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 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);
```

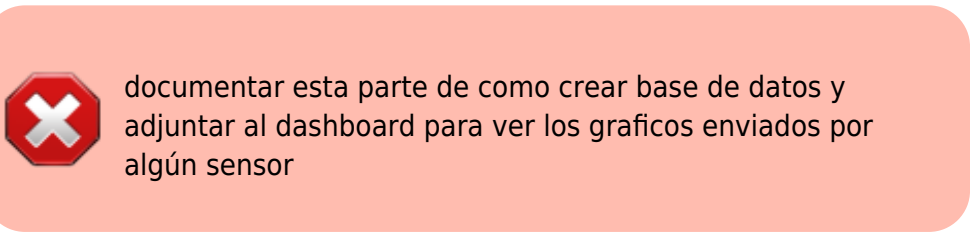
```
  // Serial.print(F("Lectura Humidity: "));
  // Serial.print(h);
  // Serial.print(F("% Lectura Temperature: "));
  // Serial.print(t);
  // Serial.print("\n");

  /*
  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"));
  */

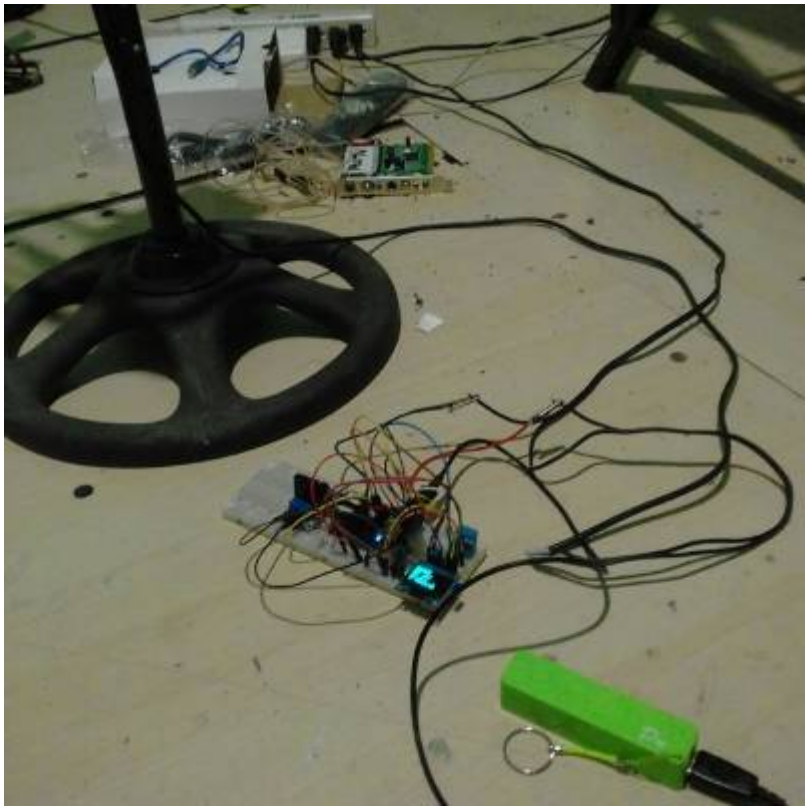
  sendDataInflux(String(h), String(t));
}
```

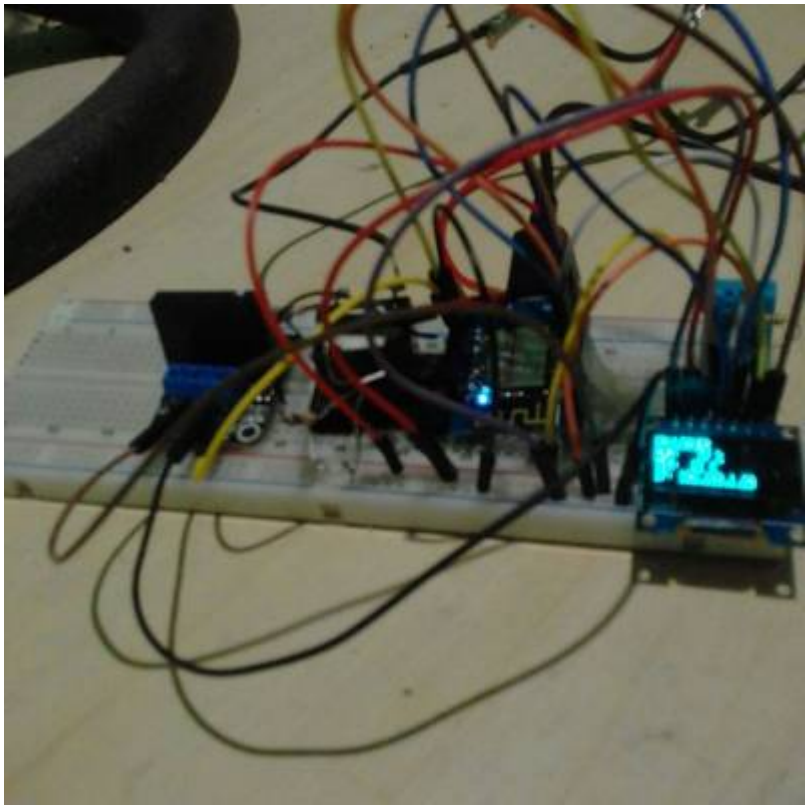
Configuración de plataforma

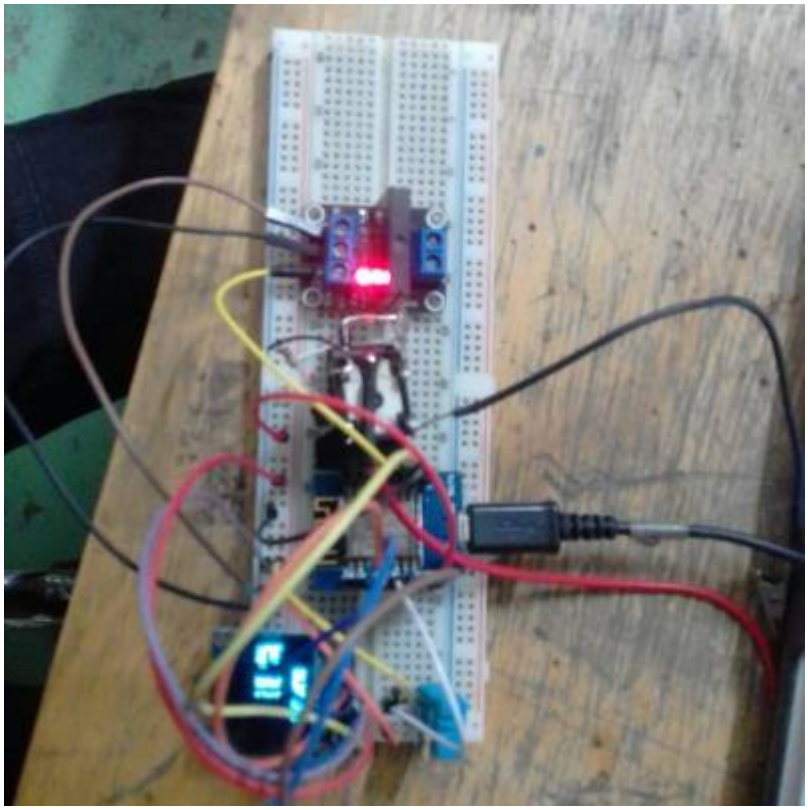
1. Se crea una base de datos



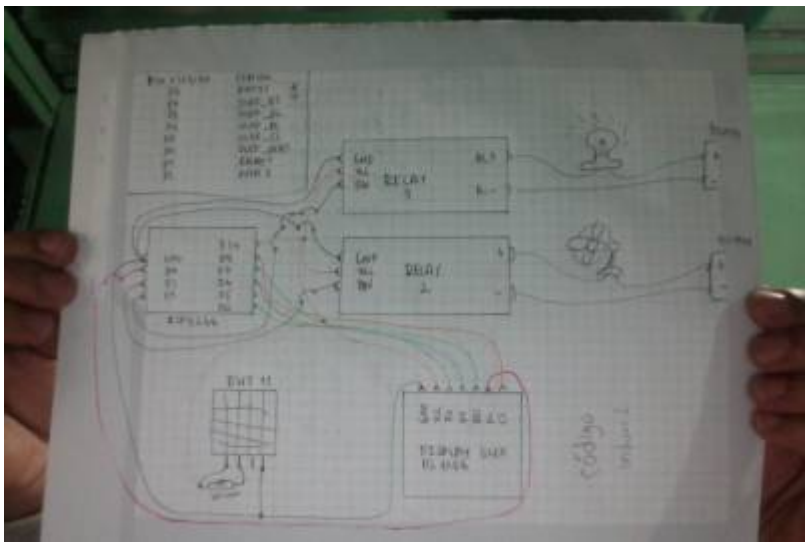
primer prototipo de control automatico

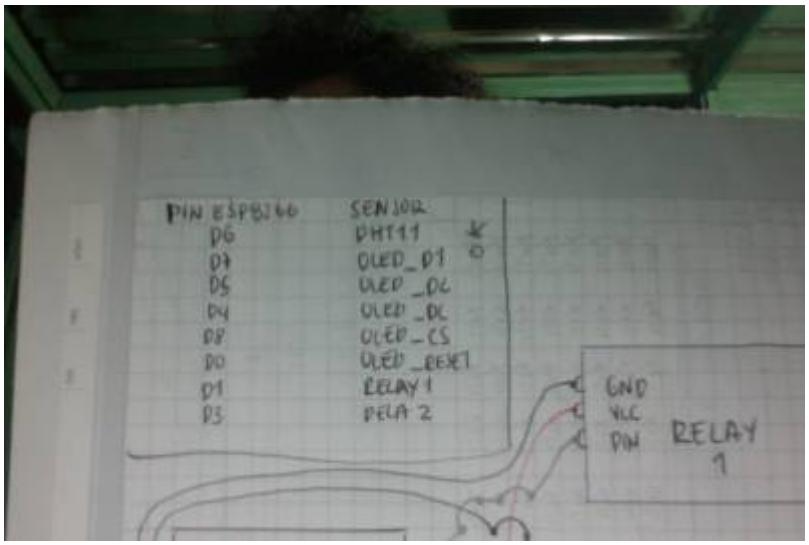







2% 1:47 AM
192.168.1.20
Automatic grow garden
RELAY 1
On
Off
Relay 2
On
Off





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.

Gist al código

Construcción física del indoor

versión barata y sencilla

la opción cara y vacana

Un resumen de una revista especializada

Código para la ventilación 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);
}
```

Código para el control automatizado de las luces

Este código posee dos funciones que segun el estado se la planta se pueden cambiar para vegetativo o floración. Es un proyecto en platformio

```
/*
  Este código toma la hora de internet usando un servidor NTP y
  enciende algo. Tomadpo de aqui
https://lastminuteengineers.com/esp8266-ntp-server-date-time-tutorial/

  !!! importante
  You need to adjust the UTC offset for your timezone in milliseconds.
  Refer the list of UTC time offsets.  Here are some examples for different
timezones:
https://upload.wikimedia.org/wikipedia/commons/8/88/World_Time_Zones_Map.png

  For UTC -5.00 : -5 * 60 * 60 : -18000
  For UTC +1.00 : 1 * 60 * 60 : 3600
  For UTC +0.00 : 0 * 60 * 60 : 0

  here -> const long utcOffsetInSeconds = 3600;
*/
#include <Arduino.h>
#include <NTPClient.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

const char *ssid = "el nombre de la red";
const char *password = "el password de la red";
const long utcOffsetInSeconds = -18000; // colombia UTC -5
char daysOfTheWeek[7][12] = {
  "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"
};
// Define NTP client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);
int ledTrigger = D6;
```

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("Wifi connected!");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  timeClient.begin();
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(ledTrigger, OUTPUT);
}

void statusWIFI() {
  // cuando esta pegado a internet el status es 3
  // la idea de esto es que mande un color u otro si tiene internet o no
  Serial.print("Estatus wifi is: ");
  Serial.println(WiFi.status());
}

void lucesVegetativo() {
  // 18 horas luz, 6 horas oscuridad
  int hours = timeClient.getHours();
  //  se prenden a las 6 de la mañana y se apagan a las 12 de la noche
  if ( hours < 6 ) {
    digitalWrite(ledTrigger, LOW);
    Serial.println("Luces OFF!");
  } else {
    digitalWrite(ledTrigger, HIGH);
    Serial.println("Luces ONN!");
  }
}

void lucesFloracion() {
  // 12 horas luz, 12 horas oscuridad
  int hours = timeClient.getHours();
  //  a las 6 de la mañana se prenden y a las 6 de la tarde se apagan
  if ( (hours >= 6) && (hours < 18) ) {
    digitalWrite(ledTrigger, HIGH);
    Serial.println("Luces ONN!");
  } else {
    digitalWrite(ledTrigger, LOW);
    Serial.println("Luces OFF");
  }
}

// the loop function runs over and over again forever
```

```
void loop() {
  timeClient.update();
  Serial.print(daysOfTheWeek[timeClient.getDay()]);
  Serial.print(", ");
  Serial.print(timeClient.getHours());
  Serial.print(":");
  Serial.print(timeClient.getMinutes());
  Serial.print(":");
  Serial.print(timeClient.getSeconds());
  Serial.println("");
  statusWIFI();
  //lucesVegetativo();
  lucesFloracion();
  delay(1000);

  /*
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage
level)}
  digitalWrite(ledTrigger, HIGH);
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the
voltage LOW
  digitalWrite(ledTrigger, LOW);
  delay(1000);                       // wait for a second
  */
}
```

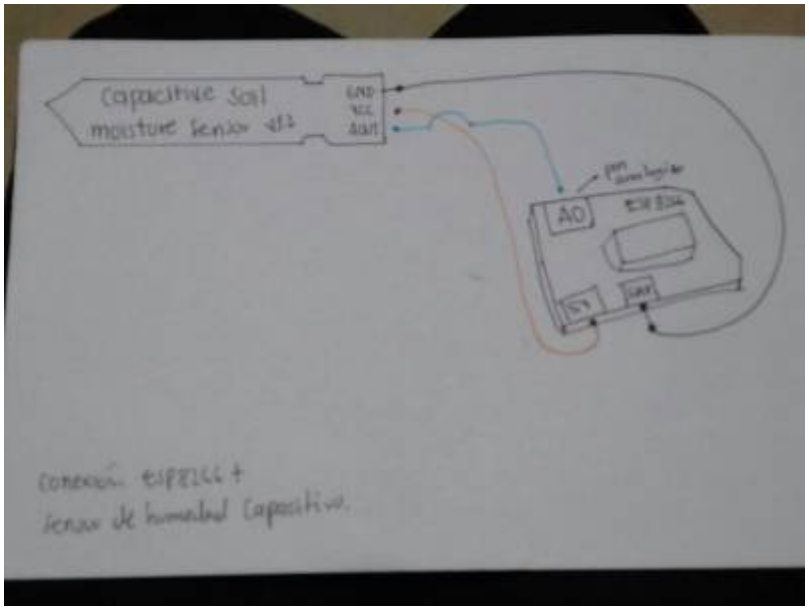
el platformio.ini

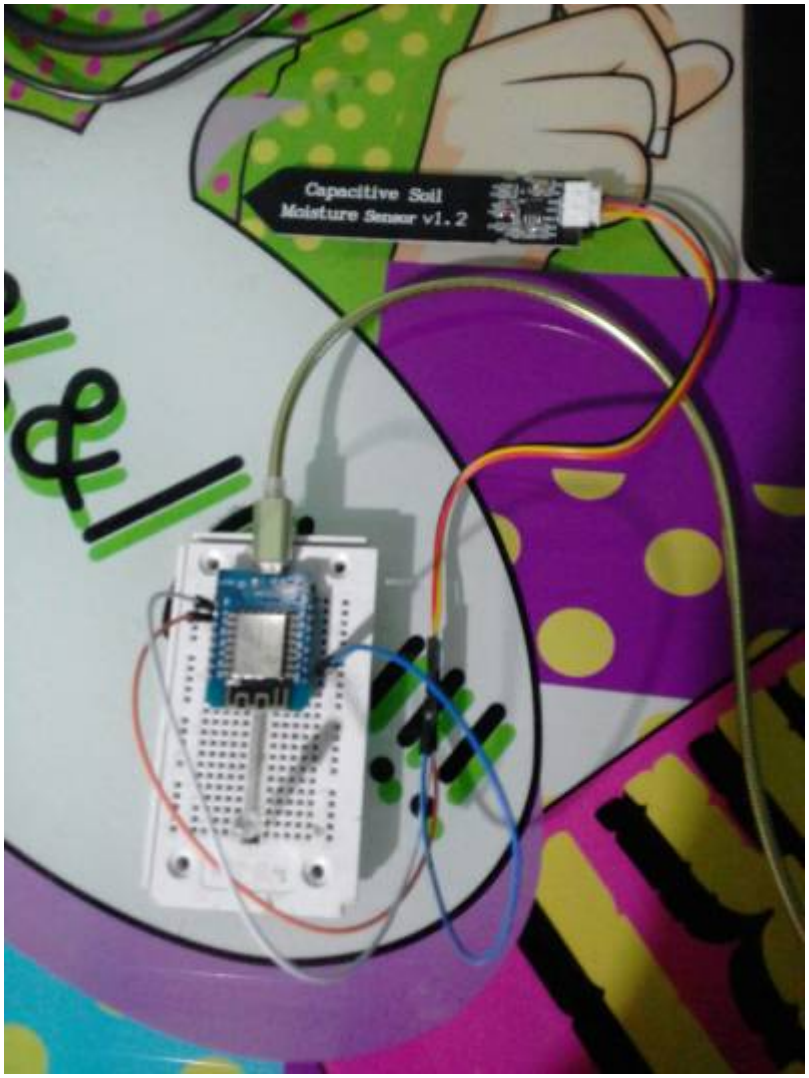
```
;PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:d1_mini_lite]
platform = espressif8266
board = d1_mini_lite
framework = arduino
lib_deps = NTPClient
```

control de humedad y temperatura

Se inicia de aqui conectar sensor de humedad capacitivo






```
#include <Arduino.h>

int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = D4;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor
```

```
void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
  Serial.begin(115200);
}

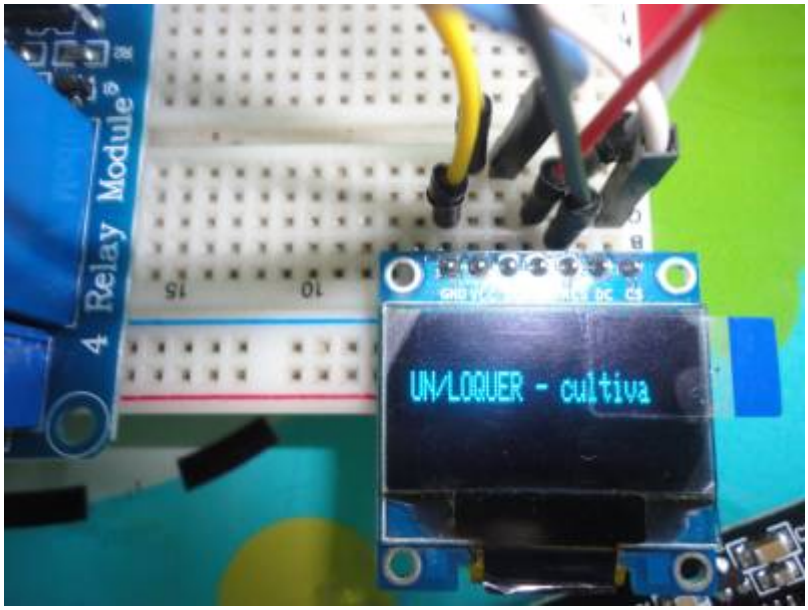
void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

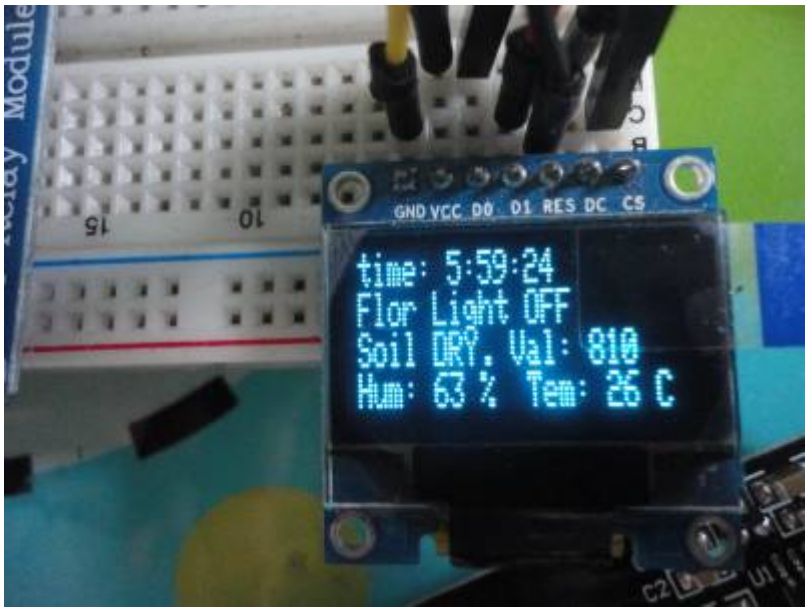


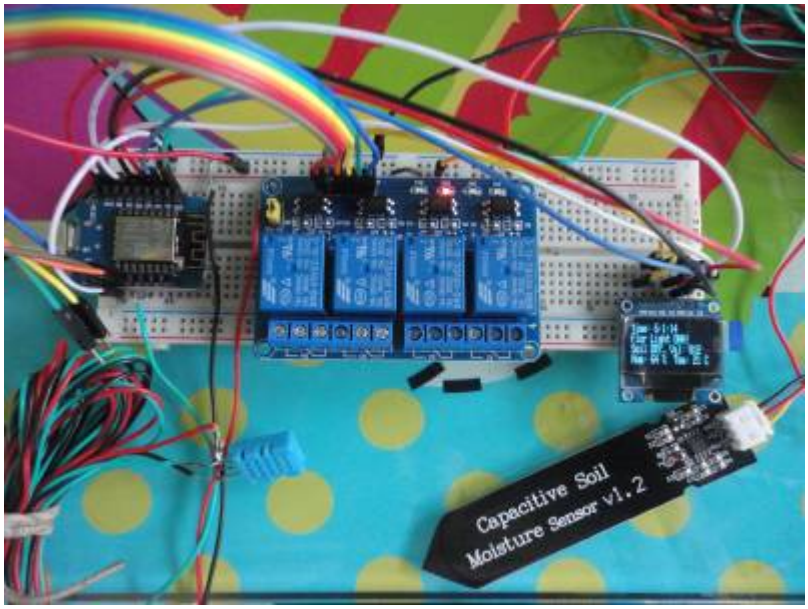
Sensor totalmente seco: 833 Sensor totalmente humedo: 482

actualizacion nov 3 2019

He creado una nueva versión del modulo que contiene un **capacitive moisture sensor** para medir la humedad en la tierra, un **module relay x 4** para controlar las luces y la ventilación. Para el ciclo solar de las luces estoy usando la libreria **time.h**, me di cuenta que es mejor usar sin el **timeAlarms.h** porque se pueden customizar mejor los ciclos de la luz y es mejor, o hasta ahora me funciona a mi asi.

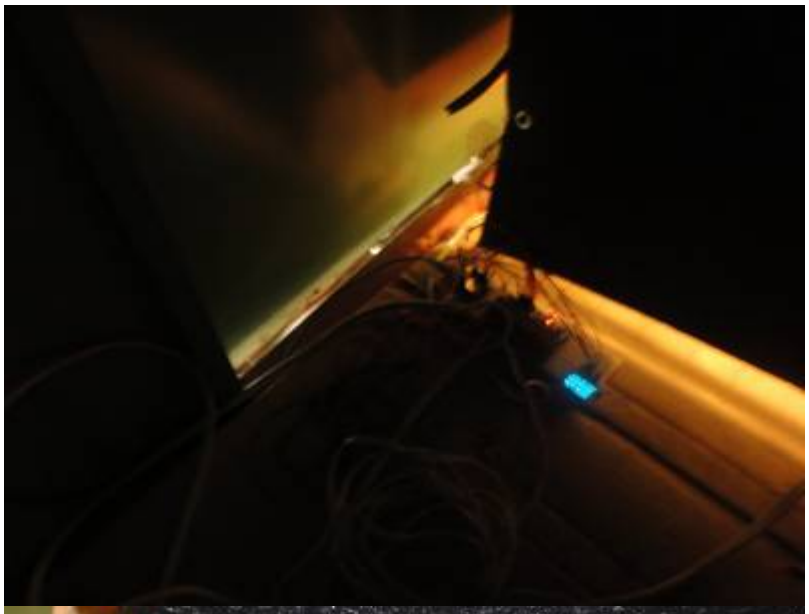


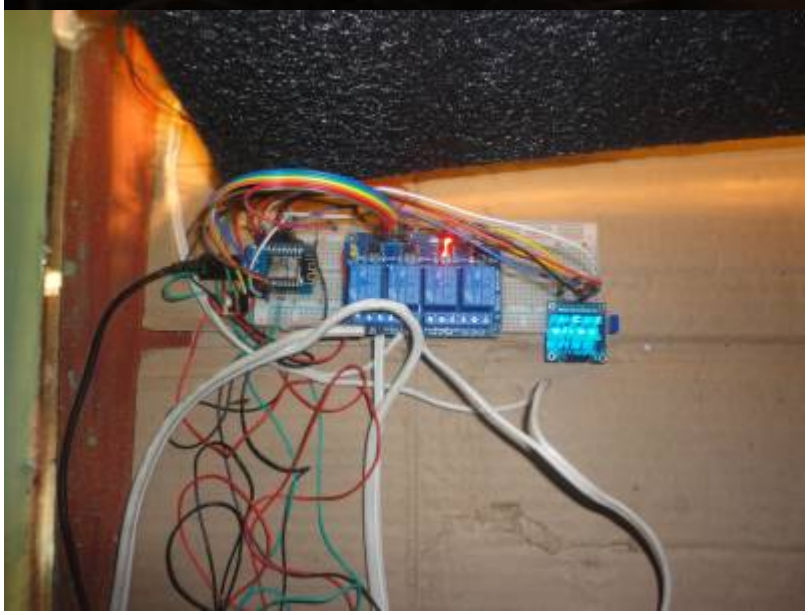




Mi abuelo me ha ayudado en la creación del indoor, una estructura de 90cm x 90cm x 1.5m. En su

interior he añadido el DHT11 para “medir” la temperatura y humedad interna del lugar. Y a la planta mas grande he anclado en la tierra el sensor capcitivo.













Link hacia el codigo del indoor

Actualización nov 9 2019

Hasta ahora la libreria time.h a funcionado muy bien, reemplazandome por completo un rtc.

Los datos se puede ver en:

Enlace al influx

Actualización nov 12 2019

Logro adjuntar a la trama de datos la humedad en la tierra de un sensor de humedad capacitivo.

Actualización 1 diciembre 2019

Por alguna razón con el código que tengo aqui ... se presenta el problema de que el modulo funciona bien 5 o 6 dias y despues deja de funcionar bien... no apagando la luz cuando debe de estar apagada o viceversa, dejando la luz prendida cuando debe de estar apagada. El problema es que solo se fija una vez el tiempo en el esp... si estre se reinicia o se va la energia... esto causa que el tiempo se reinicie... ocacionando que el tiempo de las alarmas de las luces no este sincronizado con el tiempo real.

*Solución: *Se usan las librerias NTP y Time simultaneamente... en resumen se fija al inicio el tiempo

local (libreria time) con request al servidor NTP... luego de eso el tiempo local se va actualizando cada 10 miniutos con un request al servidor NTP.

TODO

1. *RAPIDO*: integrar al código actual al actualizacion con el servidor NTP
2. integrar una web en la flash para programar la fecha del rtc digital time.h, (integrar esa parte que necesito de upayakuwasi y las alarmas)
3. pensar en una interfaz para pedir al usuario la programacion de la luz sea floración o vegetación o esquejes.
4. sacar una tarjetica en fritzing y pasarsela al brol o a uber, con el convertidor de la luz
5. actualizar el firmware por medio de ota

```
/* Library used
 * https://github.com/PaulStoffregen/Time
 * https://github.com/PaulStoffregen/TimeAlarms
 * https://learn.adafruit.com/dht/using-a-dhtxx-sensor
 *
 * resources from sensor
 *
https://www.switchdoc.com/2018/11/tutorial-capacitive-moisture-sensor-grove/
 *
  config NTP
  this code take the time from server using NTP
  https://lastminuteengineers.com/esp8266-ntp-server-date-time-tutorial/

  !!! IMPORTANT
  You need to adjust the UTC offset for your timezone in milliseconds.
  Refer the list of UTC time offsets.  Here are some examples for different
timezones:
https://upload.wikimedia.org/wikipedia/commons/8/88/World_Time_Zones_Map.png

  For UTC -5.00 : -5 * 60 * 60 : -18000
  For UTC +1.00 : 1 * 60 * 60 : 3600
  For UTC +0.00 : 0 * 60 * 60 : 0
  used -> const long utcOffsetInSeconds = 3600;
*/

#include <Arduino.h>
#include <NTPClient.h>
#include <DHT.h>
#include <Time.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <string.h>
```

```
// pin out oled ssd1306
#define OLED_MOSI 14   //D5
#define OLED_CLK 16    //D0
#define OLED_DC 13     //D7
#define OLED_CS 15     //D6
#define OLED_RESET 12  //D8

#define lights D1
char stateLights = 'F';

// config NTP variables
int time_hours = 0;
int time_minutes = 0;
int time_seconds = 0;
const int *ntp_hours, *ntp_minutes, *ntp_seconds;
const long utcOffsetInSeconds = -18000; // colombia UTC -5
char daysOfTheWeek[7][12] = {
  "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"
};

const char *ssid = "TP-LINK_Extender_C464C2";
const char *pass = "alex1988alex";

// Define NTP client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);

// config display
Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);
#if (SSD1306_LCDHEIGHT != 64)
//#error("Height incorrect, please fix Adafruit_SH1106.h!");
#endif

// define template function
void firstSync();
void syncNtptoTime();
void stateVegetative();
void stateFlowering();

void setup() {
  // display
  Serial.begin(115200);
  setTime(time_hours,time_minutes,time_seconds,30,11,19);
  WiFi.begin(ssid, pass);
  display.begin(SSD1306_SWITCHCAPVCC);
  display.clearDisplay();
  Serial.println();
  Serial.println();
  delay(1000);
```

```
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,10);
  display.println("Init ... ");
  display.display();
  delay(1000);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(0,10);
    display.setTextColor(BLACK, WHITE);
    display.println("connecting wifi ");
    display.print(".");
    display.display();
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0,10);
  display.setTextColor(WHITE);
  display.println("Wifi connected");
  display.println(WiFi.localIP());
  display.display();
  timeClient.begin();
  delay(5000);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,10);
  display.println("first sync NTP to Time hardware...");
  display.display();
  firstSync();
  delay(5000);
  //Serial.print(WiFi.localIP());
  //pinMode(ledTrigger, OUTPUT);
  pinMode(lights, OUTPUT);
}

void loop() {
  display.clearDisplay();
  display.setTextSize(0.5);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.print(daysOfTheWeek[timeClient.getDay()]);
  if (stateLights == 'V') {
    display.print(" | lights Onn");
  } else {
    display.print(" | lights Off");
  }
  display.println();
  display.setCursor(0,8);
```

```
  // time library
  display.print("rtc d: ");
  display.print(hour());
  display.print(":");
  display.print(minute());
  display.print(":");
  display.print(second());

  // time ntp
  display.setCursor(0,16);
  display.print("last sync: ");
  display.print(*ntp_hours);
  display.print(":");
  display.print(*ntp_minutes);
  display.print(":");
  display.print(*ntp_seconds);

  // state lights
  display.setCursor(0,24);

  syncNtptoTime();
  // stateFlowering();
  stateVegetative();

  // update display
  display.display();
  delay(1000);
}

void firstSync() {
  // pointer update with what is in ntp
  timeClient.update();
  ntp_hours = &time_hours;
  ntp_minutes = &time_minutes;
  ntp_seconds = &time_seconds;
  // first update ntp to time variables
  time_hours = timeClient.getHours();
  time_minutes = timeClient.getMinutes();
  time_seconds = timeClient.getSeconds();
  // frist sync Ntp time to local time, get values from value pointers
  setTime(*ntp_hours,*ntp_minutes,*ntp_seconds,30,11,19);
  timeClient.end();
}

void syncNtptoTime() {
  // update local time from NTP server over get minutes and seconds
  // from local time hardware
  if ( (minute() == 10 && second() == 58) ||
       (minute() == 20 && second() == 58) ||
       (minute() == 30 && second() == 58) ||
```

```
      (minute() == 40 && second() == 58) ||
      (minute() == 50 && second() == 58)
    ) {
    timeClient.update();
    time_hours = timeClient.getHours();
    time_minutes = timeClient.getMinutes();
    time_seconds = timeClient.getSeconds();
    timeClient.end();
    // TODO: how to update days, month and year ?
    setTime(*ntp_hours,*ntp_minutes,*ntp_seconds,30,11,19);
    delay(500);
  }
}

void stateVegetative() {
  // 18 hours lights, 6 hours night
  // they turn on 6 in the morning and turn off at the 12 last night
  if ( hour() < 6 ) {
    digitalWrite(lights, LOW);
    stateLights = 'F';
  } else {
    digitalWrite(lights, HIGH);
    stateLights = 'V';
  }
}

void stateFlowering() {
  // 12 hours lights, 12 hours night
  //  a las 6 de la mañana se prenden y a las 6 de la tarde se apagan
  if ( (hour() >= 6) && (hour() < 18) ) {
    digitalWrite(lights, HIGH);
    stateLights = 'V';
  } else {
    digitalWrite(lights, LOW);
    stateLights = 'F';
  }
}
```

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

Permanent link:
https://wiki.unloquer.org/personas/johnny/proyectos/indoor_diy_autosostenible?rev=1575186014

Last update: **2019/12/01 07:40**