THOMAS MORE

Documentatie
Eindproject Plantenbak

Senne Van Dingenen

2023 - 2024

Inhoud

1. Doel

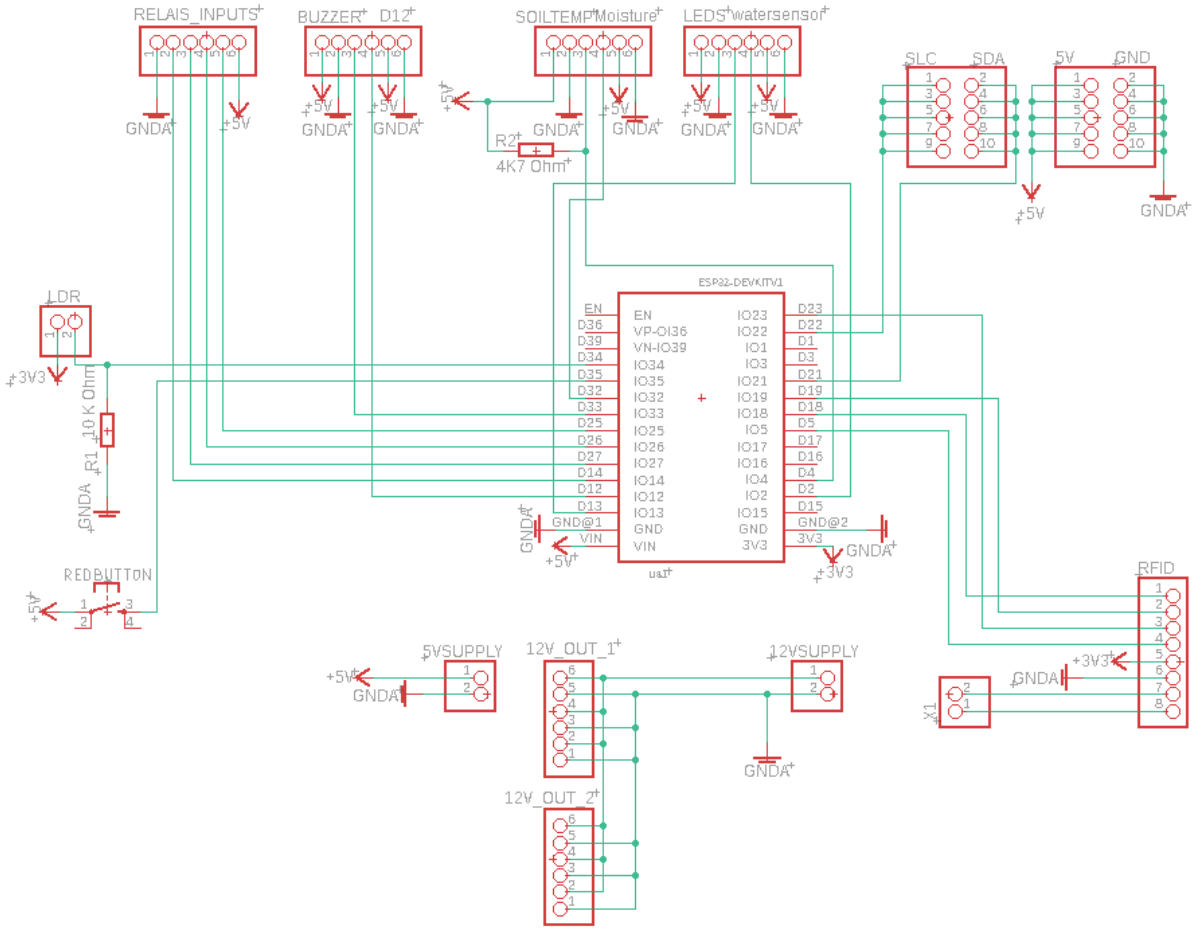
Dit project heeft als doel een geautomatiseerde plantenbak te maken, zodat je met een RFID tag kan kiezen welke plant dat je wilt, en dat dan alles automatisch geregeld wordt zodat je zelf alleen het plantje moet zaaien en de tag scannen!

2. Hardware

- ESP32 Devkit v1
  - Om de code uit te voeren
- Water sensor
  - Om te meten of er nog water in het reservoir zit
- PN532 RFID reader
  - Om te bepalen welk plantje we groot brengen
- LDR
  - Voor de led strip aan te doen en
- (Capacitive) soil moisture sensor
  - Om de grondvochtigheid te meten
- DS18B20 soil temperature sensor
  - Om de grond-temperatuur te meten
- LCD 20,04
  - To show all the information gathered on the display
- WS2812B Led strip
  - Voor meer licht te hebben als het nodig is
- BME280
  - Om de lucht temperatuur en vochtigheid te meten
- Water pump
  - Om de planten automatisch water te geven
- 12V DC Fan
  - Om hopelijk de temperatuur te doen dalen
- 4-channel (5v) relay module
  - Om de 12 apparatuur te schakelen
- Power supplies (12v)
  - Voor 5 en 12 V binnen te krijgen (adapter)
- Wires
  - Om alles te verbinden
- Buzzer
  - Zodat je het hoort als je je tag hebt gescand
- Relais
  - Gebruikt voor 12V door te schakelen met een arduino output
- 12V-5V adapter
  - Zodat je maar 1 stekker moet insteken voor 12 en 5 V te krijgen.

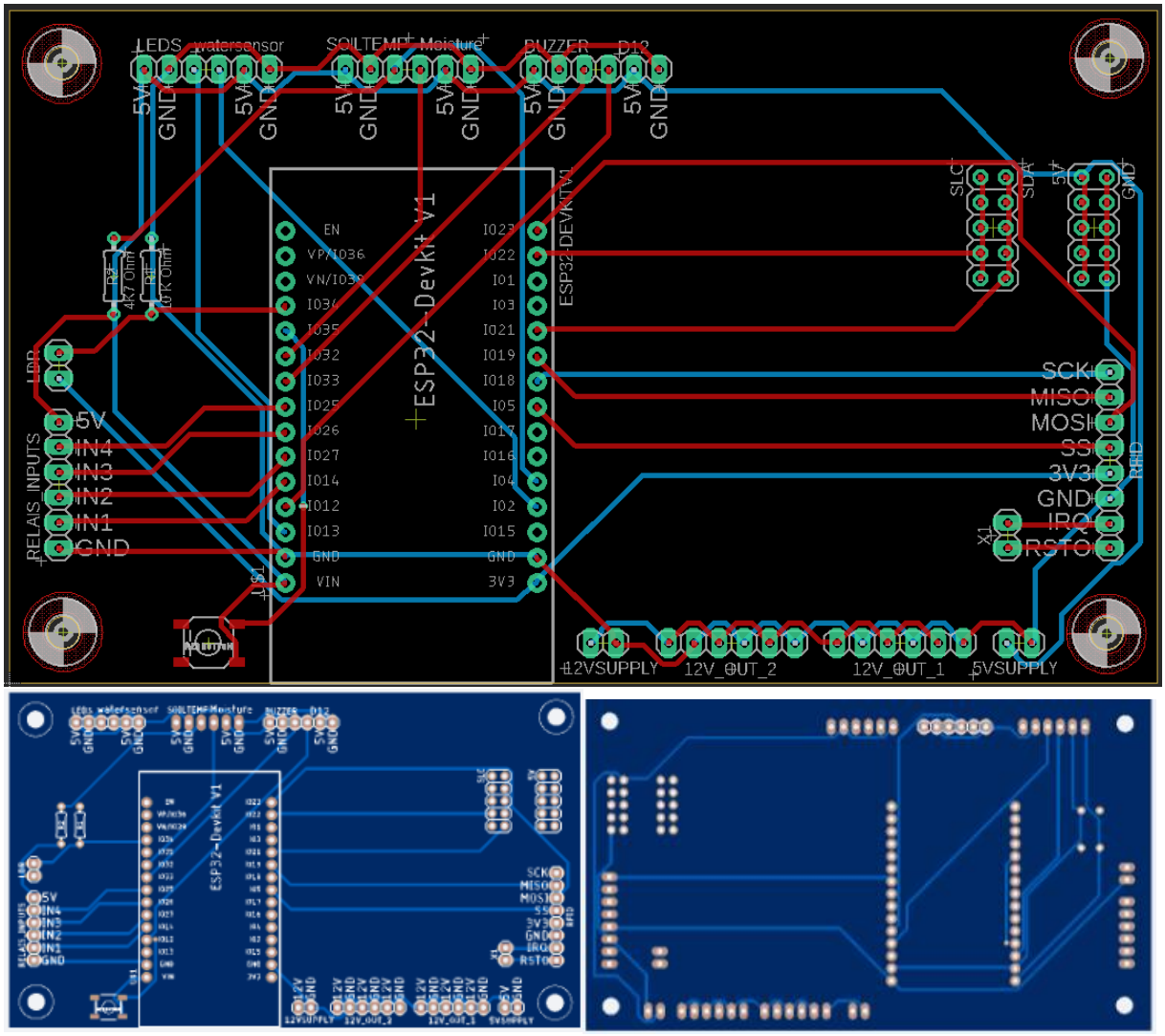
3. Schema

De LCD en BME280 sensor werken met IC2, ik kan er 5 van deze aansluiten.
Er is plek voorzien voor de Buzzer, led-strip, water sensor, bodem temperatuur sensor, grondvochtigheidssensor, de LDR, RFID, drukknop en een extra (D12) voor als je nog iets vergeten bent en dit later wilt bij plaatsen.



4. PCB

Voor mijn PCB heb ik vooral gaatjes waarop ik klemmen of draden ga aan solderen, met die gaatjes heb je altijd nog een beetje speling om iets mee te doen. Er zijn een aantal kleine aanpassingen gemaakt waardoor de drukknop momenteel niet meer in gebruik is.



4.1. Veranderingen als ik het nog eens zou doen

Als ik mijn PCB zou kunnen her-maken zou ik de SCL/SDA en de 5V/GND dezelfde pinnen geven als al de rest. Omdat deze zeer moeilijk waren om te solderen.
Ook zou ik extra gaatjes nemen voor de zekerheid bij mijn ESP32, zo kan er altijd iets worden op aangesloten als de nood uit breekt, en 100% andere “board connector pins” kopen, want diegene dat ik heb smelten de plastic rond hun als ik die 5 seconden warm maak.
Ten slotte de pinnetjes van de RFID reader omdraaien, omdat ik deze omgekeerd heb laten maken…

5. Installaties op Raspberry Pi

5.1. Grafana

Eerst downloaden en instaleren we de repositorysleutel met
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

Nu voegen we hun server-URL toe aan de APT-bronnenlijst met
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

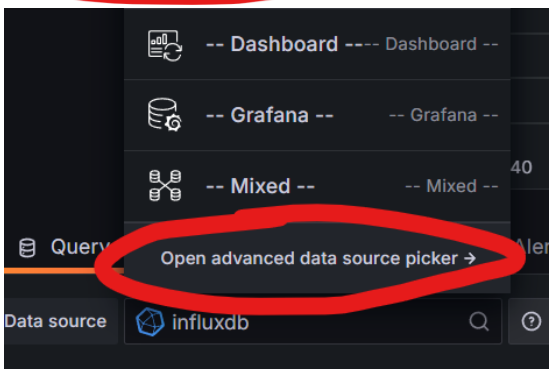
Dan gaan we Grafana zelf installeren met
sudo apt update
en
sudo apt install grafana

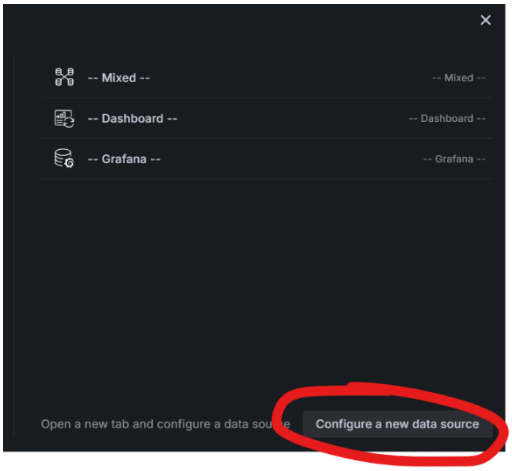
Nu starten we grafana
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable grafana-server
sudo /bin/systemctl start grafana-server

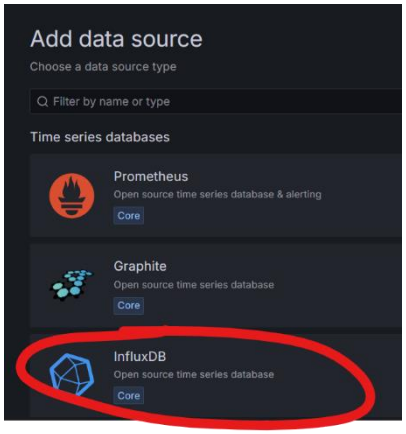
je kan de status bekijken met
sudo /bin/systemctl status grafana-server

Nu kan je op het internet naar deze site gaan
http://<RASPBERRYPI_IP>:3000
Hier kan je inloggen **met username admin wachtwoord admin**. Als je dat hebt gedaan kan je zelf je eigen wachtwoord en username instellen, en daarna je dashboard maken.

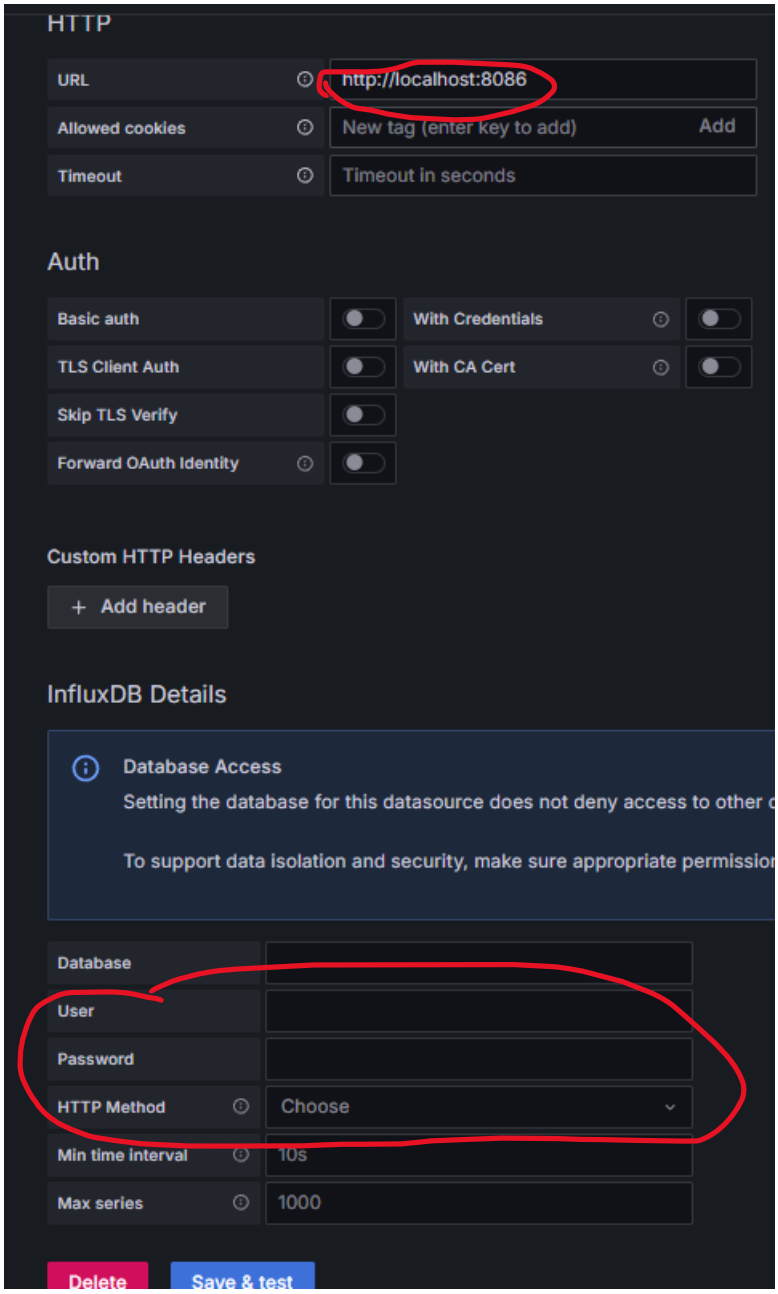
Je moet ook nog de soort database kiezen, dit doe je door naar je paneel te gaan, deze te bewerken en dan druk je hier op: **(Dit kan pas na dat je influxDB hebt geïnstalleerd.)**



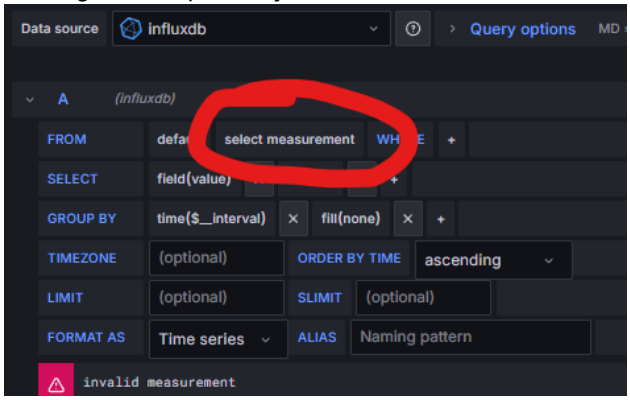




Op het einde krijg je dit scherm, hier moet je “http://localhost:8086” invullen en daarna je database naam, gebruikers naam en passwoord ingeven. Deze informatie is dezelfde informatie als toen je influxDB hebt ingesteld.
Als je die 4 hebt ingesteld heb je je database gelinkt get grafana.



De volgende stap is om je “measurement” te selecteren.



Druk op select measurement en nu kies je welke je wilt. Nu ben je helemaal klaar!

5.2. MQTT

Als je dit nog niet hebt gedaan doe je
sudo apt update && sudo apt upgrade
en druk je op Y.

Installeer nu de Mosquitto Broker met
sudo apt install -y mosquitto mosquitto-clients

Zorg er nu voor dat de mosqitto service automatisch start bij opstart.
sudo systemctl enable mosquitto.service

Test de installatie
mosquitto -v

stel nu een wachtwoord en username in
sudo mosquitto_passwd -c /etc/mosquitto/passwd YOUR_USERNAME
vul nu je password in.

Verander de configuration file
sudo nano /etc/mosquitto/mosquitto.conf

zet dit **VAN BOVEN** aan het bestand
per_listener_settings true

en zet deze **VAN ONDER** aan het bestand
allow_anonymous
false listener 1883
password_file /etc/mosquitto/passwd

Herstart het system zodat je veranderingen in werking komen
sudo systemctl restart mosquito

Kijk nu of het werkt
sudo systemctl status mosquitto

5.3. InfluxDB

Dit zou je normaal gezien al gedaan moeten hebben maar zo niet gebruik je deze om je systeem up to date te houden
sudo apt update
sudo apt upgrade -y

Installeer Influxdb
wget -qO- 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

Update apt met de nieuwe repositories en installeer.
sudo apt update && sudo apt install -y influxdb

Start en run de influxdb service
sudo systemctl unmask influxdb.service
sudo systemctl start influxdb

sudo systemctl enable influxdb.service

open influx nu
influx

Maak een database aan
Create database DATABASE_NAAM

Gebruik deze database
Use DATABASE_NAAM

Maak een admin gebruiker aan met een wachtwoord
Create user USER_NAAM with password WACHTWOORD with all privileges

Geef jezelf nu rechten op je database
Grant all privileges on DATABASE_NAAM to USER_NAAM

Je kan nu uit influx gaan met
exit

Om zeker te zijn dat je de laatste versie hebt kan je deze commando's runnen
pip install influxdb
pip3 install influxdb
pip install influxdb-client
pip3 install influxdb-client

6. Grafana Server/Data Logging

Wat er eigenlijk gebeurt om data weer te geven in Grafana gaat als volgt:
De ESP32 meet alle statussen van de sensoren waarop die ze doorstuurt naar de Raspberry pi met gebruik van MQTT. Wanneer de Raspberry pi deze aankrijgt, zal die ze in een InfluxDB database steken. Grafana kan in deze database gaan kijken en zo de data grafisch weergeven.

6.1. Demo werking MQTT, influx en Grafana

Hier zie je Hoe het werkt:
Dit stuurt de gegevens via MQTT naar de raspberry pi in een topic.

```
if (SoilTemperatureC == -999)
{
}
else
{
  String SoilTemperature = String(SoilTemperatureC);
  client.publish("home/plantenbak/Soiltemperature", SoilTemperature.c_str());
}

// Publish sensor readings to MQTT topics
client.publish("home/plantenbak/temperature", GlobalTemperatureString.c_str());
client.publish("home/plantenbak/humidity", GlobalHumidityString.c_str());
client.publish("home/plantenbak/soilmoisture", localSoilMoisture.c_str());
client.publish("home/plantenbak/led", Ledstrip.c_str());
client.publish("home/plantenbak/waterPump", waterPUMP.c_str());
client.publish("home/plantenbak/watercontainer", watercontainer.c_str());
client.publish("home/plantenbak/fan", fan.c_str());
```

In de raspberry pi worden ze ontvangen, en in de influxDB database gezet.

```
senne@senne3030: ~/Documents
Soil temperature data written to InfluxDB: 22.56
Temperature data written to InfluxDB: 21.36
Humidity data written to InfluxDB: 59.6
Soil moisture data written to InfluxDB: 75.0
led data written to InfluxDB: 0
pump data written to InfluxDB: 0
water container data written to InfluxDB: 1
fan data written to InfluxDB: 0
Soil temperature data written to InfluxDB: 22.56
Temperature data written to InfluxDB: 21.37
Humidity data written to InfluxDB: 59.67
Soil moisture data written to InfluxDB: 75.0
led data written to InfluxDB: 0
pump data written to InfluxDB: 0
water container data written to InfluxDB: 1
fan data written to InfluxDB: 0
```

(deel code)

```
elif msg.topic == MQTT_TOPIC_SOIL_TEMPERATURE:
    soil_temperature = float(msg.payload.decode())
    json_body = [
        {
            "measurement": "soil_temperature",
            "fields": {
                "value": soil_temperature
            }
        }
    ]
    influxdb_client.write_points(json_body)
    print(f"Soil temperature data written to InfluxDB: {soil_temperature}")

elif msg.topic == MQTT_TOPIC_SOIL_MOISTURE:
    soil_moisture = float(msg.payload.decode())
    json_body = [
        {
            "measurement": "soil_moisture",
            "fields": {
                "value": soil_moisture
            }
        }
    ]
    influxdb_client.write_points(json_body)
    print(f"Soil moisture data written to InfluxDB: {soil_moisture}")
```

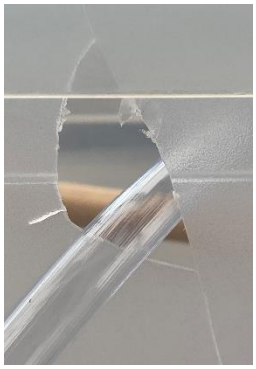
Dan kan grafana deze uit de database halen en grafisch weergeven.

7. Plantenbak maken

Om de plantenbak te maken heb je bak nodig waar je de plantjes in kan kweken, dit kan een plastieken doos zijn, of iets anders dat geen water door laat (Aan de bodem). Voor mijn plantenbak gebruik ik een grote plastieken doos en als water reservoir gebruik ik een kleinere plastieken doos.

Eerst gaan we een gat boren voor de ventilator, Hiervoor heb ik een (houtboor) gatenzaag van ongeveer de diameter van de ventilator gebruikt, en daarna de ventilator daar aan geplakt met een warm lijmpistool.

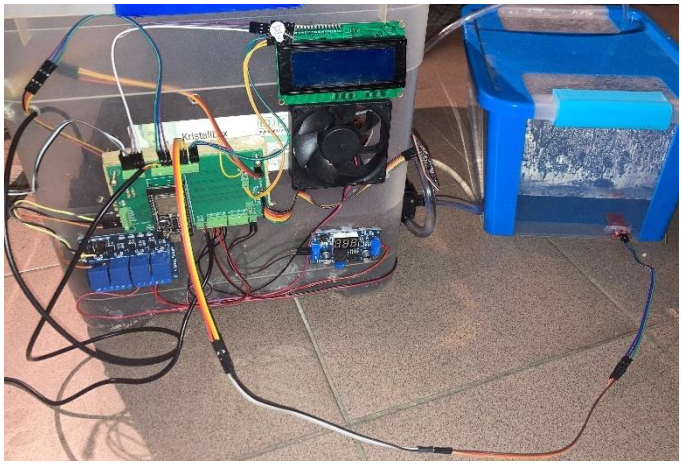
Daarna heb ik gaten geboord voor mijn waterbuisjes (dezelfde diameter als de buitenkant van de buis) Deze gaatjes moeten **ergens boven** zijn **als** je **de pomp NIET in het water** hebt (zoals ik). Dit moet omdat de pomp water nodig heeft om goed te werken. **Zorg dat je rustig boort**, veel toeren draaien en weinig druk zetten of je hebt hetzelfde voor als ik:



En ten slotte maak je een aantal gaatjes voor je draden.

De volgende stap is om alles erin te installeren.
Je plakt de PCB en de sensoren/actuatoren binnen in of buiten de doos. **Binnen** in de plantenbak: BME280, Soil Moisture sensor, Soil temperature sensor, ledstrip. De watersensor plaats je in het water reservoir, plaats deze een beetje boven de grond zodat die een melding geeft voor dat die 100% leeg is.
Buiten de plantenbak: LDR, RFID reader, LCD scherm, relais, buzzer, omvormer, (Waterpomp mag in het water reservoir of buiten de plantenbak).

Ik heb ook de grond er aan toegevoegd. Dit is het prototype. Zorg dat de adapter op 5.1 V staat. (Dit kan je veranderen door aan de schroef tegen de klok in te draaien.)
Zorg ervoor dat je de watersensor een klein beetje schuin zet zodat water er vanaf loopt wanneer deze leeg loopt.



8. Onderhoud

Het enige wat je moet doen nadat je de plantenbak zelf hebt gemaakt is de plantjes planten en water in de bak doen als die (zo goed als) leeg is. Je kan dit aflezen van de watersensor die je in de bak hebt geplaatst.

9. Technische Uitdagingen en Oplossingen

Om ervoor te zorgen dat de RFID reader kan lezen zonder de hele code te laten wachten wordt er gebruik gemaakt van een task manager, die er voor zorgt dat er meerdere codes tegelijkertijd runnen.
Wanneer mijn LED strip aan was ging mijn buzzer de hele tijd af, om dit te fixen heb ik simpelweg een andere buzzer genomen en het probleem was opgelost.
Om ervoor te zorgen dat MQTT werkt heb ik een ingang met een uitgang verwisseld omdat die ingang (analoog) anders niet zou werken.
Doordat het voltage hoger word als de ventilator opstaat, stijgt de waarde van de soil moisture sensor, dus heb ik de code hier aan aangepast.

10. Codes

10.1. Python

#Simpel gezegd wat deze code doet is het opslagen van gegevens in influxdb die worden aangekregen door MQTT.

```
import paho.mqtt.client as mqtt
from influxdb import InfluxDBClient

# MQTT settings
MQTT_BROKER = "senne3030.local"
MQTT_PORT = 1883
MQTT_TOPIC_TEMPERATURE = "home/plantenbak/temperature"
MQTT_TOPIC_HUMIDITY = "home/plantenbak/humidity"
MQTT_TOPIC_SOIL_TEMPERATURE = "home/plantenbak/Soiltemperature"
MQTT_TOPIC_SOIL_MOISTURE = "home/plantenbak/soilmoisture"
MQTT_TOPIC_LED = "home/plantenbak/led"
MQTT_TOPIC_WATER_PUMP = "home/plantenbak/waterPump"
MQTT_TOPIC_WATER_SENSOR = "home/plantenbak/watercontainer"
MQTT_TOPIC_FAN = "home/plantenbak/fan"
MQTT_USERNAME = "senne"
MQTT_PASSWORD = "senne123"

# InfluxDB settings
INFLUXDB_ADDRESS = "senne3030.local"
INFLUXDB_PORT = 8086
INFLUXDB_DATABASE = "DATA"
INFLUXDB_USER = "senne"
```

```
INFLUXDB_PASSWORD = "senne123"

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker!") #If there are no errors subscribe to those topics
        client.subscribe(MQTT_TOPIC_TEMPERATURE)
        client.subscribe(MQTT_TOPIC_HUMIDITY)
        client.subscribe(MQTT_TOPIC_SOIL_TEMPERATURE)
        client.subscribe(MQTT_TOPIC_SOIL_MOISTURE)
        client.subscribe(MQTT_TOPIC_LED)
        client.subscribe(MQTT_TOPIC_WATER_PUMP)
        client.subscribe(MQTT_TOPIC_WATER_SENSOR)
        client.subscribe(MQTT_TOPIC_FAN)
    else:
        print(f"Failed to connect, return code {rc}")

def on_message(client, userdata, msg): #If it has a certain topic save it to that certain measurement. This is
true for all of those values.
    if msg.topic == MQTT_TOPIC_TEMPERATURE:
        temperature = float(msg.payload.decode())
        json_body = [
            {
                "measurement": "temperature",
                "fields": {
                    "value": temperature
                }
            }
        ]
        influxdb_client.write_points(json_body)
        print(f"Temperature data written to InfluxDB: {temperature}")
    elif msg.topic == MQTT_TOPIC_HUMIDITY:
        humidity = float(msg.payload.decode())
        json_body = [
            {
                "measurement": "humidity",
                "fields": {
                    "value": humidity
                }
            }
        ]
        influxdb_client.write_points(json_body)
        print(f"Humidity data written to InfluxDB: {humidity}")
    elif msg.topic == MQTT_TOPIC_SOIL_TEMPERATURE:
        soil_temperature = float(msg.payload.decode())
        json_body = [
            {
                "measurement": "soil_temperature",
                "fields": {
                    "value": soil_temperature
                }
            }
        ]
        influxdb_client.write_points(json_body)
        print(f"Soil temperature data written to InfluxDB: {soil_temperature}")

    elif msg.topic == MQTT_TOPIC_SOIL_MOISTURE:
```

```
    soil_moisture = float(msg.payload.decode())
    json_body = [
        {
            "measurement": "soil_moisture",
            "fields": {
                "value": soil_moisture
            }
        }
    ]
    influxdb_client.write_points(json_body)
    print(f"Soil moisture data written to InfluxDB: {soil_moisture}")
elif msg.topic == MQTT_TOPIC_LED:
    led_status = int(msg.payload.decode())
    json_body = [
        {
            "measurement": "led_status",
            "fields": {
                "value": led_status
            }
        }
    ]
    influxdb_client.write_points(json_body)
    print(f"led data written to InfluxDB: {led_status}")
elif msg.topic == MQTT_TOPIC_WATER_PUMP:
        water_pump = int(msg.payload.decode())
        json_body = [
            {
                "measurement": "water_pump",
                "fields": {
                    "value": water_pump
                }
            }
        ]
        influxdb_client.write_points(json_body)
        print(f"pump data written to InfluxDB: {water_pump}")
elif msg.topic == MQTT_TOPIC_WATER_SENSOR:
    water_container = int(msg.payload.decode())
    json_body = [
        {
            "measurement": "water_container",
            "fields": {
                "value": water_container
            }
        }
    ]
    influxdb_client.write_points(json_body)
    print(f"water container data written to InfluxDB: {water_container}")
elif msg.topic == MQTT_TOPIC_FAN:
    fan = int(msg.payload.decode())
    json_body = [
        {
            "measurement": "fan",
            "fields": {
                "value": fan
            }
        }
```

```
        ]
        influxdb_client.write_points(json_body)
        print(f"fan data written to InfluxDB: {fan}")


if __name__ == '__main__': # InfluxDB connection
    influxdb_client = InfluxDBClient(
        host=INFLUXDB_ADDRESS,
        port=INFLUXDB_PORT,
        username=INFLUXDB_USER,
        password=INFLUXDB_PASSWORD,
        database=INFLUXDB_DATABASE
    )

    influxdb_client.create_database(INFLUXDB_DATABASE)

    mqtt_client = mqtt.Client()
    mqtt_client.username_pw_set(MQTT_USERNAME, MQTT_PASSWORD)
    mqtt_client.on_connect = on_connect
    mqtt_client.on_message = on_message

    mqtt_client.connect(MQTT_BROKER, MQTT_PORT, 60)

    mqtt_client.loop_forever()
```

10.1.1. Automatisch opstarten - systemctl

We gaan met systemctl een automatisch startend script maken zodat het script op start wanneer we de Raspberry pi insteken.
Eerst maak je een bestand aan met dit command
sudo nano /lib/systemd/system/<SCRIPT NAAM>.service

Daar zet je deze tekst in:
[Unit]
Description=automatated start for plantenbak
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python3 /home/senne/Documents/<DE NAAM VAN JE SCRIPT>.py #(Dit moet je eigen path zijn!!)

[Install]
WantedBy=multi-user.target

Geef jezelf rechten
sudo chmod 644 /lib/systemd/system/<SCRIPT NAAM>.service

her-laad de daemon
sudo systemctl daemon-reload

Zet de service aan
sudo systemctl enable <SCRIPT NAAM>.service

(extra)
Je kan de status hier van checken door dit te doen

sudo systemctl status <SCRIPT NAAM>.service

Je kan de service ook zelf starten met
sudo systemctl start <SCRIPT NAAM>.service

10.2. C++

```
// Eindporject Plantenbak -- Made in Visual Studio Code -- Senne Van Dingenen
1IoT //

// On startup disconnect the GND on the relay module so it doesn't pump all
the water out for no reason.

#include <Arduino.h> // Include the needed libraries
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_PN532.h>
#include <LiquidCrystal_I2C.h>
#include <FastLED.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_BME280.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <WiFi.h>
#include <PubSubClient.h>

// WiFi and MQTT Server Configuration
const char *ssid = "WIFI_SSID";          // Change to your WiFi SSID
const char *password = "WIFI_PASSWORD";       // Change to your WiFi password
const char *mqtt_server = "senne3030.local"; // Change to your MQTT broker
address
const char *mqtt_username = "MQTT_USERNAME";         // Change to your MQTT
username
const char *mqtt_password = "MQTT_PASSWORD";      // Change to your MQTT
password
const int mqtt_port = 1883;

WiFiClient espClient;           // WiFi client
PubSubClient client(espClient); // MQTT client

#define BME_SDA 21
#define BME_SCL 22
Adafruit_BME280 bme;

#define OneWireBus 4
OneWire oneWire(OneWireBus);
DallasTemperature sensors(&oneWire);
```

```
#define LED_PIN 13
#define NUM_LEDS 100
CRGB leds[NUM_LEDS];

#define SS_PIN 5   // Use any digital pin for SS (Slave Select)
#define SCK_PIN 18 // Use the hardware SPI pins for SCK, MISO, MOSI
#define MOSI_PIN 23
#define MISO_PIN 19
Adafruit_PN532 nfc(SCK_PIN, MISO_PIN, MOSI_PIN, SS_PIN);

#define LCD_ADDRESS 0x27
#define LCD_COLUMNS 20
#define LCD_ROWS 4
LiquidCrystal_I2C lcd(LCD_ADDRESS, LCD_COLUMNS, LCD_ROWS);

unsigned long previousMillis = 0;
const long WaitTime = 30000; // 30 seconds

const int LDR = 34;
const int MoistureSensor = 32;
const int WaterSensor = 33;
// const int RedButton = 35;
const int Buzzer = 2;
const int VentilatorK1 = 14;
const int WaterPumpK2 = 27;

float SoilMoisture = 0;
float SoilTemperatureC = 0;
float GlobalHumidity = 0;
float GlobalTemperature = 0;
int LCDSetting = 0;
int LASTLCDNUMBER = 5;
String WaterStatus = "";
String LedStripStatus = "";
String SunLightStatus = "";
String Access = "";
uint8_t CARD[4];
uint8_t UnknownCARD[4];
String cardString = "";
String Plant = "";
int VentilatorStatus = 0;
int WaterPumpStatus = 0;

int LedSetting = 0;
int VentilatorSetting = 0;
int WaterSetting = 0;
```

```
// RFID tags that are known by the system:
uint8_t authorizedUID1[] = {0x3, 0x45, 0xC5, 0x18};
uint8_t authorizedUID2[] = {0x3, 0xD4, 0xA, 0x19};
uint8_t authorizedUID3[] = {0xD3, 0xA8, 0xA5, 0x18};

// Function prototypes to make sure it knows those voids exist
void RFID_Reader();
void LCD_Print();
void LEDSTRIP();
void SoilMoistureSensor();
void SoilTemperature();
void WaterLevel();
void BME280Sensor();
void LCD_Setting();
void rfidReadTask(void *parameter);
void BuzzerSound1();
void BuzzerSound2();
void CheckTask(void *parameter);
void setup_wifi();
void reconnect();
void Ventilator();
void WaterPump();
void SendData();
void PlantSelect();

void setup_wifi() // Function to set up wifi
{
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  lcd.setCursor(0, 0);
  lcd.print("Connecting to       ");
  lcd.setCursor(0, 1);
  lcd.print(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());
```

```
  lcd.setCursor(0, 2);
  lcd.print("WiFi connected      ");
}

void reconnect() // Function to set up the MQTT connection
{
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Client", mqtt_username, mqtt_password))
    {
      Serial.println("connected");
    }
    else
    {
      lcd.setCursor(0, 0);
      lcd.print("Proces paused.      ");
      lcd.setCursor(0, 1);
      lcd.print("Attempting MQTT     ");
      lcd.setCursor(0, 2);
      lcd.print("connection...       ");
      lcd.setCursor(0, 3);
      lcd.print("Failed, State: " + String(client.state()) + "   ");

      VentilatorSetting = 0;
      WaterSetting = 0;

      WaterPump();
      Ventilator();

      Serial.print("failed, state: ");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() // Function to set everything up
{
  Serial.begin(9600);
  Serial.println("Starting..");

  ////////////// LCD ////////////
  lcd.init();
  lcd.backlight();
```

```
  ////////////////////////////////

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

  // pinMode(RedButton, INPUT_PULLUP);
  pinMode(Buzzer, OUTPUT);
  pinMode(VentilatorK1, OUTPUT);
  pinMode(WaterPumpK2, OUTPUT);

  //////// BME280 Sensor ////////
  Wire.begin(BME_SDA, BME_SCL);

  if (!bme.begin(0x76))
  {
    Serial.println("Could not find BME280 sensor!");
    lcd.setCursor(0, 0);
    lcd.print("Couldn't find BME280");
    while (1)
      ;
  }

  bme.setSampling(Adafruit_BME280::MODE_NORMAL,     // Operating mode
                  Adafruit_BME280::SAMPLING_X2,     // Temperature
oversampling
                  Adafruit_BME280::SAMPLING_X16,    // Pressure oversampling
                  Adafruit_BME280::SAMPLING_X16,    // Humidity oversampling
                  Adafruit_BME280::FILTER_X16,      // Filtering
                  Adafruit_BME280::STANDBY_MS_500); // Standby time
  ////////////////////////////////

  /////// Soil Temperature ///////
  sensors.begin();
  ////////////////////////////////

  ////////////// LEDS //////////////
  FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, NUM_LEDS);
  ////////////////////////////////

  ////////////// RFID //////////////
  nfc.begin();

  uint32_t versiondata = nfc.getFirmwareVersion();
  if (!versiondata)
  {
    Serial.println("Didn't find PN53x board");
    lcd.setCursor(0, 0);
```

```
    lcd.print("Couldn't find RFID  ");
    while (1)
      ;
  }

  Serial.print("Found chip PN5");
  Serial.println((versiondata >> 24) & 0xFF, HEX);
  Serial.print("Firmware ver. ");
  Serial.print((versiondata >> 16) & 0xFF, DEC);
  Serial.print('.');
  Serial.println((versiondata >> 8) & 0xFF, DEC);

  nfc.SAMConfig();
  Serial.println("RFID card reader ready");
  ////////////////////////////////////////////////

  ///////// rfidReadTask /////////
  xTaskCreate(
      rfidReadTask,     // Task function
      "RFID Read Task", // Task name
      4096,             // Stack size (bytes)
      NULL,             // Task parameter
      2,                // Priority (0 is lowest)
      NULL              // Task handle
  );
  ////////////////////////////////

  ////////// CheckTask //////////
  xTaskCreate(
      CheckTask,    // Task function
      "Check Task", // Task name
      4096,         // Stack size (bytes)
      NULL,         // Task parameter
      1,            // Priority (0 is lowest)
      NULL          // Task handle
  );
  ////////////////////////////////

  Serial.println("Project Plantenbak");
}

void loop() // Main loop
{
  if (!client.connected())
  {
    reconnect();
  }
```

```
  client.loop();

  Serial.println("------------------------");

  PlantSelect();

  LEDSTRIP();
  SoilMoistureSensor();
  SoilTemperature();
  WaterLevel();
  BME280Sensor();

  Ventilator();
  WaterPump();

  LCD_Print();
  SendData();

  delay(1000);
}

void RFID_Reader() // Function to read which RFID card was scanned
{
  uint8_t success;
  uint8_t uid[7] = {0, 0, 0, 0, 0, 0, 0}; // Buffer to store the returned UID
  uint8_t uidLength;                      // Length of the UID (4 or 7 bytes
depending on ISO14443A card type)

  // Wait for an RFID card to be present
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);

  if (success)
  {
    Serial.println("-----------------");
    Serial.println("Found an RFID card!");

    Serial.print("UID Length: ");
    Serial.print(uidLength, DEC);
    Serial.println(" bytes");
    Serial.print("UID Value: ");
    for (uint8_t i = 0; i < uidLength; i++)
    {
      Serial.print(" 0x");
      Serial.print(uid[i], HEX);
    }
    Serial.println("");
```

```
    if (memcmp(uid, authorizedUID1, uidLength) == 0)
    {
      BuzzerSound1();
      Access = "Access authorized";
      Plant = "Strawberries";
      memcpy(CARD, authorizedUID1, sizeof(authorizedUID1));
      Serial.println(Plant);
      Serial.println("Access authorized");
      Serial.println("-----------------");
    }
    else if (memcmp(uid, authorizedUID2, uidLength) == 0)
    {
      BuzzerSound1();
      Access = "Access authorized";
      Plant = "Lettuce";
      memcpy(CARD, authorizedUID2, sizeof(authorizedUID2));
      Serial.println(Plant);
      Serial.println("Access authorized");
      Serial.println("-----------------");
    }
    else if (memcmp(uid, authorizedUID3, uidLength) == 0)
    {
      BuzzerSound1();
      Access = "Access authorized";
      Plant = "Carrots";
      memcpy(CARD, authorizedUID3, sizeof(authorizedUID3));
      Serial.println(Plant);
      Serial.println("Access authorized");
      Serial.println("-----------------");
    }
    else
    {
      BuzzerSound2();
      memcpy(UnknownCARD, uid, sizeof(uid));
      memcpy(CARD, UnknownCARD, sizeof(UnknownCARD)); // I do this 2 times
because when I do this one time Access gets corrupted for some reason

      Access = "Access denied";
      Plant = "Unknown in system";
      Serial.println(Plant);
      Serial.println("Access denied");
      Serial.println("-----------------");
    }

    cardString = "";

    for (size_t i = 0; i < sizeof(CARD); i++)
```

```
    {
      if (CARD[i] < 0x10)
      {
        cardString += "0";
      }
      cardString += String(CARD[i], HEX);
      if (i < sizeof(CARD) - 1)
      {
        cardString += " ";
      }
    }

    delay(1000);
    return;
  }
}

void LCD_Print() // Function to Print all values on the LCD
{
  Serial.println("Printing on LCD");
  lcd.clear();

  if (LCDSetting != 0)
  {
    lcd.setCursor(19, 0);
    lcd.print(LCDSetting);
  }

  if (LCDSetting == 1)
  {
    lcd.setCursor(0, 0);
    lcd.print("Temp: ");
    lcd.setCursor(6, 0);
    lcd.print(GlobalTemperature);
    lcd.setCursor(12, 0);
    lcd.print("C");

    lcd.setCursor(0, 1);
    lcd.print("Humid: ");
    lcd.setCursor(7, 1);
    lcd.print(GlobalHumidity);
    lcd.setCursor(13, 1);
    lcd.print("%");

    lcd.setCursor(0, 2);
    lcd.print("Soil Temp: ");
    lcd.setCursor(11, 2);
```

```
    lcd.print(SoilTemperatureC);
    lcd.setCursor(17, 2);
    lcd.print("C");

    lcd.setCursor(0, 3);
    lcd.print("Soil Moist: ");
    lcd.setCursor(12, 3);
    lcd.print(SoilMoisture);
    lcd.setCursor(18, 3);
    lcd.print("%");
  }
  else if (LCDSetting == 2)
  {
    lcd.setCursor(0, 0);
    lcd.print("Water: ");
    lcd.setCursor(7, 0);
    lcd.print(WaterStatus);

    lcd.setCursor(0, 1);
    lcd.print("Pump: ");
    lcd.setCursor(6, 1);
    if (WaterPumpStatus == 1)
    {
      lcd.print("On");
    }
    else
    {
      lcd.print("Off");
    }

    lcd.setCursor(0, 2);
    lcd.print("Natural Light: ");
    lcd.setCursor(15, 2);
    lcd.print(SunLightStatus);

    lcd.setCursor(0, 3);
    lcd.print("Led strip: ");
    lcd.setCursor(11, 3);
    lcd.print(LedStripStatus);
  }
  else if (LCDSetting == 3)
  {
    lcd.setCursor(0, 0);
    lcd.print("Ventilator: ");
    lcd.setCursor(12, 0);
    if (VentilatorStatus == 1)
    {
```

```
      lcd.print("On");
    }
    else
    {
      lcd.print("Off");
    }

    lcd.setCursor(1, 3);
    lcd.print("Project Plantenbak");
  }
  else if (LCDSetting == 4)
  {
    lcd.setCursor(0, 0);
    lcd.print("Last RFID card: ");

    lcd.setCursor(0, 1);
    lcd.print(cardString);

    lcd.setCursor(0, 2);
    lcd.print(Plant);

    lcd.setCursor(0, 3);
    lcd.print(Access);
  }
  else if (LCDSetting == 0)
  {
    lcd.setCursor(0, 0);
    // lcd.print("Hold the red button");
    lcd.print("All systems working");
    lcd.setCursor(0, 1);
    lcd.print("Starting..");
    // lcd.print("to change the menu");
    lcd.setCursor(0, 3);
    lcd.print("Scan your RFID card");
  }
}

void LCD_Setting() // every 4 seconds change "page" (show other data)
{
  LCDSetting++;
  if (LCDSetting >= LASTLCDNUMBER)
  {
    LCDSetting = 1; // 0 of you wanna see the text in the begining and 1 if
you want data
  }
  Serial.print("LCDSetting: ");
  Serial.println(LCDSetting);
```

```
}

void LEDSTRIP() // Function to turn ledstrip on and off
{
  int ValueNumberLDR = 0;

  int LDRvalue = analogRead(LDR);
  Serial.print("LDR value: ");
  Serial.println(LDRvalue);

  if (LedSetting == 0)
  {
    ValueNumberLDR = 0;
  }
  else if (LedSetting == 1)
  {
    ValueNumberLDR = 2500;
  }
  else if (LedSetting == 2)
  {
    ValueNumberLDR = 1000;
  }
  else if (LedSetting == 3)
  {
    ValueNumberLDR = 2000;
  }

  if (LDRvalue <= ValueNumberLDR) // leds go on whenever
  {
    fill_solid(leds, NUM_LEDS, CRGB(50, 0, 100)); // Purple
    FastLED.show();
    LedStripStatus = "On";

    Serial.print("LED strip: ");
    Serial.println(LedStripStatus);
  }
  else
  {
    fill_solid(leds, NUM_LEDS, CRGB(0, 0, 0)); // off
    FastLED.show();
    LedStripStatus = "Off";

    Serial.print("LED strip: ");
    Serial.println(LedStripStatus);
  }

  if (LDRvalue <= 2500)
```

```
  {
    SunLightStatus = "Some";

    if (LDRvalue <= 1000)
    {
      SunLightStatus = "None";
    }
  }
  else
  {
    SunLightStatus = "A lot";
  }
  Serial.print("SunLightStatus: ");
  Serial.println(SunLightStatus);
}

void SoilMoistureSensor() // Function to get soil moisture data
{

  float sensorValue = analogRead(MoistureSensor);
  float moisturePercentage = map(sensorValue, 0, 4095, 0, 100);

  Serial.print("Soil Moisture: ");
  Serial.print(moisturePercentage);
  Serial.println("%");
  SoilMoisture = moisturePercentage;
}

void SoilTemperature() // Function to get soil temperature data
{
  sensors.requestTemperatures();

  float temperatureCelsius = sensors.getTempCByIndex(0);

  if (temperatureCelsius != DEVICE_DISCONNECTED_C)
  {
    Serial.print("Soil Temperature: ");
    Serial.print(temperatureCelsius);
    Serial.println(" °C");
    SoilTemperatureC = temperatureCelsius;
  }
  else
  {
    SoilTemperatureC = -999;
    Serial.print("Soil Temperature: ");
    Serial.println("Error: ");
    Serial.println(SoilTemperatureC);
```

```
  }
}

void WaterLevel() // Function to get water container data
{
  int sensor = analogRead(WaterSensor);
  Serial.print("WaterSensor: ");
  Serial.println(sensor);

  if (sensor == 0 || sensor < 3500)
  {
    Serial.println("Water reservoir: EMPTY ");
    WaterStatus = "EMPTY";
  }
  else if (sensor >= 3500)
  {
    Serial.println("Water reservoir: Functional ");
    WaterStatus = "Good";
  }
}

void BME280Sensor() // Function to get temperature and humidity data
{
  float temperature = bme.readTemperature();
  // float pressure = bme.readPressure() / 100.0; // Convert pressure to hPa
  float humidity = bme.readHumidity();

  GlobalTemperature = temperature;
  GlobalHumidity = humidity;

  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" °C");

  // Serial.print("Pressure = ");
  // Serial.print(pressure);
  // Serial.println(" hPa");

  Serial.print("Humidity = ");
  Serial.print(humidity);
  Serial.println(" %");
}

void BuzzerSound1() // Brief one time beeping sound
{
  tone(Buzzer, 1000);
  delay(250);
```

```
  noTone(Buzzer);
}

void BuzzerSound2() // 2 beeps, longer beeping sound
{
  tone(Buzzer, 600);
  delay(300);
  tone(Buzzer, 580);
  delay(350);
  noTone(Buzzer);
}

void rfidReadTask(void *parameter) // Function to use an apart loop to run
RFID_reader
{
  while (1)
  {
    RFID_Reader();

    vTaskDelay(pdMS_TO_TICKS(100));
  }
}

void CheckTask(void *parameter) // Function to use an apart loop to run
LCD_Setting
{
  while (1)
  {
    if (!client.connected())
    {
    }
    else
    {
      delay(4000);

      LCD_Setting();

      vTaskDelay(pdMS_TO_TICKS(200));
    }
  }
}

void Ventilator() // Function to cool the plantbox whenever it's too hot
{
  int WantedTemperature;

  if (VentilatorSetting == 0)
```

```
  {
    WantedTemperature = 999;
  }
  else if (VentilatorSetting == 1)
  {
    WantedTemperature = 21;
  }
  else if (VentilatorSetting == 2)
  {
    WantedTemperature = 18;
  }
  else if (VentilatorSetting == 3)
  {
    WantedTemperature = 19;
  }

  if (GlobalTemperature < WantedTemperature)
  {
    digitalWrite(VentilatorK1, HIGH);
    VentilatorStatus = 0;
    Serial.println("K1 HIGH (NOT OPPERATIONAL)");
  }
  else
  {
    digitalWrite(VentilatorK1, LOW);
    VentilatorStatus = 1;
    Serial.println("K1 LOW (OPPERATIONAL)");
  }
}

void WaterPump() // Function to pump water into the plantbox whenever it needs
water
{
  float Moisturenumber = 50.00;
  float MoisturenumberWithVentilator = 70.00;
  int TimeWatering;
  int activated;

  unsigned long currentMillis = millis();

  if (WaterSetting == 0)
  {
    TimeWatering = 0;
    activated = 0;
  }
  else if (WaterSetting == 1)
  {
```

```
    TimeWatering = 2000;
    activated = 1;
  }
  else if (WaterSetting == 2)
  {
    TimeWatering = 3000;
    activated = 1;
  }
  else if (WaterSetting == 3)
  {
    TimeWatering = 4000;
    activated = 1;
  }

  if (currentMillis - previousMillis >= WaitTime)
  {
    if (activated == 1)
    {
      if (VentilatorStatus == 1)
      {
        if (SoilMoisture <= MoisturenumberWithVentilator)
        {
          digitalWrite(WaterPumpK2, LOW);
          Serial.println("K2 LOW (OPPERATIONAL)");
          WaterPumpStatus = 1;
          SendData();
          delay(TimeWatering);
          SendData();
          digitalWrite(WaterPumpK2, HIGH);
          Serial.println("K2 HIGH (Timer concluded) " + String(TimeWatering) +
" milliseconds");
          WaterPumpStatus = 0;
          previousMillis = currentMillis;
        }
        else
        {
          digitalWrite(WaterPumpK2, HIGH);
          WaterPumpStatus = 0;
          Serial.println("K2 HIGH (NOT OPPERATIONAL)");
        }
      }
      else
      {

        if (SoilMoisture <= Moisturenumber)
        {
          digitalWrite(WaterPumpK2, LOW);
```

```
          Serial.println("K2 LOW (OPPERATIONAL)");
          WaterPumpStatus = 1;
          SendData();
          delay(TimeWatering);
          SendData();
          digitalWrite(WaterPumpK2, HIGH);
          Serial.println("K2 HIGH (Timer concluded) " + String(TimeWatering) +
" milliseconds");
          WaterPumpStatus = 0;
          previousMillis = currentMillis;
        }
        else
        {
          digitalWrite(WaterPumpK2, HIGH);
          WaterPumpStatus = 0;
          Serial.println("K2 HIGH (NOT OPPERATIONAL)");
        }
      }
    }
    else
    {
      digitalWrite(WaterPumpK2, HIGH);
      WaterPumpStatus = 0;
      Serial.println("K2 HIGH (NOT ACTIVATED)");
    }
  }
  else
  {
    digitalWrite(WaterPumpK2, HIGH);
    WaterPumpStatus = 0;
    Serial.println("K2 HIGH (" + String(WaitTime) + " millisecond timer
running)");
  }
}

void SendData() // Function to send all data to raspberry pi via MQTT
{
  // change all to a string so it can be sent
  String Ledstrip;
  String watercontainer;
  String fan = String(VentilatorStatus);
  String waterPUMP = String(WaterPumpStatus);
  String GlobalTemperatureString = String(GlobalTemperature);
  String GlobalHumidityString = String(GlobalHumidity);
  String localSoilMoisture = String(SoilMoisture);

  if (VentilatorK1 == HIGH)
```

```
  {
    fan = "1";
  }
  else if (VentilatorK1 == LOW)
  {
    fan = "0";
  }

  if (LedStripStatus == "On")
  {
    Ledstrip = "1";
  }
  else if (LedStripStatus == "Off")
  {
    Ledstrip = "0";
  }

  if (WaterStatus == "Good")
  {
    watercontainer = "1";
  }
  else if (WaterStatus == "EMPTY")
  {
    watercontainer = "0";
  }

  if (SoilTemperatureC == -999)
  {
  }
  else
  {
    String SoilTemperature = String(SoilTemperatureC);
    client.publish("home/plantenbak/Soiltemperature",
SoilTemperature.c_str());
  }

  // Publish sensor readings to MQTT topics
  client.publish("home/plantenbak/temperature",
GlobalTemperatureString.c_str());
  client.publish("home/plantenbak/humidity", GlobalHumidityString.c_str());
  client.publish("home/plantenbak/soilmoisture", localSoilMoisture.c_str());
  client.publish("home/plantenbak/led", Ledstrip.c_str());
  client.publish("home/plantenbak/waterPump", waterPUMP.c_str());
  client.publish("home/plantenbak/watercontainer", watercontainer.c_str());
  client.publish("home/plantenbak/fan", fan.c_str());
}
```

```
void PlantSelect() // selects what plant you want to grow (with RFID)
{
  if (Plant == "Unknown in system" || Plant == "")
  {
    LedSetting = 0;
    VentilatorSetting = 0;
    WaterSetting = 0;
  }
  else if (Plant == "Strawberries")
  {
    LedSetting = 1;
    VentilatorSetting = 1;
    WaterSetting = 1;
  }
  else if (Plant == "Lettuce")
  {
    LedSetting = 2;
    VentilatorSetting = 2;
    WaterSetting = 2;
  }
  else if (Plant == "Carrots")
  {
    LedSetting = 3;
    VentilatorSetting = 3;
    WaterSetting = 3;
  }
  Serial.print("Selected: ");
  Serial.println(Plant);
}
```

11. Evaluatie en Reflectie

Ik vond het wel een leuk, maar ook stressvol project, zeker als het ging over dingen bestellen. Er zijn een aantal dingen misgelopen waarvoor ik voor de meeste een oplossing heb gevonden zoals de PCB die niet 100% correct was, maar toch bruikbaar is. Toch vind ik dat ik een goed resultaat heb, en heb bijgeleerd.

12. GitHub

Alle codes en bestanden die je nodig hebt staan hier op GitHub.
Link naar de GitHub pagina: https://github.com/Senne3030/Project-Plantenbak

13. Bronnen

https://raspberrytips.com/install-grafana-raspberry-pi/
https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/
https://simonhearne.com/2020/pi-influx-grafana/
https://stackoverflow.com/questions/76756160/influxdb-client-stopped-working-modulenotfounderror-no-module-named-influxdb-c