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);
```

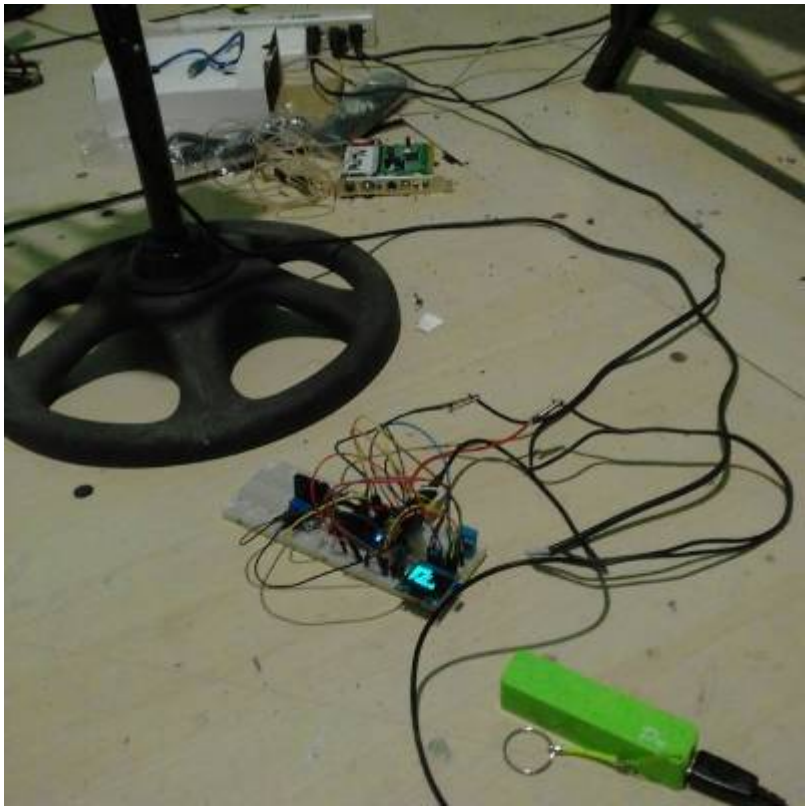
```
  // Serial.print(F("Lectura Humidity: "));
  // Serial.print(h);
  // Serial.print(F("% Lectura Temperature: "));
  // Serial.print(t);
  // Serial.print("\n");

  /*
  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));
}
```

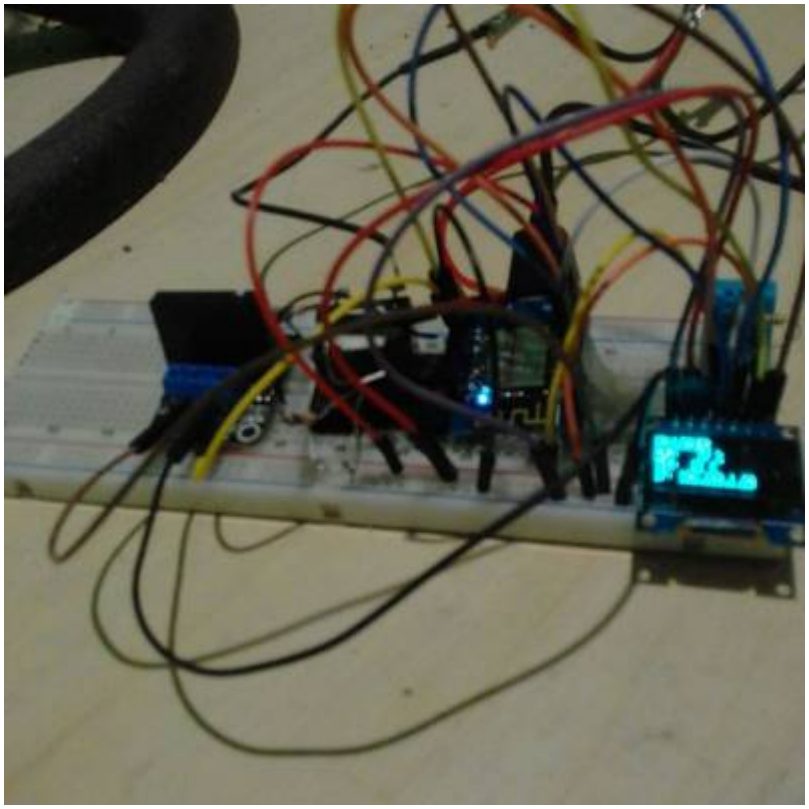
Configuración de plataforma

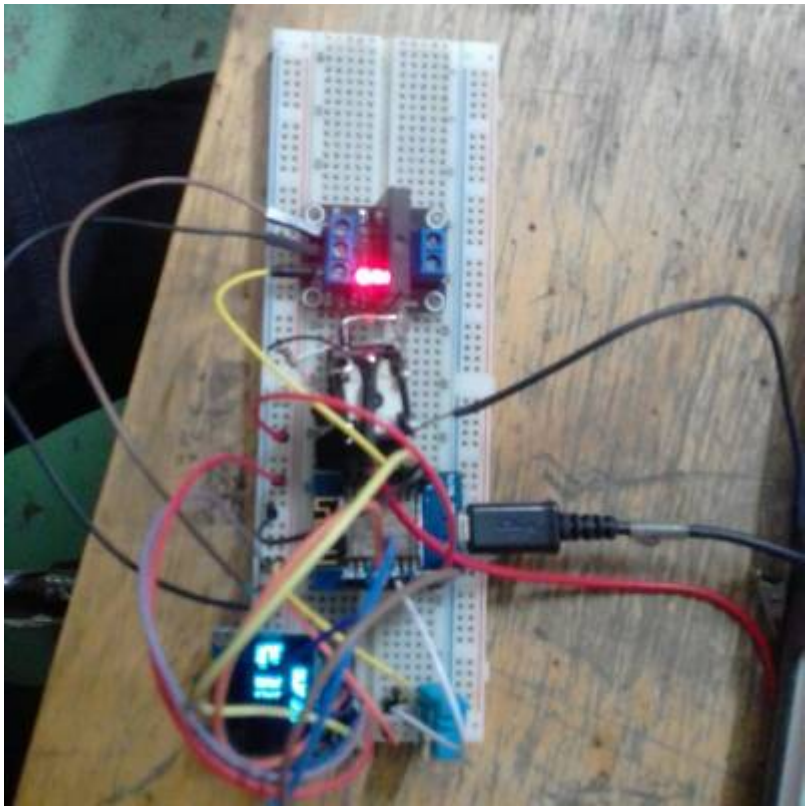
1. Se crea una base de datos

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

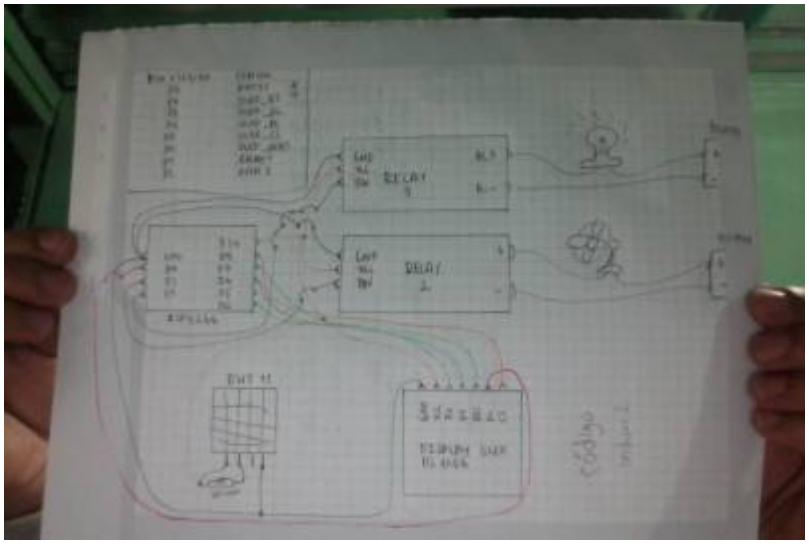
primer prototipo de control automatico

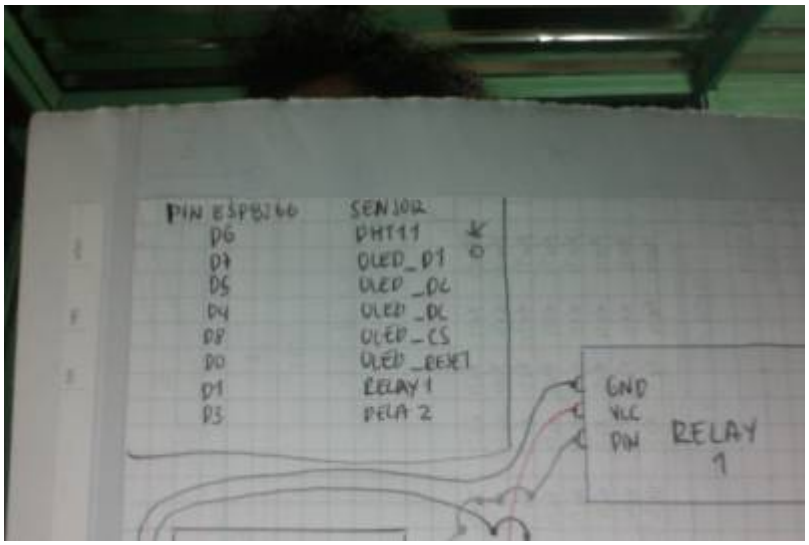






2% 1:47 AM
192.168.1.20
Automatic grow garden
RELAY 1
On
Off
Relay 2
On
Off





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.

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
```

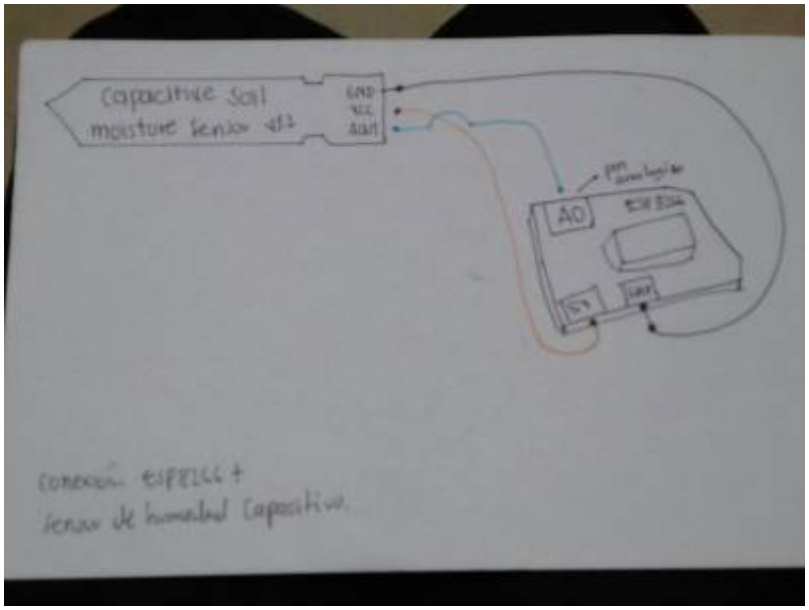
```
void loop() {
  timeClient.update();
  Serial.print(daysOfTheWeek[timeClient.getDay()]);
  Serial.print(", ");
  Serial.print(timeClient.getHours());
  Serial.print(":");
  Serial.print(timeClient.getMinutes());
  Serial.print(":");
  Serial.print(timeClient.getSeconds());
  Serial.println("");
  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
  */
}
```

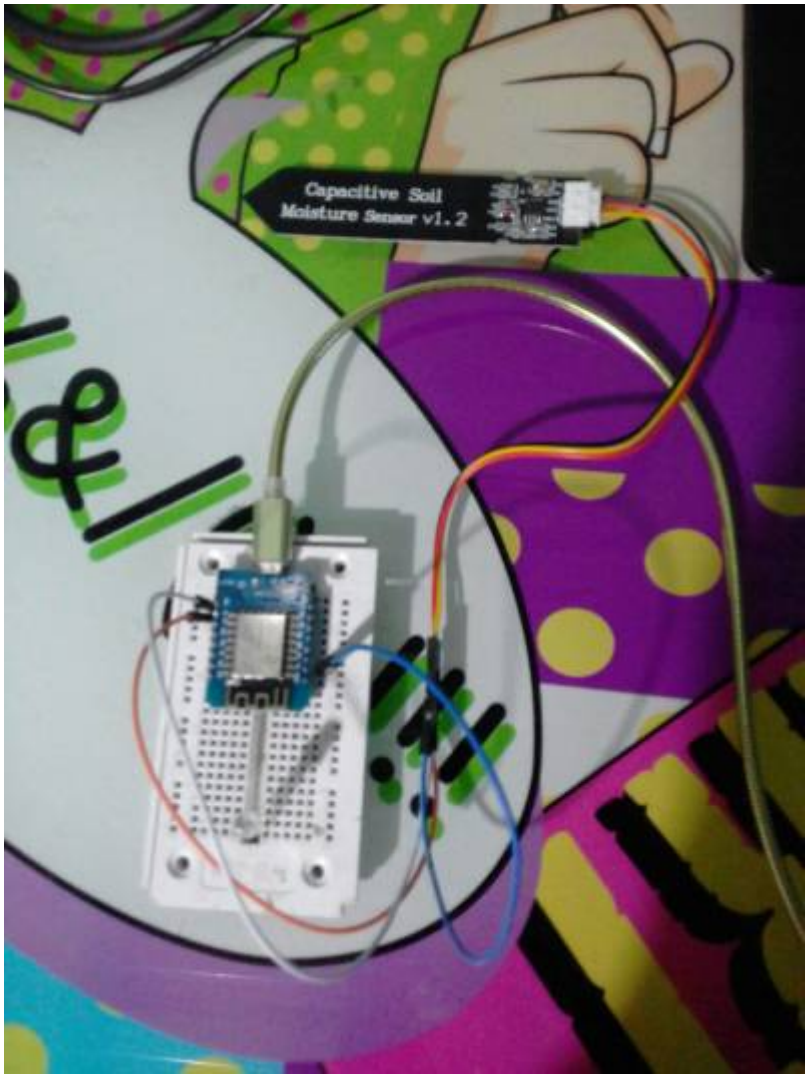
el platformio.ini

```
;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





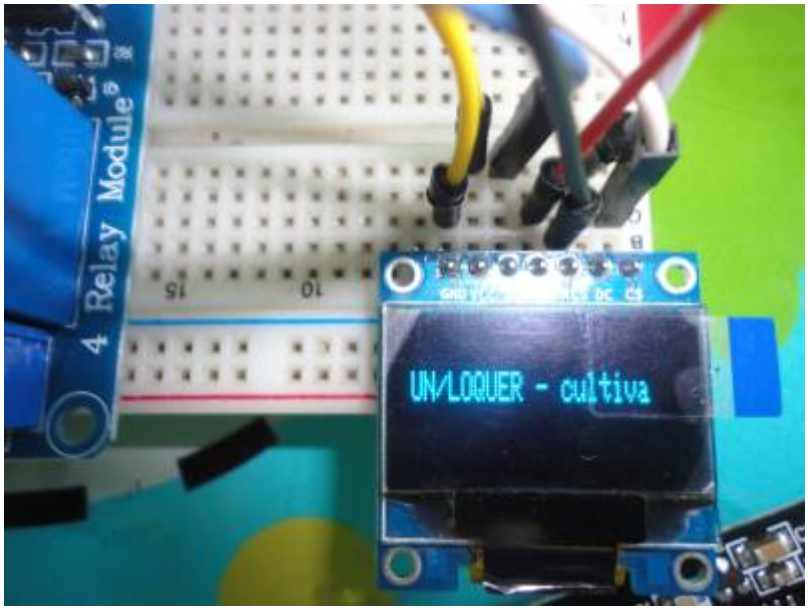

```
#include <Arduino.h>

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
```

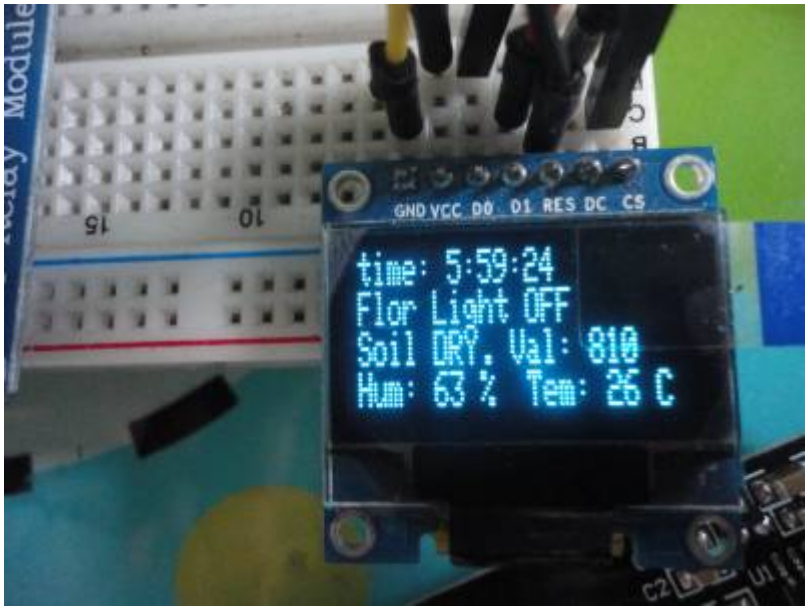
```
void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
  Serial.begin(115200);
}

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);
}
```

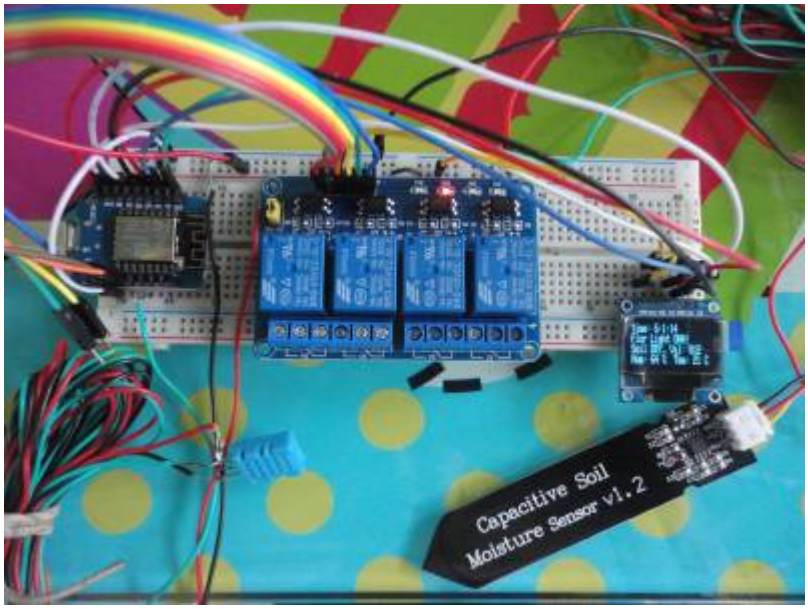
Sensor totalmente seco: 833 Sensor totalmente humedo: 482

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.

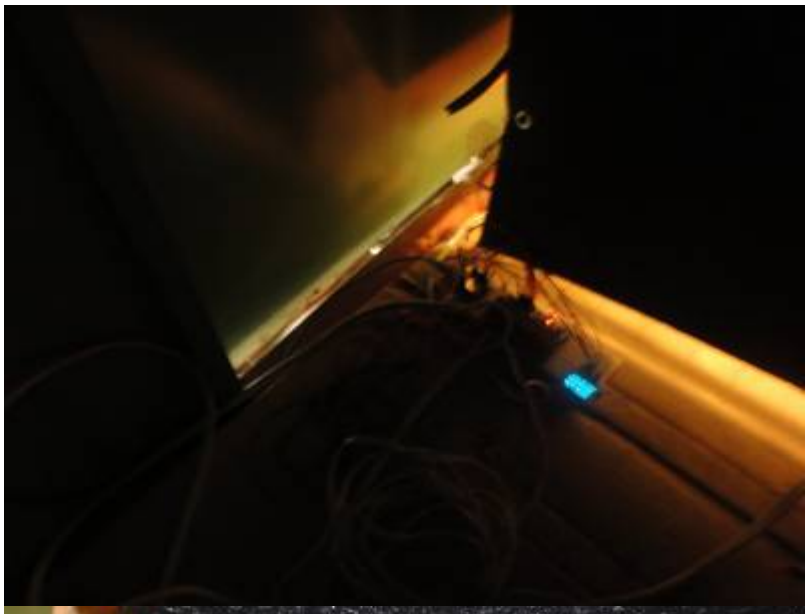


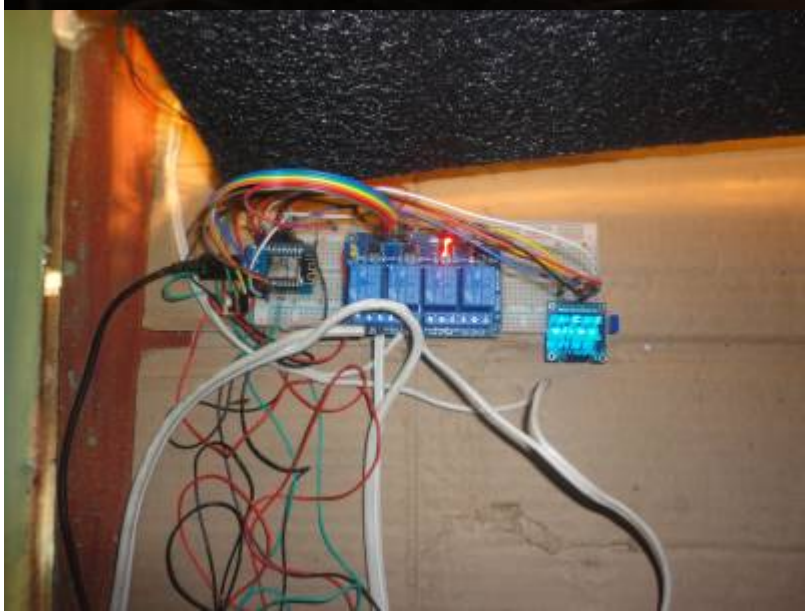




Mi abuelo me ha ayudado en la creación del indoor, una estructura de 90cm x 90cm x 1.5m. En su

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

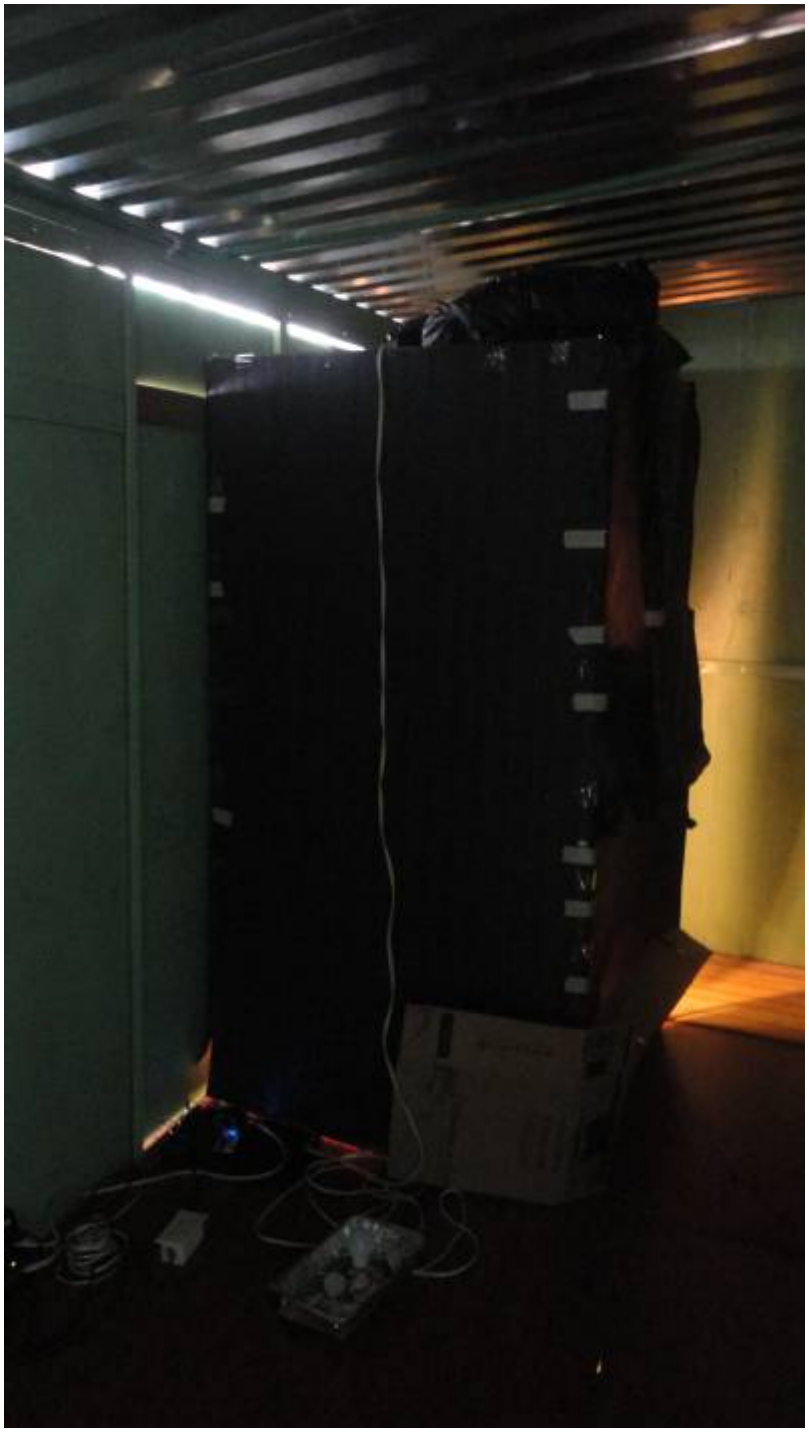
1. *RAPIDO*: integrar al código actual al actualizacion con el servidor NTP
2. integrar una web en la flash para programar la fecha del rtc digital time.h, (integrar esa parte que necesito de upayakuwasi y las alarmas)
3. pensar en una interfaz para pedir al usuario la programacion de la luz sea floración 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

Actualización 13 de julio 2020

En todo este tiempo he estado aprendiendo mas que todo sobre las plantas, conociendolas y mirando como los diferentes factores que inciden en ella(Temperatura y humedad, calidad de la tierra, calidad del agua, lummens necesarios, etc...) Entonces antes de seguir con el codigo y el prototipo de placa. Estuve un tiempo pensando en la mejoras que se le arian al indoor.

He dispuesto un extrator de 125v para sacar el aire caliente y regular la temperatura del indoor en la parte superior. Tambien arregle el espacio y le trabaje junto con mi tio para volverlo mejor.

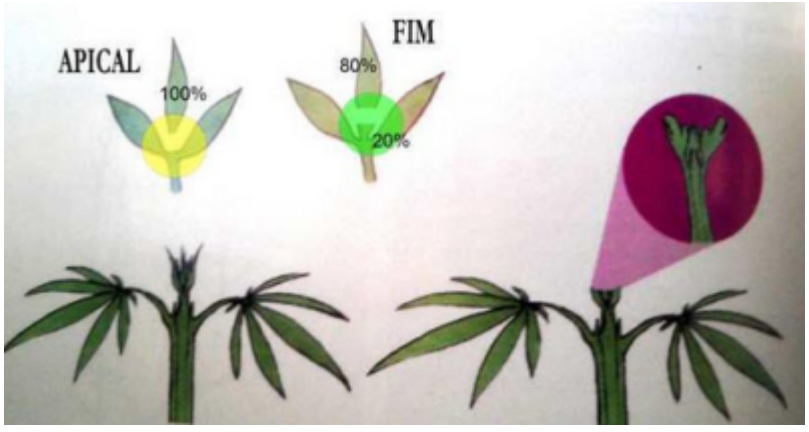
ANTES



AHORA

Tambien mi tio me ayudo a construir un reflector para adaptarle hasta 6 bombillos a este mismo. Este cambio ayudo mucho a las plantas pero tambien afecto bastante otros factores que detallaremos mas adelante.

En este cultivo tambien estuve experimentando y aprendiendo sobre las podas. En especial la fim y la apical.



Entonces esta fue mi planta experimental... a la cual le hice de todo... apicales y fim... dando espacio minimo de un mes entre poda y poda para que la planta se recuperara y finalmente siempre podando

en cuarto menguante.

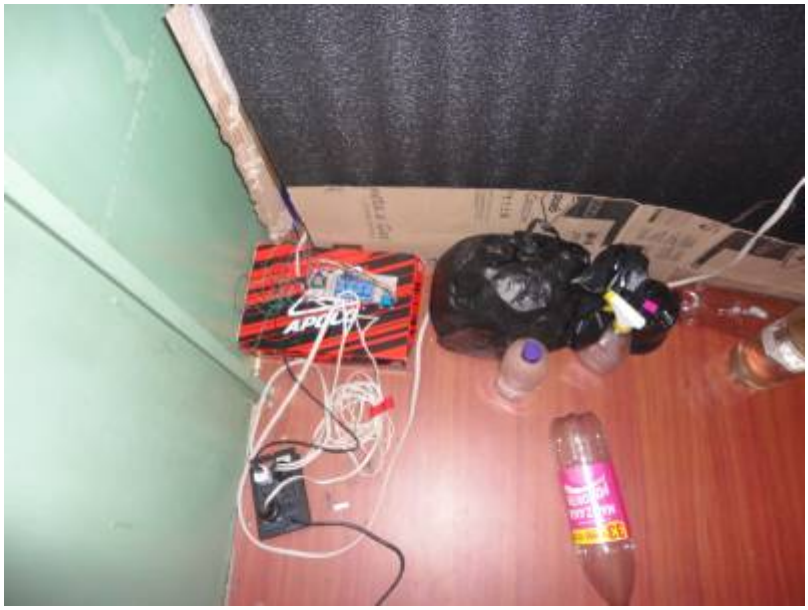
Dadas las experimentaciones, la planta desarrollo un brazo mas grande que el otro. Tambien se observa como queda la cicatriz del corte, pero la planta a medida que pasa el tiempo se va curando.

Este experimento fue para aprender a realizar un cultivo tipo scrog... donde son necesarias este tipo de podas para hacer que las plantas sean mas eficientes a la hora del cultivo... pero el unico contra es que se alarga el tiempo de vegetacion para que las plantas se recuperen.

Pasamos la placa base a baquela y actualizaciones y fixes en el código

Primero hablaremos de la placa. Entonces antes teniamos este hardware, 4 reles (de los cuales solo estoy usando 1) y una pantallita para ver las lecturas del dht11.

Antes



Cuando teniamos este circuito. Disponiamos de problemas de interrumciones de corriente... porque como vemos los cables estan pegados con mocos y mi casa a veces se mueve el piso, ocacionando que los cables se desconenten.

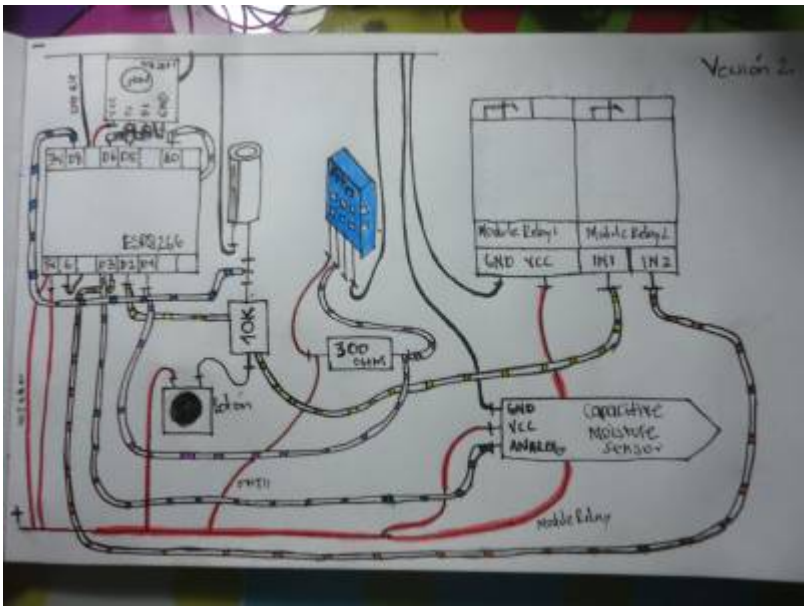
Por otro lado estabamos haciendo un request al server **NTP** cada minuto y con esta misma hora era que se configuraba todo en el codigo... ocacionando que cuando se cae el internet... no exista hora y se volviera loca la alarma...

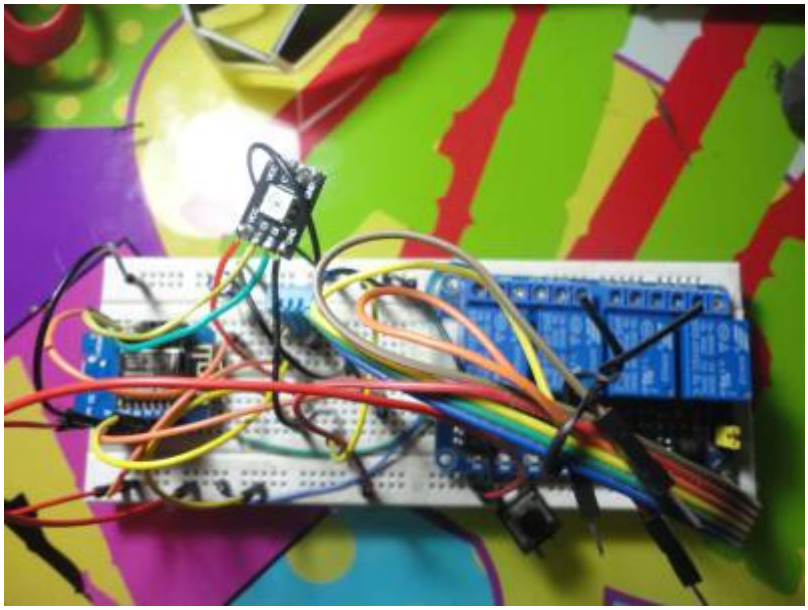
Otra cosa grave era que cuando se hiba la luz... entonces como la alarma del codigo estaba seteada con una hora quemada... coacionaba que los ciclos de foto periodo se corrieran o no fueran los esperados.

Otro problema que existia era que cuando las plantas ya necesitan el cambio de fotoperiodo... era necesario ingresar de nuevo el firmware a la placa con el cambio de periodo... entonces era tedidoso cada cierto tiempo estar ingresando el codigo a la base.

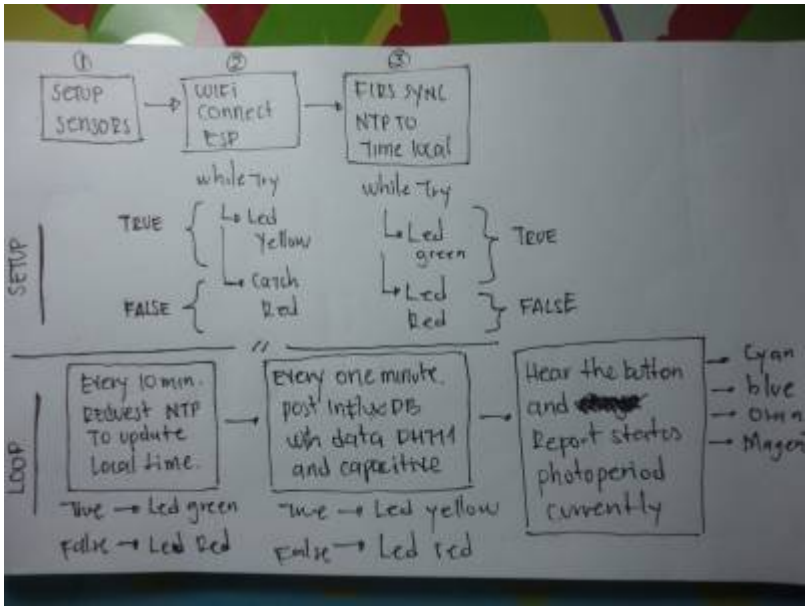
Ahora

Circuito actualmente





Arquitectura del firmware



Resumidamente cuando se trata de conectar el led parpadea amarillo, si falla entonces parpadeara rojo. Luego hara el request al NTP para sincronizar el timeAlarms y setear la alarma inicialmente. En este proceso el led parpadeara verde, si falla parpadeara rojo.

Luego de esto el led cada cierto tiempo empezara a parpadear en un color especifico... inicialmente el cyan. Y cada minuto el led parpadeara amarillo, para indicar que esta inviando las lecturas del capacitivo y el dht al influxdb.

Mientras eso ocurre no es adecuado presionar el boton para cambiar el ciclo de floracion. 60%>

Como funciona actualmente

Tenemos basicamente estos estados.

1. **Cyan:**Indica estado vegetativo de dia. Las luces se encianden a las 6 y se apagan a las 24
2. **Azul:**Indica estado vegetativo de noche. Las luces se encienden a las 18 y se apagan a las 12

del otro dia

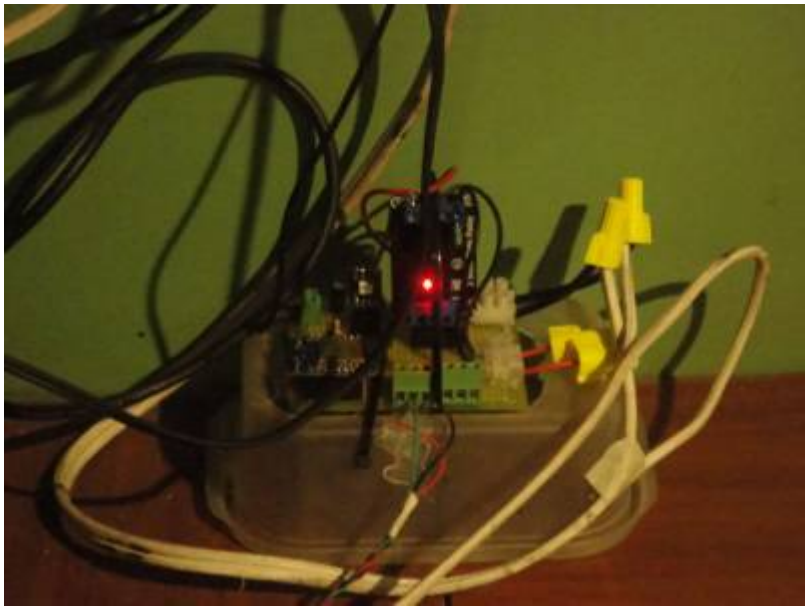
3. **Naranja:**Indica estado floracion de dia. Las luces se encienden a las 6 y se apagan a las 18
4. **Magenta:**Indica estado floracion de noche. Las luces se encienden a las 18 y se apagan a las 6 del otro dia.
5. **Amarillo:**Indica que el esp esta haciendo un POST. **En este momento no deberiamos de cambiar el estado de fotoperiodo de las luces. (no apretar el boton)**
6. **Verde:**Indica primera conexion del esp a la red, tambien primera sincronizacion del tiempo local por medio del NTP o actualizacion de este mismo. **En este momento no deberiamos de cambiar el estado de fotoperiodo de las luces. (no apretar el boton)**
7. **Rojo:** Indica que el esp no se pude conectar a la red. Si el la primera vez... Si ocurre durante un update del tiempo local desde el ntp... no pasa nada... no se actualizara... este test se hizo forzozamente durante un dia sin conexion en el lugar. La alarma local siguio funcionando normalmente.

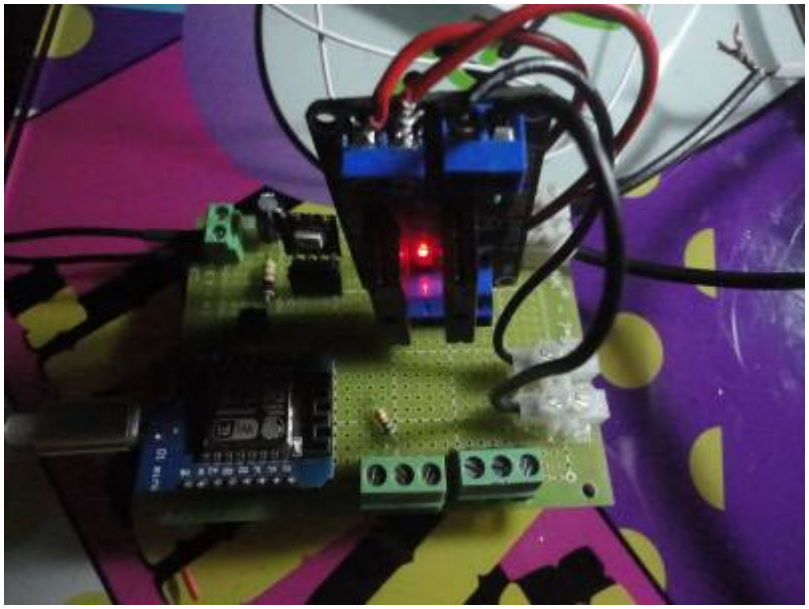
Estos estados fueron ideados y pensados principamente por dos razones:

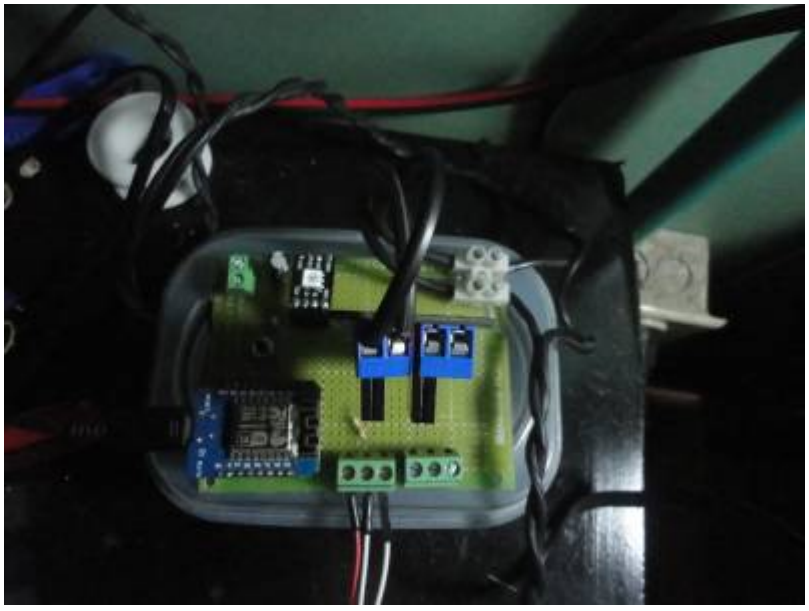
- No tener que estar programando el esp para cambiar el fotoperido de las luces.
- Favorecer las condiciones de temperatura y humedad en el cuarto de cultivo. Ya que se ha investigado que apagando las luces durante el dia en lugares muy calurosos, se reduce la temperatura. Estos datos son corroborados mirando la trama de datos del influx db... donde se nota claramente que en el dia se superanban temperaturas de 35 grados con luces encendidas... notandose quema de hojas en las puntas.

Con luces apagadas en el dia... osea configuracion azul, la temperatura no supera los 25, 26 grados. Datos adecuados.... no ideales pero mas cercanos a los necesarios.

Ahora la placa sobre la baquelita 😮







Finalmente estas son las planticas con el dht y el scrog

Aun queda pendiente mucho trabajo... configurar el esp en modo sta-ap para que podamos encender o apagar ventiladores... o controlar la velocidad por medio de dimmers y una pagina web como cliente.

Pero ahora lo mas importante seria el riego y poder medir con el capacitivo cuando seria un punto ideal para el riego. Asi como el dht definio puntos vitales para la configuracion de nuevas alarmas

Actualización 9 de agosto 2020

[https://blog.usejournal.com/automated-smart-home-irrigation-system-9061c391f8e2[http://example.com|inspiracion]] He arreglado las lecturas del dht haciendo 4 lecturas en un minuto y promediando ese dato. Luego es enviado al influx para arreglar las interpolaciones generadas en la grafica del influx.

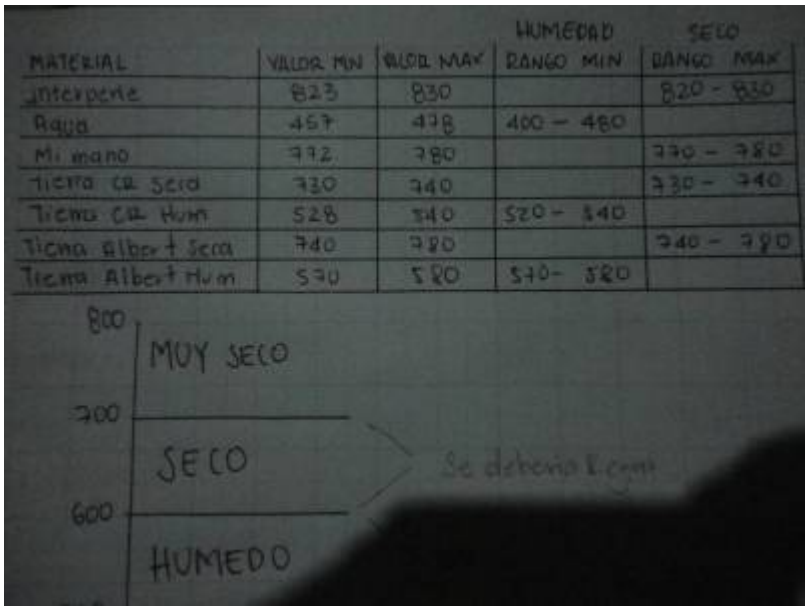
Ademas de eso. He añadido las lecturas de un sensor de humedad capacitivo **(Soil moisture sensor v1.2)** promediando el dato tambien y enviandolo al influx.

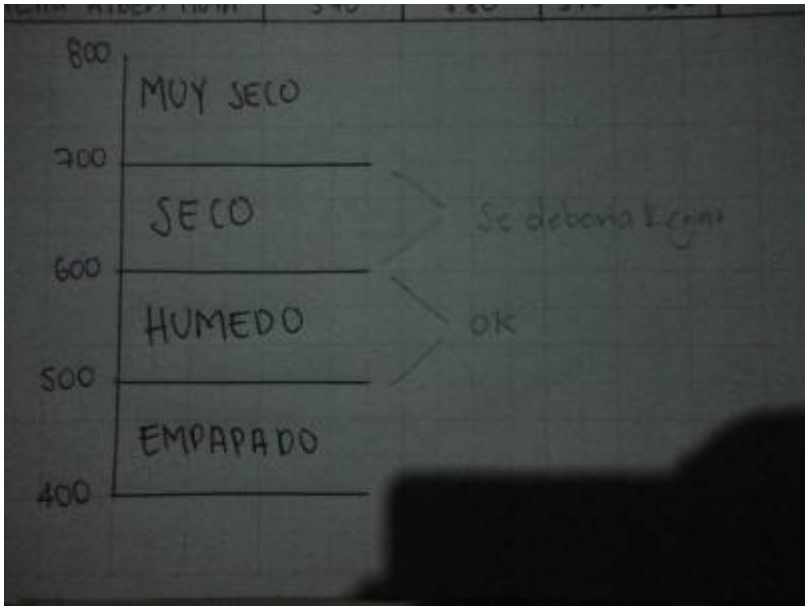
Codigo inicial para empezar a usar el sensor.

```
const int numReadings = 30;
const int sendingInterval = 1000;

int capacitive_sensor = A0;

int readings_capacitive_sensor[numReadings];
int readIndex = 0;
int total = 0;
int average = 0;

void setup() {
    Serial.begin(115200);
```

```
  // asigned all index to 0
  for(int thisReading=0; thisReading < numReadings; thisReading++){
    readings_capacitive_sensor[thisReading] = 0;
  }
}

void loop() {
  int currentReading = analogRead(capacitive_sensor);
  if( currentReading > 25 && currentReading < 1000) {
    calculateAverage(currentReading);
  } else {
    delay(10);
  }
}

void calculateAverage(int currentReading) {
  total = total - readings_capacitive_sensor[readIndex];
  readings_capacitive_sensor[readIndex] = currentReading;
  total = total + readings_capacitive_sensor[readIndex];
  readIndex++;
  average = total / numReadings;
  if (readIndex >= numReadings) {
    Serial.println(average);
    // envio dato
    readIndex = 0;
    delay(sendingInterval);
  }
  delay(sendingInterval / numReadings);
}
```

Calibración amateur





El resumen seria que cuando se **superen valores de 600 se deberia de regar**. Pero importante tambien **no regar que se baje hasta llegar a valores cercanos a 490-500**

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

Permanent link:
https://wiki.unloquer.org/personas/johnny/proyectos/indoor_diy_autosostenible?rev=1596952656

Last update: **2020/08/09 05:57**