

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 enviar 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_B0", "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 esquema 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 14340555620000000000
    */
}

```

```
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); // Enviamos los datos haciendo un POST

if(httpCode > 0) {
    String payload = http.getString();
    Serial.println(payload);
    Serial.println("Envío de datos con éxito!");
} 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));  
}
```

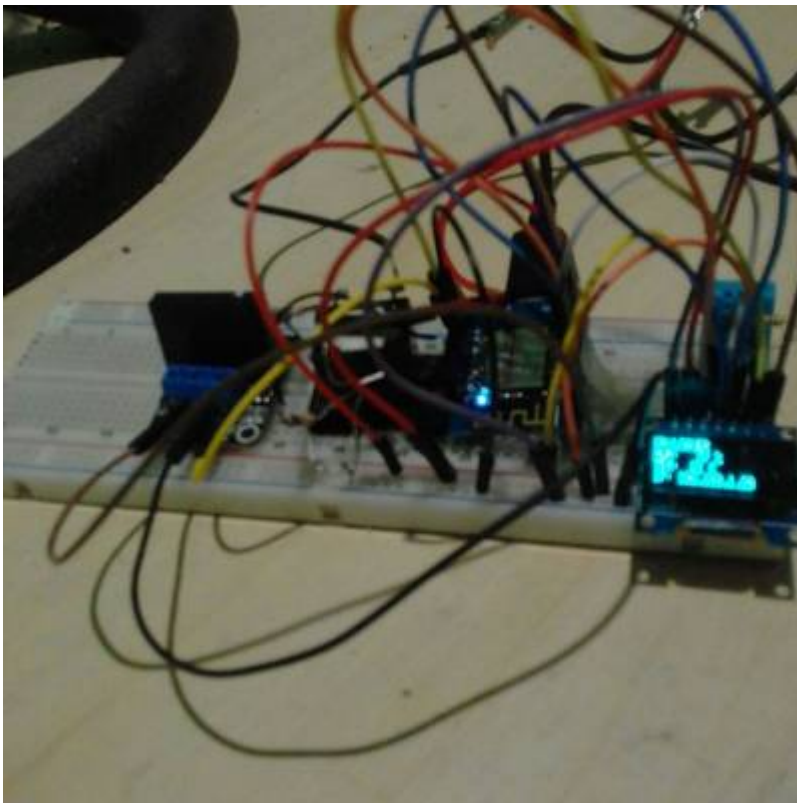
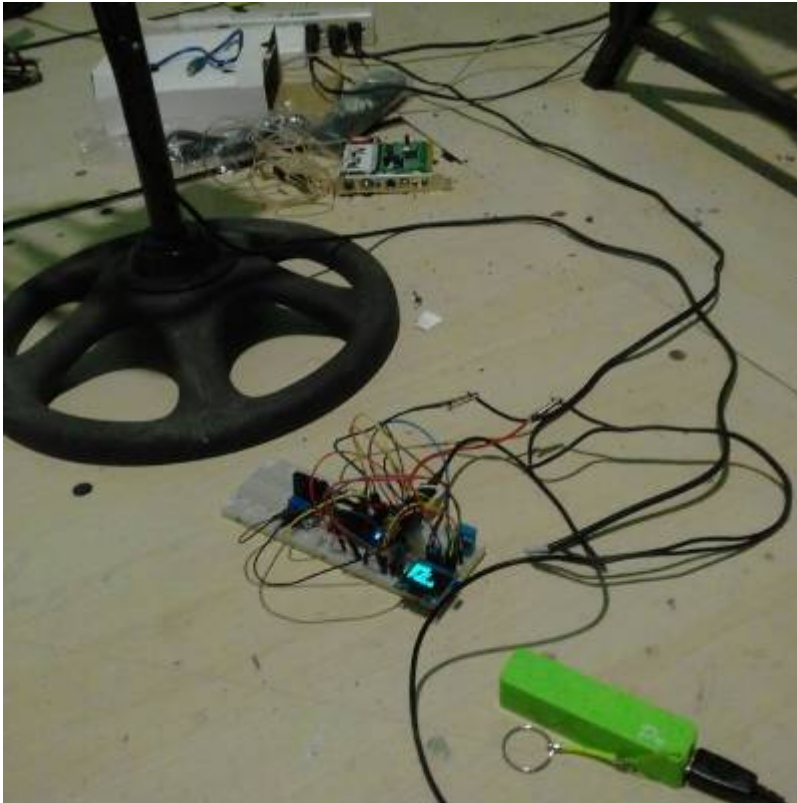
Configuración de plataforma

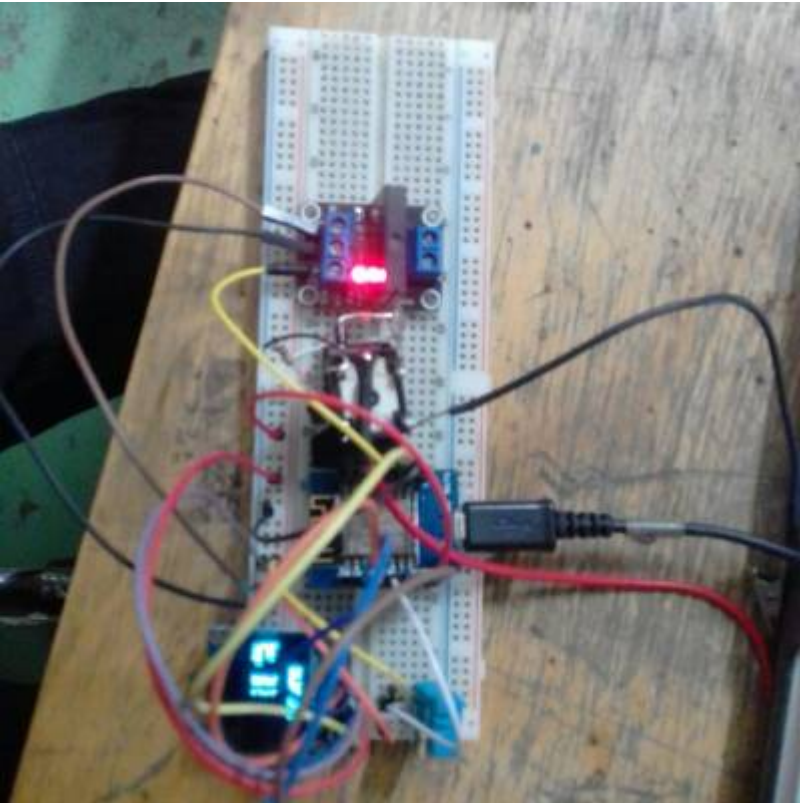
1. Se crea una base de datos



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



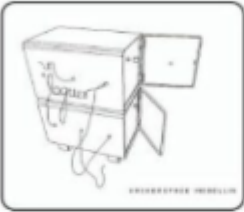




192.168.1.20

1





Automatic grow garden

RELAY 1

On

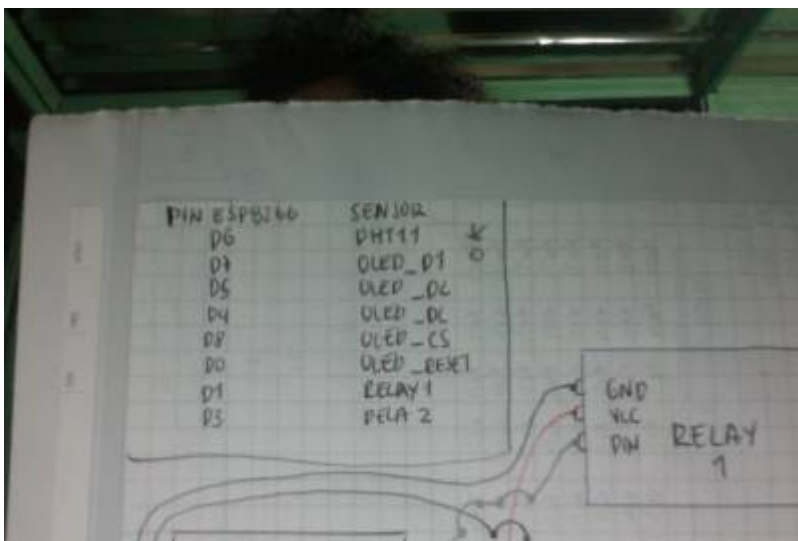
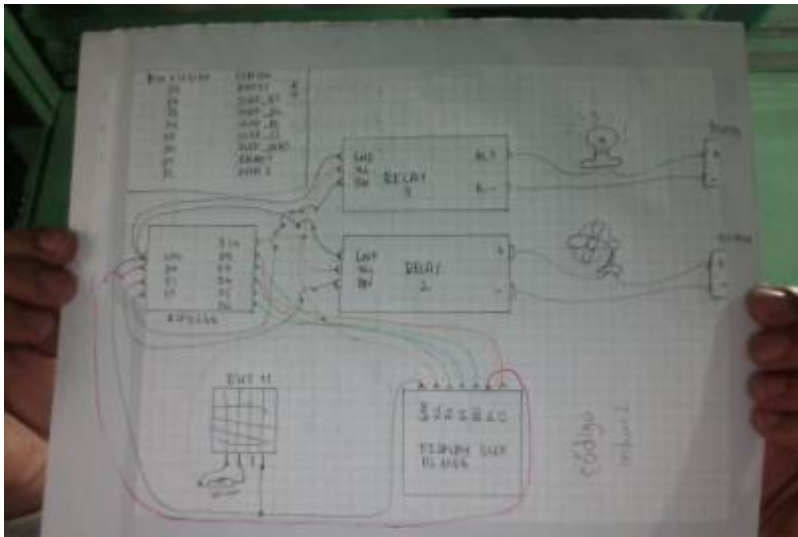
Off

Relay 2

On

Off

- <https://wiki.unloquer.org/>



Se intenta manipular relays, mostrar datos en pantalla 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 websocket;
var webSocketInflux; se tiene que mandar los datos por un websocket function start() { websocket = new
WebSocket('ws:' + window.location.hostname + ':81');

```

```

websocket.onopen = function(evt) { console.log('websocket open'); };
websocket.onclose = function(evt) { console.log('websocket close'); };
websocket.onerror = function(evt) { console.log(evt); };
websocket.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) {
```

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

```
</script> </head> <body onload="javascript:start();">  <h4>Automatic grow garden</h4>

<b>RELAY 1</b>

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

<b>Relay 2</b>

<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(LED0FF1));
 }
}
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(LED0FF, (const char *)payload) == 0) {
 writeLED(false);
}
else if (strcmp(LEDON1, (const char *)payload) == 0) {
 writeLED1(true);
}
else if (strcmp(LED0FF1, (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 esquema 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 SÓLO 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 14340555620000000000
*/
```

```
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); Enviamos los datos haciendo un POST

if(httpCode > 0) {

```
String payload = http.getString();
Serial.println(payload);
Serial.println("Envío de datos con éxito!");
} 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>

From:  
<https://wiki.unloquer.org/> -

Permanent link:  
[https://wiki.unloquer.org/personas/johnny/proyectos/indoor\\_diy\\_autosostenible?rev=1560929805](https://wiki.unloquer.org/personas/johnny/proyectos/indoor_diy_autosostenible?rev=1560929805)

Last update: **2019/06/19 07:36**

