191  
connecting to api.wunderground.com  
URL de la petición: http://api.wunderground.com:80/api/tu clave API de wunderground.com/conditions/lang:SP/q/autoip.js  
Memoria libre en el ESP8266: 51128 Bytes  
HTTP/1.1 503 Service Unavailable  
Server: amselb/2.0  
Content-Type: text/plain; charset=utf-8  
Content-Length: 19  
Expires: Sun, 04 Oct 2020 01:33:46 GMT  
Cache-Control: max-age=0, no-cache  
Pragma: no-cache  
Date: Sun, 04 Oct 2020 01:33:46 GMT  
Connection: close  
X-Origin-Hint: Default  
Service Unavailable  
Cerrando la conexión

Autoscroll Show timestamp Newline 115200 baud Clear output

## B. Evidencias Planeación



## Implementación de internet de las cosas

