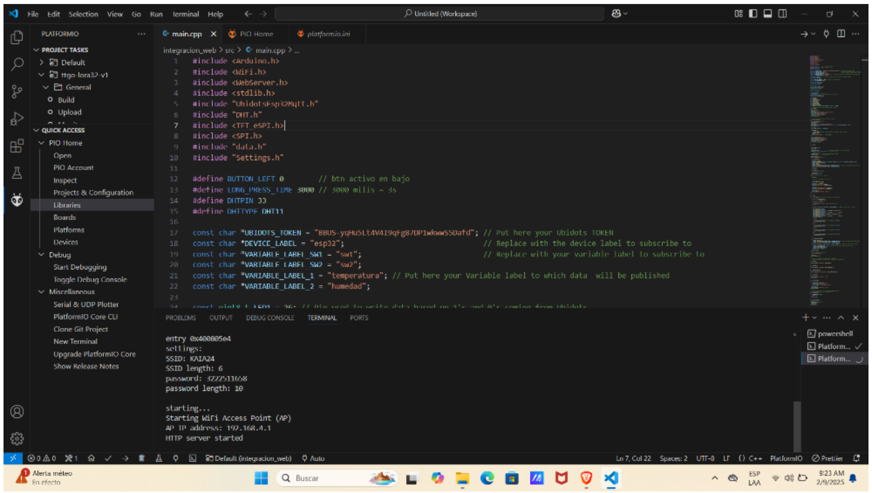
Práctica 3 – Diseño Electrónico

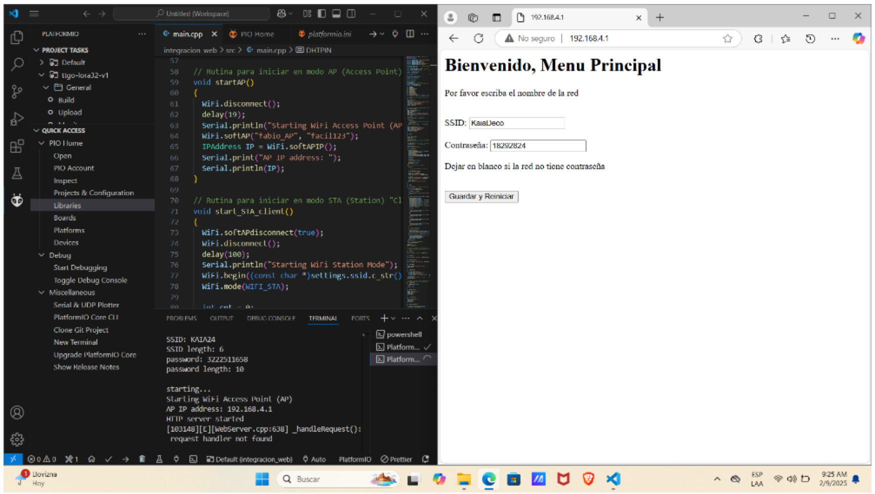
Evidencias de Funcionamiento



integracion_web > src > main.cpp > ...

```cpp
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <stdlib.h>
#include "UbidotsEsp32Mqtt.h"
#include "DHT.h"
#include <TFT_eSPI.h>
#include <SPI.h>
#include "data.h"
#include "Settings.h"

#define BUTTON_LEFT 0        // btn activo en bajo
#define LONG_PRESS_TIME 3000 // 3000 milis = 3s
#define DHTPIN 33
#define DHTTYPE DHT11

const char *UBIDOTS_TOKEN = "BBUS-yqHu5Lt4V419qFg87DP1wkwwS5Dafd"; // Put here your Ubidots TOKEN
const char *DEVICE_LABEL = "esp32";                                 // Replace with the device label to subscribe to
const char *VARIABLE_LABEL_SW1 = "sw1";                             // Replace with your variable label to subscribe to
const char *VARIABLE_LABEL_SW2 = "sw2";
const char *VARIABLE_LABEL_1 = "temperatura"; // Put here your Variable label to which data  will be published
const char *VARIABLE_LABEL_2 = "humedad";
```

```
entry 0x400805e4
settings:
SSID: KAIA24
SSID length: 6
password: 3222511658
password length: 10

starting...
Starting WiFi Access Point (AP)
AP IP address: 192.168.4.1
HTTP server started
```



integracion_web > src > main.cpp > DHTPIN

```cpp
// Rutina para iniciar en modo AP (Access Point)
void startAP()
{
  WiFi.disconnect();
  delay(19);
  Serial.println("Starting WiFi Access Point (AP
  WiFi.softAP("fabio_AP", "facil123");
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
}

// Rutina para iniciar en modo STA (Station) "Cl
void start_STA_client()
{
  WiFi.softAPdisconnect(true);
  WiFi.disconnect();
  delay(100);
  Serial.println("Starting WiFi Station Mode");
  WiFi.begin((const char *)settings.ssid.c_str()
  WiFi.mode(WIFI_STA);
```

```
SSID: KAIA24
SSID length: 6
password: 3222511658
password length: 10

starting...
Starting WiFi Access Point (AP)
AP IP address: 192.168.4.1
HTTP server started
[103148][E][WebServer.cpp:638] _handleRequest():
 request handler not found
```

192.168.4.1

Bienvenido, Menu Principal

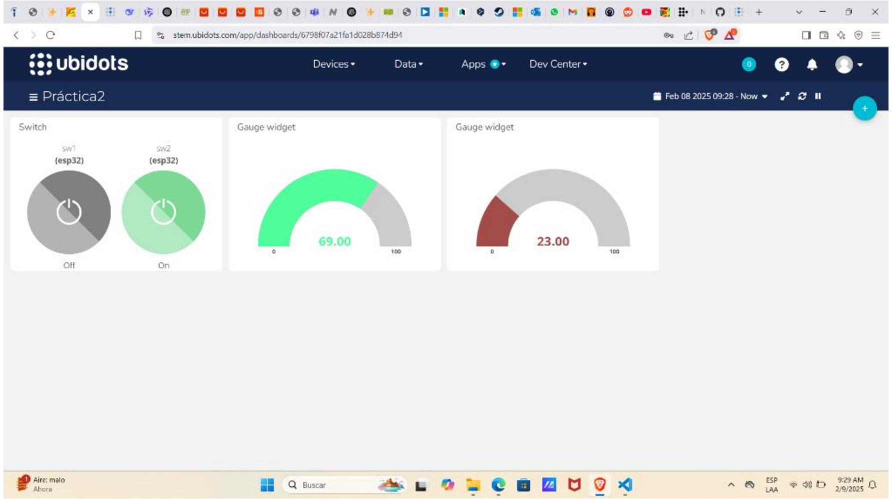
Por favor escriba el nombre de la red

SSID: KaiaDeco

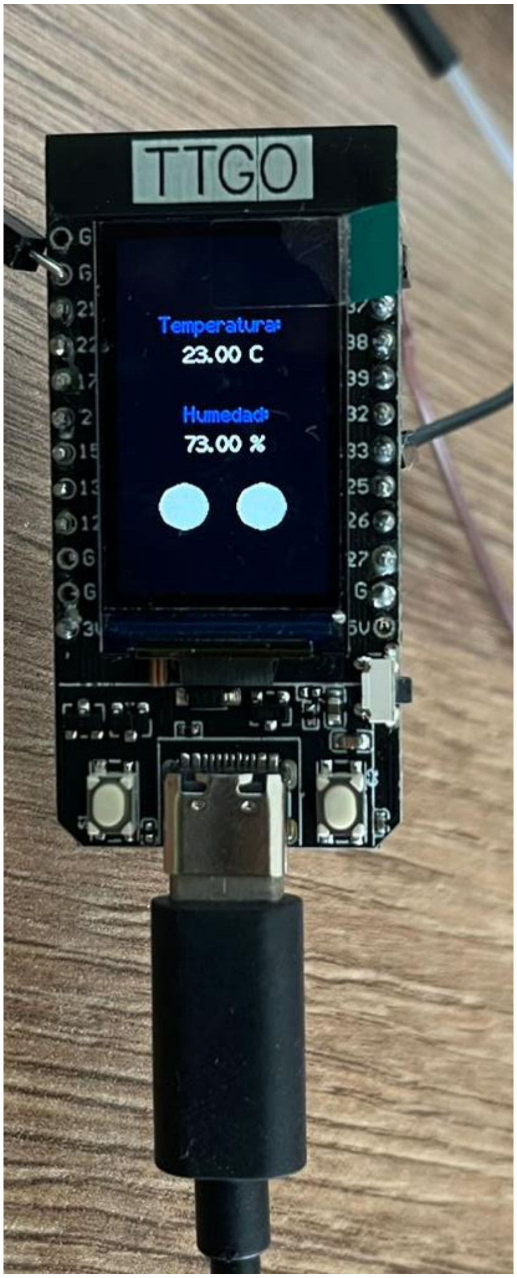
Contraseña: 18292824

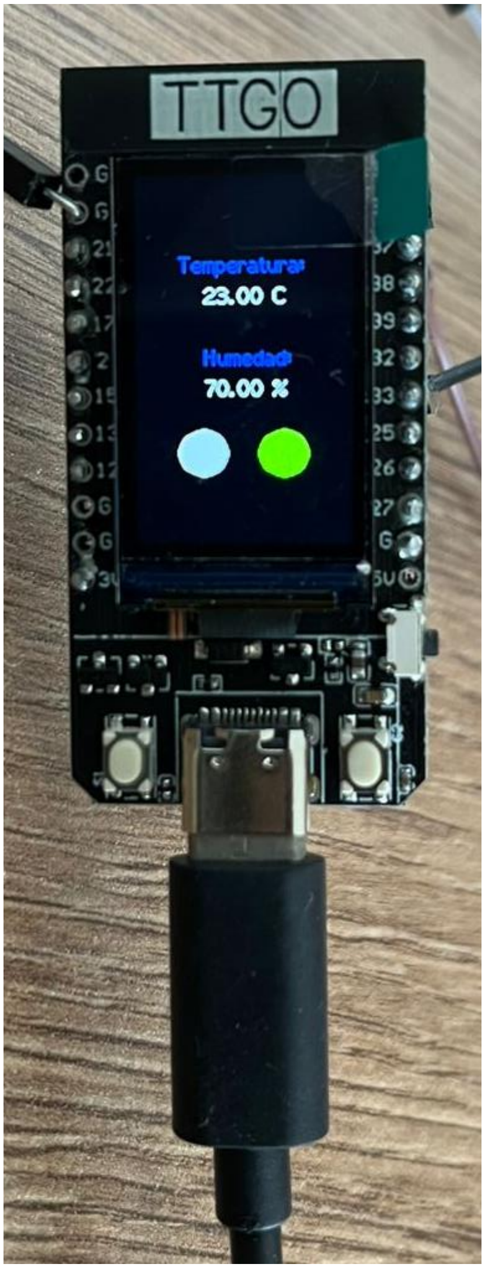
Dejar en blanco si la red no tiene contraseña

Guardar y Reiniciar



stem.ubidots.com/app/dashboards/6798f07a21fa1d028b874d94

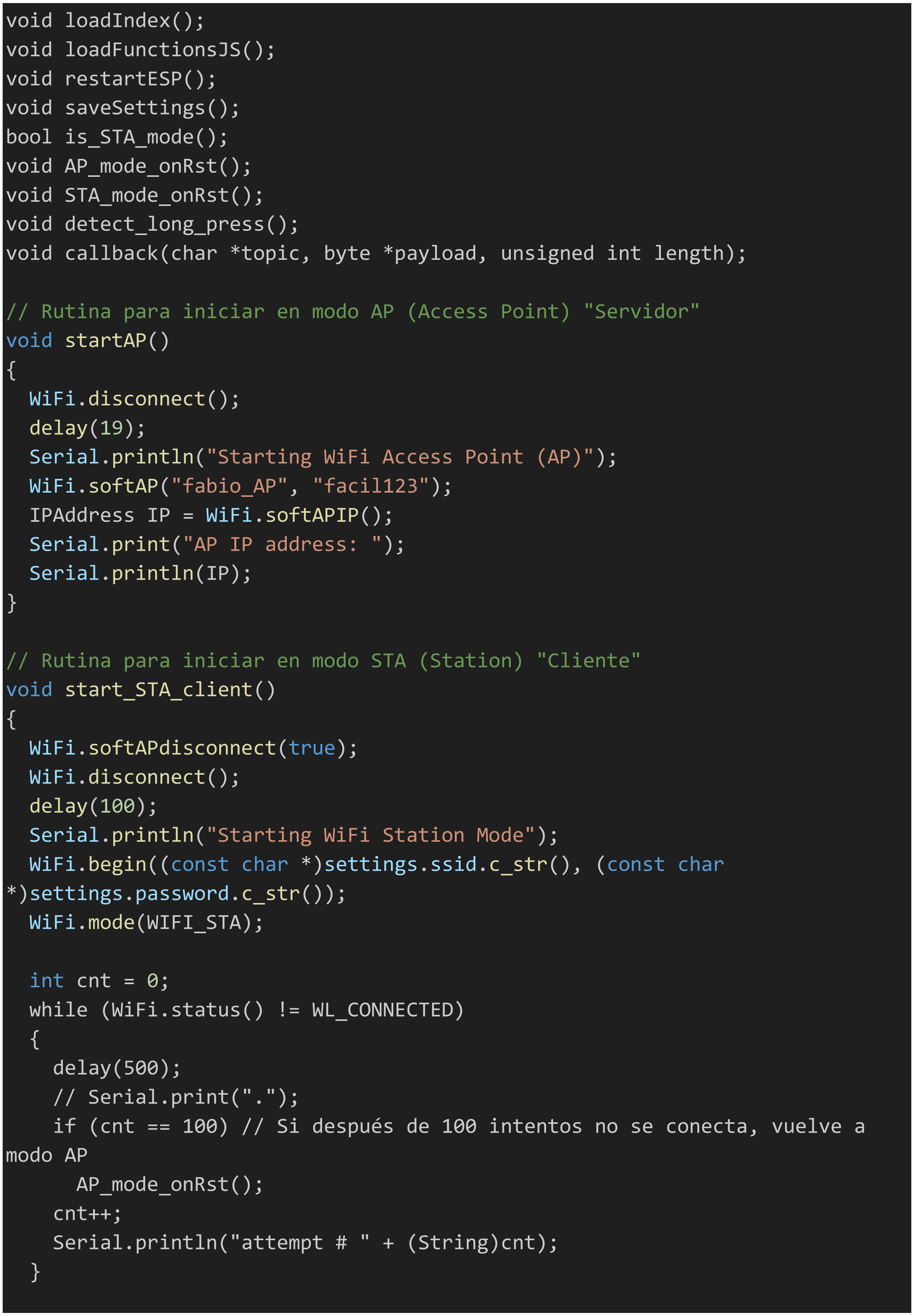




Código

```
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <stdlib.h>
#include "UbidotsEsp32Mqtt.h"
#include "DHT.h"
#include <TFT_eSPI.h>
```

```
#include <SPI.h>
#include "data.h"
#include "Settings.h"

#define BUTTON_LEFT 0        // btn activo en bajo
#define LONG_PRESS_TIME 3000 // 3000 milis = 3s
#define DHTPIN 33
#define DHTTYPE DHT11

const char *UBIDOTS_TOKEN = "BBUS-yqHu5Lt4V4I9qFg87DP1wkwwS5Dafd"; // Put
here your Ubidots TOKEN
const char *DEVICE_LABEL = "esp32";                                //
Replace with the device label to subscribe to
const char *VARIABLE_LABEL_SW1 = "sw1";                            //
Replace with your variable label to subscribe to
const char *VARIABLE_LABEL_SW2 = "sw2";
const char *VARIABLE_LABEL_1 = "temperatura"; // Put here your Variable
label to which data  will be published
const char *VARIABLE_LABEL_2 = "humedad";

const uint8_t LED1 = 26; // Pin used to write data based on 1's and 0's
coming from Ubidots
const uint8_t LED2 = 25;

const int PUBLISH_FREQUENCY = 5000; // Update rate in milliseconds
const int OLED_FREQUENCY = 1000;

unsigned long publish_timer;
unsigned long oled_timer;

bool sw1, sw2 = false;

Ubidots ubidots(UBIDOTS_TOKEN);
TFT_eSPI tft = TFT_eSPI();
DHT dht(DHTPIN, DHTTYPE);

WebServer server(80);

Settings settings;
int lastState = LOW; // para el btn
int currentState;    // the current reading from the input pin
unsigned long pressedTime = 0;
unsigned long releasedTime = 0;

void load404();
```



```
void loadIndex();
void loadFunctionsJS();
void restartESP();
void saveSettings();
bool is_STA_mode();
void AP_mode_onRst();
void STA_mode_onRst();
void detect_long_press();
void callback(char *topic, byte *payload, unsigned int length);

// Rutina para iniciar en modo AP (Access Point) "Servidor"
void startAP()
{
  WiFi.disconnect();
  delay(19);
  Serial.println("Starting WiFi Access Point (AP)");
  WiFi.softAP("fabio_AP", "facil123");
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
}

// Rutina para iniciar en modo STA (Station) "Cliente"
void start_STA_client()
{
  WiFi.softAPdisconnect(true);
  WiFi.disconnect();
  delay(100);
  Serial.println("Starting WiFi Station Mode");
  WiFi.begin((const char *)settings.ssid.c_str(), (const char
*)settings.password.c_str());
  WiFi.mode(WIFI_STA);

  int cnt = 0;
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    // Serial.print(".");
    if (cnt == 100) // Si después de 100 intentos no se conecta, vuelve a
modo AP
      AP_mode_onRst();
    cnt++;
    Serial.println("attempt # " + (String)cnt);
  }
  WiFi.setAutoReconnect(true);
  Serial.println(F("WiFi connected"));
  Serial.println(F("IP address: "));
  Serial.println(WiFi.localIP());
  pressedTime = millis();
  // Rutinas de Ubidots
}

void setup()
{

  Serial.begin(115200);
  delay(2000);

  EEPROM.begin(4096);                  // Se inicializa la EEPROM con su
tamaño max 4KB
  pinMode(BUTTON_LEFT, INPUT_PULLUP); // btn activo en bajo

  // settings.reset();
  settings.load(); // se carga SSID y PWD guardados en EEPROM
  settings.info(); // ... y se visualizan

  Serial.println("");
  Serial.println("starting...");

  if (is_STA_mode())
  {
    start_STA_client();
    ubidots.connectToWifi((const char *)settings.ssid.c_str(), (const char
*)settings.password.c_str());
    ubidots.setCallback(callback);
    ubidots.setup();
    ubidots.reconnect();
    ubidots.subscribeLastValue(DEVICE_LABEL, VARIABLE_LABEL_SW1); // Insert
the device and variable's Labels, respectively
    ubidots.subscribeLastValue(DEVICE_LABEL, VARIABLE_LABEL_SW2);

    dht.begin();

    tft.init();
    tft.fillScreen(TFT_BLACK);

    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
  }
  else // Modo Access Point & WebServer
  {
    startAP();

    /* ========== Modo Web Server ========== */

    /* HTML sites */
    server.onNotFound(load404);

    server.on("/", loadIndex);
    server.on("/index.html", loadIndex);
    server.on("/functions.js", loadFunctionsJS);

    /* JSON */
    server.on("/settingsSave.json", saveSettings);
    server.on("/restartESP.json", restartESP);

    server.begin();
    Serial.println("HTTP server started");
  }
}

void loop()
{
  if (is_STA_mode()) // Rutina para modo Station (cliente Ubidots)
  {
    float h = dht.readHumidity();
    float t = dht.readTemperature();

    if (isnan(h) || isnan(t))
    {
      Serial.println(F("Failed to read from DHT sensor!"));
      return;
    }

    if (abs((int)(millis() - oled_timer)) > OLED_FREQUENCY) // triggers the routine every 5 seconds
    {
      tft.fillScreen(TFT_BLACK);
      tft.setTextColor(TFT_BLUE);
      tft.drawString("Temperatura: ", tft.width() / 2 - 5 * 8, 50, 2);
      tft.setTextColor(TFT_WHITE);
      tft.drawString(String(t) + " C", tft.width() / 2 - 3 * 8, 70, 2);
      tft.setTextColor(TFT_BLUE);
      tft.drawString("Humedad: ", tft.width() / 2 - 3 * 8, 110, 2);
      tft.setTextColor(TFT_WHITE);
      tft.drawString(String(h) + " %", tft.width() / 2 - 3 * 8, 130, 2);
      oled_timer = millis();

      if (sw1)
      {
        tft.fillCircle(tft.width() / 2 - 25, 180, 15, TFT_GREEN);
        digitalWrite(LED1, HIGH);
      }
      else
      {
        tft.fillCircle(tft.width() / 2 - 25, 180, 15, TFT_SILVER);
        digitalWrite(LED1, LOW);
      }
      if (sw2)
      {
        tft.fillCircle(tft.width() / 2 + 25, 180, 15, TFT_GREEN);
        digitalWrite(LED2, HIGH);
      }
      else
      {
        tft.fillCircle(tft.width() / 2 + 25, 180, 15, TFT_SILVER);
        digitalWrite(LED2, LOW);
      }
    }

    if (!ubidots.connected())
    {
      ubidots.reconnect();
      ubidots.subscribeLastValue(DEVICE_LABEL, VARIABLE_LABEL_SW1); //
Insert the device and variable's Labels, respectively
      ubidots.subscribeLastValue(DEVICE_LABEL, VARIABLE_LABEL_SW2);
    }
    if (abs((int)(millis() - publish_timer)) > PUBLISH_FREQUENCY) //
triggers the routine every 5 seconds
    {
      ubidots.add(VARIABLE_LABEL_1, t); // Insert your variable Labels and
the value to be sent
      ubidots.add(VARIABLE_LABEL_2, h);
      ubidots.publish(DEVICE_LABEL);
      publish_timer = millis();
    }
    ubidots.loop();
  }
  else // rutina para AP + WebServer
```

```
    server.handleClient();

  delay(10);
  detect_long_press();
}

// funciones para responder al cliente desde el webserver:
// load404(), loadIndex(), loadFunctionsJS(), restartESP(), saveSettings()

void load404()
{
  server.send(200, "text/html", data_get404());
}

void loadIndex()
{
  server.send(200, "text/html", data_getIndexHTML());
}

void loadFunctionsJS()
{
  server.send(200, "text/javascript", data_getFunctionsJS());
}

void restartESP()
{
  server.send(200, "text/json", "true");
  ESP.restart();
}

void saveSettings()
{
  if (server.hasArg("ssid"))
    settings.ssid = server.arg("ssid");
  if (server.hasArg("password"))
    settings.password = server.arg("password");

  settings.save();
  server.send(200, "text/json", "true");
  STA_mode_onRst();
}

// Rutina para verificar si ya se guardó SSID y PWD del cliente
// is_STA_mode retorna true si ya se guardaron
bool is_STA_mode()
```

```
{
  if (EEPROM.read(flagAdr))
    return true;
  else
    return false;
}

void AP_mode_onRst()
{
  EEPROM.write(flagAdr, 0);
  EEPROM.commit();
  delay(100);
  ESP.restart();
}

void STA_mode_onRst()
{
  EEPROM.write(flagAdr, 1);
  EEPROM.commit();
  delay(100);
  ESP.restart();
}

void detect_long_press()
{
  // read the state of the switch/button:
  currentState = digitalRead(BUTTON_LEFT);

  if (lastState == HIGH && currentState == LOW) // button is pressed
    pressedTime = millis();
  else if (lastState == LOW && currentState == HIGH)
  { // button is released
    releasedTime = millis();

    // Serial.println("releasedtime" + (String)releasedTime);
    // Serial.println("pressedtime" + (String)pressedTime);

    long pressDuration = releasedTime - pressedTime;

    if (pressDuration > LONG_PRESS_TIME)
    {
      Serial.println("(Hard reset) returning to AP mode");
      delay(500);
      AP_mode_onRst();
    }
  }

  // save the the last state
  lastState = currentState;
}

void callback(char *topic, byte *payload, unsigned int length)
{
  if (strstr(topic, "sw1") != NULL)
  {
    if ((char)payload[0] == '1')
    {
      sw1 = true;
    }
    else
    {
      sw1 = false;
    }
  }
  if (strstr(topic, "sw2") != NULL)
  {
    if ((char)payload[0] == '1')
    {
      sw2 = true;
    }
    else
    {
      sw2 = false;
    }
  }
}
```