Indoor para autocultivo de marihuana

la idea principal de este indoor es que sea pueda estar pendiente de las necesidades básicas de las plantas y proporcionarlas mientras el dueño no esta.

Son los principales items que requieren las plantas son:

1. Agua
2. Luz
3. Aire
4. humedad y temperatura ideales en ambiente
5. nutrientes

Materiales que se pueden explorar

- Display QVGA 2.2 TFT SPI 240x320
- DHT22 digital temperature and humidity sensor
- Soil Hygrometer Humidity Detection Module
- Dispenser Flowmeter Flow Sensor. Inner diameter 3mm DC 5-24v
- Bomba de riego a 12v 60w 5L/min
- Bomba peristáltica de 5v

Aquí se escribirán ideas sueltas para llevar a cabo, que a largo plazo; serán implementadas en el indoor.

Cómo envíar datos a influxdb de algún sensor

Firmware para el ESP8266

Parte del código se toma de acá

```
// Mirar los ejemplos de código que trae el dht adafruit sensor para
entender lo concerniente al dht11

#include "DHT.h"
#include <ESP8266HTTPClient.h>
#include <ESP8266Wifi.h>

#define DHTPIN D5 // Pin que va conectado al sensor
#define DHTTYPE DHT11 // Tipo de sensor que estamos usando
#define HTTP_TIMEOUT 1000 * 60 // cada minuto

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  Serial.println(F("DHTxx test!"));
  dht.begin();
  // nombre del wifi y clave del wifi al cual se va a conectar el esp
  WiFi.begin("RAMIREZ_BO", "10101973");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("connection successfull !");
}

// función que prepara la trama de datos para hacer un POST a endpoint del
influx
String influxFrame( String dht11_humidity, String dht11_temperature ) {
  // este es el nombre del sensor
  // Siempre que se quema la primera vez, se debe de cambiar el nombre del
sensor
  const String SENSOR_ID = "DHT11_llanadas"; // Nombre del sensor en la
plataforma

  const String STR_COMMA = ",";
  const String STR_SLASH = "/";
  const String STR_DOT = ".";
  const String STR_COLON = ":";
  const String STR_NULL = "NULL";
  const String STR_ZERO = "0";
  const String STR_SPACE = " ";

  // El primer dato en el squema de la DB es el id del sensor
  String frame = SENSOR_ID + STR_COMMA + "id=" + SENSOR_ID +  STR_SPACE;

  // Add GPS data
  frame += "lat=";
```

```
  frame += "6.2563143" + STR_COMMA; // coordenada GSP lat
  frame += "lng=";
  frame += "-75.5386472" + STR_COMMA; // coordenada lng lat
  frame += "altitude=";
  frame += STR_ZERO + STR_COMMA;
  frame += "course=";
  frame += STR_ZERO + STR_COMMA;
  frame += "speed=";
  frame += STR_ZERO + STR_COMMA;

  //Add DHT11 data
  //if
    frame += "humidity=";
    frame += dht11_humidity + STR_COMMA;
    frame += "temperature=";
    frame += dht11_temperature + STR_COMMA;
  // } else {
  //   frame += "humidity=" + STR_NULL + STR_COMMA + "temperature=" +
STR_NULL + STR_COMMA;
  // }

  // Add Plantower data
    // if
    frame += "pm1=";
    frame += STR_ZERO + STR_COMMA;
    frame += "pm25=";
    frame += STR_ZERO + STR_COMMA;
    frame += "pm10=";
    frame += STR_ZERO;
  // } else {
  //   frame += "pm1=" + STR_NULL + STR_COMMA + "pm25=" + STR_NULL +
STR_COMMA + "pm10=" + STR_NULL;
  // }

  return frame;
}

// función que envía la trama de datos
void sendDataInflux ( String humidity, String temperature ) {
  /*
  El post a la base de datos tiene una trama siguiente:
  // volker0001,id=volker0001
lat=6.268115,lng=-75.543407,altitude=1801.1,course=105.55,speed=0.00,humidit
y=37.00,temperature=25.00,pm1=22,pm25=31,pm10=32
  Para nuestro caso que SOLO es el envío de datos del dht_11 que es humedad
y temperatura la trama es la siguiente
  // DHT11_llanadas, id=DHT11_llanadas, lat=6.2563143, lng=-75.5386472,
altitude=0, course=0, speed=0, humidity=37.00, temperature=25.00, pm1=0,
pm25=0, pm10=0 1434055562000000000
  */
```

```
  HTTPClient http;
  // _testsensorhumedad es el nombre de la DB donde se almacenan estos datos
  http.begin("http://aqa.unloquer.org:8086/write?db=_testsensorhumedad"); //
endPoint final, '_testsensorhumedad' es el nombre de la base de datos
  http.setTimeout(HTTP_TIMEOUT);
  http.addHeader("Content-Type", "--data-binary");

  String frame = influxFrame(humidity, temperature); // Construimos el
request POST

  int httpCode = http.POST(frame); // Envíamos los datos haciendo un POST

  if(httpCode > 0) {
    String payload = http.getString();
    Serial.println(payload);
    Serial.println("Envío de datos con exito!");
  } else {
    Serial.print("[HTTP] failed, error;");
    Serial.println(http.errorToString(httpCode).c_str());
  }

  http.end();
  delay(60000); // cada minuto se envía un POST al influx
}

void loop() {
  // esperamos 5 segundos entre lecturas y lectura
  // El sensor de humedad o temperatura toma alrededor de 250 milisegundos
  // o hasta dos segundos entre lectura y lectura. Es un sensor muy lento
  // por eso se añade este de 2000
  delay(2000);

  float h = dht.readHumidity(); // leemos la temperatura en grados celcius
(Esta es la default del sensor)
  float t = dht.readTemperature();
  float f = dht.readTemperature(true); // Si queremos la temperatura en
fahrenheit, ponemos este en true

  // Si las lecturas fallan, salimos, no mandamos nada y volvemos a
intentarlo
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  //float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  //float hic = dht.computeHeatIndex(t, h, false);
```

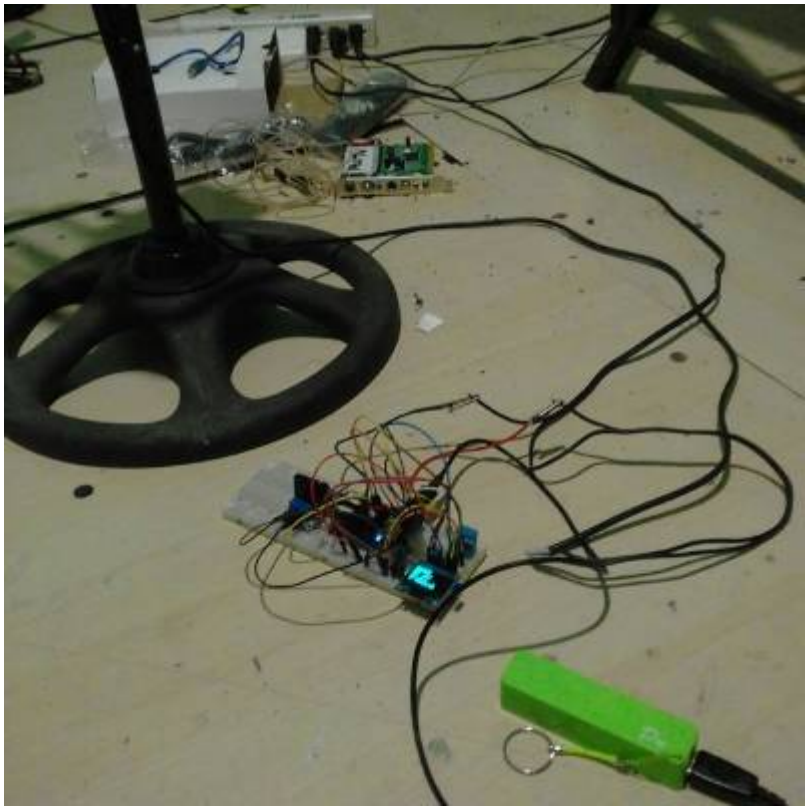
```
  // Serial.print(F("Lectura Humidity: "));
  // Serial.print(h);
  // Serial.print(F("% Lectura Temperature: "));
  // Serial.print(t);
  // Serial.print("\n");

  /*
  Serial.print(f);
  Serial.print(F("Â°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("Â°C "));
  Serial.print(hif);
  Serial.println(F("Â°F"));
  */

  sendDataInflux(String(h), String(t));
}
```

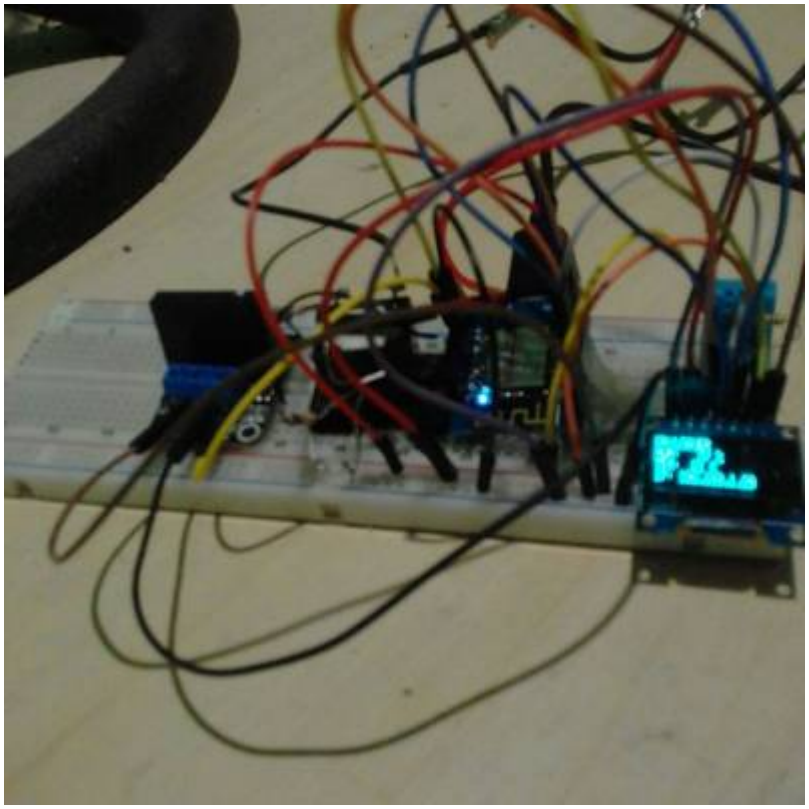
Configuración de plataforma

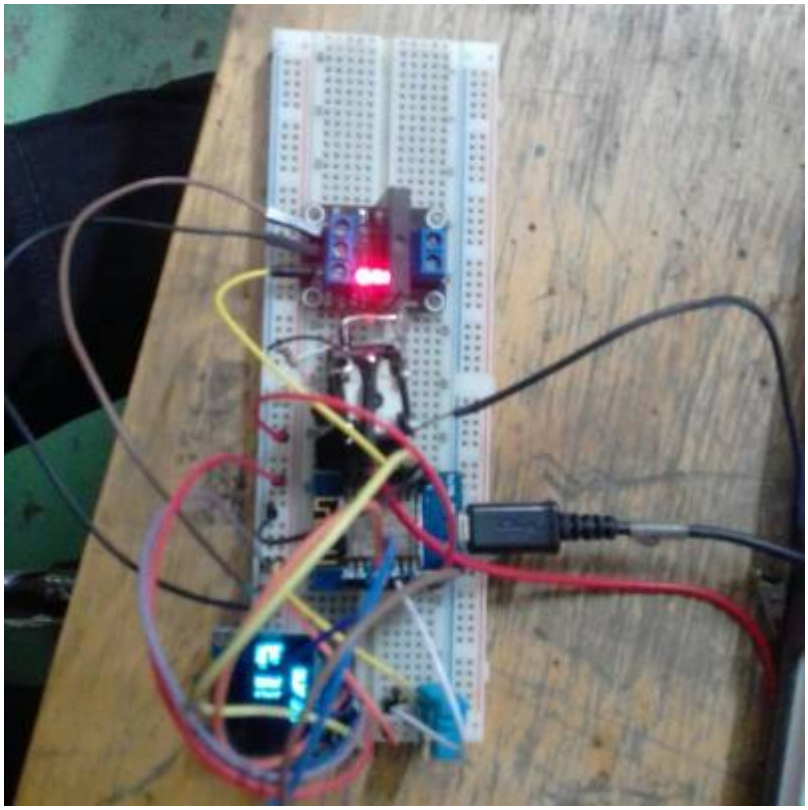
1. Se crea una base de datos

documentar esta parte de como crear base de datos y adjuntar al dashboard para ver los graficos enviados por algún sensor

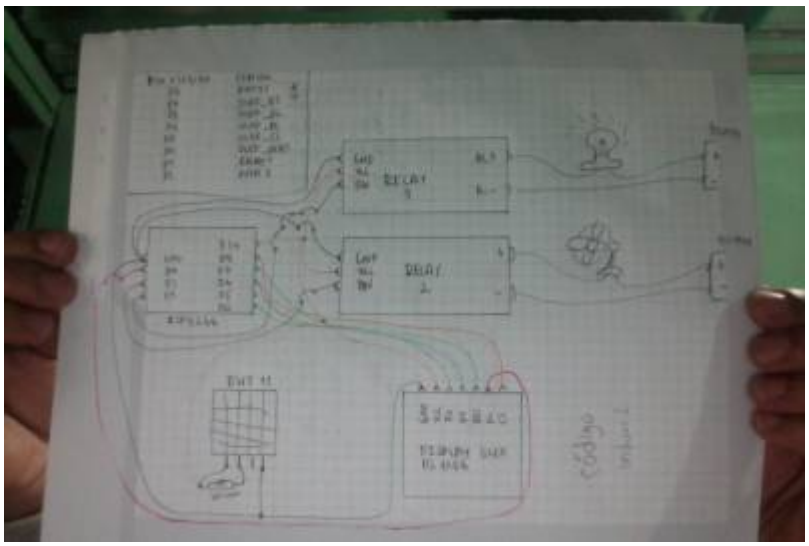
primer prototipo de control automatico

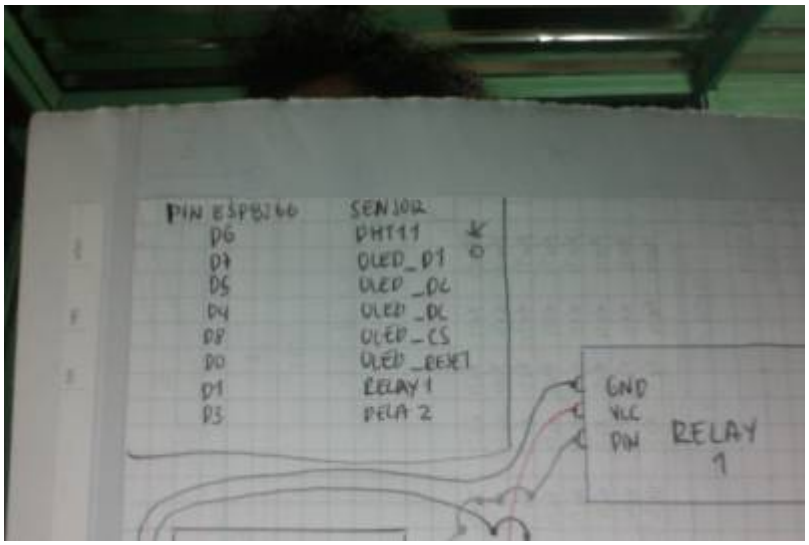







2% 1:47 AM
192.168.1.20
Automatic grow garden
RELAY 1
On
Off
Relay 2
On
Off





Se intenta manipular relays, mosttar datos en pantallay enviar datos a una base de datos influxdb

A ESTE CÖDIGO FALTA IMPLEMENTAR ENVIO DE DATOS AL INFLUX CON WEBSOCKETS <code c++> #include <ESP8266WiFi.h> #include <ESP8266WiFiMulti.h> #include <WebSocketsServer.h> #include <Hash.h> #include <ESP8266WebServer.h> #include <ESP8266mDNS.h> #include <SPI.h> #include <Wire.h> #include <Adafruit_GFX.h> #include <Adafruit_SSD1306.h> #include <ESP8266HTTPClient.h> #include “DHT.h”

#define DHTPIN D6 *Pin que va conectado al sensor #define DHTTYPE DHT11* Tipo de sensor que estamos usando #define HTTP_TIMEOUT 1000 * 60 *cada minuto DHT dht(DHTPIN, DHTTYPE);* If using software SPI (the default case): #define OLED_MOSI D7 *D1 #define OLED_CLK D5* D0 #define OLED_DC D4 #define OLED_CS D8 #define OLED_RESET D0

Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);

static const char ssid[] = “d(O_O)b”; static const char password[] = “alex1988alex”; MDNSResponder mdns;

static void writeLED(bool);

ESP8266WiFiMulti WiFiMulti;

ESP8266WebServer server(80); WebSocketsServer webSocket = WebSocketsServer(81);

static const char PROGMEM INDEX_HTML[] = R“rawliteral(<!DOCTYPE html> <html> <head> <meta name = “viewport” content = “width = device-width, initial-scale = 1.0, maximum-scale = 1.0, user-scalable=0”> <title>ESP8266 WebSocket Demo</title> <style> “body { background-color: #808080; font-family: Arial, Helvetica, Sans-Serif; Color: #000000; }” </style> <script> var websock; var webSockInflux; *se tiene que mandar los datos por un websocket function start() { websock = new WebSocket('ws:*' + window.location.hostname + ':81/');

```
websock.onopen = function(evt) { console.log('websock open'); };
websock.onclose = function(evt) { console.log('websock close'); };
websock.onerror = function(evt) { console.log(evt); };
websock.onmessage = function(evt) {
  console.log(evt);
  var e = document.getElementById('ledstatus');
  var f = document.getElementById('ledstatus2');
  if (evt.data === 'ledon') {
    e.style.color = 'red';
  }
  else if (evt.data === 'ledoff') {
    e.style.color = 'black';
  }
  else if (evt.data === 'ledon1') {
    f.style.color = 'red';
  }
  else if (evt.data === 'ledoff1') {
    f.style.color = 'black';
  }
  else {
    console.log('unknown event');
  }
};
```

}

function buttonclick(e) {

```
websock.send(e.id);
```

}

</script> </head> <body onload=“javascript:start();”> <img src=“

XTyZbM9Hb4uI+urhSgp/ky2Z6O3xcR9dPJlsz0dvi4j66uFKCn+TLZno7fFxH108mWzPR2+LiPrq4UoKf5Mt
mejt8XEfXTyZbM9Hb4uI+urhSgp/ky2Z6O3xcR9dPJlsz0dvi4j66uFKCn+TLZno7fFxH108mWzPR2+LiPrq4
UoKf5Mtmejt8XEfXTyZbM9Hb4uI+urhSgp/ky2Z6O3xcR9dPJlsz0dvi4j66uFKCnHxZ7PHm45Y25mSacMP
WCWNcT4vFbKdftMz4nAswThZNZYCdFMjfnT1n5aA36o/b+CWbDTROOK8bqfcbHrBsfZRXcGvqKV8T2R4
wMRHBDGNQkaKCdTxVA119VKlWL74pPwuDrl/UerjVO8Un4XB1y/qPVxqslKUoFKVxzbRQEqt5HH5UGYj1
MeSn8xFB2UqPMc78phEvQlmc9bMMq9BAB9TV6Gyk3h57/2+IPyLEfKorupXAcA/NiZwOi2GPzaMn516GH
mG6YHtxqf6StB20rgWPEgayQMea0UifPhWrLSYkbo4G6SZZF+XBN31R3Uri+0TA6w37Min+oLWBj358N
OPXfDH5CQmiO6lcJ2mByo5x/4pW/oBp+1Y+fOO1HMv9Sig7qVxftfD3sZ4gegugPuJrfBi435Do1t+VlPdQb
qUpQKUpQKUpQKUpQKUpQK04zzb9hu41urTjPNv2G7jQfmbC8heyO6s1jC8heyO6s1lt9t8Un4XB1y/qPVx
qneKT8Lg65f1Hq41phhmA1OlRB8IImYpC0bsN5MiIo9p4x61Uj11s8IQTEBdwDJGGyBC1iwAADgqbtlBvzE
1ztM40Lzm43SYYye/glA9lRXWmDMmsspcH8qXRPkcze029VdsUSqAqgKBuAAAHUBVcaKP/qLhCQdS+Ek
itb1uTWVkjN8pwwBtbg8ZNGT6rKot76EWalQKIwAyJiSf4MSsnzmevYaUXu+MHWmDYD4a3NCJulQZx0gt
94QOfPg8Vc+0Mo+VejtcjQy4YnmDF4j1m+ahE1SomLajHcMOx5xHiA1vegreuPf0eW3SDhz3SXoR30ri/a
S88cw/wDFK3zUEVj9rw2uzhe2GT+oCqjurFq0RY+JtFljJ9Tqe41y7R2kVligAeZhcD8qKdOEkI3JvsN7EWHO
QHnau0ihWKFc87jirrZRuMjkbkHvJ0HqoGKkk+0RyO8ipMqxFmsGcvNGBKE1GXKGsdR5v1CvouzNnCIEli8
jWMkjWzMRu7Ki+ijQe8n51BeabBqkUgyPGQQ4a6iSQSS3uci3hPE9dhuqNYuCeCYBBXFYsW5hItj1jLb5Uf
wXYn/nMVb1MAflYfKrEKzSJVdPg7KORjJx2izH5MK9NsTEW0xbX6SJ/naYCrBSkKr0eyMWDf7UpHNdZzb/AD
bH3Vhtn48HTFREdBQjvzVYqUhVWmxWMglgEzxSLLKlrKLWurNmPFH7lt+9hoatNV3bvHxmEjsLWkkOv5k
aF0/wpL7qsQoFKUqoVpxnm37DdxrdWnGebfsN3Gg/M2F5C9kd1ZrGF5C9kd1ZrLb7b4pPwuDrl/UerjVO8
Un4XB1y/qPVxrTDm2jAXidRvKnL6mGqn2ECveEnDorjcyhh7QDW6uHZRsHj/ckYDstaRAOpXA9lFd1q8sg
O8A9deMQzAXRQx6C2X52NcWL2wseHbEMr5UuGWy5gQ2Rhvtob63tYXvaojofZsJNzDET0lEv77Vr/AGTD
ayplH8DOn9JFesNjizBHjdCVLLmyG4BAbkk2IzLoenrt2UVw/s0DkyTD/wAsp+TEin2J+bETe0QHvS9d1KCN
kwMhvx4mv+/CG99mF65zsjQDgcG3TaMx+62apqlCoX7C9yeBA6ODxWIHyyqBXng3G6PGDqmgf9SQ91
TlKQqAlW9+EOJltqGw8UgPw4yTWvwSMaQGViimaaUm4EdyrtFGoU7il40FukHpqx2qo7P2L9ow6ESFLSY
m+ga6vNIxGu7m/wDw2FBZ58bGhtJli9plX1c56apGwEU4qDLbzMR0tbKftDA+0s59dWLaGxs7u3BQSZo1
jBl5S2zbuI1wc1+aq94Esh4Nk/NJGxNt4+yHKerLb3Gir7SlKrJStGOxHBozhS2UXsN5rGAxPCIHKlb30PqJFx0
g2uD0EbqDopSuXaszJDI8YLOqMVABN2ANhYanWgh4hn2k5vcRwovZcZm+aYj/AA1YqrPgdd3xMz2zNKE
bX80QyE+0ZfYBVmqLpSlKqFacZ5t+w3ca3Vpxnm37DdxoPzNheQvZHdWaxheQvZHdWay2+2+KT8Lg6
5f1Hq41TvFJ+Fwdcv6j1ca0wVwcnEeqSP8AxRnvIl9yV31wbW0CSf8AxyK38rXjcn1BHZv5aK7zUJtPZQxEc
8cc+VZbq9gr5XAC3HRuW6+rTKSTU1UFjNlrEDkZxw0kKNqNE4QsUFgDY53W5JPGGugtB24nSaAk80iX6
WYKw+UbVI1DYXAL9/CoyIHR0C2GQlVN0FrCzqW3WuT010JtApxcQCpH51VijfxXF8nU3PoCd9Bl0qKxm2
BlPAK7tpYiOYqASAzXAs1hc2BubWGprs2fOzxhnUqTfQ3G4kBrHUXABsdRex1qo6aUpQKUJoDQKh/BM3w
qdcl+vhHv871MGobwT8ww5hiMWo6lxM6j5AVFTBr5x4u2W8OQgjgMKpsb8b7PIcpPStmHXevoG0fNSW/
cbuNfOvBxYjtGI4cqY7aGPiqxXDx3YrvtZltf+L13aY+m0pSqhSlKBWvEzBEZ23KCx6gLnurZUT4VOBhZQdz
gRfFIj/30GjwMiK4VC98zEI7i3GHEY+0rf21O1x7IjywxgixyAnrOrfMmuyilKUohWnGebfsN3Gt1acZ5t+w3ca
D8zYXkL2R3VmsYXkL2R3Vmstvtvik/C4OuX9R6uNU7xSfhcHXL+o9XGtMFasVAHRkbkspU9TCx+RrbSg5
dmTF4kZuVazdscVx/eBrXtqBnhYILspWRBe13iZZEW/MCyAe2vGCkCO8RIF2MiDddZOM1uk8IX946a68Xi
BGjO17KCTYXOnMBznoFRUJgdqgtK8SSS8I4KKmS5RUjBYl2VV4xYWJBuD0Gu9NrqdFSQvqGjC8ZSLXza2
G8WN+MDcXrhh2TOjcLC8Kuy/eRujMmYku2RlYEcZnOt75j6raoMJiYXlndsOpdbyv96wtHfIVj4uXKhIPGN9
DpzhM4PaUUvIcE8YEaggrYOCp1BBZQei9dEEyuodGVlYXDKQQR0gjfVW2CvBxxyzI5MySNIUjdzwkj53BCZ
iLiwHMMgF91+PYtwkGHbhUk4OPlcJEqyszviTfQPJfcgzCxB5JJoReCa4J9tYdR56MnmVWDsT+6qLcsfUBUU
gEuNlhlu8SqGRGsVDjI0hK/mW0sWXNexVrc1Sc6rw8KAAWWWUWAGq5I+6ZqClxkeJxiFMrQxNYnMCpsD
fKwuHa9hcDg7XNnap7ZmC4GMRgkgXPPpc3sLkkAX0BJ6666UA1D+DK2SUf9ziCP5pGf/camDUNsVxHH
O0jABZ8QzHcAudmHuW3uoPPhPjmVFgh1mnJjT+FbXklOhsFW+8EXK1WfBfDxS5mwuJKm5gRsraxQqka
BDe1yI84JzXVgeY21baxEkqzMFPCTQStrf7rCRgkjTc0jWQa87sDYVu8GMAJo5pMOVuJDwer5HjzM6LfcBdiy
vHYKWuBqykqaOx8Z+XFn+a7e/QVmTA7RA4mJi9q7/ehtXZsba+f7uS6uCV41gcwFyjAaCQDWw0ZeMtx
e0zRKrZh2koFpYGPQR/qFXT/29YM+0wB93hyeeyn/AFmFqstKQqtR4zaV+NhoLdoD/ea0bQjxuICxS4eNEz
qXdZcxsDuyFRbmN8x3btdLZShWAKzSlVClKUCtOM82/YbuNbq04zzb9hu40H5mwvIXsjurNYwvIXsjurNZb
fbfFJ+Fwdcv6j1cap3ik/C4OuX9R6uNaYKUpQQm341aSESFgn3hFkWTj8UAEMjADKZNbdOtQ+MwsX3Uc
MuHGeVBaMGFuITKSeCcbzGBydL3qybW0CSfuSK38pvG56grk+ytW341Ma5lVvvoBYgEWaWNTv8AUTU

Vz8BiF3GS2tgksUnVczRg/wCKuLa2PmAETg/eNaxhlBCKMz3aNnBBCldALZxoeebOyYfypk/sy0f9BFR32Jji
HKzSgRRKq3KObyFmkUl1JItHCd/TQeNj7WCxRJZHIRF4k0JJsouSHKG9SMmOjZcskcljvDROy+0qCvzrRgYp
jDHcwOvBpZSjrzC92zNf3V4ODtr9kUf2EuU2G7mj76DswuIw5YmNoc5ABylAxC6AHnsL7uahF8SP4IW/zG
S36JqOmRLAOcUg51kjEy9JzOVcD+9UbsyJGkmMc2GJzhFCM8bEKBzxyafeNNpY6689BcqVB8FiE3cL6gkk
MvVczKp/xVn9ozLyxf1GDELoN5MicIlCJuqPNic7TQsWWKOeWSZgDrxgyRrzkm67vzEAXysKsEG28xy5Y3b
nWKeJyOgWfIb1Xtn4N5sTNA8bRxic4iQMUJYOF4JbozC1xIbX9YtYXDXNjYosLinxByT4mBnC5T93EUdMND
caCy81+U7HnFTXgNEogJQAAlTpz5kR83rvmrZ4XbLWTC4k3YM0DiynRiqsUFjuOY6EWO7WtfgM18OSBY
MY3UdAeCBwPZmt7KLx2bb2Rwn3kdhKABqWCyKDmEbldRY6q44yNqN7K3nY21c33clw4JXjBQ2YC5RwN
BIBrpxWHGXQ2EzUPtrZOf72KwlAAlJIEiqbhGI1Ug6q41U66gkEiYpUZsLaHCoQ18ymxuADzjjAaBgyuhA0zI
1tCKk6qFKUoFKUoFKUoFacZ5t+w3ca3Vpxnm37DdxoPzNheQvZHdWaxheQvZHdWay2+2+KT8Lg65f1
Hq41TvFJ+Fwdcv6j1ca0wUpSg1YmEOjI25lKnqIse+onGSl8KjtoVkgaT1GKaMze7I/uqbqOw8QzzwsAVaz2I
0yyghgem7pIT2qipEVBY+bIcWBymjRk5rs4aJVB6cyDqzjprZLszEBo+CxbhEYZldI2zJzqWsGJtcA3vrc3IqRn
wSOys6gshupPNuPeqnrUHeBQbYI8qhRuAA92le6VrnmVFLuyqqi5ZiAABvJJ3CqhiJQis7blBY9Q1PdXHsvCf
8PGsqgsVDOCLjO3Gc2P8TN7687ZcPCEUgiYrGLEG6uRnI6fu859lSQqK4f2RDuWMJ/Zlot+p1jIodnsORPMv
qJRx73Un513UqiPlw0p0ZoZFvueNh88xF/5a4PBGlAYiyopGJkXKgAUZQoCjQaeznqfqG8G0scV/wDakPvW
M/61B07e8w4F+NlTS1xwjKhYX0uM19eiovxfSZsDEbflRfUeDRIiR6iYzXb4UzhcOwJsZCI1IvcFjq4tqcihn05k
NR3i9lAwkce5goLLpoWALAW0srEr7jqCpI4tFKV5drC55qqIHwY1edubPlP8/Etp7HWrBUB4GqeBzE3LcGx6
2iiZv8RY+2p+pi6UpSqhSlKBSlKBWnGebfsN3Gt1acZ5t+w3caD8zYXkL2R3VmsYXkL2R3Vmstvtvik/C4O
uX9R6uNU7xSfhcHXL+o9XGtMFKUoFcGL4s0L/AL2eI/zDhAT1cEQO366760YzDiRSpJGoII3hlIZWHrBAOun
TQb6VwB8Qu9In6SGZCepCGA/v14h2uCCWimUBmXkZ9UJVvNlrC6nfaoqSqN8Ih/w7jXUoNCQdXUXBG46
7668PjY35Do3SAwJHqI5qjfCua2HZVPHcqE0Jscy8Yj90aE+7eRQe8NAvCqkYtHh1ygan7xgABc63WM/5vq
qWrRg8KI1Crc7ySd5JN2Y+skk+2t9VCo3C7XV8RLhwkgaIKWYrxTmCsLG/Q49zdBrdjcSRZIwDI26+5RuLt/
COjnNhpvGzBYURrYEk72Y72Y72b19WgFgAAAKiuioXwfPHxQ/7hj71Qf7amqrEePEC46TS6zcUHndlTIOolh
Qxz7VJxWNWAebi5XrY2Ln2LZQRqGepDbeCMbDFQ6FR94oG9RoGsN5UbwNSumpVLePA/AZEZ2uWYkEn
eSCc7H1l736coNWKg5sBi1lQOvtFwbHouN/MQdxBBGhrxtibJh5nH5YpG9yk/wClQzIcJMMoYwyHQC5sdSY
wOneydIzpzRipDb8gbCyZSCHTKpBBB4SyqQeg5hrQZ8HYwsNgNzuv9wmMfJBUpXFsfzQP7xdh1MzMPka
7aoUpSiFKUoFKUoFacZ5t+w3ca3Vpxnm37DdxoPzNheQvZHdWaxheQvZHdWay2+2+KT8Lg65f1Hq41T
vFJ+Fwdcv6j1ca0wUpSgUpSgVwvg3Us0UlsxvlZQy3tqRYhtTryumu6lBWv/5++IlxGIaMo/BsyBb6xAgMW
O4WsSPVvtodeJ2aPs0k5MiNkaQKGNkQZXWMK11GiJew1N+awE1t0/csp3OUjO7dKyxn5Oa97XgzwSxj80
bqObepAqLWDHONzxuOhlZT/eUkdP5a0YnajxgcJA+Y3ChGVwzWJCjc24E3y2FjXZHjFMQmuAhQPfmCkZr
9Vq0YGFmbhpBZiLlp/Im+x/iawLdQGuW5Dn2fi4V5cqiVzds4aNieZVSSxyjmHt3kky9eXQEWIuDvBqAiwk
URdWWWLjsytHwyplOoJ4Pi6XPKHTQWGvn+PjeTaEsK8hXjmJJOszIiRC3OqqruQd4VtNBUnitsypiI4cPMkyy
WHHCMQSJSbGPLoODF7gm199Smz9lMuIlxEhW75bKpJAlUlXJlGtlt6rnpoeJXDQhFVF3KAo6hoK2UpVRoxu
EWVGjkF1YdJBHOGBGoYEAgjUEAiqrippIrYeXX72JweKMyrKjtIo3WNuMo5LG9sri1xrnxeBiltwsaPlNxnVWs
bFbi40NiR1E1FeNkx5YIlO8RoD12F666ClVClKUClKUClKUCtOM82/YbuNbq04zzb9hu40H5mwvIXsjurNYw
vIXsjurNZbfbfFJ+Fwdcv6j1cap3ik/C4OuX9R6uNaYKUpQKUpQKUpQcG3R9yzXsEaOQ9mN1kb5Ka7qMtx
Y7qjo8JMgCRSJkGg4RGZlHMLhxmA5r69JO+orjgUiUYQ2yp98NRrHm+7QjeMr39ka77kCeFR/7KFtWbhL5
uF0zZrWuNLWtplta3TXrC4tg3BTAB7cUi+VwN5W+49Knd0ka0CXEsZRFHbihWkY62UkhVUc7HK2vMBz3
Fd1R2OGSWKUaZjwT+tWBKH1kOAB/aNUjVR4ES3zWF+mwv7690pQKUpQKUpQKUpQKUpQKUpQKUpQ
K04zzb9hu41urTjPNv2G7jQfmbC8heyO6s1jC8heyO6s1lt9t8Un4XB1y/qPVxqneKT8Lg65f1Hq41pgpSlAp
SlApSlApSlArg22LRF+eMiUb7/dkMbW6VDDqJrvrDC9BG41xJJHEuuVhK9vyqt8ntZsthzgN0VJ1pwuESMZY
0VRe9lAAud50rdQKUrVib5Wy8qxtu320+dBtpVcmTHGOwNnMguRweiCC+l7/APWAHtPNXdAZuGXPwmX
lt7cDkzWOfNfjXva1tPnUWJWlVxftn2Y5s/D3i5PAcm6cJlvpe2a9+fdzVvZcTwsJUuIwicJfgrlrtmzDqC3sdL3F
6CcpUARiyMQAXBJHBMeBA1Zr5QL7ly6tv6L3r1iocWOCCSZrXMjWQZuOmVdf4C+otfLfTdQTtKg8T9p/4nJ
wmbJJwPmMmbKOCy342a9+Vp8q79k5+DHC581zfPkvvNuTputQdtKrhTHBb5rk4aewAiBE/wB0YbnnPnQ
LcWw11rGN+18AnBcNwl5M3/LX5L8HmubZM+S9taEWSlQ8fD/aXzcJwVhktwWTkre/5r5s/wAq8YfEYjg8M
XjlzaCdfuL3yEFtGtlz2Ohv6qCbrTjPNv2G7jUPtT7ZlJg5Qlcgfd6x8G2Ua8+e1vXa+l6lsTfgmvvyHuNB+aML
yF7I7qzWMLyF7I7qzUafa/FS4XBthzy8PPNC46CHZr9RzfKrnVN27sbE4fEtj9nKrs4AxGHJyiULoroeaQD3+8
HEXjJwY0xIxGGk545YZgw/ug3FaZXOlVDymbM9JPwsT9FPKZsz0k/CxP0URb6VUPKZsz0k/CxP0U8pmzPS
T8LE/RQW+lVDymbM9JPwsT9FPKZsz0k/CxP0UFvpVQ8pmzPST8LE/RTymbM9JPwsT9FBb6VUPKZsz0k/C
xP0U8pmzPST8LE/RQW+lVDymbM9JPwsT9FPKZsz0k/CxP0UFvpVQ8pmzPST8LE/RTymbM9JPwsT9FBb6

VUPKZsz0k/CxP0U8pmzPST8LE/RQW+lVDymbM9JPwsT9FPKZsz0k/CxP0UFvpVQ8pmzPST8LE/RTymbM9JPwsT9FBb6VUPKZsz0k/CxP0U8pmzPST8LE/RQW+lVDymbM9JPwsT9FPKZsz0k/CxP0UFvpVQ8pmzPST8LE/RTymbM9JPwsT9FBb64NvYxYcNNK5sqRux9gNh1k6e2q83jM2d+SZ3bmVYcQST0AFRrXDLBidrOonhfDYBWDmOTSXEFdVDL+SO/N37wVTdkeAOJkgikAFnjRxe+5lBHN66V9uVbCw3ClSLX/9k=” width=“140px”/> <h4>Automatic grow garden</h4>

RELAY 1

<button id=“ledon” type=“button” onclick=“buttonclick(this);”>On</button> <button id=“ledoff” type=“button” onclick=“buttonclick(this);”>Off</button>

Relay 2

<button id=“ledon1” type=“button” onclick=“buttonclick(this);”>On</button> <button id=“ledoff1” type=“button” onclick=“buttonclick(this);”>Off</button>

</body> </html>)rawliteral”;

GPIO#0 is for Adafruit ESP8266 HUZZAH board. Your board LED might be on 13. const int LEDPIN = D1; const int LEDPIN1 = D3; Current LED status bool LEDStatus; bool LEDStatus1;

Commands sent through Web Socket const char LEDON[] = “ledon”; const char LEDOFF[] = “ledoff”; Commands sent through Web Socket const char LEDON1[] = “ledon1”; const char LEDOFF1[] = “ledoff1”;

void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {

```
Serial.printf("webSocketEvent(%d, %d, ...)\r\n", num, type);
switch(type) {
  case WStype_DISCONNECTED:
    Serial.printf("[%u] Disconnected!\r\n", num);
    break;
  case WStype_CONNECTED:
    {
      IPAddress ip = webSocket.remoteIP(num);
      Serial.printf("[%u] Connected from %d.%d.%d.%d url: %s\r\n", num,
ip[0], ip[1], ip[2], ip[3], payload);
      // Send the current LED status
      if (LEDStatus) {
        webSocket.sendTXT(num, LEDON, strlen(LEDON));
      }
      else {
        webSocket.sendTXT(num, LEDOFF, strlen(LEDOFF));
      }
      // Send the current LED status
      if (LEDStatus1) {
        webSocket.sendTXT(num, LEDON1, strlen(LEDON1));
```

```
      }
      else {
        webSocket.sendTXT(num, LEDOFF1, strlen(LEDOFF1));
      }
    }
    break;
  case WStype_TEXT:
    Serial.printf("[%u] get Text: %s\r\n", num, payload);
```

if (strcmp(LEDON, (const char *)payload) == 0) {

```
      writeLED(true);
    }
    else if (strcmp(LEDOFF, (const char *)payload) == 0) {
      writeLED(false);
    }
    else if (strcmp(LEDON1, (const char *)payload) == 0) {
      writeLED1(true);
    }
    else if (strcmp(LEDOFF1, (const char *)payload) == 0) {
      writeLED1(false);
    }
    else {
      Serial.println("Unknown command");
    }
    // send data to all connected clients
    webSocket.broadcastTXT(payload, length);
    break;
  case WStype_BIN:
    Serial.printf("[%u] get binary length: %u\r\n", num, length);
    hexdump(payload, length);
```

echo data back to browser webSocket.sendBIN(num, payload, length); break; default: Serial.printf(“Invalid WStype [%d]\r\n”, type); break; } } void handleRoot() { server.send_P(200, “text/html”, INDEX_HTML); } void handleNotFound() { String message = “File Not Found\n\n”; message += “URI: ”; message += server.uri(); message += “\nMethod: ”; message += (server.method() == HTTP_GET)?“GET”:“POST”; message += “\nArguments: ”; message += server.args(); message += “\n”; for (uint8_t i=0; i<server.args(); i++){ message += “ ” + server.argName(i) + “: ” + server.arg(i) + “\n”; } server.send(404, “text/plain”, message); } static void writeLED(bool LEDon) { LEDStatus = LEDon; Note inverted logic for Adafruit HUZZAH board

```
if (LEDon) {
  digitalWrite(LEDPIN, 1);
}
else {
  digitalWrite(LEDPIN, 0);
}
```

}

static void writeLED1(bool LEDon) {

```
LEDStatus1 = LEDon;
// Note inverted logic for Adafruit HUZZAH board
if (LEDon) {
  digitalWrite(LEDPIN1, 1);
}
else {
  digitalWrite(LEDPIN1, 0);
}
```

}

función que prepara la trama de datos para hacer un POST a endpoint del influx String influxFrame(String dht11_humidity, String dht11_temperature) { este es el nombre del sensor

```
// Siempre que se quema la primera vez, se debe de cambiar el nombre del
sensor
const String SENSOR_ID = "DHT11_llanadas"; // Nombre del sensor en la
plataforma
```

const String STR_COMMA = “,”;

```
const String STR_SLASH = "/";
const String STR_DOT = ".";
const String STR_COLON = ":";
const String STR_NULL = "NULL";
const String STR_ZERO = "0";
const String STR_SPACE = " ";
```

El primer dato en el squema de la DB es el id del sensor String frame = SENSOR_ID + STR_COMMA + “id=” + SENSOR_ID + STR_SPACE; Add GPS data

```
frame += "lat=";
frame += "6.2563143" + STR_COMMA; // coordenada GSP lat
frame += "lng=";
frame += "-75.5386472" + STR_COMMA; // coordenada lng lat
frame += "altitude=";
frame += STR_ZERO + STR_COMMA;
frame += "course=";
frame += STR_ZERO + STR_COMMA;
frame += "speed=";
frame += STR_ZERO + STR_COMMA;
```

Add DHT11 data if

```
  frame += "humidity=";
  frame += dht11_humidity + STR_COMMA;
  frame += "temperature=";
  frame += dht11_temperature + STR_COMMA;
// } else {
```

```
//   frame += "humidity=" + STR_NULL + STR_COMMA + "temperature=" + STR_NULL
+ STR_COMMA;
// }
```

Add Plantower data if

```
  frame += "pm1=";
  frame += STR_ZERO + STR_COMMA;
  frame += "pm25=";
  frame += STR_ZERO + STR_COMMA;
  frame += "pm10=";
  frame += STR_ZERO;
// } else {
//   frame += "pm1=" + STR_NULL + STR_COMMA + "pm25=" + STR_NULL + STR_COMMA
+ "pm10=" + STR_NULL;
// }
```

return frame; }

función que envía la trama de datos void sendDataInflux (String humidity, String temperature) { / El post a la base de datos tiene una trama siguiente:* volker0001,id=volker0001 lat=6.268115,lng=-75.543407,altitude=1801.1,course=105.55,speed=0.00,humidity=37.00,temperature=25.00,pm1=22,pm25=31,pm10=32

```
Para nuestro caso que SOLO es el envío de datos del dht_11 que es humedad y
temperatura la trama es la siguiente
// DHT11_llanadas, id=DHT11_llanadas, lat=6.2563143, lng=-75.5386472,
altitude=0, course=0, speed=0, humidity=37.00, temperature=25.00, pm1=0,
pm25=0, pm10=0 1434055562000000000
*/
HTTPClient http;
// _testsensorhumedad es el nombre de la DB donde se almacenan estos datos
http.begin("http://aqa.unloquer.org:8086/write?db=_testsensorhumedad"); //
endPoint final, '_testsensorhumedad' es el nombre de la base de datos
http.setTimeout(HTTP_TIMEOUT);
http.addHeader("Content-Type", "--data-binary");
```

String frame = influxFrame(humidity, temperature); *Construimos el request POST int httpCode = http.POST(frame);* Envíamos los datos haciendo un POST

if(httpCode > 0) {

```
  String payload = http.getString();
  Serial.println(payload);
  Serial.println("Envío de datos con exito!");
} else {
  Serial.print("[HTTP] failed, error;");
  Serial.println(http.errorToString(httpCode).c_str());
}
```

http.end();

```
delay(60000); // cada minuto se envía un POST al influx
```

}

void setup() {

```
pinMode(LEDPIN, OUTPUT);
pinMode(LEDPIN1, OUTPUT);
writeLED(false);
writeLED1(false);
```

Serial.begin(115200);

```
// init pantalla
display.begin(SSD1306_SWITCHCAPVCC);
display.display();
delay(1000);
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
```

```
Serial.println(F("DHTxx test!"));
dht.begin();
```

Serial.setDebugOutput(true); Serial.println(); Serial.println(); Serial.println(); for(uint8_t t = 4; t > 0; t–) { Serial.printf(“[SETUP] BOOT WAIT %d…\r\n”, t); Serial.flush(); delay(1000); } WiFiMulti.addAP(ssid, password); while(WiFiMulti.run() != WL_CONNECTED) { Serial.print(“.”); delay(100); } Serial.println(“”); Serial.print(“Connected to ”); Serial.println(ssid); Serial.print(“IP address: ”); Serial.println(WiFi.localIP()); if (mdns.begin(“espWebSock”, WiFi.localIP())) { Serial.println(“MDNS responder started”); mdns.addService(“http”, “tcp”, 80); mdns.addService(“ws”, “tcp”, 81); } else { Serial.println(“MDNS.begin failed”); } Serial.print(“Connect to http://espWebSock.local or http:”);

```
Serial.println(WiFi.localIP());
```

server.on(“/”, handleRoot);

```
server.onNotFound(handleNotFound);
```

server.begin();

webSocket.begin();

```
webSocket.onEvent(webSocketEvent);
```

}

void loop() {

```
webSocket.loop();
server.handleClient();
```

```
// dht11 y pantall
static unsigned int h = 0;
static unsigned int t = 0;
```

```
h = dht.readHumidity();
t = dht.readTemperature();
/*
// Si las lecturas fallan, salimos, no mandamos nada y volvemos a intentarlo
if (isnan(h) || isnan(t)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
}
*/
```

```
display.clearDisplay();
display.setCursor(0,0);
display.print("UN/LOQUER \n");
display.print("HUM:   ");
display.print(h);
display.print(" % \n");
display.print("TEM:  ");
display.print(t);
display.print(" C \n");
display.print("IP: ");
display.print(WiFi.localIP());
display.display();
```

```
// sendDataInflux(String(h), String(t)); // se tiene que mandar los datos
por websokect + no por protocolo http
```

} </code>