

# 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("name wifi", "wifi password");

  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));  
}
```

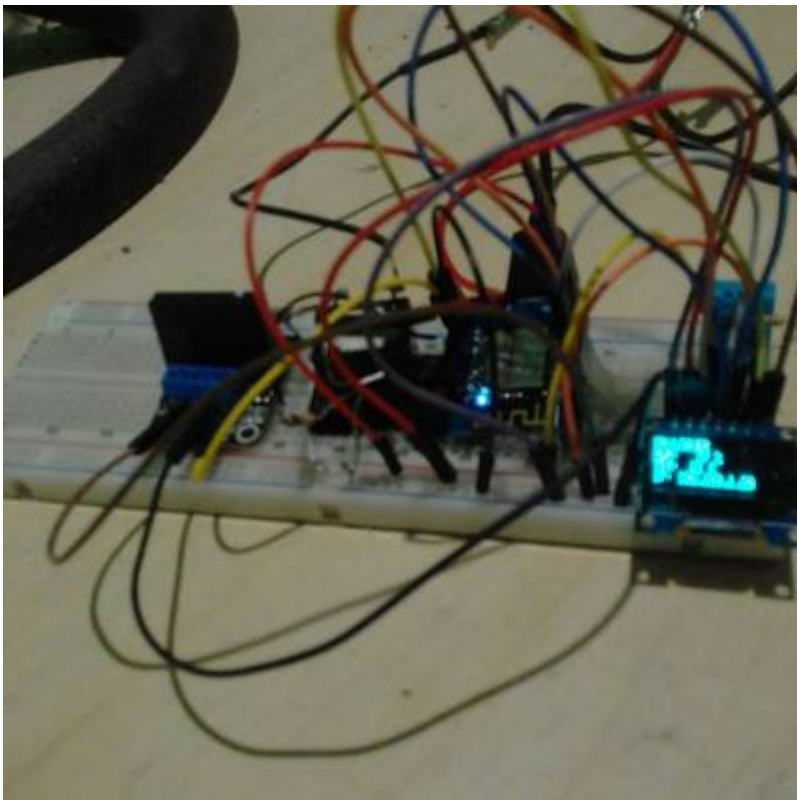
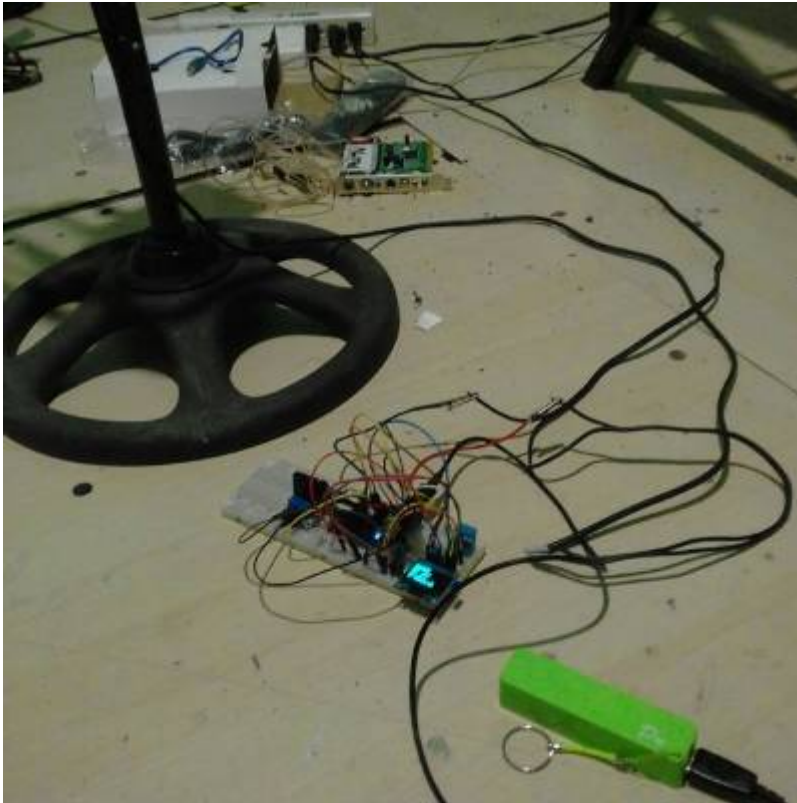
## Configuración de plataforma

1. Se crea una base de datos



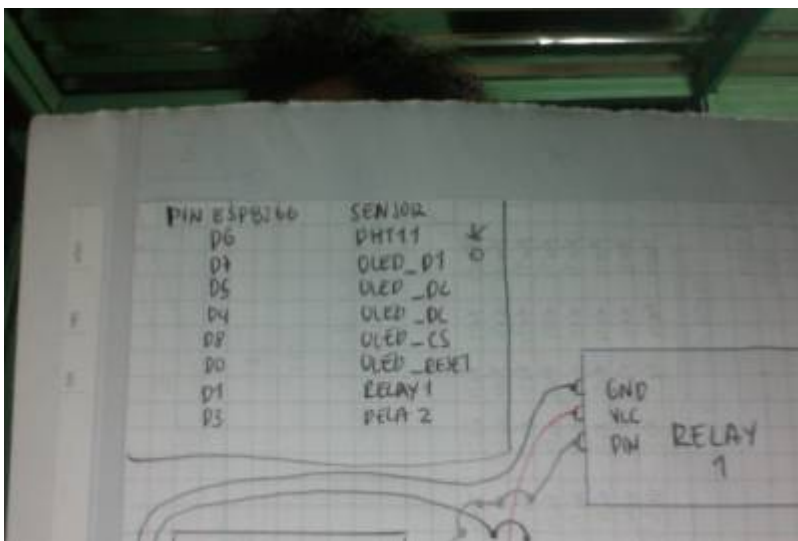
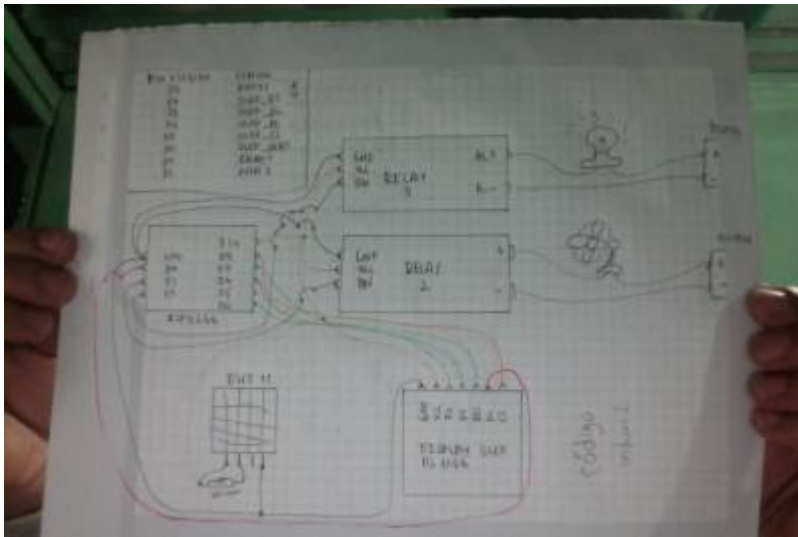
documentar esta parte de como crear base de datos y adjuntar al dashboard para ver los graficos enviados por algún sensor

## primer prototipo de control automatico









Se intenta manipular relays, mostrar datos en pantalla y enviar datos a una base de datos influxdb  
A ESTE C DIGO FALTA IMPLEMENTAR ENVIO DE DATOS AL INFLUX CON WEBSOCKETS.

[Gist al c digo](#)

## Construcci n f sica del indoor

[versi n barata y sencilla](#)

[la opci n cara y vacana](#)

[Un resumen de una revista especializada](#)

# C digo para la ventilaci n usando



# timeAlarms

```
// https://github.com/PaulStoffregen/TimeAlarms
// Librerias
#include <Time.h>
#include <TimeAlarms.h>

// pin que controla
int pin = 13;

int alarma = 900; // cada 15 min

void setup() {

  Serial.begin(9600);
  //fijamos el tiempo inicial del esp
  // (08:30:00 25/05/17)
  setTime(8,10,0,28,5,19);

  //Creamos las alarmas
  //Alarm.alarmRepeat(8,init15,0,EveningAlarm); Alarma que se inicia cada
día
  //Alarm.alarmRepeat(8,end15,0,apagarVentilador); Alarma que termina cada
día

  Alarm.timerRepeat(alarma, Repeats); // Timer cada 15 segundos

  //Alarm.alarmRepeat(17,45,0,EveningAlarm); 5:45pm cada día
  //Alarm.alarmRepeat(dowSaturday,8,30,30,WeeklyAlarm); 8:30:30 cada sabado
  //Alarm.timerRepeat(alarma, Repeats); Timer cada 15 segundos
  //Alarm.timerOnce(10, OnceOnly); Llamado una vez despues de 10 segundos

  pinMode(pin, OUTPUT);
}

void loop() {
  digitalClockDisplay();
  Alarm.delay(1000);
}

// encender ventilador
void prenderVentilador(){
  Serial.println("Ventilando");
  digitalWrite(pin, HIGH);
}

// encender ventilador
void apagarVentilador(){
```

```
Serial.println("apagando ventilador");
digitalWrite(pin, LOW);
}

// función que enciende el riego
void prenderSensorRiegoManana(){
    Serial.println("Alarm: - Sensor encendido y regando");
    digitalWrite(pin, HIGH);
}

// función que apaga el riego
void apagarSensorRiegoManana(){
    Serial.println("Alarm: - Sensor apagado y riego apagado");
    digitalWrite(pin, LOW);
}

void EveningAlarm(){
    Serial.println("Alarm: - turn lights on");
}

void WeeklyAlarm(){
    Serial.println("Alarm: - its Monday Morning");
}

void ExplicitAlarm(){
    Serial.println("Alarm: - this triggers only at the given date and time");
}

void Repeats(){
    digitalWrite(pin, HIGH);
    Alarm.delay(60000);
    Serial.println("ventilando x un minuto");
    digitalWrite(pin, LOW);
    Alarm.delay(1000);
    Serial.println("apagando ventilador");
}

void OnceOnly(){
    Serial.println("This timer only triggers once");
}

void digitalClockDisplay()
{
    // digital clock display of the time
    Serial.print(hour());
    printDigits(minute());
    printDigits(second());
    Serial.println();
}
```

```
void printDigits(int digits)
{
    Serial.print(":");
    if(digits < 10)
        Serial.print('0');
    Serial.print(digits);
}
```

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

Permanent link:  
[https://wiki.unloquer.org/personas/johnny/proyectos/indoor\\_diy\\_autosostenible?rev=1568579970](https://wiki.unloquer.org/personas/johnny/proyectos/indoor_diy_autosostenible?rev=1568579970)

Last update: **2019/09/15 20:39**

