

IOT PROJECT

Automatische serre ESP32 + RaspberryPi

Dylan Geldhof

Inhoud

1. BESCHRIJVING	2
2. HARDWARE	3
2.1. ESP32	3
2.2. Power supply en LM2596 step down buck converter	3
2.3. DS18B20	4
2.4. RCWL-1601	4
2.5. DHT11	5
2.6. FC-28 Hygrometer	5
2.7. Rfid RC522 module	6
2.8. RGB Ledring	7
2.9. Raspberry Pi 4	7
2.10. Overige hardware	7
3. AANSLUITSCHEMA	8
4. INFLUXDB	10
5. GRAFANA	10
6. CODE	12

1. Beschrijving

Deze documentatie zal je laten zien hoe je een kleine mini serre kan bouwen die verschillende automatiseringen heeft. Zo kan het allerlei sensor data lezen, deze laten uitschrijven op een LCD, en kan het deze ook laten afbeelden op Grafana met een Raspberry pi.

Naast deze uitlezing van de data kan hij zichzelf ook bijsturen waar nodig. Bijvoorbeeld als de grond niet meer vochtig genoeg is, dan zal de ESP32 een waterpomp aanzetten die water naar de grond brengt.

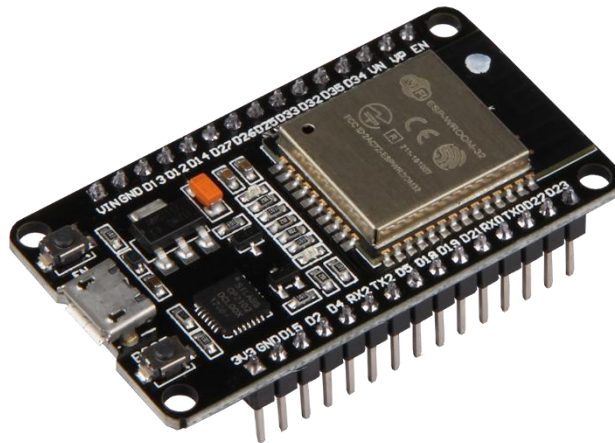
Er is ook een pcb-design bij deze documentatie die je kan laten namaken zodat je geen breadbord hoeft te gebruiken.

Ook hoeft de ESP32 niet voortdurend verbonden te zijn met een laptop, eens de code opgeladen is kan je het met de power adapter bekrachtigen.

2. Hardware

2.1. ESP32

Centraal staat de ESP32 als hoofd van het systeem. Deze komt met een Wi-Fi module en heeft GPIO's die gebruikt kunnen worden als analoge of digitale input. Dit stuurt heel het systeem aan.



Figuur 1 ESP32

2.2. Power supply en LM2596 step down buck converter

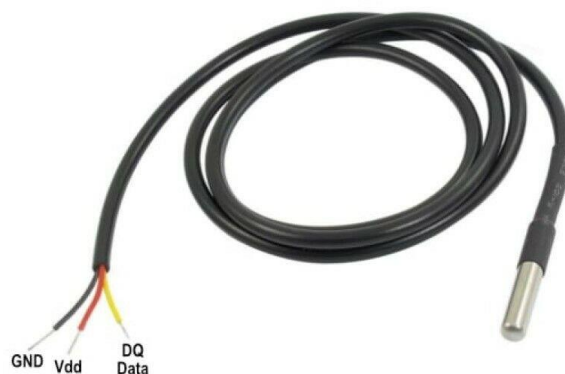
Om heel het systeem te voeden koos ik voor een laptop power supply met 20V spanning en 2A stroom. Deze paste perfect op dc-connector zodat het hele systeem kan draaien via 1 voeding. Om aan de ESP32 5V input spanning te geven wordt er gebruik gemaakt van een LM2596 IC (Instruments, sd). Deze reguleert een inputspanning van 3.2-40V DC naar een spanning van 1.25-35V DC met een maximale stroom van 2A. Ook deze wordt aangepast via een kleine potentiometer op het IC. Het IC zelf is ook voor 80% energie efficiënt.



Figuur 2 LM2596

2.3. DS18B20

Dit wordt gebruikt als tempsensor voor in de grond zelf. De sensor zelf is volledig waterdicht ingesloten en heeft lange waterdichte kabels. Ideaal om in de grond te steken en te gebruiken bij zowel hoge als koude (vries) temperaturen.



Figuur 3 DS18B20 Digitale Temp Sensor

2.4. RCWL-1601

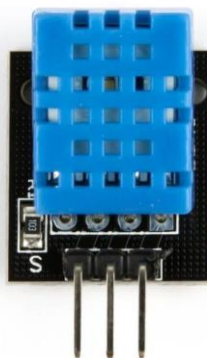
Deze sensor kan op een redelijk accurate manier een afstand meten. Deze wordt dan ook gebruikt om te kijken of er een waterreservoir bijna leeg zou zijn. Zo kan deze tijdig bijgevuld worden en komen de planten niet zonder water te staan.



Figuur 4 RCWL-1601 Distance Sensor

2.5. DHT11

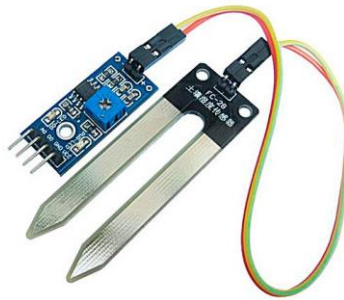
DHT11 heeft een redelijk accurate luchtvochtigheid sensor. Deze is goedkoop en meervoudig beschikbaar dus ideaal voor dit project.



Figuur 5 DhT 11 Sensor

2.6. FC-28 Hygrometer

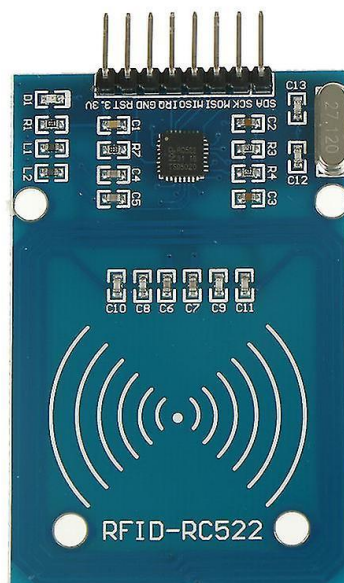
De vochtigheid van de bodem meten is belangrijk om te weten wanneer er water toegevoegd moet worden aan het systeem. Met de FC-28 Hygrometer kunnen we een bereik van 0-4095 mappen op een analoge sensor. Hierdoor kunnen we met hoge nauwkeurigheid bepalen wanneer we water al dan niet moeten toevoegen aan het systeem.



Figuur 6 FC-28 Hygrometer

2.7. Rfid RC522 module

Deze module laat toe om rfid tags te scannen met de esp32. Hierdoor kunnen we instellen welke plant onze serre gaat gebruiken



Figuur 7 RC522 Module

2.8. RGB Ledring

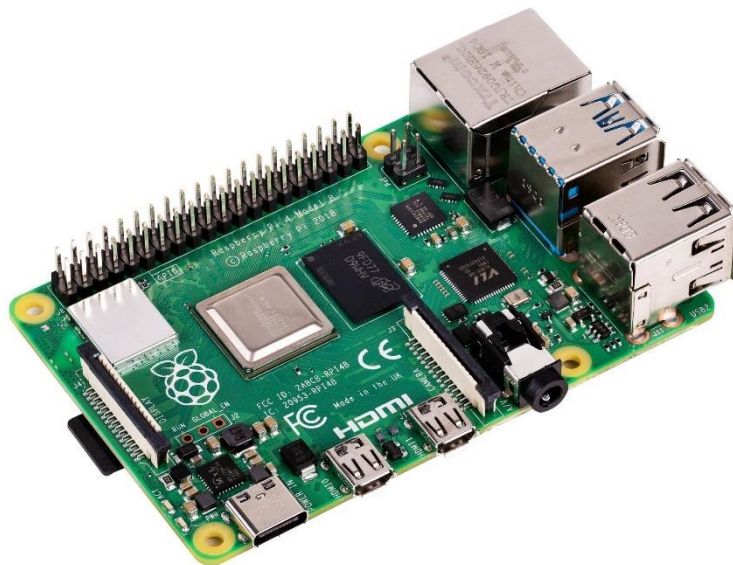
Deze ledring laat de serre toe extra licht naar de plant te schijnen wanneer het nodig zou zijn. Als het te bewolkt zou zijn, of het staat in een donkere kamer. Het is ook heel belangrijk dat dit een RGB-led is en niet een gewone witte led want deze hebben geen effect op de groei van de plant. Deze ledring is van AZ Delivery maar valt wat klein.



Figuur 8 RGB Ledring

2.9. Raspberry Pi 4

Deze computer laat ons toe om de data op te slaan en deze weer te geven in een grafiek die je kan bekijken met laptop/smartphone als je op dezelfde wifi zit.

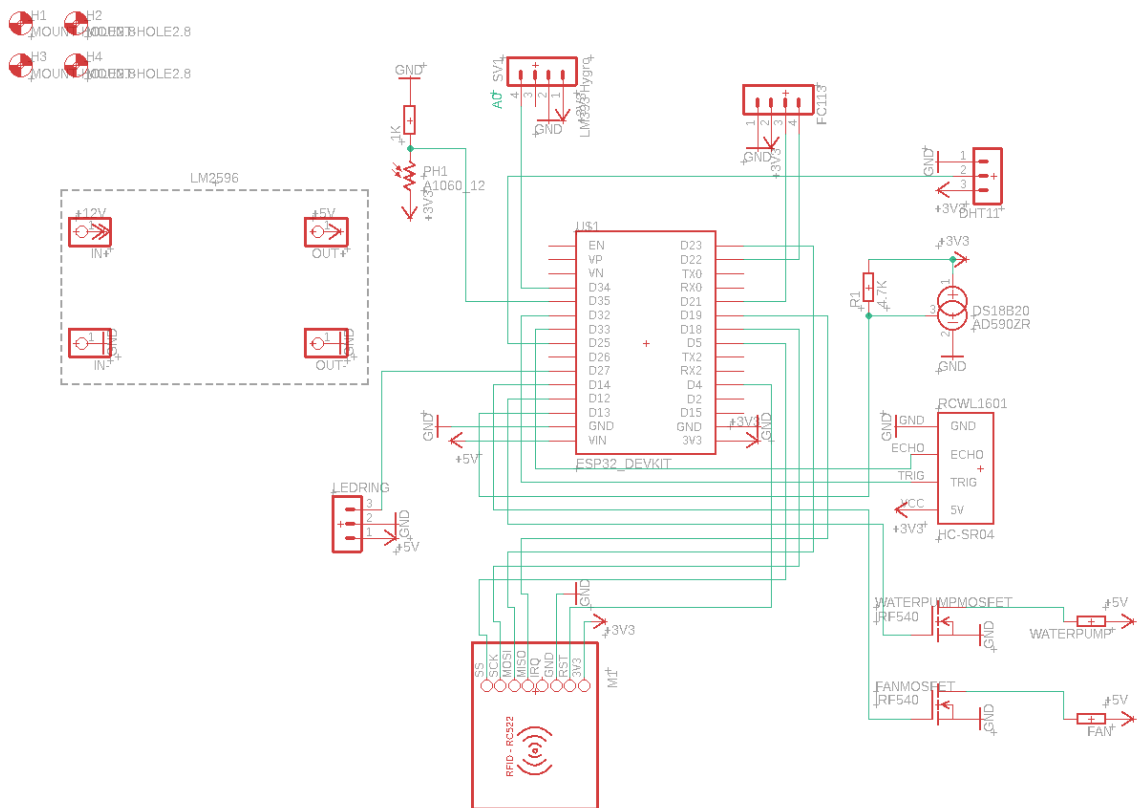


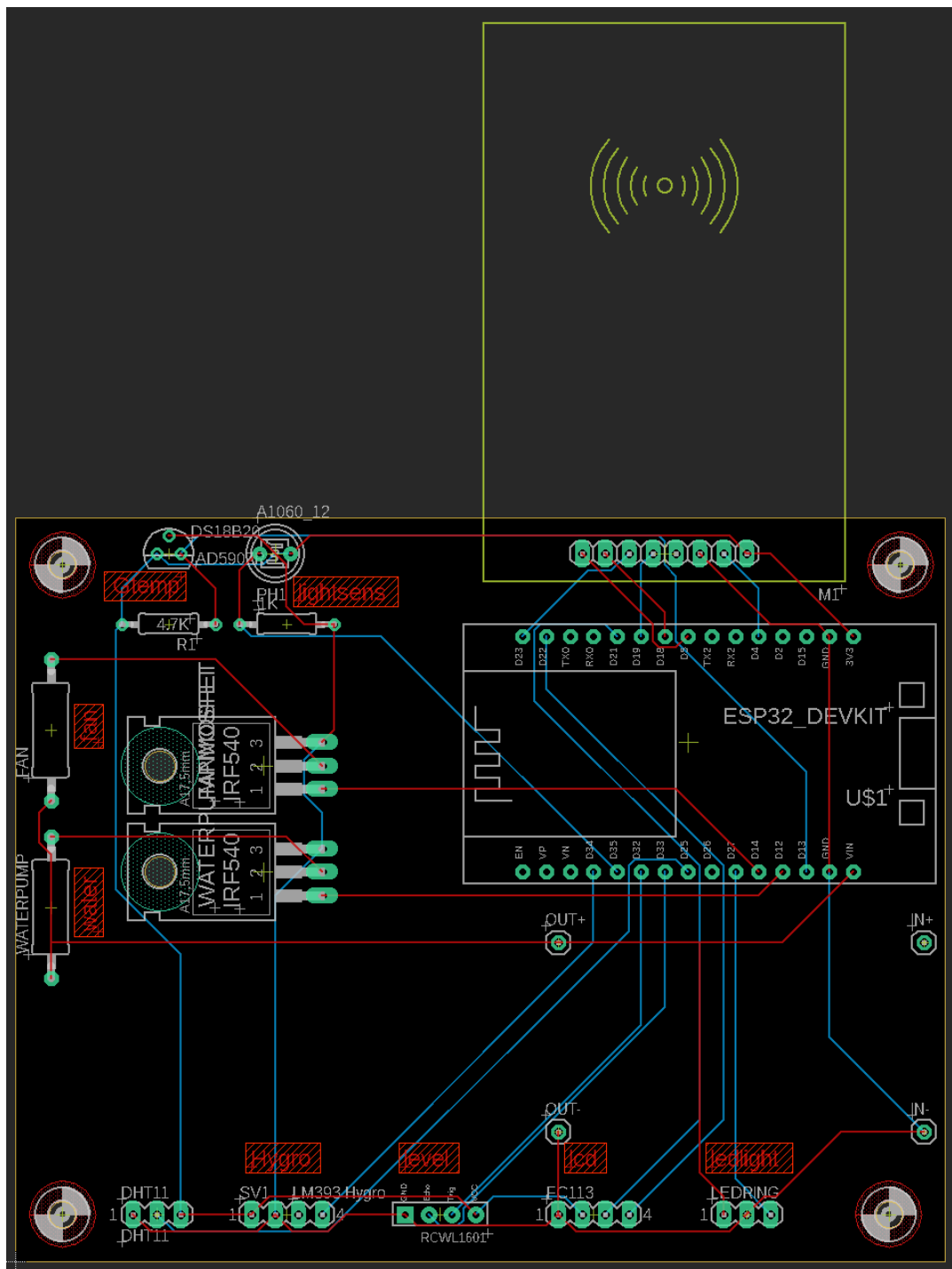
Figuur 9 Raspberry Pi 4

2.10. Overige hardware

Er werd ook gebruik gemaakt van enkele simpele componenten zoals mosfets, weerstanden, kabels, LDR...

3. Aansluitschema





In bovenstaande PCB Schema staat een 'klein' fout. De pinnen van de dc to dc-converter zijn gespiegeld. In de weergave van mijn project is er ook 1 mosfet vervangen door een relay.

4. InfluxDB

InfluxDB staat in als database van ons project, het slaagt de sensordata op samen met een tijdscode zo kan je precies zien wanneer welk data gemeten is geweest. Om onze data naar InfluxDB te sturen gebruiken we een python script die de overbrugging maakt tussen ESP32 en InfluxDB. Ook deze pythoncode zal in de documentatie folder staan en op Github.

Om InfluxDB op te stellen refereer ik naar de handleiding van de docent Van Eerdenbruh. In essentie moet je via de CLI van de Raspberry pi InfluxDB downloaden en installeren. Volgende commando's invoeren

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/os-release
echo "deb https://repos.influxdata.com/debian $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list

dan

sudo apt-get update
sudo apt-get install influxdb

dan

sudo systemctl unmask influxdb
sudo systemctl start influxdb
sudo systemctl enable influxdb
```

De config file passen we aan zodat er ook met HTTP gewerkt wordt.
Nu kunnen we een database maken in InfluxDB met

```
influx
CREATE DATABASE Plant
```

Nu maken we hierover een gebruiker aan, zo kan Grafana aan onze data.

```
CREATE USER dylanpi WITH PASSWORD dylanpi
GRANT ALL ON Plant TO dylanpi
```

Dit is al een handige start om te beginnen. Natuurlijk kan je andere namen gebruiken voor de user. Degene die in deze documentatie staan, werken wel met de bijgeleverde bridge.py bestand. Voor je die bridge bestand kan gebruiken zul je ook volgende commando's moeten invoeren.

```
sudo pip3 install Paho-mqtt
sudo pip3 install influxdb
```

Je kan nu ook een bash script maken zodat je automatisch de bridge kan activeren naar de influxdb bij het opstarten van de Raspberry Pi.

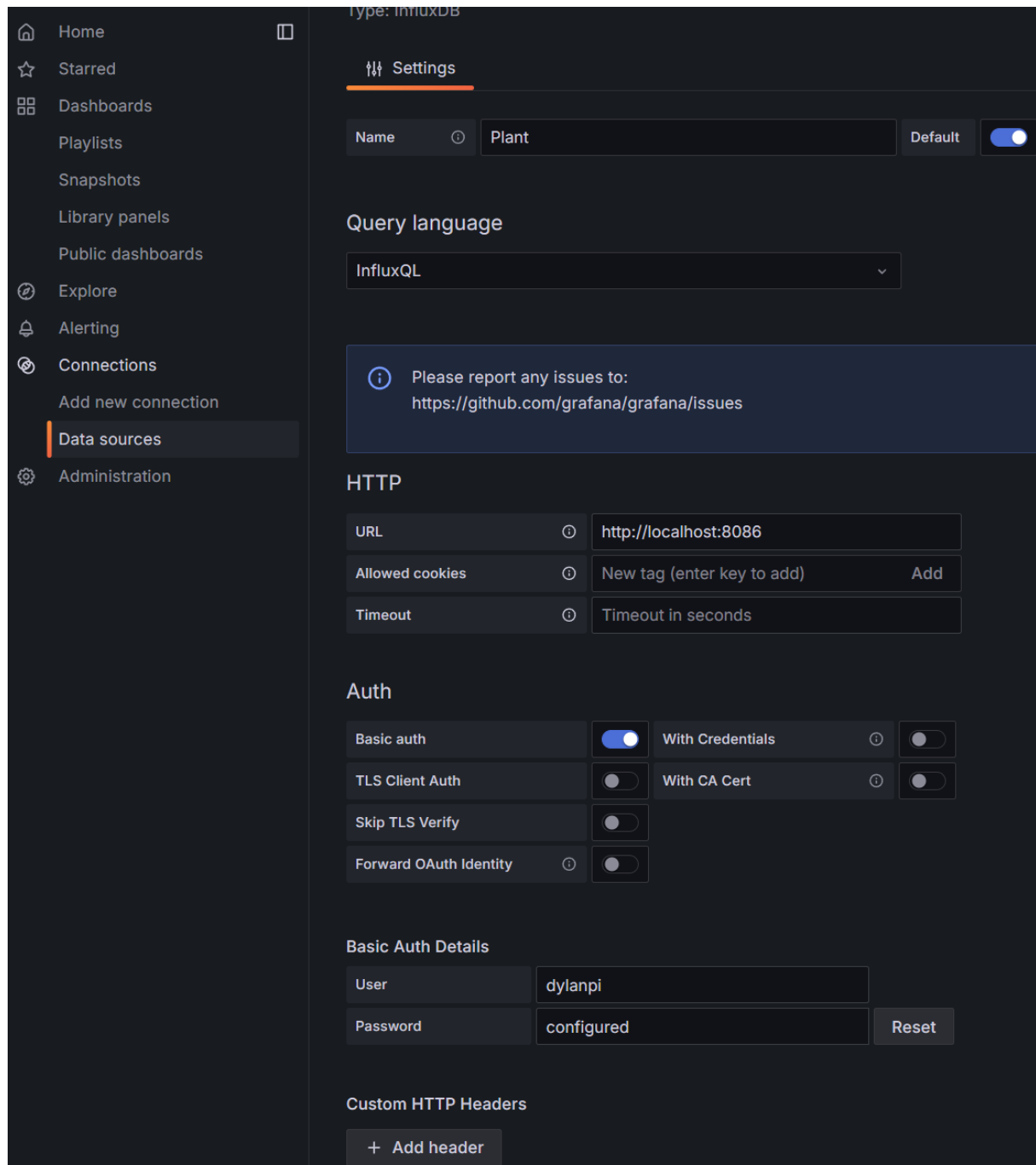
5. Grafana

Met Grafana kunnen we onze data live weergeven op de browser en kunnen we ook een historiek zien van wat er gebeurd is. Voor onderzoekers kan dit eventueel ook handig zijn om nieuwe omgevingen te testen op bepaalde planten. Om het in te stellen gebruiken we volgende commando's

```
sudo dpkg -i grafana_6.2.2_armhf.deb
```

```
sudo apt-get update
sudo apt-get install grafana
sudo systemctl grafana-server start
```

Nu kunnen we Grafana openen in de browser en inloggen met admin/admin. Hierna kan je zelf de visualisaties instellen zoals je wil. Voor je dit kan doen met je wel een bron geven van data. Volgende screenshot zal tonen hoe dit moet.




Basic Auth Details

User	dylanpi	
Password	configured	Reset

Custom HTTP Headers




+ Add header

InfluxDB Details

 Database Access

Setting the database for this datasource does not deny access to other databases. The or `SELECT * FROM "_internal".."database" LIMIT 10`

To support data isolation and security, make sure appropriate permissions are configur

Database	Plant	
User	dylanpi	
Password	configured	Reset
HTTP Method	 GET	
Min time interval	 10s	
Max series	 1000	

Delete

Save & test

6. Code

Ik kan momenteel alleen een pdf-bestand uploaden dus post ik hierin ook de code van het project.

```
// Total project
#include <WiFi.h>
#include <PubSubClient.h>
```

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
#include "DHT.h"
#include <OneWire.h>
#include <DallasTemperature.h>

// WiFi credentials
const char* ssid = "telenet-FC959";
const char* password = "wb6u7xsW75ap";

const char* temperatureground_topic = "home/livingroom/temperature_ground";
const char* temperatureair_topic = "home/livingroom/temperatureair";
const char* humidityair_topic = "home/livingroom/humidityair";
const char* humidityground_topic = "home/livingroom/humidityground";
const char* light_topic = "home/livingroom/light";
const char* waterlevel_topic = "home/livingroom/waterlevel";
const char* water_percent_topic = "home/livingroom/waterpercent";

const char* mqtt_server = "192.168.0.139";
const int mqtt_port = 1883;
const char* mqtt_user = "dylanpi";           // If using authentication
const char* mqtt_password = "dylanpi";       // If using authentication
const char* clientID = "client_livingroom";  // MQTT client ID

WiFiClient espClient;
PubSubClient client(espClient);

// LCD settings
#define LCD_ADDR 0x27 // Change if needed
#define LCD_COLUMNS 20
#define LCD_ROWS 4

// RFID settings
#define SS_PIN 5
#define RST_PIN 4 // Changed from 22 to 4 to avoid conflict with I2C SCL

#define DHT11PIN 25
// ground temp sensor
#define ONE_WIRE_BUS 13
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
float groundtemp;
```

Dit is de koptekst in stijl 'Koptekst'

```
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522
instance
LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLUMNS, LCD_ROWS); // Create LCD
instance
DHT dht(DHT11PIN, DHT11);

float AirHum;
float AirTem;
float WaterPercent;
int LCDState = 0;

#define GroundHumPin 34
int GroundHumValue;

#define LightPin 35
int LightValue;

#define trigPin 32
#define echoPin 33
float WaterLevel;

#define FanPin 14
#define WaterPump 12
#define RGBLight 27

int TargetGroundTemp;
int TargetAirTemp;
int TargetGroundHum;
int TargetAirHum;
int TargetLight;
int TargetWater;
long tagValue;
//////////////////////////////////// test waarden
int value1 = 10;
int value2 = 10;
int value3 = 10;
int value4 = 10;
int value5 = 10;
int value6 = 10;
////////////////////////////////////

void setup() {
  Serial.begin(115200);
  // Initialize I2C LCD
  lcd.init();
  lcd.backlight();
}
```

Dit is de voettekst in stijl 'Voettekst'

```
lcd.print("Scan an RFID tag");

setup_wifi();
client.setServer(mqtt_server, mqtt_port);

// Initialize SPI and RFID reader
SPI.begin();
mfrc522.PCD_Init();

/* Start the DHT11 Sensor */
dht.begin();
sensors.begin();
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(FanPin, OUTPUT);
pinMode(WaterPump, OUTPUT);
pinMode(RGBLight, OUTPUT);

Serial.println("Ready to scan an RFID tag...");
}

void loop() {
  client.loop();

  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed with state ");
      Serial.print(client.state());
      delay(5000);
    }
  }
}

// Continuously check for RFID tags
if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
  // Show UID on Serial Monitor and LCD
  Serial.print("Card UID:");
  String content = "";
  //long tagValue = 0;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
}
```

```

        tagValue = tagValue * 256 + mfrc522.uid.uidByte[i];
    }
    Serial.println();
    Serial.print("Tag value in decimal: ");
    Serial.println(tagValue);

    // Halt briefly to avoid immediate re-reading of the same tag
    delay(300);
}
////////////////////////////////////
String payload = String("sensors value1=") + String(value1) + ",value2=" +
String(value2) + ",value3=" + String(value3) + ",value4=" + String(value4) +
",value5=" + String(value5) + ",value6=" + String(value6);

client.publish("esp32/data", (char*)payload.c_str());

////////////////////////////////////
// Perform other tasks here if necessary
AirTempHumSensor();

GroundTemp();

GroundMoisture();

LightSens();

WaterLevelSens();

SendDataToMqtt();

LCDDisplay();

// Small delay to prevent excessive CPU usage
delay(500);
}
void LCDDisplay(){
    if (LCDState == 0){
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print(AirHum);
        lcd.setCursor(4,0);
        lcd.print("Per");
        lcd.setCursor(0,1);
        lcd.print(AirTem);
        lcd.setCursor(4,1);
    }
}

```



```
    lcd.print("Deg A");
    lcd.setCursor(0,2);
    lcd.print(groundtemp);
    lcd.setCursor(4,2);
    lcd.print("Deg G");
    lcd.setCursor(0,3);
    lcd.print(GroundHumValue);
    lcd.setCursor(4,3);
    lcd.print("Hum Per G");
    LCDState = 1;
}
else if (LCDState ==1){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(LightValue);
    lcd.setCursor(4,0);
    lcd.print("Per");
    lcd.setCursor(0,1);
    lcd.print(WaterLevel);
    lcd.setCursor(4,1);
    lcd.print("Cm");
    lcd.setCursor(0,2);
    lcd.print(WaterPercent);
    lcd.setCursor(4,2);
    lcd.print("Per");
    LCDState = 0;
}
}

void IDCheck(){
    // Give each plant dedicated target values it want's to reach
    if (tagValue == 1666537181){
        //Basil plant
        TargetGroundTemp = 26;
        TargetAirTemp = 26;
        TargetGroundHum = 80;
        TargetAirHum = 75;
        TargetLight = 80;
    }
    else if (tagValue == -479294755){
        // CherryTomato
        TargetGroundTemp = 29;
        TargetAirTemp = 29;
        TargetGroundHum = 10;
        TargetAirHum = 70;
        TargetLight = 80;
    }
}
```

```
else if (tagValue == -1016820723)
// Cilantro
    TargetGroundTemp = 20;
    TargetAirTemp = 20;
    TargetGroundHum = 30;
    TargetAirHum = 60;
    TargetLight = 80;
}

void AirTempHumSensor() {
    AirHum = dht.readHumidity();
    AirTem = dht.readTemperature();
    Serial.print("Temperature: ");
    Serial.print(AirTem);
    Serial.print("°C ");
    Serial.print("Humidity: ");
    Serial.println(AirHum);
    if (AirHum > TargetAirHum){
        digitalWrite(FanPin, HIGH);
    }
    else {
        digitalWrite(FanPin, LOW);
    }
}

void GroundTemp() {
    // call sensors.requestTemperatures() to issue a global temperature
    // request to all devices on the bus
    Serial.print("Requesting temperatures...");
    sensors.requestTemperatures(); // Send the command to get temperatures
    Serial.println("DONE");
    // After we got the temperatures, we can print them here.
    // We use the function ByIndex, and as an example get the temperature from
    the first sensor only.
    groundtemp = sensors.getTempCByIndex(0);

    // Check if reading was successful
    if (groundtemp != DEVICE_DISCONNECTED_C) {
        Serial.print("Temperature for the device 1 (index 0) is: ");
        Serial.println(groundtemp);
    } else {
        Serial.println("Error: Could not read temperature data");
    }
    delay(10);
}

void GroundMoisture() {
```

```
GroundHumValue = analogRead(GroundHumPin);
GroundHumValue = map(GroundHumValue, 0, 4095, 100, 0);
Serial.print("Mositure : ");
Serial.print(GroundHumValue);
Serial.println("%");
delay(10);

if (GroundHumValue < TargetGroundHum){
    digitalWrite(WaterPump, HIGH);
    delay(1000);
    digitalWrite(WaterPump, LOW);
}
}

void LightSens() {
    LightValue = analogRead(LightPin);
    LightValue = map(LightValue, 0, 4095, 0, 100);
    Serial.println(LightValue);
    delay(10);
    if (LightValue < TargetLight){
        digitalWrite(RGBLight, HIGH);
    }
    else {
        digitalWrite(RGBLight, LOW);
    }
}

void WaterLevelSens() {
    long duration, distance;
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);

    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = (duration / 2) / 29.1;
    WaterLevel = (float)distance;
    Serial.print(distance);
    Serial.println(" cm");
    WaterPercent = map(distance, 0,8, 100,0);
    Serial.println(WaterPercent);
}
```

```
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

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

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
            Serial.println("connected");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}

void SendDataToMqtt() {
    // Stuur data naar de Raspberry Pi

    String LightMqtt = String((float)LightValue);
    String GroundTempMqtt = String((float)groundtemp);
    String AirtempMqtt = String((float)AirTem);
    String AirHumMqtt = String((float)AirHum);
    String GroundHumMqtt = String((float)GroundHumValue);
    String WaterLevelMqtt = String((float)WaterLevel);
    String WaterPercentMqtt = String((float)WaterPercent);

    client.publish(light_topic, LightMqtt.c_str());
}
```

```
delay(1000);
client.publish(temperatureground_topic, GroundTempMqtt.c_str());
delay(1000);
client.publish(temperatureair_topic, AirtempMqtt.c_str());
delay(1000);
client.publish(humidityair_topic, AirHumMqtt.c_str());
delay(1000);
client.publish(humidityground_topic, GroundHumMqtt.c_str());
delay(1000);
client.publish(waterlevel_topic, WaterLevelMqtt.c_str());
delay(1000);
client.disconnect(); // disconnect from the MQTT broker
}
```