SYSTEMS ON A CHIP

Introducción

Sistema de detección de temperatura y humedad con aviso acústico mediante un buzzer y de activación de ventiladores mediante disparadores y desde la plataforma ThingsBoard.

Materiales:

- 4 leds de colores
- Buzzer pasivo
- Sensor temperatura y humedad DHT11
- 2 transistores 2N 2222
- 7 resistencias 220 ohm
- Protoboard
- Cables y conectores
- Placa nodemcu 8266
- 2 Ventiladores 3,6-6V

Software utilizado:

- Plataforma web de ThingsBoard
- IDE de arduino
- Librerias necesarias

Funcionamiento:

Tenemos un sistema con un sensor de temperatura y humedad que tiene a sus disposición un sistema de aviso si sobrepasa ciertos umbrales de temperatura.

Mediante unos leds y un buzzer nos avisa de la temperatura del sensor.

Los colores y el sonido van cambiando según el rango de temperatura.Si sube demasiado se activa un ventilador conectado al ESP8266.

Desde la plataforma ThingsBoard se obtiene cada 2 segundos las lecturas de temperatura y humedad mediante graficas.

Tambien podemos controlar 2 controladores desde la plataforma:

1 led azul a modo de ejemplo y un segundo ventilador de apoyo.

Configuración de la plataforma ThingBoard:

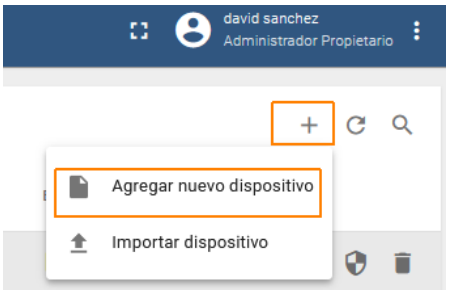
Lo primero que tenemos que hacer es crearnos una cuenta en la plataforma y loguearnos.

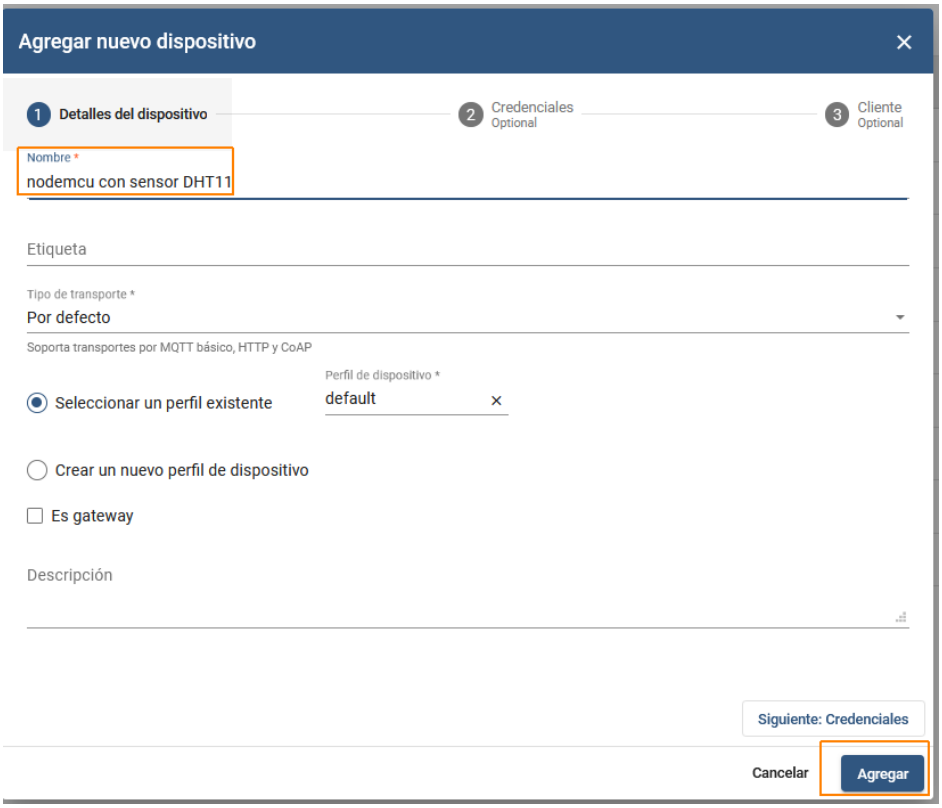
Una vez logueados,tenemos que crear un dispositivo(o elegir uno creado)y un panel donde se visualizará las telemetrías.

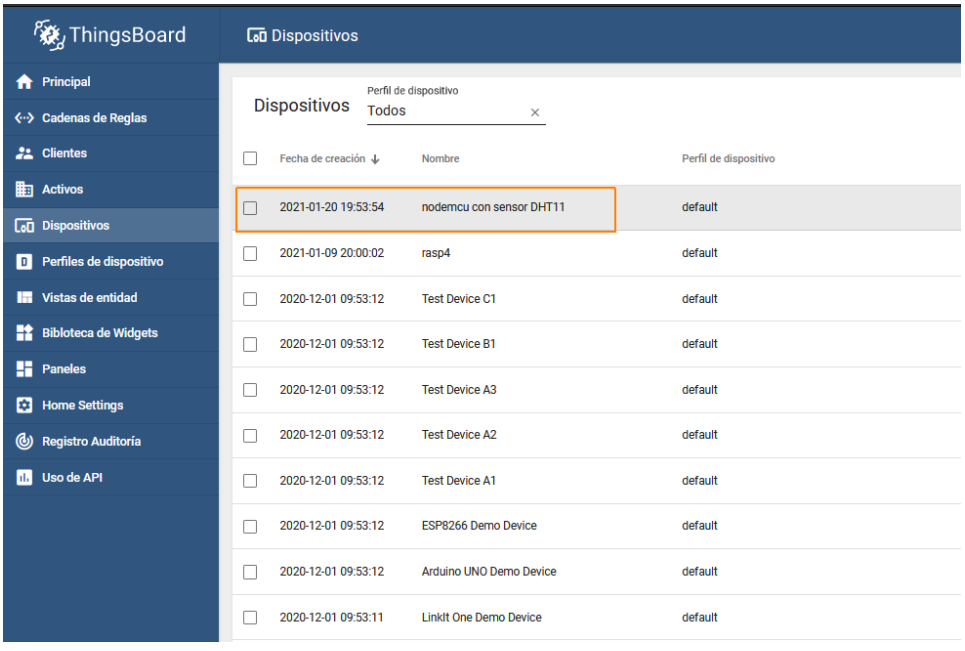
Creación dispositivo:

Desde el panel principale elegimos “dispositivos” y se nos despliegue la lista con los dispositivos disponibles y algunas caracteristicas de los mismos.

Para crear un dispositivo nuevo le damos a “+” en la parte derecha del panel y seguimos las indicaciones.


david sanchez
Administrador Propietario
Agregar nuevo dispositivo
Importar dispositivo


Agregar nuevo dispositivo
1 Detalles del dispositivo
2 Credenciales Optional
3 Cliente Optional
Nombre *
nodemcu con sensor DHT11
Etiqueta
Tipo de transporte *
Por defecto
Soporta transportes por MQTT básico, HTTP y CoAP
Seleccionar un perfil existente
Perfil de dispositivo *
default
Crear un nuevo perfil de dispositivo
Es gateway
Descripción
Siguiente: Credenciales
Cancelar
Agregar


ThingsBoard
Dispositivos
Principal
Cadenas de Reglas
Clientes
Activos
Dispositivos
Perfiles de dispositivo
Vistas de entidad
Biblioteca de Widgets
Paneles
Home Settings
Registro Auditoría
Uso de API
Dispositivos
Perfil de dispositivo
Todos
Fecha de creación
Nombre
Perfil de dispositivo
2021-01-20 19:53:54 nodemcu con sensor DHT11 default
2021-01-09 20:00:02 rasp4 default
2020-12-01 09:53:12 Test Device C1 default
2020-12-01 09:53:12 Test Device B1 default
2020-12-01 09:53:12 Test Device A3 default
2020-12-01 09:53:12 Test Device A2 default
2020-12-01 09:53:12 Test Device A1 default
2020-12-01 09:53:12 ESP8266 Demo Device default
2020-12-01 09:53:12 Arduino UNO Demo Device default
2020-12-01 09:53:11 LinkIt One Demo Device default

Con esto ya tendriamos creado el dispositivo.

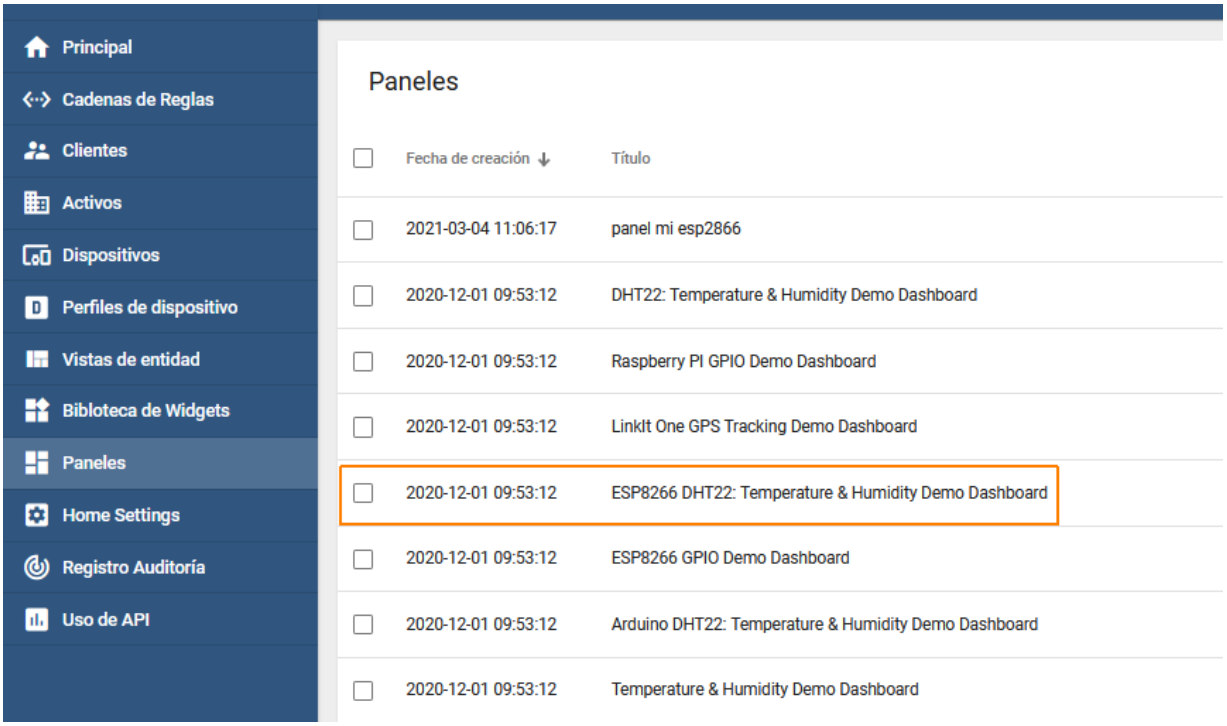
Para poder conectarnos a el ,tenemos que obtener las credenciales del mismo y para ello le damos al icono del lado derecho que se muestra en la imagen inferior.Apuntamos el contenido que hay debajo de “Token de acceso” porque lo necesitaremos después para introducirlo en el codigo de conexión.

Creación de un panel:

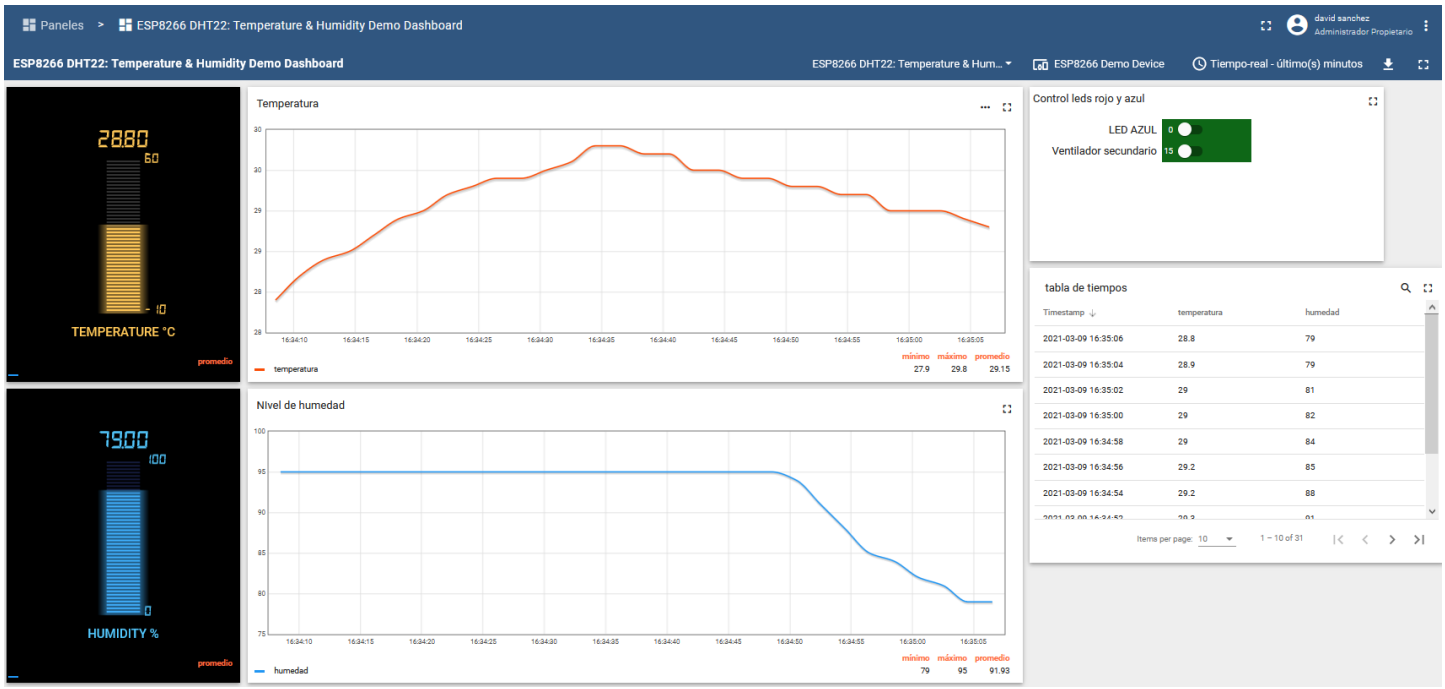
Para crear un panel donde veremos las telemetrias y gestionaremos el control del led y ventilador secundario de neustro nodemcu esp8266 elegimos un panel ya creado y lo modificaremos a nuestro gusto.

Priemramente elegimos en el panel principal de la plataforma el apartado “Paneles” y en el menu central elegimos

“**ESP8266 DHT22 Temperature & humidity Demo Dashboard**”.



Este es el aspecto de mi panel ya terminado:



Partes del panel:

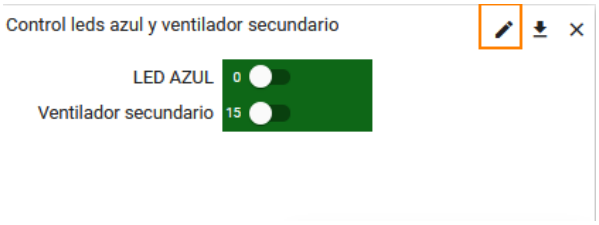
Para poder acceder a la parte de creación y edición de los widget tenemos que dar al boton naranja de la parte inferior derecha de la plataforma.

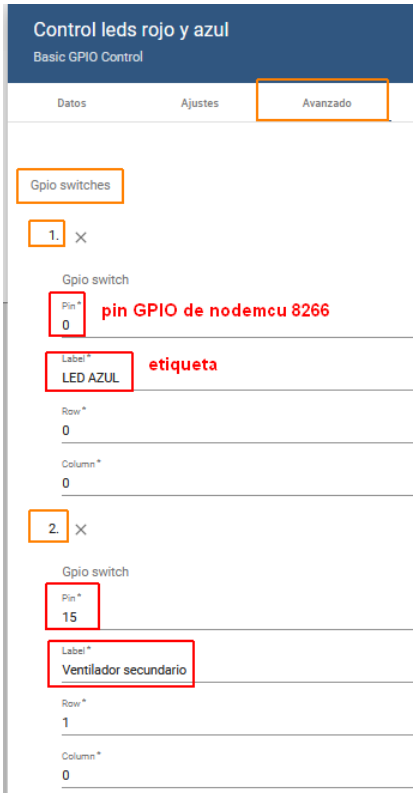
Una vez hecho esto ya podemos editar cada widget creado(aparece una especia de lapiz):

Apartado de control de led y ventilador:

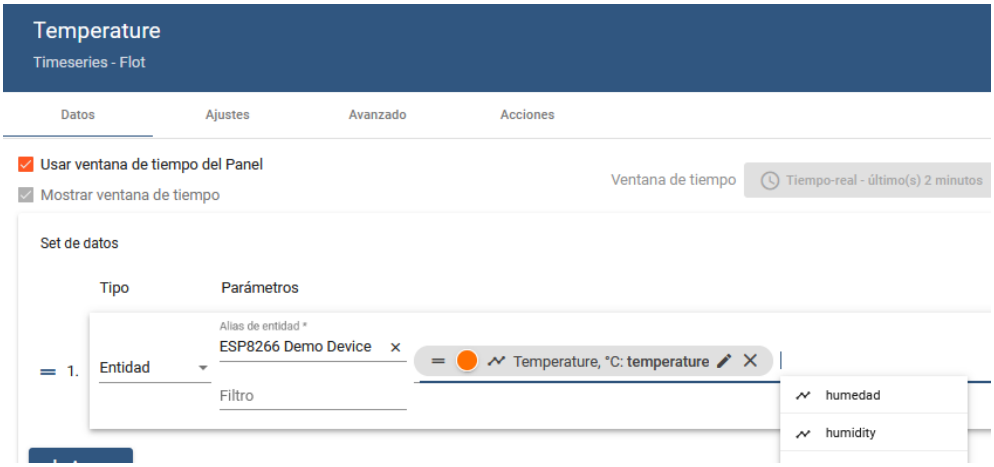
En el apartado “Avanzado” de la configuración del widget tenemos la sección de gestión de los siwtches de los Gpio de nuestro nodemcu esp8266.

Aquí figuran los pines Gpio del led azul conectado al pin D3(Gpio 0) y el ventilador conectado al pin D1 (Gpio 5).

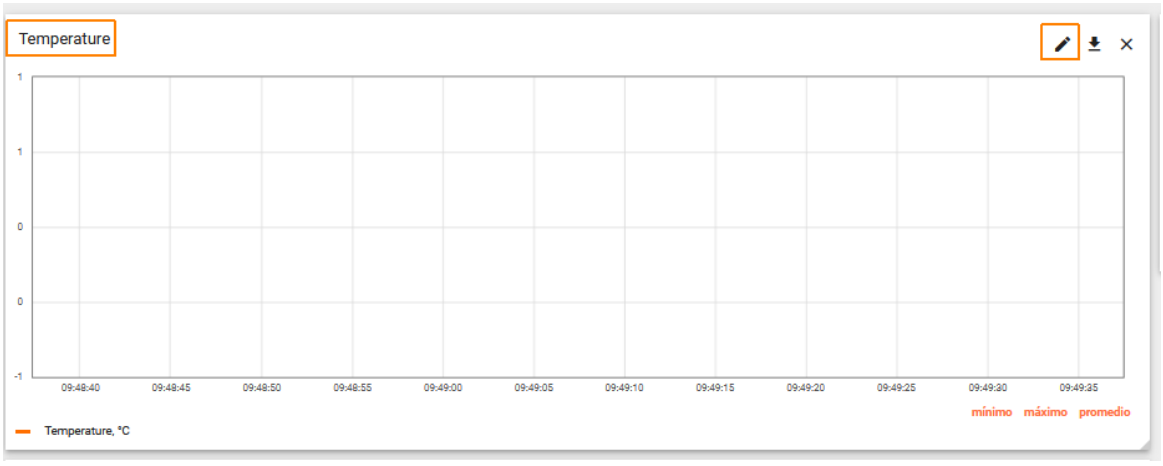




Widget de ventana de tiempo de la temperatura:

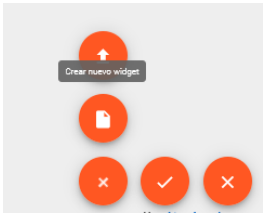


Aquí elegimos desde que dispositivo vamos a leer las lecturas y de que tipo(en este caso la temperatura):

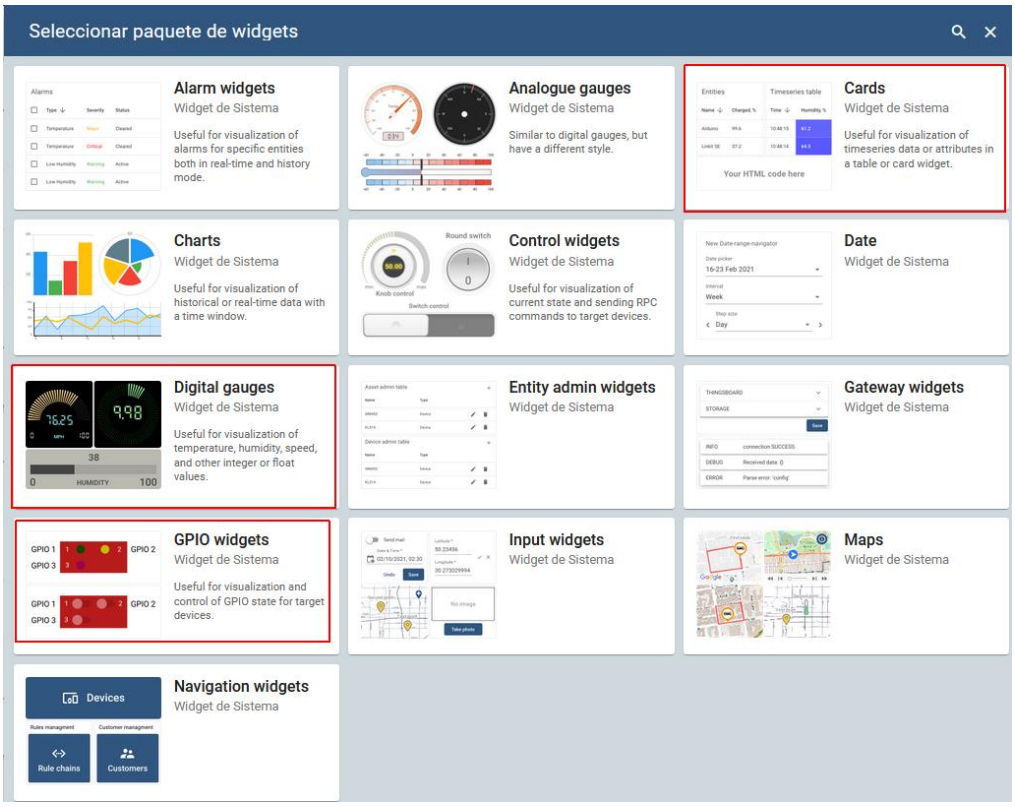


E igualmente el widget de la medición de la humedad:

Para poder añadir mas widget tenemos que dar al botón naranja antes mencionado y cuando se abre el desplegable elegir "**crear nuevo widget**":



Se nos abre un panel con todas las opciones posibles:



En rojo aparecen las opciones que he elegido yo para la creación de mi panel.

Creación del script de conexión y envio de telemetría:

Parte importación librerias y creación variables globales

scriptTerminado

```
#include <ArduinoJson.h>
#include <PubSubClient.h>          // Protocolo MQTT
#include <ESP8266WiFi.h>           // Include the Wi-Fi-Multi library
#include <DHT.h>                   // Include DHT11
#include <Adafruit_Sensor.h>       // Dependencia de DHT.h

#define WIFI_AP "[redacted]"                  // Nombre del wifi
#define WIFI_PASSWORD "[redacted]"            // Contraseña del wifi
#define TOKEN "[redacted]"                    // ThingsBoard Token
#define LED_AZUL 0
#define VENTILADOR D1
#define VENTILADOR_SEC 15
#define DHTPIN D2
#define LEDVERDE D4
#define LEDAMARILLO D5
#define LEDROJO D6
#define BUZZER D7
#define DHTTYPE DHT11
```

Parte conexión plataforma,inicio sensor DHT y genstion pines Gpio:

```
char thingsboardServer[] = "demo.thingsboard.io";
WiFiClient wifiClient;
PubSubClient client(wifiClient);

// ponemos los gpio en estado LOW
boolean gpioState[] = {false, false};

// Inicializamos el sensor DHT
DHT dht(DHTPIN, DHTTYPE);
int status = WL_IDLE_STATUS;
unsigned long lastSend;
void setup() {
  Serial.begin(115200);
  // Set output mode for all GPIO pins
  pinMode(LED_AZUL, OUTPUT);
  pinMode(VENTILADOR, OUTPUT);
  pinMode(VENTILADOR_SEC, OUTPUT);
  pinMode(LEDVERDE, OUTPUT);
  pinMode(LEDAMARILLO, OUTPUT);
  pinMode(LEDROJO, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  delay(10);
  InitWiFi();
  dht.begin();
  client.setServer( thingsboardServer, 1883 );
  client.setCallback(on_message);
  lastSend = 0;
}
```

Bucle continuo:

```
void loop() {
  if ( !client.connected() ) {
    Serial.println("reconectando");
    reconnect();
  }
  if ( millis() - lastSend > 2000 ) { // se actualiza cada 2 seg
    apagarLEDs();
    getAndSendTemperatureAndHumidityData();

    lastSend = millis();
   }
  client.loop();
}
```

Función de apagar los leds y captura de datos del sensor dht11 con la creación de condicionales para activar los leds y el buzzer de aviso y la activación del ventilador cuando la temperatura es mas elevada:

```
void apagarLEDs()
{
  // Apagamos todos los LEDs
  digitalWrite(VENTILADOR, LOW);
  digitalWrite(LEDVERDE, LOW);
  digitalWrite(LEDAMARILLO, LOW);
  digitalWrite(LEDROJO, LOW);
}
void getAndSendTemperatureAndHumidityData()
{
  Serial.println("Obteniendo datos de temperatura");

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (t>15 && t <=30){
  digitalWrite(LEDVERDE, HIGH);
  tone(BUZZER, 2000, 200);
  }
  if (t>30 && t <=40){
  digitalWrite(LEDAMARILLO, HIGH);
  tone(BUZZER, 2200, 225);
  }
  if (t>40){
  digitalWrite(LEDROJO, HIGH);
  tone(BUZZER, 2800, 250);
  digitalWrite(VENTILADOR, HIGH);
  }

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println("Fallo al leer el sensor DHT!");
    return;
  }

  Serial.print("Humedad: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperatura: ");
  Serial.print(t);
  Serial.print(" *C ");

  String temperature = String(t);
  String humidity = String(h);
```

Envio de telemetria a la plataforma:

```
  // Just debug messages
  Serial.print( "Enviando temperatura y humedad : [" );
  Serial.print( temperature ); Serial.print( "," );
  Serial.print( humidity );
  Serial.print( "]   -> " );

  // Prepare a JSON payload string
  String payload = "{";
  payload += "\"temperatura\":"; payload += temperature; payload += ",";
  payload += "\"humedad\":"; payload += humidity;
  payload += "}";

  // Send payload
  char attributes[100];
  payload.toCharArray( attributes, 100 );
  client.publish( "v1/devices/me/telemetry", attributes );
  Serial.println( attributes );
}
```

Función de publicación de mensajes recibidos de la plataforma:

```
// The callback for when a PUBLISH message is received from the server.
void on_message(const char* topic, byte* payload, unsigned int length) {

  Serial.println("On message");

  char json[length + 1];
  strncpy (json, (char*)payload, length);
  json[length] = '\0';

  Serial.print("Topic: ");
  Serial.println(topic);
  Serial.print("Message: ");
  Serial.println(json);

  // Decode JSON request
  StaticJsonBuffer<200> jsonBuffer;
  JsonObject& data = jsonBuffer.parseObject((char*)json);

  if (!data.success())
  {
    Serial.println("parseObject() failed");
    return;
  }

  // Check request method
  String methodName = String((const char*)data["method"]);

  if (methodName.equals("getGpioStatus")) {
    // Reply with GPIO status
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    client.publish(responseTopic.c_str(), get_gpio_status().c_str());
  } else if (methodName.equals("setGpioStatus")) {
    // Update GPIO status and reply
    set_gpio_status(data["params"]["pin"], data["params"]["enabled"]);
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    client.publish(responseTopic.c_str(), get_gpio_status().c_str());
    client.publish("v1/devices/me/attributes", get_gpio_status().c_str());
  }
}
```

Gestión de los gpio desde la paltaforma(led azul y ventilador secundario):

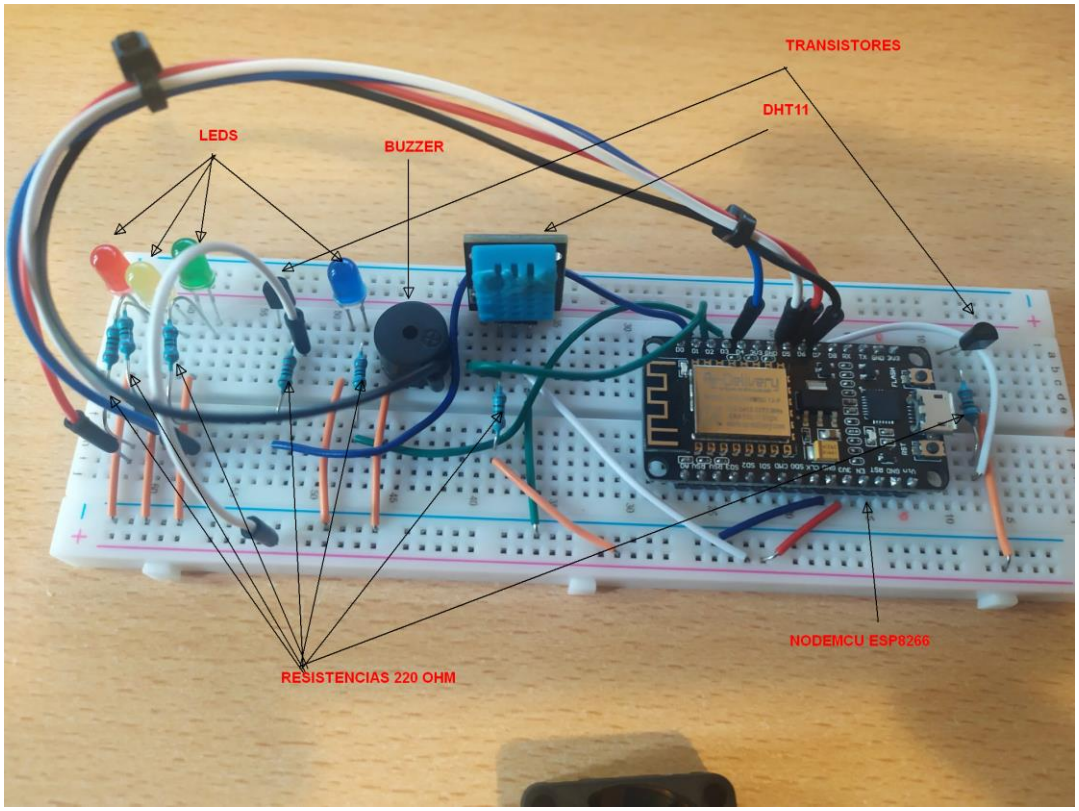
```
String get_gpio_status() {
  // Prepare gpios JSON payload string
  StaticJsonBuffer<200> jsonBuffer;
  JsonObject& data = jsonBuffer.createObject();
  data[String(LED_AZUL)] = gpioState[0] ? true : false;
  data[String(VENTILADOR_SEC)] = gpioState[1] ? true : false;
  char payload[256];
  data.printTo(payload, sizeof(payload));
  String strPayload = String(payload);
  Serial.print("Get gpio status: ");
  Serial.println(strPayload);
  return strPayload;
}
void set_gpio_status(int pin, boolean enabled) {
  if (pin == LED_AZUL) {
    // Output GPIOs state
    digitalWrite(LED_AZUL, enabled ? HIGH : LOW);
    // Update GPIOs state
    gpioState[0] = enabled;
  } else if (pin == VENTILADOR_SEC) {
    // Output GPIOs state
    digitalWrite(VENTILADOR_SEC, enabled ? HIGH : LOW);
    // Update GPIOs state
    gpioState[1] = enabled;
  }
}
```

Funciones de Wifi y reconexión:

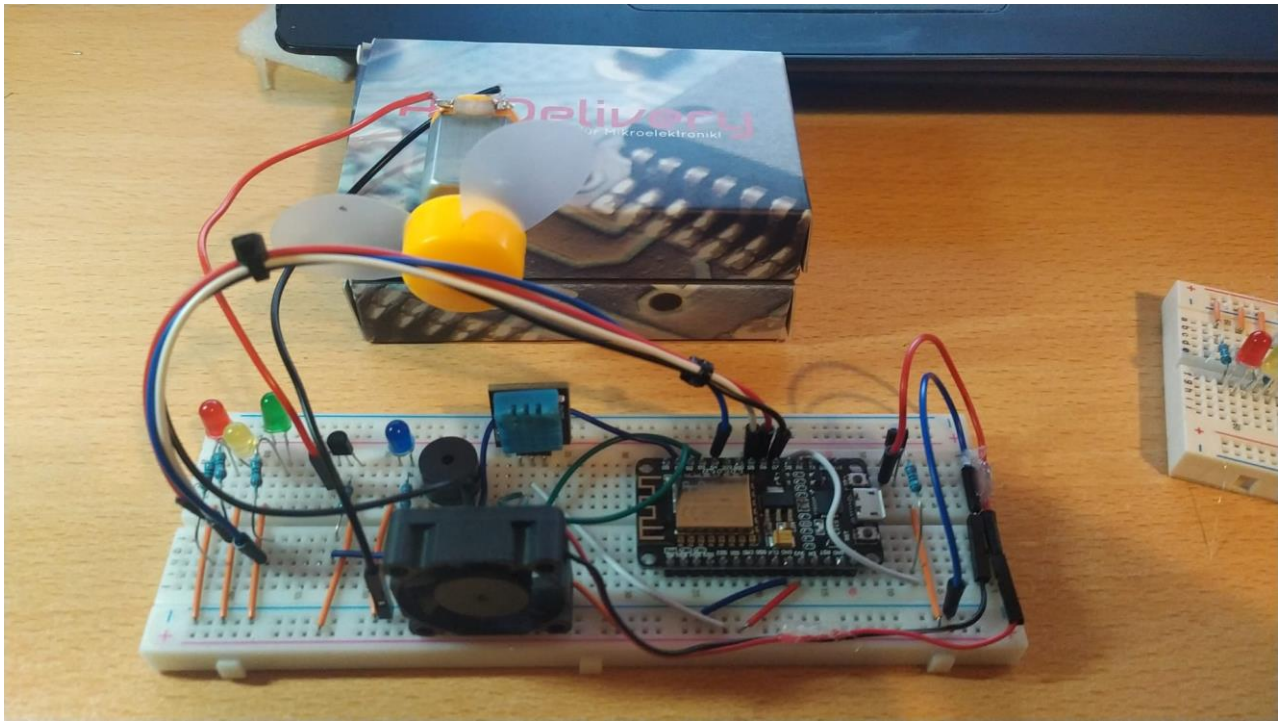
```
void InitWiFi() {
  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network

  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connectado al AP");
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    status = WiFi.status();
    if ( status != WL_CONNECTED) {
      WiFi.begin(WIFI_AP, WIFI_PASSWORD);
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      }
      Serial.println("Connectado al AP");
    }
    Serial.print("Conectando a ThingsBoard node ...");
    // Attempt to connect (clientId, username, password)
    if ( client.connect("ESP8266 Device", TOKEN, NULL) ) {
      Serial.println( "[DONE]" );
      // Subscribing to receive RPC requests
      client.subscribe("v1/devices/me/rpc/request/+");
      // Sending current GPIO status
      Serial.println("Enviando estado  GPIO actual ...");
      client.publish("v1/devices/me/attributes", get_gpio_status().c_str());
    } else {
      Serial.print( "[FAILED] [ rc = " );
      Serial.print( client.state() );
      Serial.println( " : intentando en 5 segundos]" );
      // Wait 5 seconds before retrying
      delay( 5000 );
    }
  }
}
```

Visión del hardware conectado:

Componentes en la broadboard sin ventiladores.



Con los dos ventiladores montados:

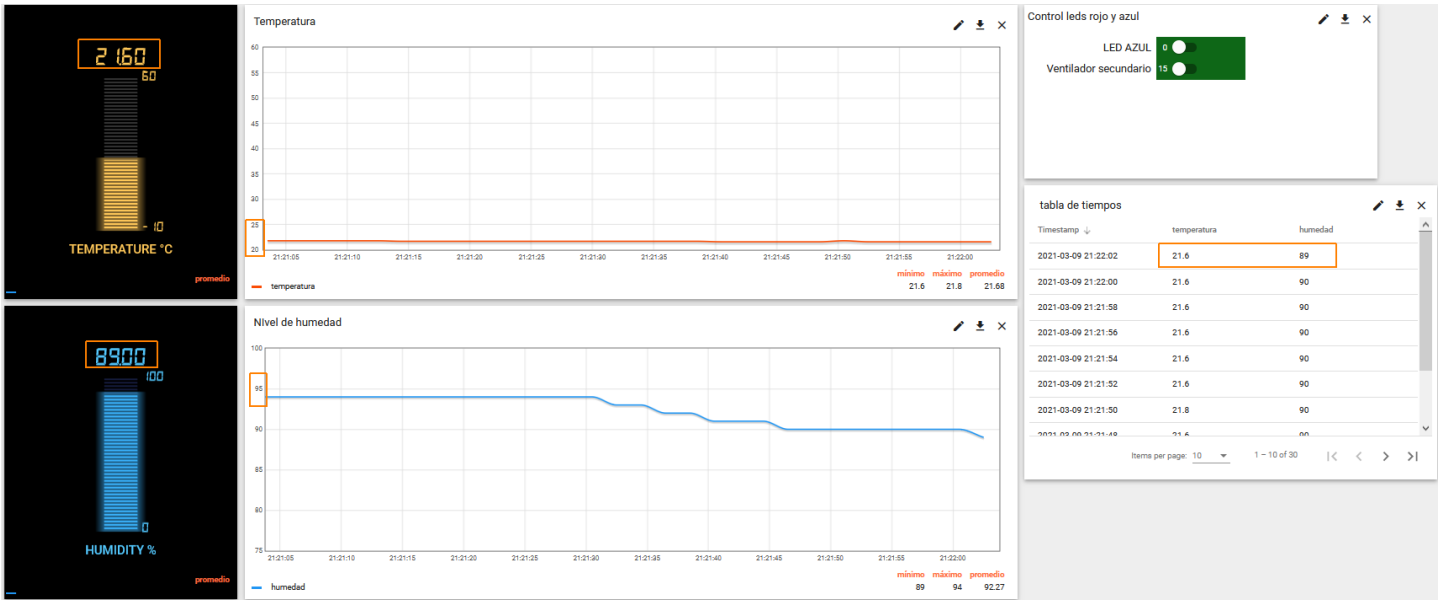


Prueba de funcionamiento.

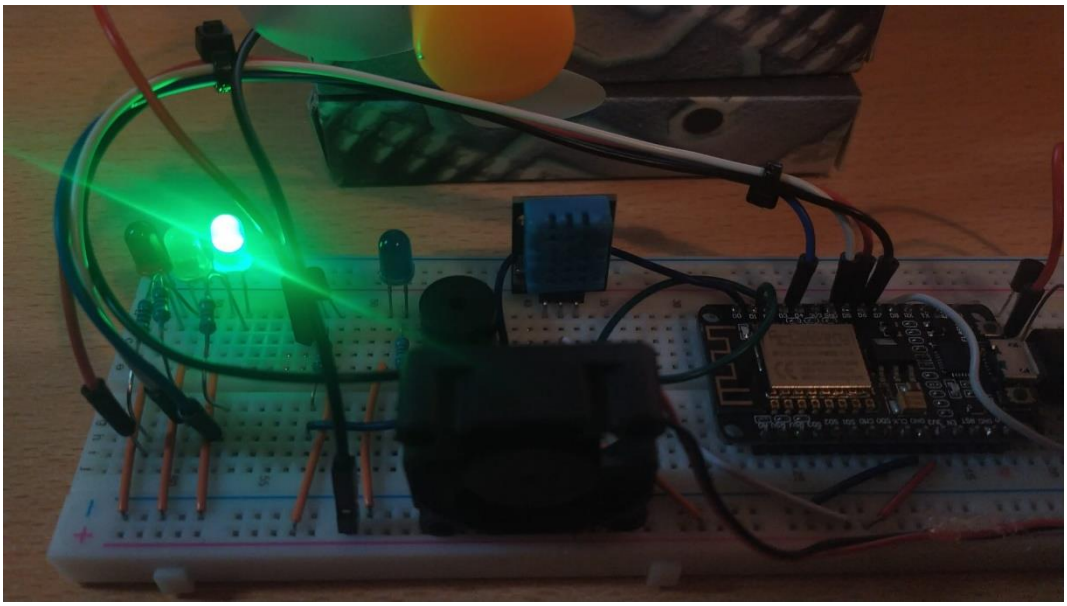
Al encender el sistema ponemos el monitor serie con el IDE de Arduino para visualizar las temperaturas:

```
COM6

Obteniendo datos de temperatura:
Humedad: 88.00 %        Temperatura: 21.20°C Enviando telemetria... {"temperatura":21.20,"humedad":88.00}
Obteniendo datos de temperatura:
Humedad: 88.00 %        Temperatura: 21.20°C Enviando telemetria... {"temperatura":21.20,"humedad":88.00}
Obteniendo datos de temperatura:
Humedad: 88.00 %        Temperatura: 21.20°C Enviando telemetria... {"temperatura":21.20,"humedad":88.00}
Obteniendo datos de temperatura:
Humedad: 88.00 %        Temperatura: 21.20°C Enviando telemetria... {"temperatura":21.20,"humedad":88.00}
Obteniendo datos de temperatura:
Humedad: 94.00 %        Temperatura: 21.20°C Enviando telemetria... {"temperatura":21.20,"humedad":94.00}
```

y desde la plataforma empiezan a llegar los datos:



Si visualizamos los leds del montajes vemos que el led verde está iluminado ya que el rango de temperatura esta en el baremo establecido(entre 20 y 30 ºC).El buzzer pita cada vez que se transmite la telemetría(2 seg)con un sonido grave.



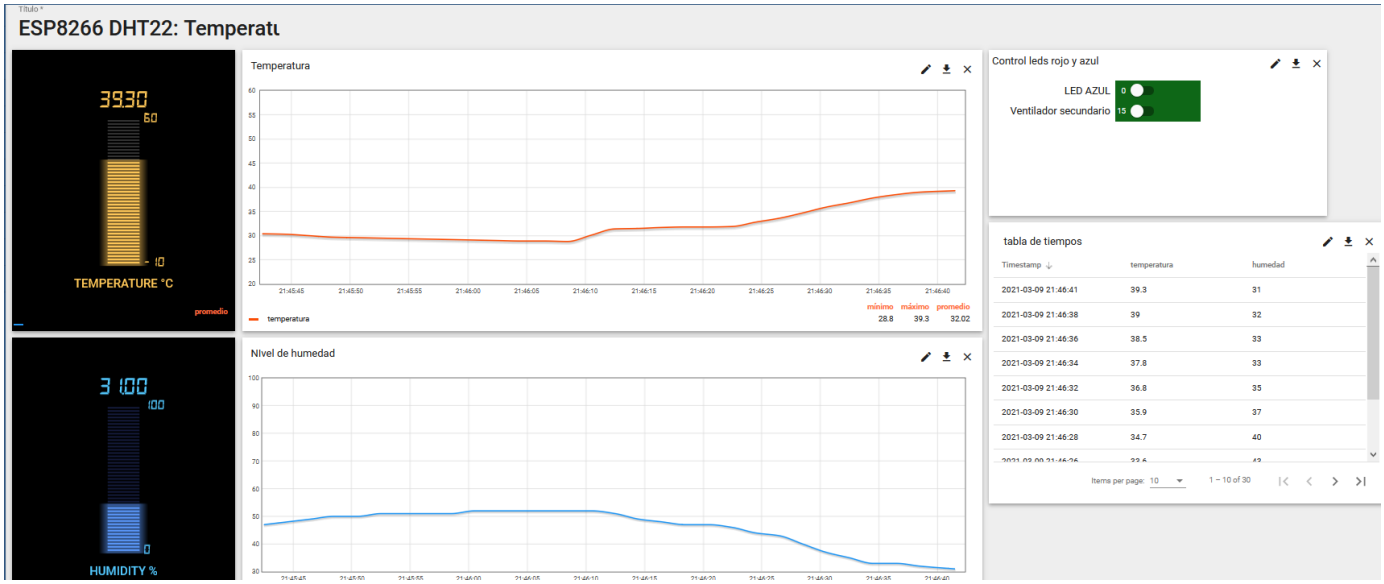
Si aumentamos un poco la temperatura del sensor,el led verde se apaga y se ilumina el led amarillo.El sonido del buzzer se hace mas agudo.Cuando se restablece la temperatura el sonido volverá a ser grave y el led amarillo se apagará para encenderse el verde.

Temperaturas en el monitor serie:

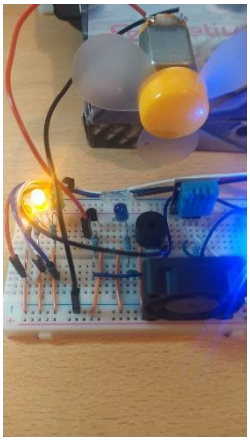
COM6

```
Obteniendo datos de temperatura:
Humedad: 82.00 %        Temperatura: 22.70°C Enviando telemetria... {"temperatura":22.70,"humedad":82.00}
Obteniendo datos de temperatura:
Humedad: 82.00 %        Temperatura: 22.70°C Enviando telemetria... {"temperatura":22.70,"humedad":82.00}
Obteniendo datos de temperatura:
Humedad: 82.00 %        Temperatura: 22.70°C Enviando telemetria... {"temperatura":22.70,"humedad":82.00}
Obteniendo datos de temperatura:
Humedad: 80.00 %        Temperatura: 23.70°C Enviando telemetria... {"temperatura":23.70,"humedad":80.00}
Obteniendo datos de temperatura:
Humedad: 74.00 %        Temperatura: 26.30°C Enviando telemetria... {"temperatura":26.30,"humedad":74.00}
Obteniendo datos de temperatura:
Humedad: 68.00 %        Temperatura: 29.30°C Enviando telemetria... {"temperatura":29.30,"humedad":68.00}
Obteniendo datos de temperatura:
Humedad: 58.00 %        Temperatura: 33.40°C Enviando telemetria... {"temperatura":33.40,"humedad":58.00}
Obteniendo datos de temperatura:
Humedad: 47.00 %        Temperatura: 37.70°C Enviando telemetria... {"temperatura":37.70,"humedad":47.00}
```

Y en la plataforma:



Led amarillo encendido:



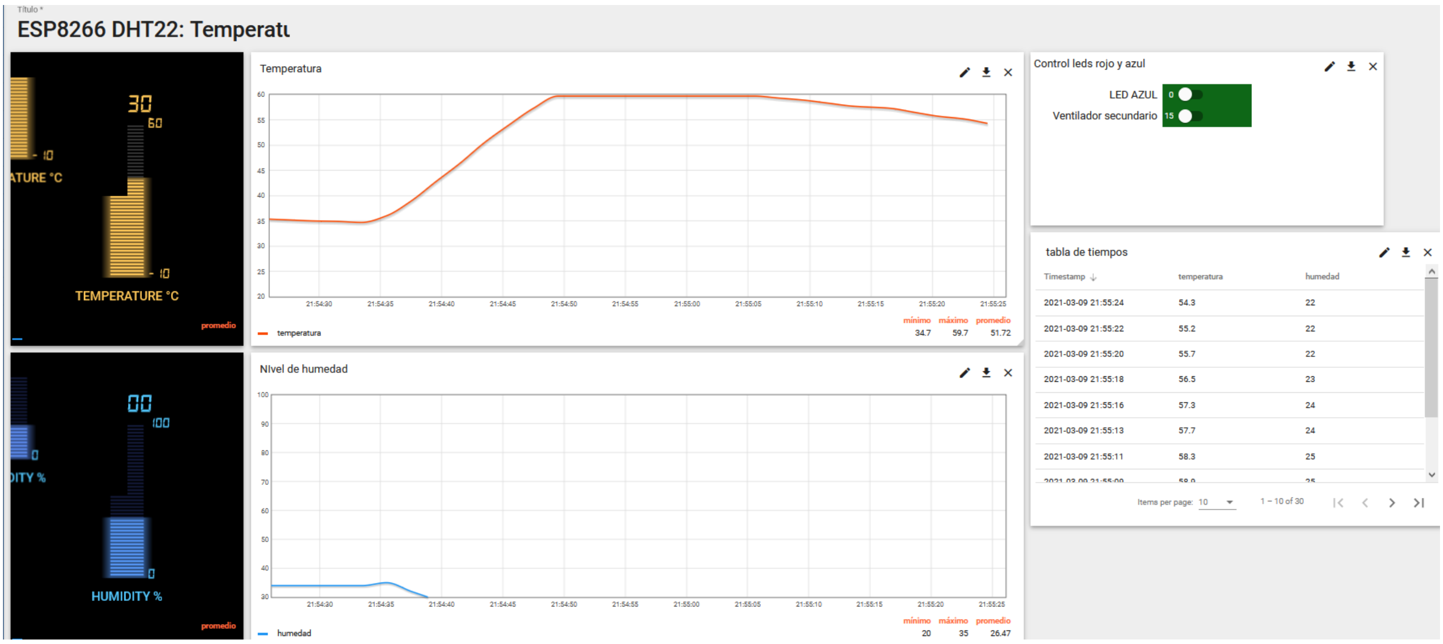
Si aumentamos la temperatura por encima de los 40 grados,se ilumina el led rojo,el buzzer pita con un sonido mas agudo y se enciende el ventilador automaticamente.Cuando baje la temperatura el ventilador se apagará y el led rojo también.

Lecturas del monitor serie:

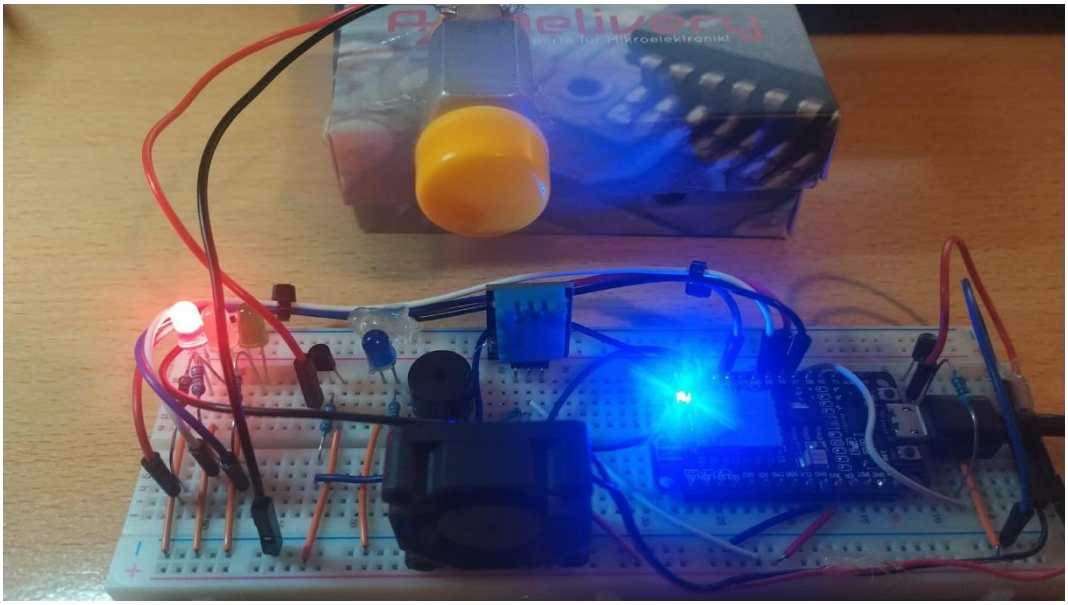

COM6

```
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 59.70°C Enviando telemetria... {"temperatura":59.70,"humedad":25.00}
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 59.70°C Enviando telemetria... {"temperatura":59.70,"humedad":25.00}
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 59.70°C Enviando telemetria... {"temperatura":59.70,"humedad":25.00}
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 59.70°C Enviando telemetria... {"temperatura":59.70,"humedad":25.00}
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 59.70°C Enviando telemetria... {"temperatura":59.70,"humedad":25.00}
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 59.70°C Enviando telemetria... {"temperatura":59.70,"humedad":25.00}
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 59.30°C Enviando telemetria... {"temperatura":59.30,"humedad":25.00}
Obteniendo datos de temperatura:
Humedad: 25.00 %        Temperatura: 58.90°C Enviando telemetria... {"temperatura":58.90,"humedad":25.00}
```

Lecturas de la plataforma:



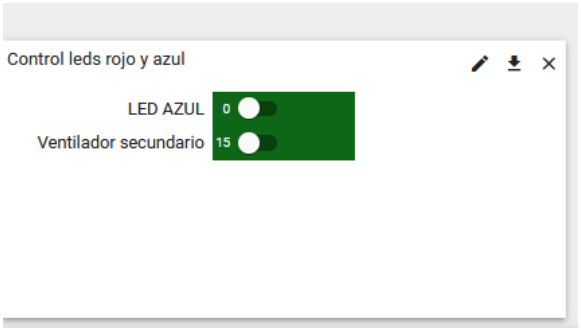
Led rojo y ventilador funcionando:

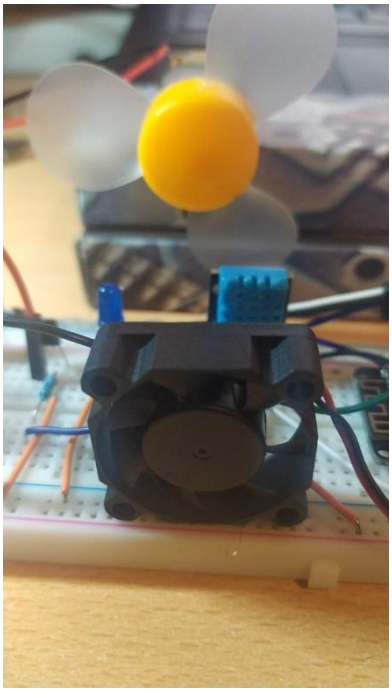


Accionamiento de led Azul y ventilador auxiliar desde la plataforma

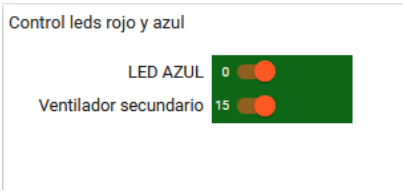
Para accionar el led azul simplemente le damos al widget del panel que se encarga de la gestion de los gpio:

Led azul y ventilador secundario apagados:





Led azul y ventilador funcionando:



Como ideas se puede montar un visor lcd para visualizar las temperaturas y la humedad sin necesidad de tener que recurrir al monitor serie o a la plataforma y la posibilidad de alimentar el proyecto con una bateria en lugar del cable usb.

David Sánchez Jaramillo