

JONAS VAN HOOL

Smart Green House

Eindproject IoT

Inhoud

1. BESCHRIJVING VAN HET PROJECT	5
2. HARDWARE	6
2.1. ESP32-AliExpress	6
2.2. 5V-UV-LEDstrip-AliExpress	6
2.3. RFID-tag-AliExpress	6
2.4. Irrigatiekit-Arduino-AliExpress	6
2.5. Ultrasone-sensor-AliExpress	6
2.6. LCD-I2CModule-AliExpress	6
2.7. DHT11-link-amazon	6
2.8. miniventilator-link-fruugo	6
2.9. DS18B20-Amazon	7
2.10. Rode Led	7
3. SENSOREN	8
3.1. DHT11	8
3.2. Ultrasone sensor	10
3.3. DS18B20	11
3.4. Vochtsensor	13
3.5. RFID-tag	14
3.5.1. UID-Tags vinden	14
3.6. LDR	15
4. ACTUATOREN	17
4.1. 5V UV-Ledstrip	17
4.2. Waterpomp	18
4.3. Mini Ventilator	19
4.4. LCD (16X2) met I2C	20
4.5. Rode LED	22
5. VOLLEDIGE CODE	23
5.1. ESP32	23
5.2. Raspberry Pi	29
6. AANSLUIT- /ELEKTRISCH-SCHEMA	31
7. PCB-DESIGN	34
8. SERRE	35
8.1. Plan	35
8.2. Benodigdheden	37
8.3. Werkwijze	37
8.3.1. Bodem	37
8.3.2. Plantenbak	41
8.3.3. Deksel	51

Dit is de koptekst in stijl 'Koptekst'

9. CONFIGURATIE RASPBERRY PI	52
9.1. Imager	52
9.2. SSH	56
9.3. Update / Upgrade.	56
9.4. MQTT	57
9.5. Influxdb	59
9.6. GRAFANA	62
9.7. Opstarten bij boot:	62

Dit is de koptekst in stijl 'Koptekst'

1. Beschrijving van het project

We hebben een zeer grote serre die onderverdeeld is in meerdere kweekbakken met verschillende soorten kruiden. Omdat de kweekbakken geregeld van kruid veranderen wil de eigenaar graag een modulair systeem om deze te automatiseren en monitoren.

Elke kweekbak regelt afzonderlijk de ideale kweekomgeving voor het gekozen kruid. Zo zal een kweekbak voor tuinkers andere temperatuur, vochtigheid en belichtingsperiodes hebben dan een kweekbak voor aardbeien.

Elke kweekbak zal dus sensoren nodig hebben die de grondvochtigheid en temperatuur monitoren. Indien een vooraf bepaalde grenswaarde overschreden wordt zal er een bepaalde automatisatie/actie ondernomen worden. (te warm -> vensters open, te droog -> water pompen, voedingstoffen toevoegen?, ...) De belichting gebeurd ook via een automatisch programma.

De data van deze sensoren en de bijhorende acties worden opgeslagen in een database en kunnen worden weergegeven met grafana.

Om te voorkomen dat de kweker zijn kweekbakken opnieuw moet programmeren bij elke kruidwisseling zal hij door middel van een RFID tag de bak automatisch kunnen instellen op de gewenste kruiden. Deze instellingen kunnen worden weergegeven op een LCD scherm.

2. Hardware

2.1. ESP32-AliExpress

De **ESP32** is een microcontroller en gaan we gebruiken om alle sensoren/actuatoren aan te sluiten en te doen werken. De **ESP32** bevat ook een WiFi-Module, deze is nodig om alle data naar een database te kunnen sturen. (Via MQTT)

2.2. 5V-UV-LEDstrip-AliExpress

Deze **5V-UV-LedStrip** gaat de benodigde lichthoeveelheid aanvullen. Aangezien deze op 5V werkt, heb je geen externe bron nodig.

2.3. RFID-tag-AliExpress

De **RFID-tag** gaat ervoor zorgen dat ons Green House op de hoogte is van welke plant er in de plantenbak zit. Aan de hand van welke badge gescand is, gaan de optimale omstandigheden voor de plant veranderen.

2.4. Irrigatiekit-Arduino-AliExpress

De **Irrigatiekit van Arduino** bevat een **vochtsensor**, een **waterpomp** en een **BatteryPack**.

De **vochtsensor** gaat de vochtigheid van de potgrond meten om te bepalen of de **waterpomp** aan of uit moet.

De **BatteryPack** kan aangesloten worden op de arduino om deze te laten werken zonder netspanning te gebruiken.

2.5. Ultrasone-sensor-AliExpress

De **Ultrasone-Sensor** gaat het water niveau meten door een ultrasoon-signaal te verzenden naar de toplaag van het water. Hoe sneller het signaal terugkomt hoe hoger het waterniveau. Aan de hand van het tijdsverschil gaat het waterpercentage op het **LCD-scherm** weergegeven worden.

2.6. LCD-I2CModule-AliExpress

De **LCD-I2C-Module** is een module die op een **LCD** kan worden aangesloten zodat er minder bekabeling nodig is voor een aansluiting op de **ESP32**. De communicatie zal dan gebeuren via bussen.

2.7. DHT11-link-amazon

De **DHT11** gaat de vochtigheid en temperatuur in de serre meten. Indien het te warm is gaat het de ventilators aanzetten.

2.8. miniventilator-link-fruugo

De **mini-Ventilator** gaat de serre koelen, als het te warm is (gemeten door **DHT11**), dan gaat de ventilator aan.

Dit is de koptekst in stijl 'Koptekst'

2.9. DS18B20-Amazon

De **DS18B20** gaat de bodemtemperatuur meten.

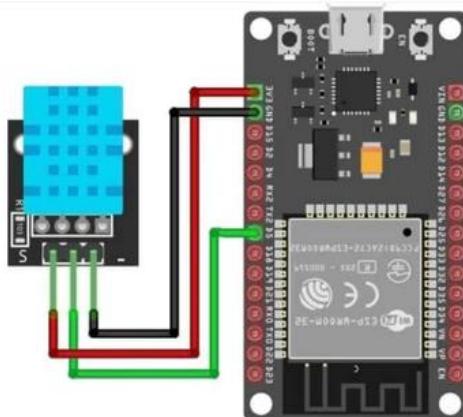
2.10. Rode Led

De **rode Led** ga ik gebruiken om te simuleren dat een warmtemat aan zou moeten gaan.
Een warmtemat is relatief duur en om deze reden heb ik deze niet aangekocht.

3. Sensoren

3.1. DHT11

Lezen van de vochtigheid en temperatuur in °C & Fahrenheit.



CODE

```
#include"DHT.h"

// Digitale pin verbonden met de DHT-sensor (aansluiting DataPin)
#defineDHTPIN26

// DHT 11 (Indien je gebruik maakt van DHT21/DHT22; verander type)
#defineDHTTYPE DHT11

// Initialiseer de DHT-sensor.
DHT dht(DHTPIN, DHTTYPE);

Voidsetup(){
    Serial.begin(9600);
    Serial.println(F("DHTxx test!"));

    dht.begin();
}

Voidloop(){
    // Wacht een paar seconden tussen metingen.
    delay(2000);

    // Lees vochtigheid
    float h = dht.readHumidity();
    // Lees temperatuur als Celsius (standaard)
    float t = dht.readTemperature();
```

Dit is de koptekst in stijl 'Koptekst'

```
// Lees temperatuur als Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);

// Controleer of er fouten zijn opgetreden bij het lezen en stop vroegtijdig
// (om opnieuw te proberen).
if(isnan(h) || isnan(t) || isnan(f)){
    Serial.println(F("Kan niet lezen vanaf DHT-sensor!"));
    return;
}

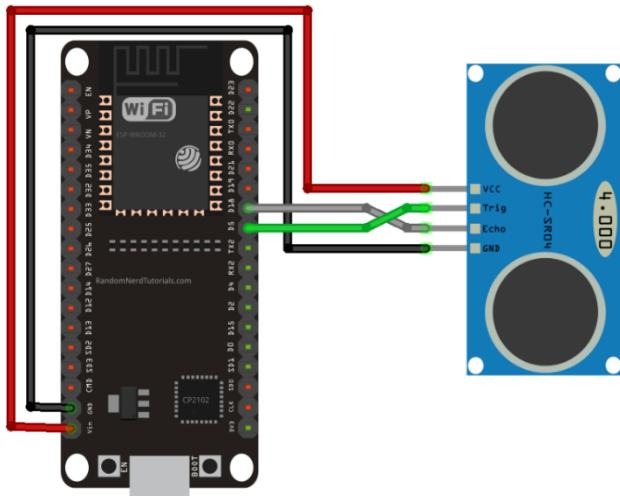
// Bereken de warmte-index in Fahrenheit (standaard)
float hif = dht.computeHeatIndex(f, h);
// Bereken de warmte-index in Celsius (isFahrenheit = false)
float hic = dht.computeHeatIndex(t, h, false);

// Printen van de resultaten op de Seriele Monitor
Serial.print(F("Vochtigheid: "));
Serial.print(h);
Serial.print(F("% Temperatuur: "));
Serial.print(t);
Serial.print(F("°C "));
Serial.print(f);
Serial.print(F("°F Warmte-index: "));
Serial.print(hic);
Serial.print(F("°C "));
Serial.print(hif);
Serial.println(F("°F"));

}
```

3.2. Ultrasone sensor

Meten van de afstand van een object, uitgedrukt in centimeter.



Code

```
const int trigPin = 5;
const int echoPin = 18;

// definieer geluidssnelheid in cm
#define SOUND_SPEED 0.034

long tijd;
float afstandCm;

void setup(){
    Serial.begin(115200); // Start de seriële communicatie
    pinMode(trigPin, OUTPUT); // Stelt trigPin in als een uitvoer
    pinMode(echoPin, INPUT); // Stelt echoPin in als een invoer
}

void loop(){
    // Maakt trigPin leeg
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Stelt trigPin in op HOOG voor 10 microseconden
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

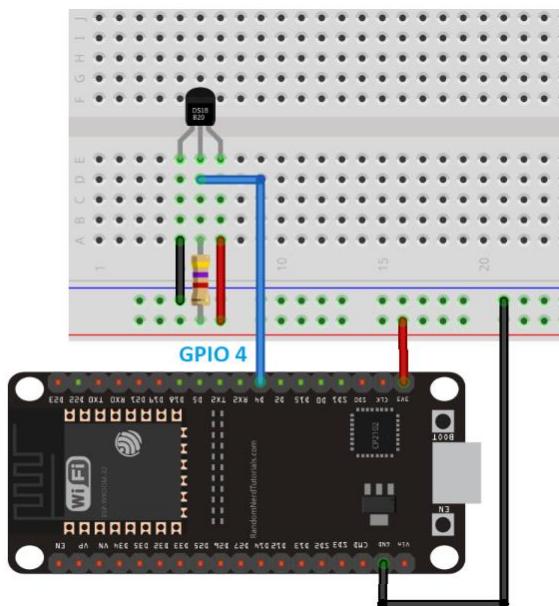
    // Wacht tot er een signaal terugkomt
    unsigned long start = micros();
    while (digitalRead(echoPin) == LOW)
        ;
```

Dit is de koptekst in stijl 'Koptekst'

```
// Leest echoPin, retourneert de reistijd van de geluidsgolf in  
microseconden  
tijd = pulseIn(echoPin, HIGH);  
  
// Bereken de afstand  
afstandCm = tijd * SOUND_SPEED/2;  
  
// Print de afstand in de seriële monitor  
Serial.print("Afstand (cm): ");  
Serial.println(afstandCm);  
delay(1000);  
}
```

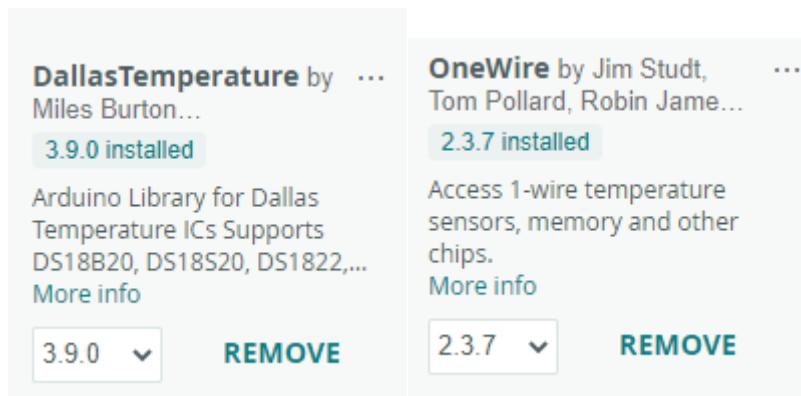
3.3. DS18B20

Grondtemperatuur meten



Library's

Dit is de koptekst in stijl 'Koptekst'



Code

```
#include<OneWire.h>
#include<DallasTemperature.h>

// GPIO waar de DS18B20 op is aangesloten
const int oneWireBus = 4;

// Stel een oneWire instantie in om te communiceren met OneWire apparaten
OneWire oneWire(oneWireBus);

// Geef onze oneWire referentie door aan de Dallas Temperature sensor
DallasTemperature sensors(&oneWire);

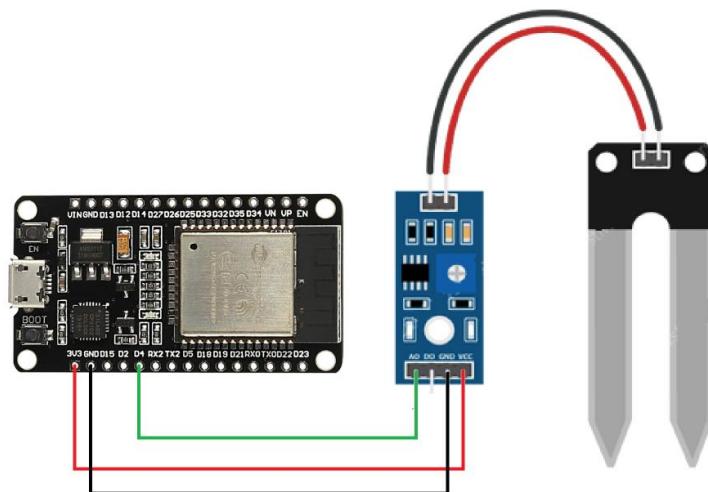
void setup(){
    // Start de Serial Monitor
    Serial.begin(115200);
    // Start de DS18B20 sensor
    sensors.begin();
}

void loop(){
    sensors.requestTemperatures();
    float temperatuurC = sensors.getTempCByIndex(0);
    Serial.print(temperatuurC);
    Serial.println("°C");
    delay(1000);
}
```

Dit is de koptekst in stijl 'Koptekst'

3.4. Vochtsensor

// Hoe hoger de waarde, hoe droger de grond



Code

```
#define soil_moisture_pin4

void setup(){
    Serial.begin(9600);
}

void loop(){
    Serial.println(analogRead(soil_moisture_pin));
    delay(500);
}
```

Dit is de koptekst in stijl 'Koptekst'

3.5. RFID-tag

Lezen welke UID-de badge heeft

Als de juiste UID wordt gelezen, print seriele monitor: authorized acces

ESP32	RFID Reader
3.3V	3.3V
14	RST
GND	GND
Not required	IRQ
19	MISO
23	MOSI
18	SCK
5	SDA

Library

MFRC522 by GithubCommunity

...

Arduino RFID Library for MFRC522 (SPI)
Read/Write a RFID Card or Tag using the ISO/IEC
14443A/MIFARE interface.

[More info](#)

1.4.11 ▾

INSTALL

<https://www.youtube.com/watch?v=pJLjFm4lpro>

Code

3.5.1. UID-Tags vinden

```
#include<SPI.h>
#include<MFRC522.h>

#define RST_PIN          14      // Configurable, see typical pin layout above
#define SS_PIN           5       // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance
```

Dit is de koptekst in stijl 'Koptekst'

```
voidsetup(){
    Serial.begin(9600); // Initializeserialcommunicationswiththe PC
    while(!Serial); // Do nothingif no serial port is opened
(addedforArduinosbased on ATMEGA32U4)
    SPI.begin(); // Init SPI bus
    mfrc522.PCD_Init(); // Init MFRC522
    delay(4); // Optional delay. Some board do need more time
afterinittobe ready, seeReadme
    mfrc522.PCD_DumpVersionToSerial(); // Show details of PCD - MFRC522 Card
Reader details
    Serial.println(F("Scan PICC tosee UID, SAK, type, and data blocks..."));
}

voidloop(){
    // Reset the loop if no new card present on the sensor/reader.
This savestheentireprocesswhenidle.
    if( ! mfrc522.PICC_IsNewCardPresent()){
        return;
    }

    // Select one of the cards
    if( ! mfrc522.PICC_ReadCardSerial()){
        return;
    }

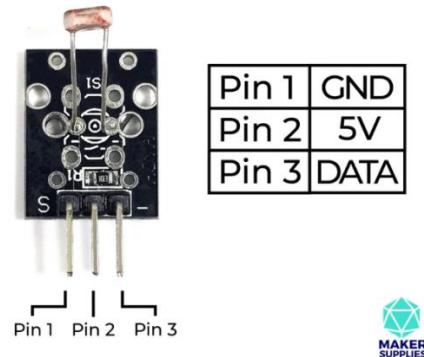
    // Dump debug info aboutthe card; PICC_HaltA() is automaticallycalled
    mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

/dev/cu.SLAB_USBtoUART																		
Card SAK: 08																		
PICC type: MIFARE 1KB																		
Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	AccessBits
15	63	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	62	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	61	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
14	59	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	58	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	57	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	56	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
13	55	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	54	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	53	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	52	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]

3.6. LDR

Op seriële monitor; de lichtwaarden meten (0-100)

Dit is de koptekst in stijl 'Koptekst'



Code

```
// Bepalen aan welke GPIO de data pin is verbonden
#define LDRPIN 13

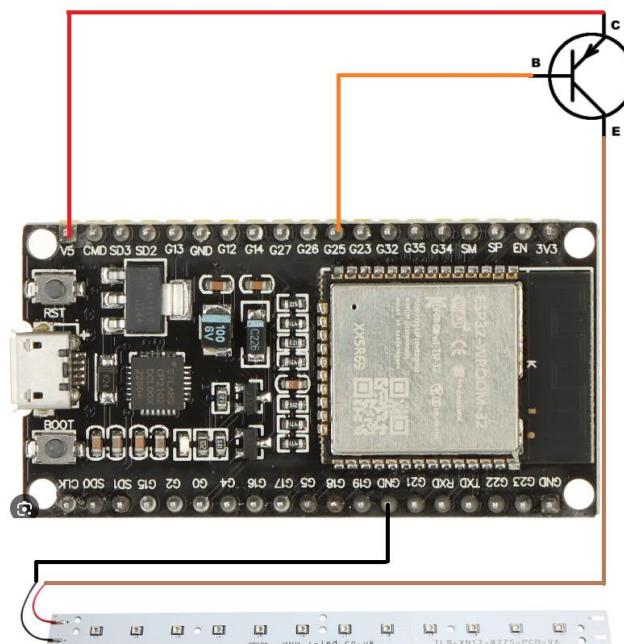
void setup(){
  Serial.begin(9600);          //Starten Seriële Communicatie
  pinMode(LDRPIN, INPUT);     //Bepalen dat LDRPIN een input is
}

void loop(){
  int ldr = analogRead(LDRPIN); // ldr = de analoog gemeten waarde van de LDRPIN
  Serial.println(ldr);         // de waarde van ldr wordt op de Seriële monitor
  geschrijven lijn per lijn
  delay(500);                 // per 0.5 seconde begint de loop opnieuw
}
```

4. Actuatoren

4.1. 5V UV-Ledstrip

Ledstrip aan/uit zetten



Code

```
#defineLedstrip27
```

```
// de setup-functie wordt één keer uitgevoerd wanneer je op reset drukt of de
voeding van het bord inschakelt
voidsetup(){
pinMode(Ledstrip, OUTPUT); //Bepalen van GPIO 27 '#defineLedstrip 27' als
OUTPUT
}

// de loop-functie wordt keer op keer continu uitgevoerd
voidloop(){
// initialiseerLedstrip als HIGH.(hierdoor gaat de ledstrip aan)
digitalWrite(Ledstrip, HIGH);

// wachttijd tot volgende regel code (in ms)
delay(1000);

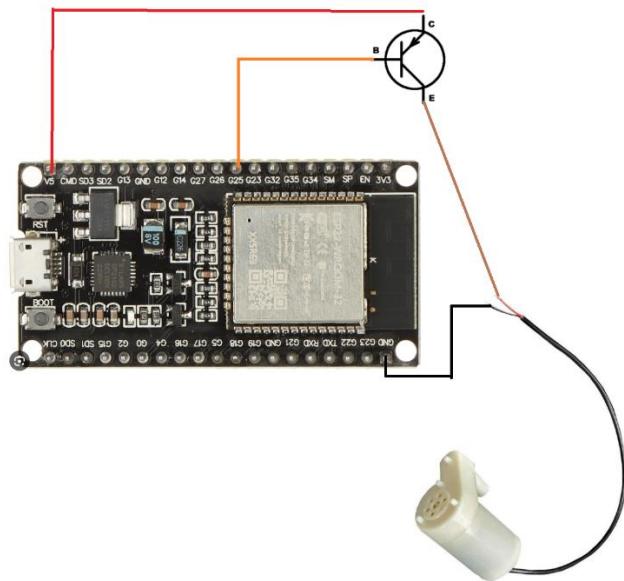
// initialiseerLedstrip als LOW. (Hierdoor gaat de ledstrip uit)
digitalWrite(Ledstrip, LOW);
```

Dit is de koptekst in stijl 'Koptekst'

```
// wachttijd tot volgende regel code (in ms)
delay(1000);
}
```

4.2. Waterpomp

Waterpomp aan/uit zetten



Code

```
#define Pomp25
```

```
// de setup-functie wordt één keer uitgevoerd wanneer je op reset drukt of de
voeding van het bord inschakelt
Voidsetup(){
}

// de loop-functie wordt keer op keer continu uitgevoerd
Voidloop(){
    // initialiseer digitale pin Pomp als een uitvoer.(hierdoor gaat de Pomp aan)
    pinMode(Ledstrip, OUTPUT);
```

Dit is de koptekst in stijl 'Koptekst'

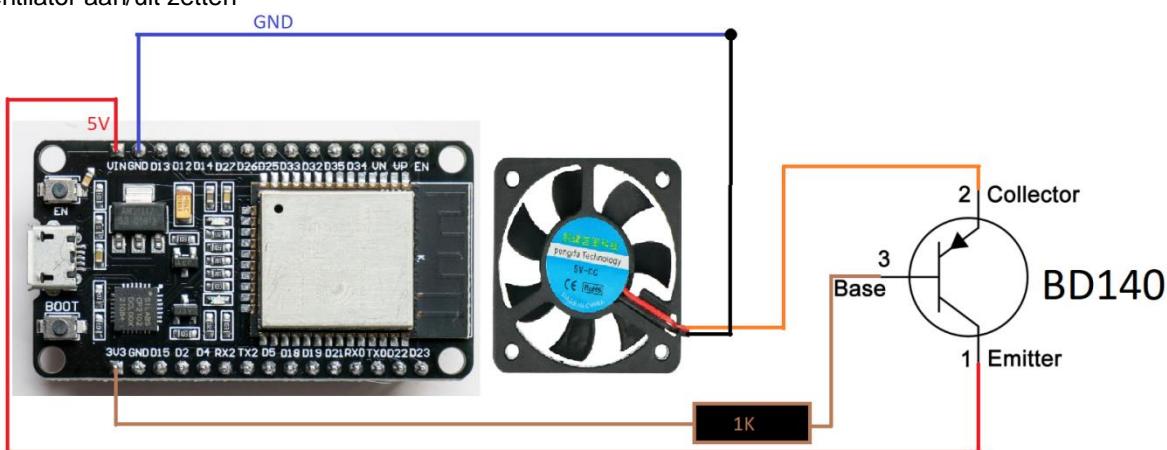
```
// wachttijd tot volgende regel code (in ms)
delay(1000);

// initialiseer digitale pin Ledstrip als een invoer. (Hierdoor gaat de Pomp
uit)
pinMode(Pomp, INPUT);

// wachttijd tot volgende regel code (in ms)
delay(1000);
}
```

4.3. Mini Ventilator

Ventilator aan/uit zetten



Code

```
#defineVentilator12

// de setup-functie wordt één keer uitgevoerd wanneer je op reset drukt of de
voeding van het bord inschakelt
Voidsetup(){

}

// de loop-functie wordt keer op keer continu uitgevoerd
Voidloop(){
    // initialiseer digitale pin Ventilator als een uitvoer.(hierdoor gaat
    ventilator aan)
```

Dit is de koptekst in stijl 'Koptekst'

```
pinMode(Ventilator, OUTPUT);

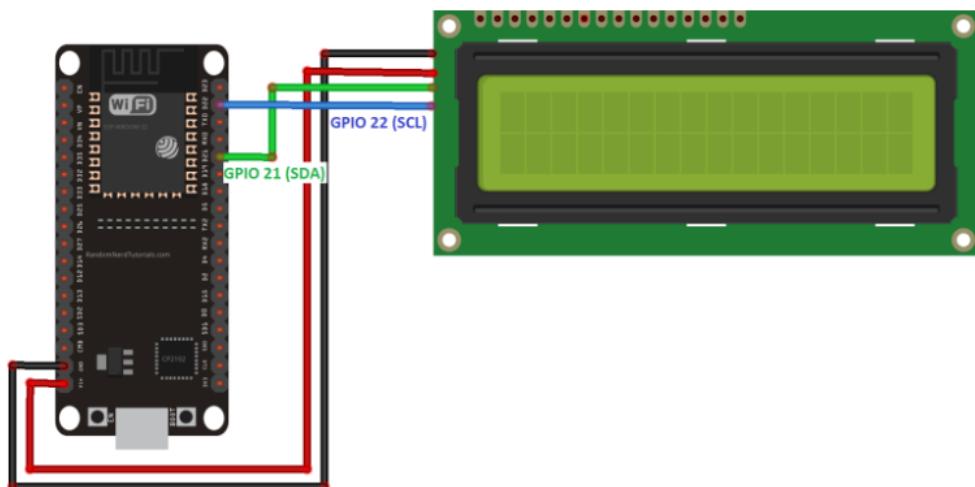
// wachttijd tot volgende regel code (in ms)
delay(1000);

// initialiseer digitale pin Ventilator als een invoer. (Hierdoor gaat
ventilator uit)
pinMode(Ventilator, INPUT);

// wachttijd tot volgende regel code (in ms)
delay(1000);
}
```

4.4. LCD (16X2) met I2C

Schrijven van: Hallo, wereld!
versprongen



Code

```
#include<LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,20,4); // stel het LCD-adres in op 0x27 voor een
weergave van 16 tekens en 2 regels

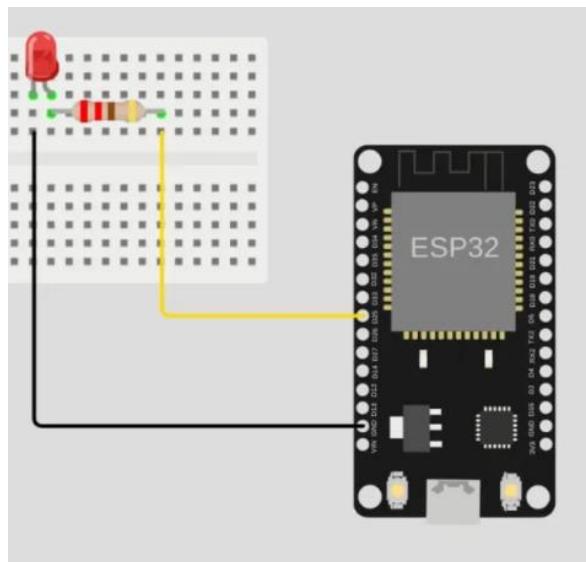
voidsetup()
```

Dit is de koptekst in stijl 'Koptekst'

```
{  
    lcd.init(); // initialiseer het lcd-scherm  
    // Print een bericht naar het LCD-scherm.  
    lcd.backlight();  
    // Zet de cursor op de eerste rij en het eerste karakter  
    lcd.setCursor(0,0);  
    lcd.print("Hallo, wereld!");  
    // Zet de cursor op de tweede rij (0,1) en het tweede karakter  
    lcd.setCursor(2,1);  
    lcd.print("versprongen");  
  
}  
  
voidloop()  
{  
}
```

4.5. Rode LED

Led aan/uit zetten



Code

```
#define LED_PIN 25

void setup(){
    pinMode(LED_PIN, OUTPUT);
}

void loop(){
    digitalWrite(LED_PIN, HIGH);
    delay(500);
    digitalWrite(LED_PIN, LOW);
    delay(500);
}
```

5. Volledige code

Hierin gaat de volledige gebruikte code geschreven staan met de uitleg wat de regels doenr kan ook verwezen worden naar voorgaande hoofdstukken.

<https://sprinklr.co/nl-be/blogs/tuinplanten-verzorging/tuinkers#:~:text=Waar%20je%20tuinkers%20zaait%20maakt,heeft%20is%20een%20beetje%20daglicht.>

5.1. ESP32

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <SPI.h>
#include <MFRC522.h>
#include "DHT.h"
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal_I2C.h>
#include <time.h>

// WiFi en MQTT server instellingen
const char* ssid = "embed";
const char* password = "weareincontrol";
const char* mqttServer = "192.168.1.34";
const int mqttPort = 1883;
const char* mqttUser = "SERRE";
const char* mqttPassword = "SERRE";
const char* clientID = "ESP32";

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

// RFID instellingen
#define SS_PIN 5
#define RST_PIN 14
MFRC522 mfc522(SS_PIN, RST_PIN);

// DHT11 sensor instellingen
#define DHTPIN 26
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float hum;
float temp;

// DS18B20 sensor instellingen
const int oneWireBus = 15;
OneWire oneWire(oneWireBus);
```

Dit is de koptekst in stijl 'Koptekst'

```
DallasTemperature sensors(&oneWire);
float bodem_temp;

// Bodemvochtsensor instelling
#define soil_moisture_pin 39
float grondwater;

// Ultrasone sensor instellingen
const int trigPin = 32;
const int echoPin = 35;
#define SOUND_SPEED 0.034
long tijd;
float afstandCM;
String level;

// LDR sensor instelling
#define LDRPIN 13
int ldr;

// Pin instellingen voor verwarming en ventilatie
#define verwarming 25
#define ventilatie 16

// LCD initialisatie
LiquidCrystal_I2C lcd(0x27, 20, 4);
String kaart;

// Pin instellingen voor pomp en LED-strip
#define pomp 4
#define strip 27

// Variabelen voor grenswaarden en tijd
int max_grondwater = 200;
int min_grondwater = 0;
int max_temp = 0;
int min_temp = 0;
int max_lichttijd = 0;

// Tijd instellingen
const long gmtOffset_sec = 3600;
const int daylightOffset_sec = 3600;

void setup() {
    Serial.begin(9600); // Initialiseer seriële communicatie
    dht.begin(); // Start de DHT sensor
    SPI.begin(); // Start SPI communicatie
    mfrc522.PCD_Init(); // Initialiseer RFID
    sensors.begin(); // Start de DS18B20 sensor
```

Dit is de koptekst in stijl 'Koptekst'

```
// Pin configuraties
pinMode(DHTPIN, INPUT);
pinMode(soil_moisture_pin, INPUT);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(LDRPIN, INPUT);
pinMode(verwarming, OUTPUT);
pinMode(strip, OUTPUT);
pinMode(pomp, OUTPUT);

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

// Verbinden met WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Verbinden met WiFi..");
}
Serial.println("Verbonden met het WiFi-netwerk");

// Instellen van de MQTT server
client.setServer(mqttServer, mqttPort);

// Tijd synchroniseren via NTP
configTime(gmtOffset_sec, daylightOffset_sec, "pool.ntp.org");
}

void loop() {
    // Uitvoeren van sensormetingen en regelingen
    RFID();
    DHT();
    DS18B20();
    Vochtsensor();
    UltraSone();
    LDR();
    temp_regeling();
    licht_regeling();
    water_regeling();
    LCD();
    publish();

    delay(1000); // Wacht een seconde

    // Verbind met de MQTT server indien niet verbonden
    client.loop();
    while (!client.connected()) {
        Serial.println("Verbinden met MQTT...");
```

Dit is de koptekst in stijl 'Koptekst'

```
if (client.connect("ESP32Client", mqttUser, mqttPassword)) {
    Serial.println("Verbonden");
} else {
    Serial.print("Verbinding mislukt, status: ");
    Serial.print(client.state());
    delay(2000);
}

void RFID() {
    // Lees RFID-kaart
    if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
        Serial.print("UID tag: ");
        String content = "";
        byte letter;
        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));
        }
        Serial.println();
        content.toUpperCase();
        lcd.clear();

        // Check welke kaart is gescand en stel de variabelen in
        if (content.substring(1) == "53 8A F6 27") {
            kaart = "Waterkers";
            max_grondwater = 30;
            min_grondwater = 5;
            max_temp = 22;
            min_temp = 20;
            max_lichttijd = 18000000; // 5 uur
        }
        if (content.substring(1) == "63 6C 58 14") {
            kaart = "Aardbei";
            max_grondwater = 200;
            min_grondwater = 50;
            max_temp = 24;
            min_temp = 20;
            max_lichttijd = 32400000; // 9 uur
        }
    }
}

void DHT() {
    // Lees de DHT11 sensor
    hum = dht.readHumidity();
```

Dit is de koptekst in stijl 'Koptekst'

```
temp = dht.readTemperature();
Serial.print("Vochtigheid: " + String(hum));
Serial.println(" Temperatuur: " + String(temp) + "°C");
}

void DS18B20() {
    // Lees de DS18B20 sensor
    sensors.requestTemperatures();
    bodem_temp = sensors.getTempCByIndex(0);
    Serial.println("Bodemtemperatuur: " + String(bodem_temp) + "°C");
}

void Vochtsensor() {
    // Lees de bodemvochtsensor
    grondwater = analogRead(soil_moisture_pin);
    Serial.println(grondwater);
}

void UltraSone() {
    // Meet de afstand met de ultrasone sensor
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    tijd = pulseIn(echoPin, HIGH);
    afstandCM = tijd * SOUND_SPEED / 2;
    Serial.print("Afstand (cm): ");
    Serial.println(afstandCM);
}

void LDR() {
    // Lees de lichtintensiteit
    ldr = analogRead(LDRPIN);
    Serial.println("Lichtintensiteit: " + String(ldr));
}

void temp_regeling() {
    // Regel de verwarming op basis van de temperatuur
    if (temp < min_temp) {
        digitalWrite(verwarming, HIGH);
    } else {
        digitalWrite(verwarming, LOW);
    }
}

void licht_regeling() {
    // Regel de LED-strip op basis van tijd en lichtintensiteit
    struct tm timeinfo;
```

Dit is de koptekst in stijl 'Koptekst'

```
if (!getLocalTime(&timeinfo)) {
    Serial.println("Tijd niet beschikbaar");
    return;
}

int hour = timeinfo.tm_hour;
int minute = timeinfo.tm_min;
int second = timeinfo.tm_sec;
long currentTime = hour * 3600 + minute * 60 + second; // Huidige tijd in
seconden sinds middernacht

long startTime = 6 * 3600; // Starttijd voor de LED-strip (bijv. 06:00 AM)
long endTime;

// Stel de eindtijd in op basis van de gescande kaart
if (kaart == "Waterkers") {
    endTime = startTime + (max_lichttijd / 1000); // 5 uur in seconden
} else if (kaart == "Aardbei") {
    endTime = startTime + (max_lichttijd / 1000); // 9 uur in seconden
} else {
    endTime = startTime; // Als geen kaart is gescand, stel het einde gelijk
aan de start
}

// Controleer of het tijd is om de LED-strip aan te zetten en of de
lichtintensiteit laag genoeg is
if (currentTime >= startTime && currentTime < endTime && ldr < 500) { //
Voeg lichtwaarde controle toe
    digitalWrite(strip, HIGH);
} else {
    digitalWrite(strip, LOW);
}

// Print de tijdsinstellingen voor debugging
Serial.print("Huidige tijd: ");
Serial.print(hour);
Serial.print(":");
Serial.print(minute);
Serial.print(":");
Serial.println(second);
Serial.print("Starttijd: ");
Serial.println(startTime);
Serial.print("Eindtijd: ");
Serial.println(endTime);
}

void water_regeling() {
    // Regel de waterpomp op basis van de grondvochtigheid
    if (grondwater < min_grondwater) {
```

Dit is de koptekst in stijl 'Koptekst'

```
    digitalWrite(pomp, HIGH);
} else {
    digitalWrite(pomp, LOW);
}
}

void LCD() {
// Update het LCD-scherm met actuele waarden
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Plant: " + kaart);
lcd.setCursor(0, 1);
lcd.print("Temp: " + String(temp) + "C");
lcd.setCursor(0, 2);
lcd.print("Vocht: " + String(hum) + "%");
lcd.setCursor(0, 3);
lcd.print("Bodem: " + String(bodem_temp) + "C");
}

void publish() {
// Alle waarden omzetten naar een String
String Shum = String((float)hum);
String Stemp = String((float)temp);
String Sbodem_temp = String((float)bodem_temp);
String Sgrondwater = String((float)grondwater);
String SafstandCM = String((float)afstandCM);
String Sldr = String((float)ldr);

// Alle strings publiceren naar juiste topic
client.publish("serre/serre/humdht11", Shum.c_str());
client.publish("serre/serre/tempdht11", Stemp.c_str());
client.publish("serre/serre/ds18b20", Sbodem_temp.c_str());
client.publish("serre/serre/bodemvochtsensor", Sgrondwater.c_str());
client.publish("serre/serre/ultrasone", SafstandCM.c_str());
client.publish("serre/serre/ldr", Sldr.c_str());
}
```

5.2. Raspberry Pi

Om python code te schrijven heb ik 1st een map aangemaakt genaamd serre.

In de 2^{de} lijn navigeer ik naar deze map.

In de 3^{de} lijn ga ik een nano bestand openen serre.py ("py" = is om aan te duiden dat het python code is)

```
SERRE@SERRE:~ $ mkdir serre
SERRE@SERRE:~ $ cd serre/
SERRE@SERRE:~/serre $ nano serre.py
```

Eenmaal in het bestand kunnen we de python code schrijven.

Dit is de koptekst in stijl 'Koptekst'

```
SERRE@SERRE: ~
GNU nano 7.2
import paho.mqtt.client as mqtt      #library voor mqtt gebruiken

#De gekozen topicnamen voor de sensors
ultrasone = "serre/serre/ultrasone"
ds18b20 = "serre/serre/ds18b20"
ldr = "serre/serre/ldr"
hum_dht11 = "serre/serre/humdht11"
temp_dht11 = "serre/serre/tempdht11"
bodemsensor = "serre/serre/bodemvochtsensor"

#Benodigdheden connectie broker(raspberry pi)
mqtt_broker = "192.168.1.34"      #ipadres
mqtt_port = 1883                  #poort
mqtt_user = "SERRE"                #gebruikersnaam
mqtt_password = "SERRE"             #wachtwoord

#Functie om te abonneren op alle onderwerpen
def on_connect(client, userdata, flags, rc, properties=None):
    print("verbonden " + str(rc))
    client.subscribe(ultrasone)
    client.subscribe(ds18b20)
    client.subscribe(ldr)
    client.subscribe(hum_dht11)
    client.subscribe(temp_dht11)
    client.subscribe(bodemsensor)

#Functie om de berichten van arduino te lezen
def on_message(client, userdata, msg):
    topic = msg.topic
    payload = msg.payload.decode("utf-8")

    if topic == ultrasone:
        print(f"Bericht ontvangen op topic {msg.topic}: {str(msg.payload.decode())}")
    if topic == ds18b20:
        print(f"Bericht ontvangen op topic {msg.topic}: {str(msg.payload.decode())}")
    if topic == ldr:
        print(f"Bericht ontvangen op topic {msg.topic}: {str(msg.payload.decode())}")
    if topic == hum_dht11:
        print(f"Bericht ontvangen op topic {msg.topic}: {str(msg.payload.decode())}")
    if topic == temp_dht11:
        print(f"Bericht ontvangen op topic {msg.topic}: {str(msg.payload.decode())}")
    if topic == bodemsensor:
        print(f"Bericht ontvangen op topic {msg.topic}: {str(msg.payload.decode())}")

mqttc = mqtt.Client() #mqtt.CallbackAPIVersion.VERSION2)   #callback functie van de mqttClient
mqttc.on_connect = on_connect                      #bevestiging connect
mqttc.on_message = on_message                      #bevestiging message
mqttc.username_pw_set(mqtt_user, mqtt_password)    #ingeven gebruikersnaam en wachtwoord
mqttc.connect(mqtt_broker, mqtt_port, 60)           #verbinden met broker

#Blijven loopen van de code tot keyboardinterrupt bij manueel opzetten. (sudo python serre.py)
try:
    mqttc.loop_forever()
except KeyboardInterrupt:
    mqttc.disconnect()
```

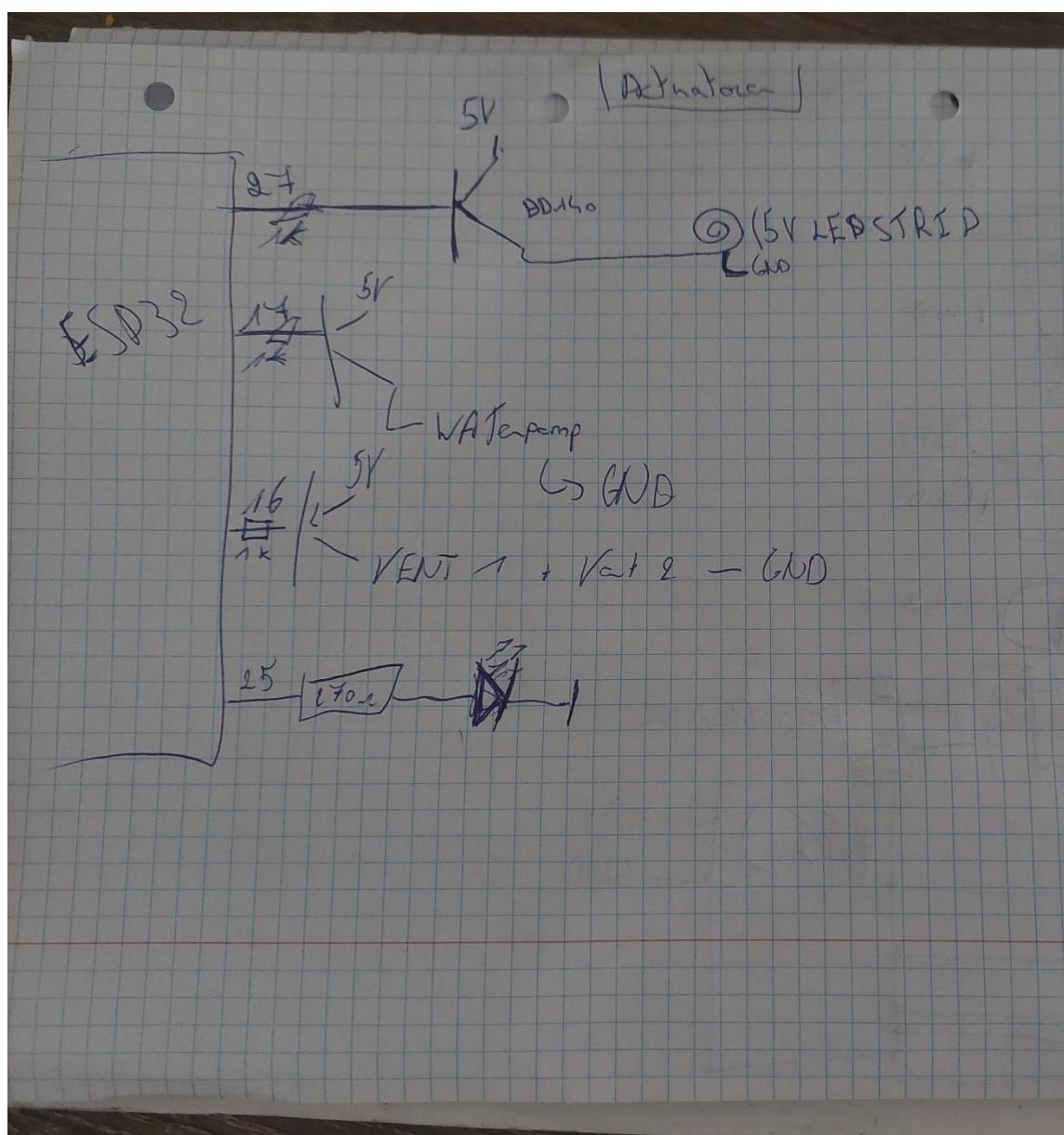
Dit is de koptekst in stijl 'Koptekst'

6. Aansluit- /elektrisch-schema

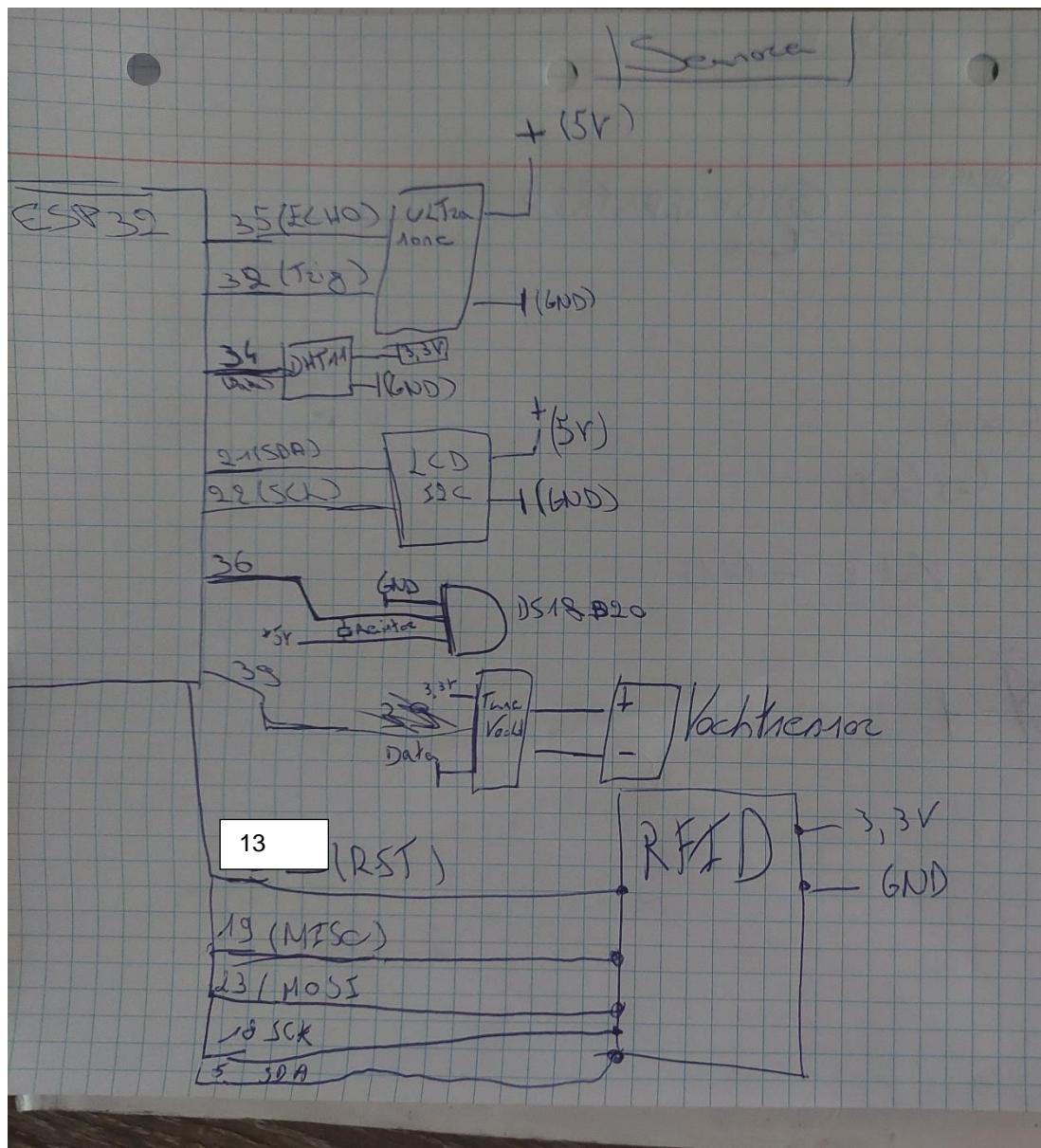
Het elektrisch schema is opgemaakt aan de hand van een vooraf bedacht aansluitschema.

Dit aansluitschema is ruwweg getekend om te kijken hoeveel en welke pinnen er benodigd zijn om de sensoren/actuatoren correct te kunnen aansluiten.

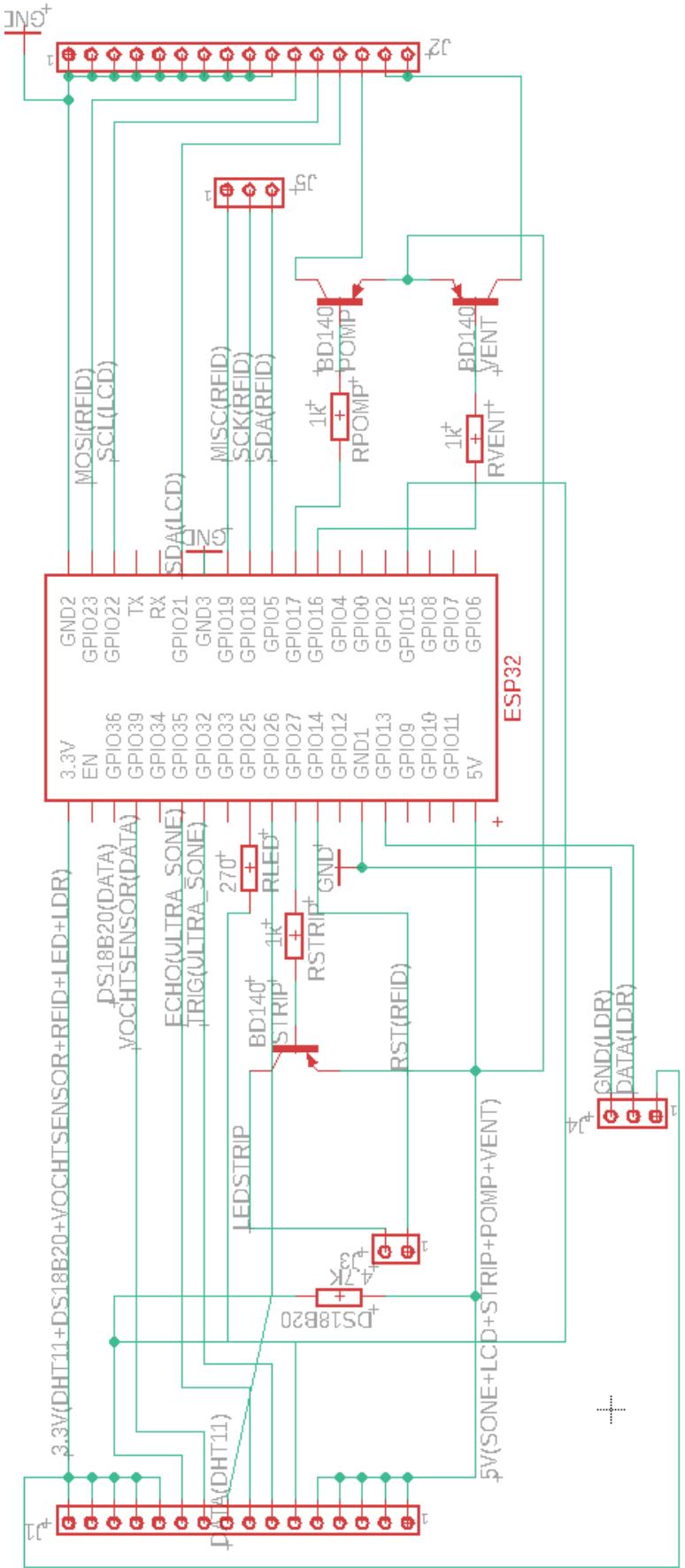
Tijdens het aanleggen van de bekabeling kan je er best voor zorgen dat deze lang genoeg is zodat je alles gemakkelijk kan solderen.



Dit is de koptekst in stijl 'Koptekst'



Dit is de koptekst in stijl ‘Koptekst’



Dit is de voettekst in stijl 'Voettekst'

Dit is de koptekst in stijl 'Koptekst'

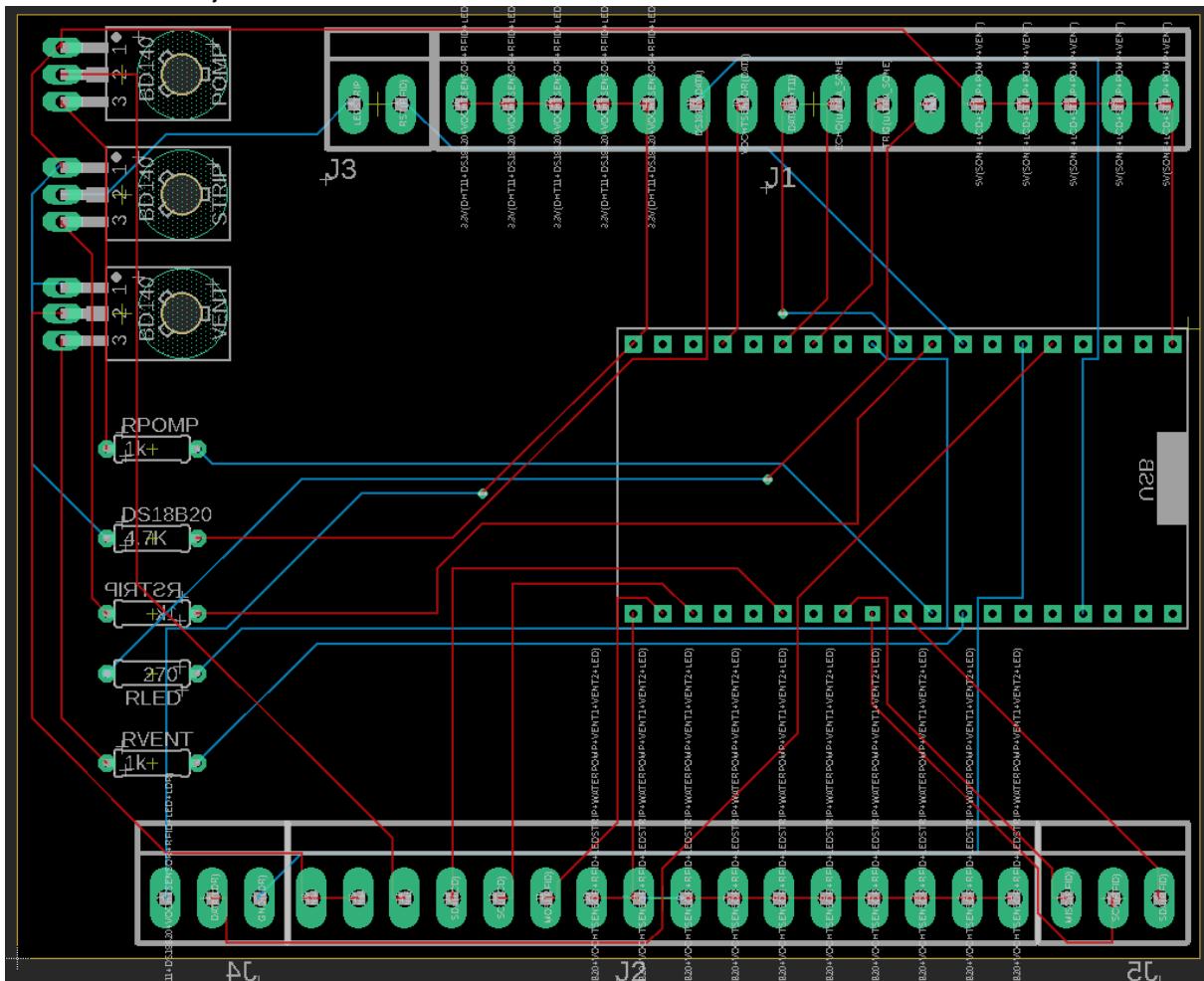
7. PCB-design

Op het onderstaande design zie je alle connectoren om de sensoren/actuatoren aan te sluiten.

Je kan 3 maal een BD140 terugvinden en 5 weerstanden.

De BD140 is een transistor om de 5V actuatoren aan te sturen (Ledstrip, Waterpomp en de Ventilatoren).

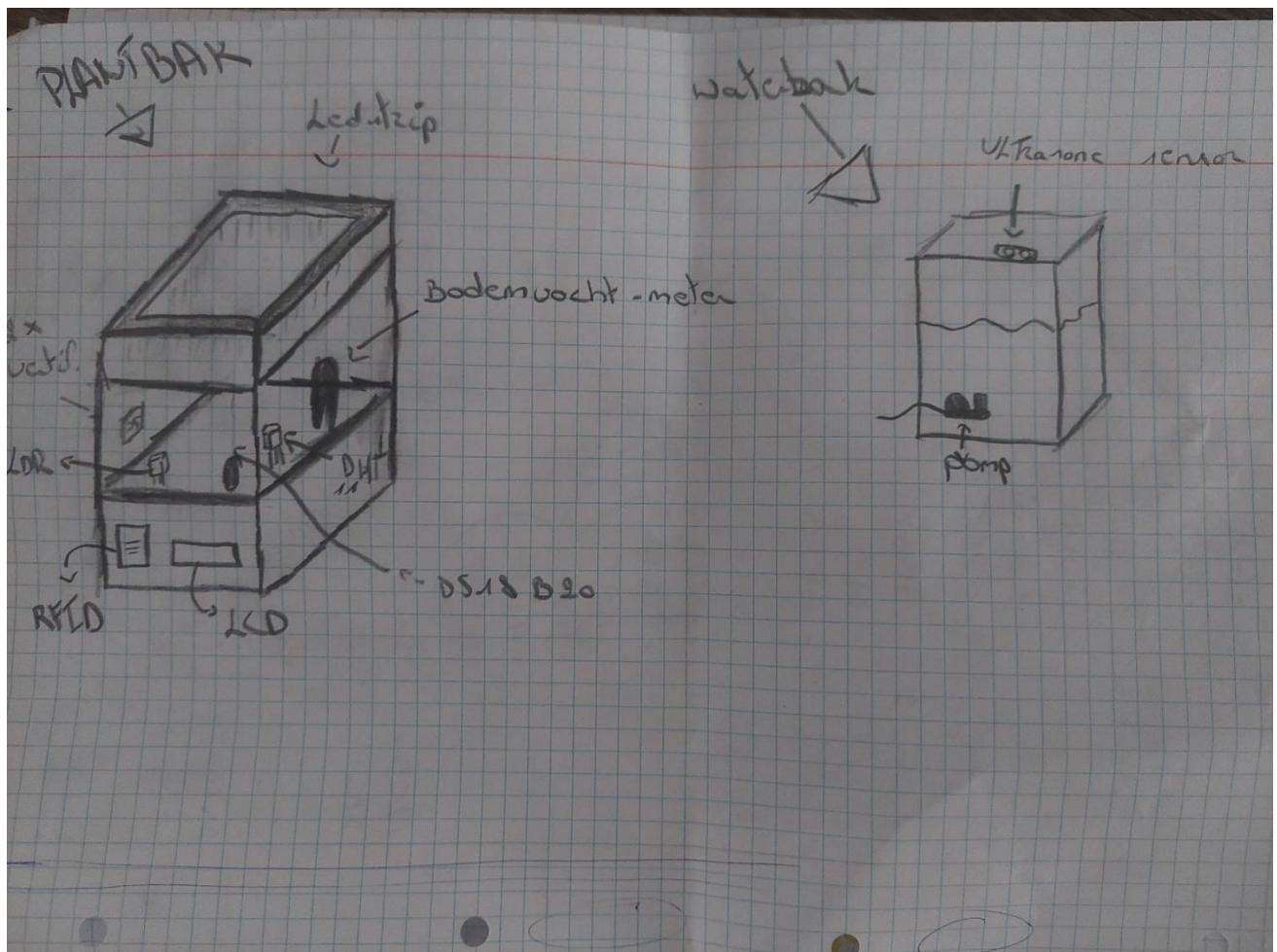
De weerstanden zijn voor zowel de transistoren alsook voor de Rode Led en DS18B20.



8. Serre

8.1. Plan

Op foto 1 kan je links zien dat ik gekozen heb om een soort gesloten plantenbak te maken waarbij de watertank (rechtse tekening) ernaast staat.

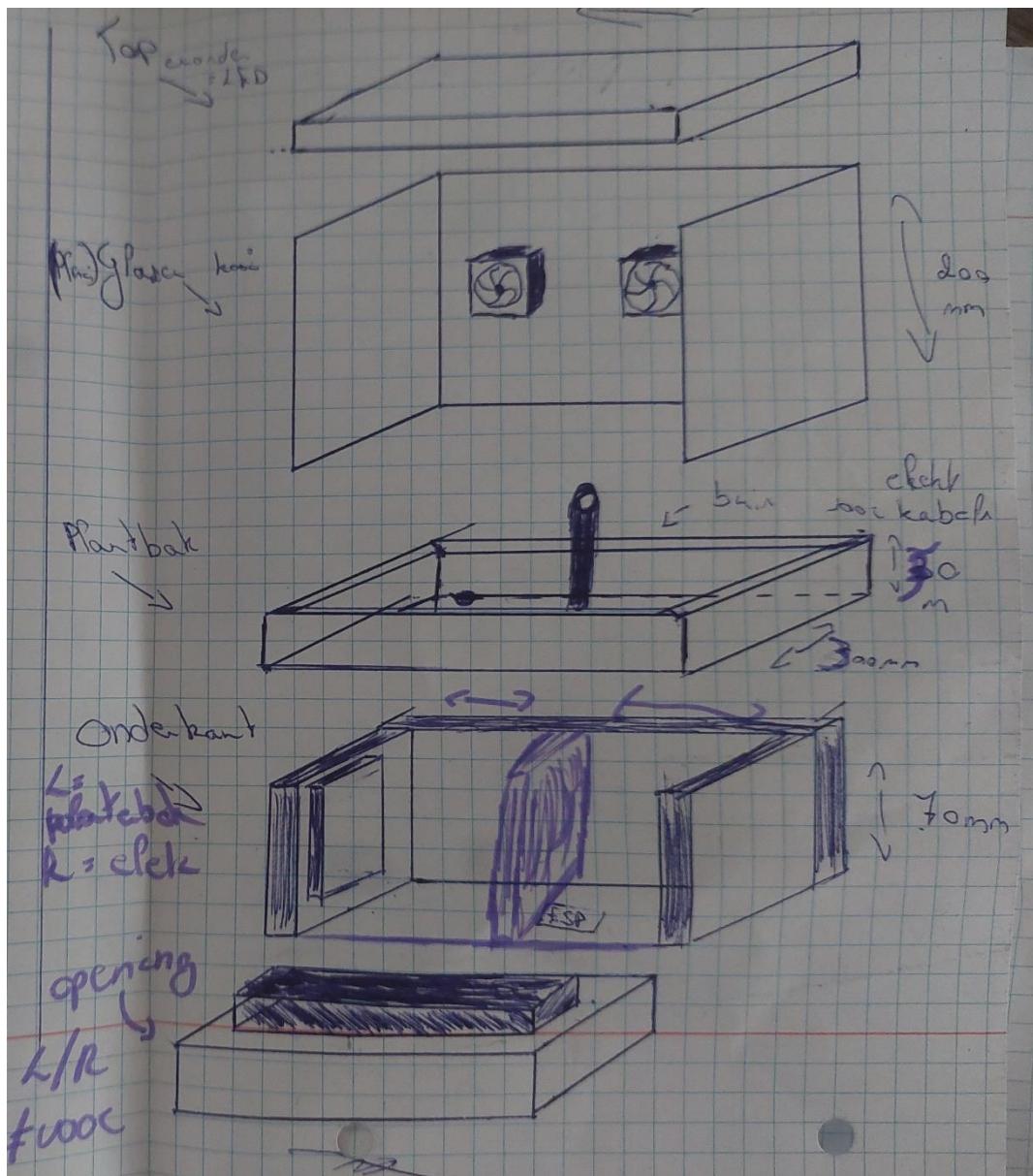


Aangezien ik mijn microcontroller en PCB nog ergens kwijt moest, heb ik het design aangepast zodat deze onder de plantenbak kan (Zie volgende foto). Een pvc-buis door de onderkant van de plantenbak tot boven de potgrond, zorgt ervoor dat de bekabeling toch tot boven geraakt.

Hierdoor kwam er ook ruimte vrij om onder de serre de watertank te steken. Het waterniveau kan langs de achterkant worden bijgevuld (in het echt aan kraan verbonden) als deze te laag wordt.

We kunnen alsnog aan de watertank en de ESP32 als we de zijkanten terug losschroeven.

Dit is de koptekst in stijl 'Koptekst'



Om de plantenbak zelf wat meer ondersteuning te geven als we een onderste zijpaneel weghalen, staat deze ook nog eens op 4 poten. (zie foto hieronder)

De scheidingswand is om het water apart te houden van de microcontroller.



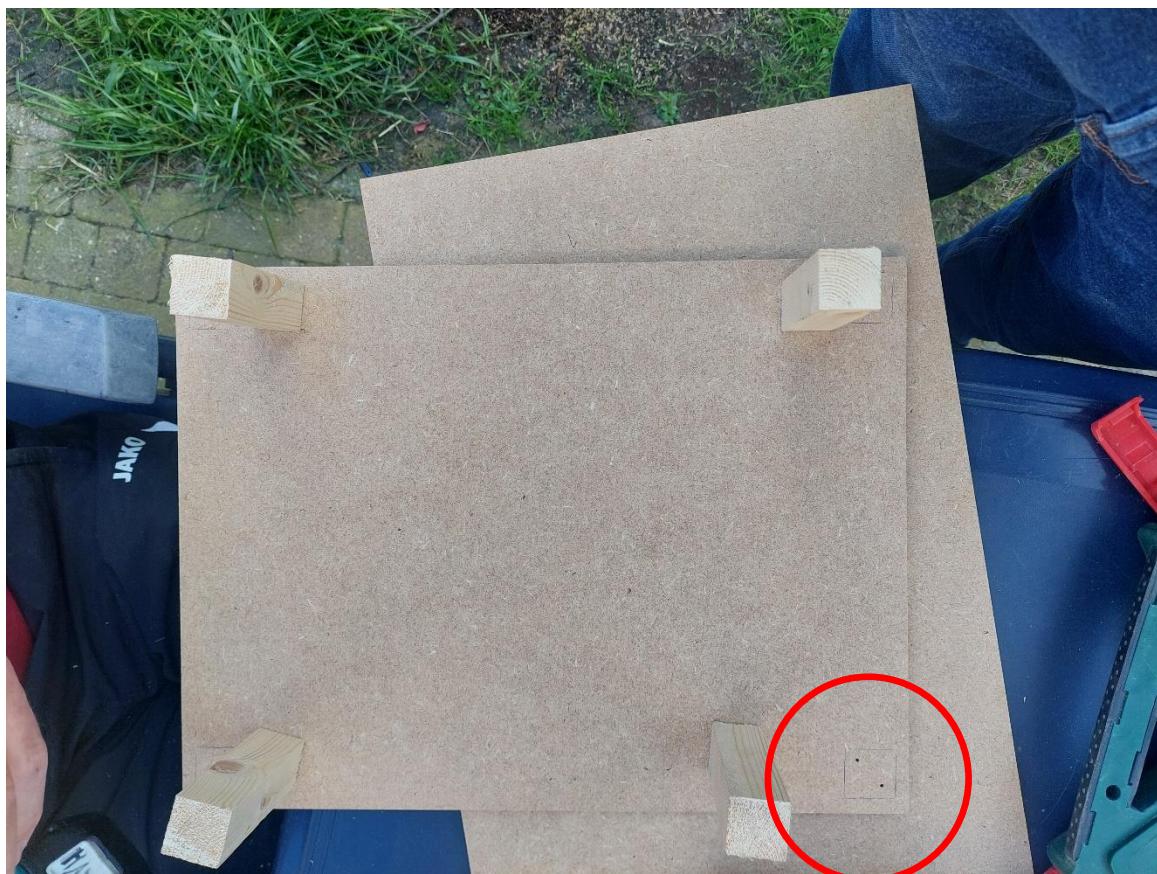
8.2. Benodigdheden

Materiaal	Hoeveelheid
(1) Houten balk (27*27)(l = 64 cm)	1
(2) Houten balk (18*92) (l = 132 cm)	1
(3) Houten plaat (400*300*6 mm)	3
(4) Houten balk (400*90*6 mm)	2
(5) Houten balk (300*90*6 mm)	2
(6) Houten plaat (280*90*6 mm)	1
(7) Plexiglazen plaat (0.5*1 m) (2.5 mm)	1
(8) Schroeven (d = 3mm) (l = 20mm)	Voldoende
(9) Schroeven (d = 3mm) (l = 30mm)	Voldoende
(10) Nagels	9
(11) Doorzichtige plexigassilicone	1 bus
(12) Houtlijm	1 bus

8.3. Werkwijze

8.3.1. Bodem

- 1) Neem (3) uit de lijst en leg deze vlak op tafel.
- 2) Meet voor elke hoek een vierkant van 27*27 die 7mm van de rand verwijderd is.
- 3) Maak in het getekende vierkant 2 gaatjes met een boor van 2 mm. (niet te dicht op de rand, is om vijzen in te vijzen).
- 4) Neem (1) en zaag 4 stukken van 9 cm.



- 5) Leg de plaat met de gaatjes op de 4 afgezaagde balkjes.

Dit is de koptekst in stijl 'Koptekst'

- - 6) Zet de balkjes zodanig dat er onder elk zelfgetekend vierkantje een balkje steekt.
 - 7) Steek 1 balkje precies onder een vierkantje en vijs deze vast met **(7)**. Doe dit totdat elke hoek vasthangt.



- 8) Neem de overige houtenbalk van **(1)** en neem ook een houten plaat **(6)**.
- 9) Leg de balk en de houten plaat zo tegen elkaar dat ze over de lengte tegen elkaar aanliggen
- 10) Vijs deze vervolgens aan elkaar vast met een aantal vijzen **(9)** verdeeld over de lengte.
- 11) Nu kan je het knutselwerkje van stap 10 vastmaken aan het knutselwerkje uit stap 7. (Zie onderstaande afbeelding)

Dit is de koptekst in stijl 'Koptekst'

Doe dit zodanig dat de scheiding op 2/3^{de} van de bak is verdeeld.



Dit is de koptekst in stijl 'Koptekst'



Dit is de koptekst in stijl 'Koptekst'

8.3.2. Plantenbak

- 1) Neem een houten plaat (3) de houten balken (4) en (5).
- 2) Leg de balken zo, waardoor de balken de buitenrand van de houten plaat volgen maar er niet overgaan.
- 3) Zaag de te lange stukken van de houten balken tot juist formaat.
- 4) En vijs deze vervolgens vast met een (9).



Dit is de koptekst in stijl 'Koptekst'



Dit is de koptekst in stijl 'Koptekst'

- 5) Vijs vervolgens deze bak aan de pootjes van de bodem met **(8)** Doe dit zo dat de vijzen lichtjes scheef naar buiten toe gaan.



Dit is de koptekst in stijl 'Koptekst'



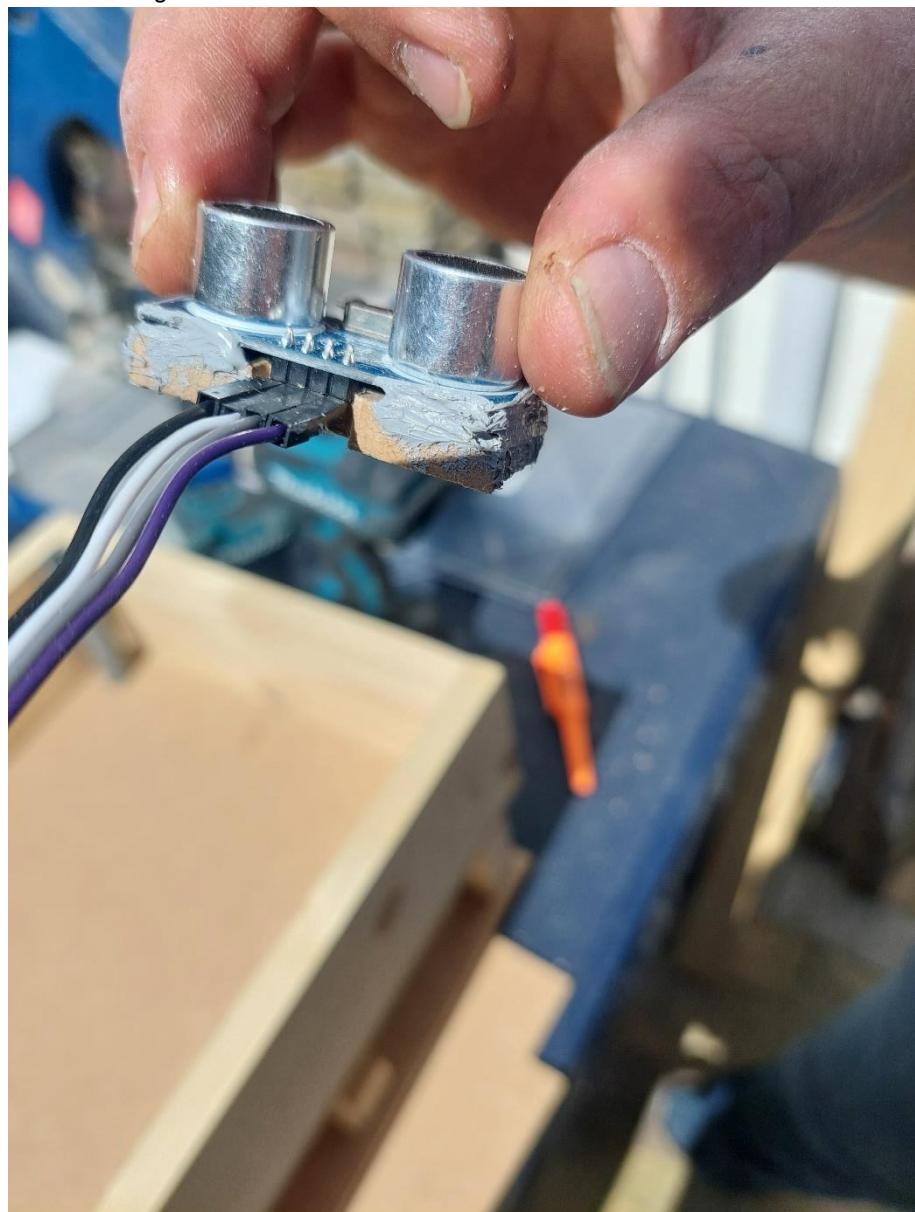
Dit is de koptekst in stijl 'Koptekst'

- 6) Boor linksboven een gat van 8 mm voor de waterslang en rechtsboven een gat 20 mm zodat er een pvc buis van 20 mm door kan. (Na het solderen komen de kabels via deze buis tot boven) Zorg ervoor dat de zijkant van de gaten minstens 2.5mm van de rand af is zodat de plexiglazen behuizing er nog tussen kan.



Dit is de koptekst in stijl 'Koptekst'

- 7) Lijm (met waterdichte silicone) aan de onderkant van de Ultra-Sone een stuk hout met een kleine opening voor de bekabeling.



- 8) Als dit gedroogd is kunnen we deze aan de onderkant van de plantenbak hangen. Met houtlijm (**9**).

Dit is de koptekst in stijl 'Koptekst'



De Houten balk met vijs onder is om deze tegen het hout aan te duwen. Van zodra de lijm droog is kan deze worden weggehaald.

Nu we de onderkant en de plantenbak aan elkaar hebben verbonden kunnen we de omkadering voor de afwerking aan de onderkant ook afmaken.

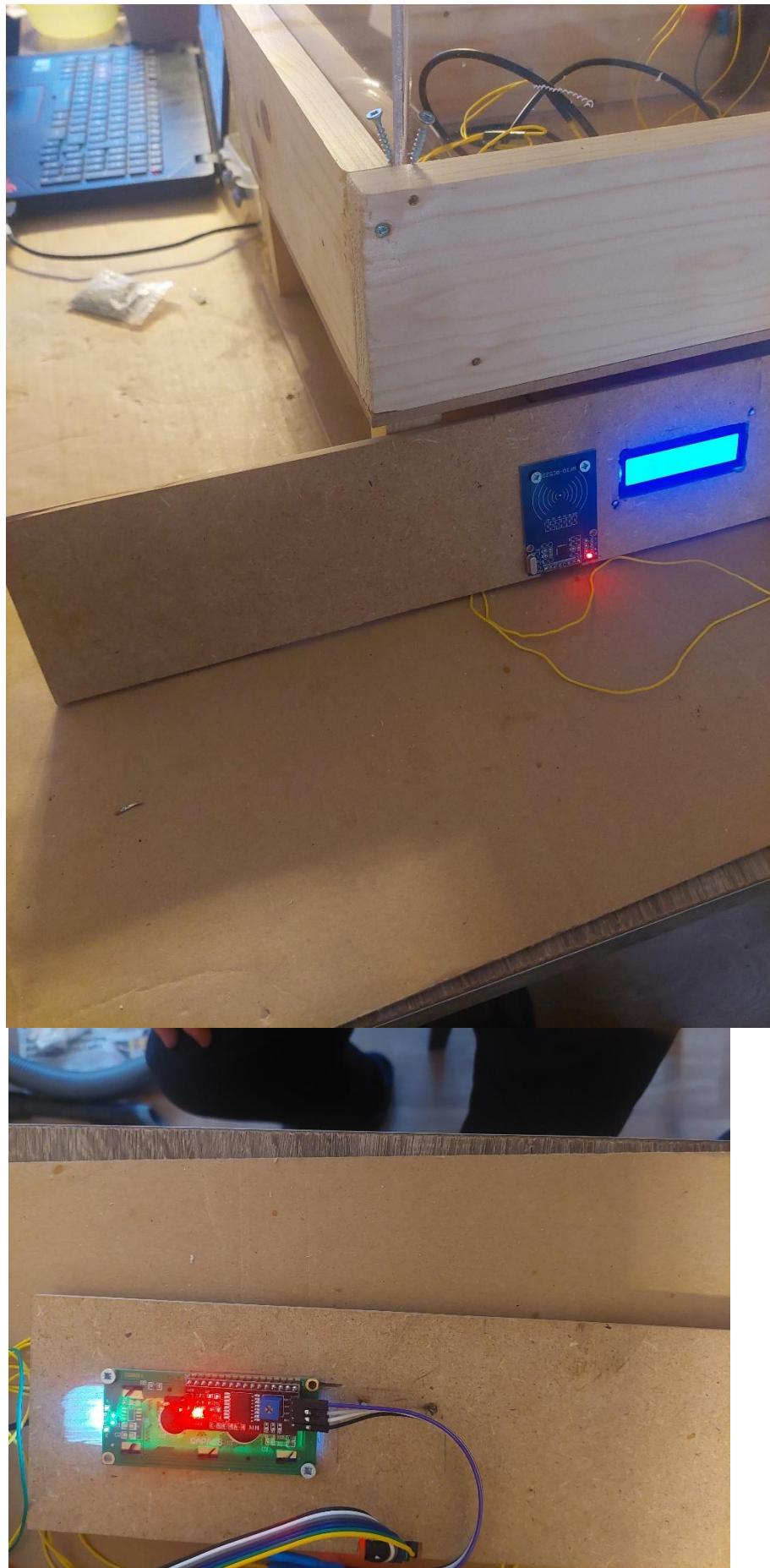
Dit is de koptekst in stijl 'Koptekst'

- 9) Neem 1 keer **(4)** en 2 maal **(5)** deze kunnen aan de achterkant en de zijkant gestoken worden. Als je dit goed hebt gedaan past deze perfect en heb je dus geen nagels nodig.
Te groot? → Schuur plankje bij
Te Klein? → Klop nagels **(10)** ter hoogte van de balkjes op de hoeken. (Achteraan; kan je ook de middenbalk gebruiken)



- 10) Neem de houten balk **(4)** en voorzie een gat waar de LCD in past en een gleuf waar de bekabeling van de RFID door kan tot aan de ESP32.

Dit is de koptekst in stijl 'Koptekst'



Dit is de voettekst in stijl 'Voettekst'

Dit is de koptekst in stijl 'Koptekst'

- 11) Snij de plexiglazen plaat zodat je **2X** een vlak van **105*364mm** en **2X** een vlak van **105*264mm** hebt.
- 12) Elke wand past nu perfect in de bak.
- 13) Voorzie in de achterste wand een gaat waar de lucht van de ventilator door kan.
- 14) Plaats deze 4 wanden correct in de bak zodat de meting 105 overal in de hoogte is geplaatst en de ventilator zich bovenaan bevindt.
- 15) Schroef deze vast aan de wand. (Van binnen naar buiten met **(8)**)
- 16) Voeg ter volledigheid en dichting nog aan de binnenkant van de hoeken doorzichtige silicone **(11)** toe.



Dit is de koptekst in stijl 'Koptekst'

8.3.3. Deksel

Gebruik voor het deksel de derde houten plaat (3).

- 1) Plak de Ledstrip hieraan vast. Met de kabels in de richting van de kabelgoot en door het pvc-buisje.



Nu kan je deze op de plexiglazen plaat leggen. Met de kabel naar binnen.

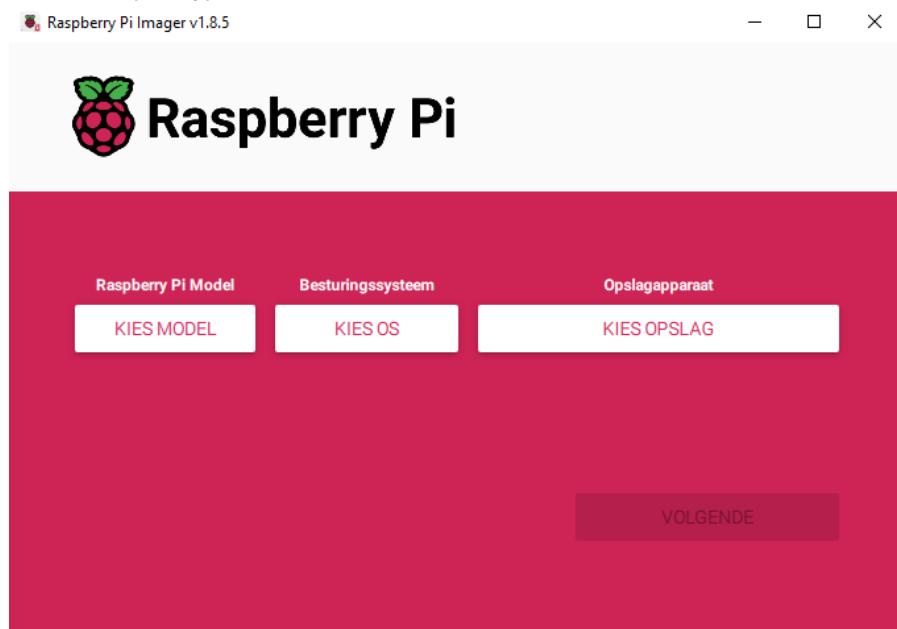
Nu kunnen we alles solderen

Dit is de koptekst in stijl 'Koptekst'

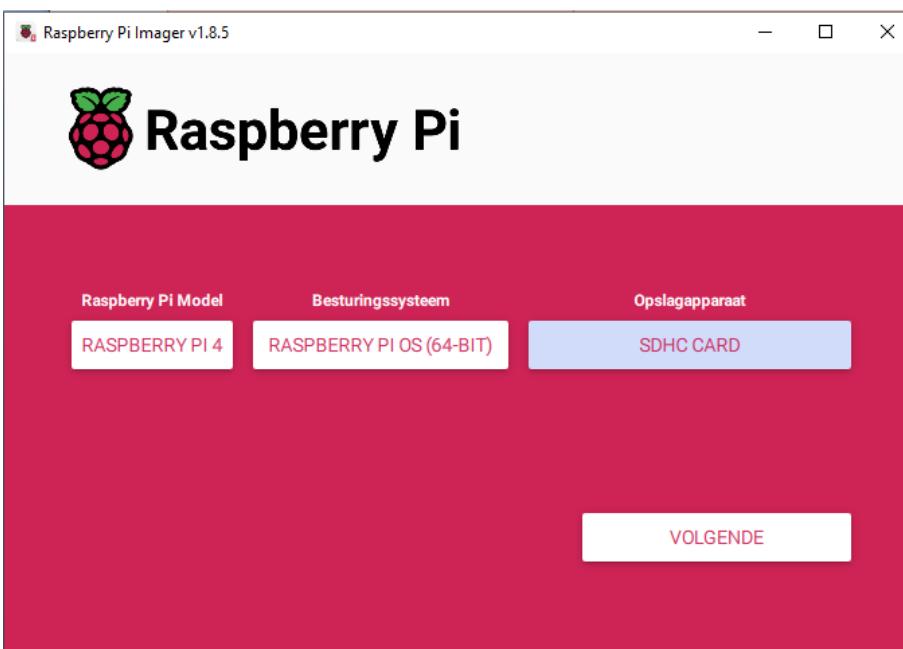
9. Configuratie Raspberry Pi

9.1. Imager

De Raspberry Pi Imager wordt gebruikt om het besturingssysteem op een opslagapparaat (usb / sd-kaart) te zetten. <https://www.raspberrypi.com/software/>



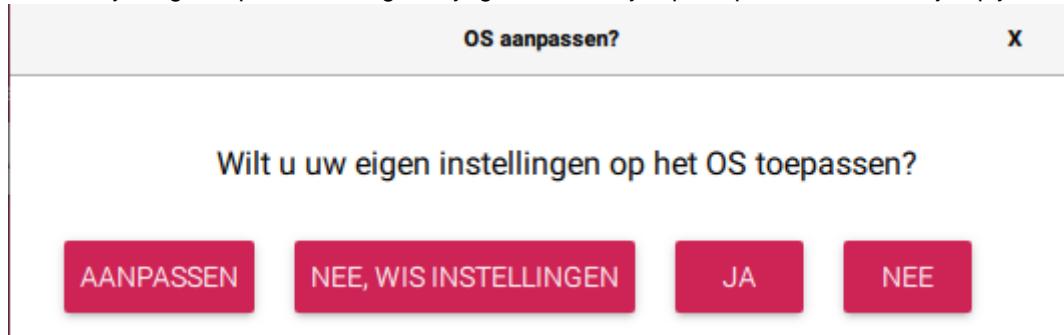
Ik heb gekozen voor Raspberry Pi 4, met OS-systeem (64-bit) dat wordt geschreven naar mijn sd-kaart.



Dit is de koptekst in stijl 'Koptekst'

Klik vervolgens op volgende.

Hier moet je nog een paar instellingen wijzigen. Dus klik je op aanpassen vooraleer je op ja klikt.



PAS AAN: Je hostnaam, gebruikersnaam, wachtwoord, SSID, wachtwoord.

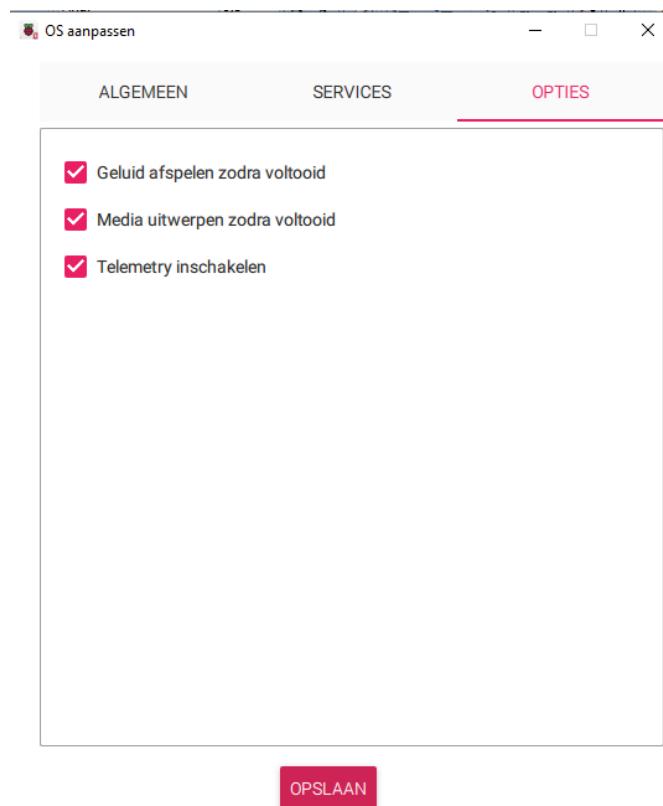
Controleer: Regio instellingen, en WiFi-land, SSH-inschakelen

Optioneel: Telemetry inschakelen (delen van gegevensgebruik met Raspberry Pi communiteit)

Dit is de koptekst in stijl 'Koptekst'

The screenshots show the configuration interface for the OS aanpassen application. The top window is titled 'OS aanpassen' and has tabs for 'ALGEMEEN', 'SERVICES', and 'OPTIES'. The 'ALGEMEEN' tab is active, showing fields for Hostnaam (SERRE.local), Gebruikersnaam (Jonas), Wachtwoord (redacted), Wifi instellen (SSID: wifi-naam, Wachtwoord: wifi-code), and Region settings (Tijdzone: Europe/Brussels, Toetsenbord indeling: be). The bottom window is also titled 'OS aanpassen' and has tabs for 'ALGEMEEN', 'SERVICES', and 'OPTIES'. The 'SERVICES' tab is active, showing options for SSH inschakelen (checked) and authentication methods (radio buttons for Gebruik wachtwoord authenticatie (selected) and Gebruik uitsluitend public-key authenticatie).

Dit is de koptekst in stijl 'Koptekst'



Klik op **OPSLAAN**.



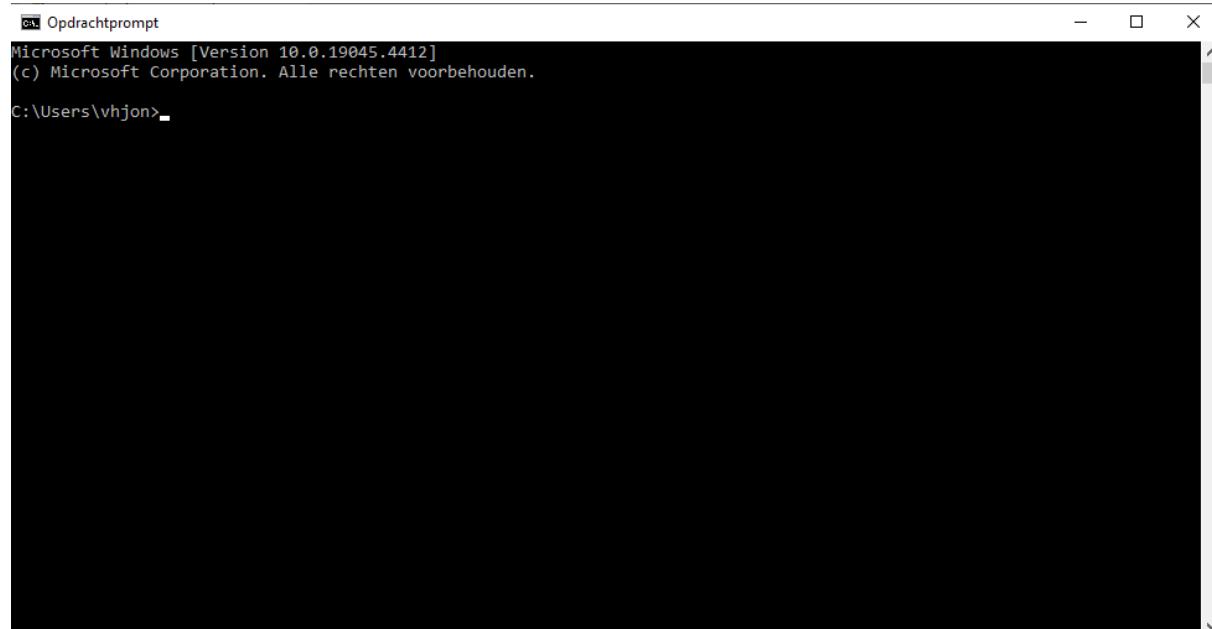
En vervolgens op **JA**.

9.2. SSH

Om in te loggen op je Raspberry Pi, steek je je opslagapparaat in de Raspberry Pi en wacht je een paar minuten totdat de Raspi is geboot.

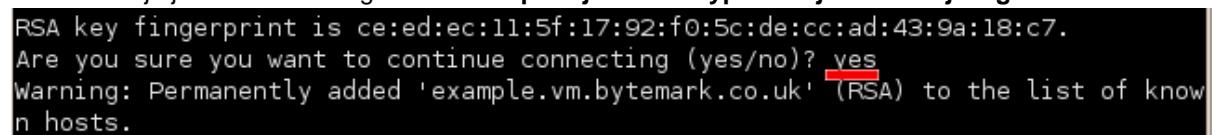
Vervolgens kan je via **SSH** inloggen op je apparaat.

Open je opdrachtprompt op de PC: druk op de Windows toets  en typ vervolgens **cmd**.



Nu kan je intypen **ssh host@user**. Pas de host en de user aan naar je gekozen inloggegevens (hfdst 9.1). Vervolgens vraagt het of je wil doorgaan: hier typ je **yes**.

Daarna kan je je wachtwoord ingegeven. **!!Let op!! Tijdens het typen zie je niet wat je ingeeft.**



Nu ben je verbonden via SSH.

9.3. Update / Upgrade.

Het 1^{ste} wat je doet na het inloggen, is je systeem updaten en upgraden. Dit doe je aan de hand van volgende commando-lijnen.

```
SERRE@SERRE:~ $ sudo apt update  
SERRE@SERRE:~ $ sudo apt upgrade
```

Dit is de koptekst in stijl 'Koptekst'

9.4. MQTT

Nu dat het systeem up-to-date is, kunnen we **mosquitto(-clients)** installeren om gebruik te maken van MQTT.

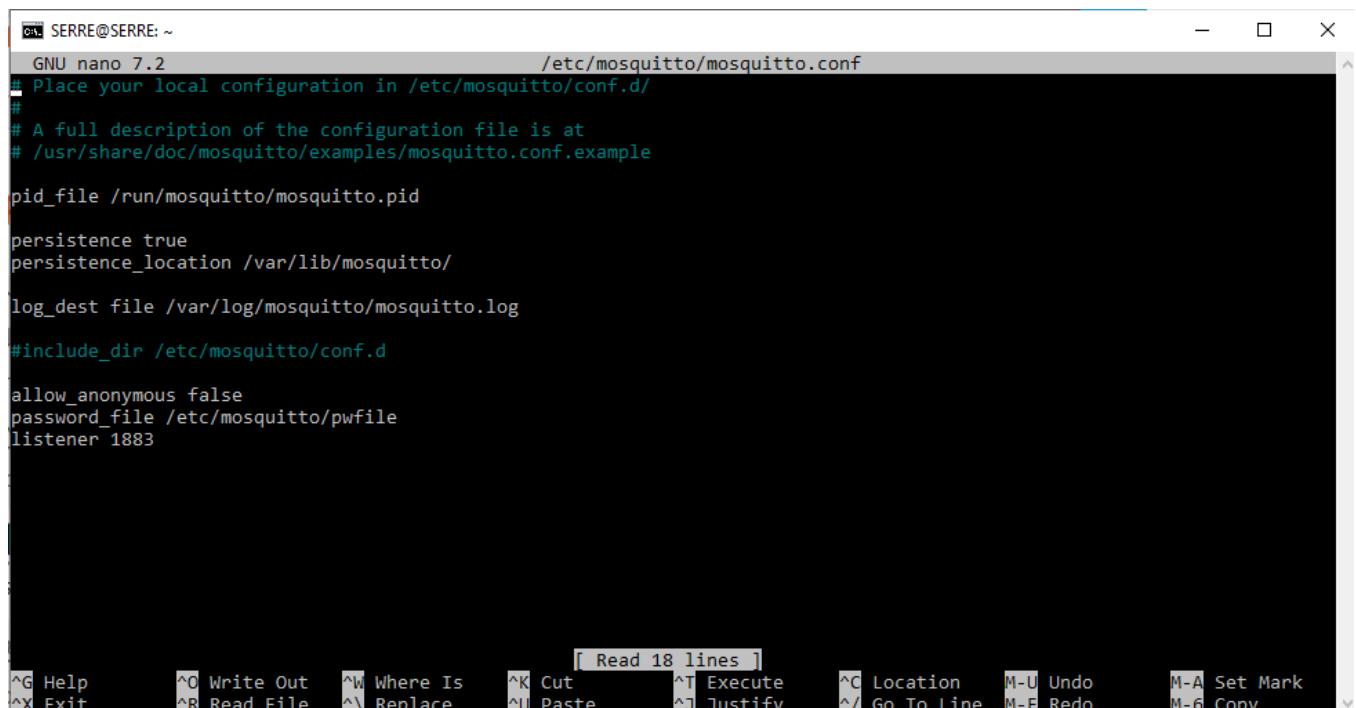
```
SERRE@SERRE:~ $ sudo apt-get install mosquitto  
SERRE@SERRE:~ $ sudo apt-get install mosquitto-clients
```

Na deze installatie moeten we nog een paar dingen doen vooraleer we MQTT kunnen gebruiken.

```
SERRE@SERRE:~ $ sudo nano /etc/mosquitto/mosquitto.conf
```

Hiermee open je volgend blad.

Zorg ervoor dat er bij jezelf **exact hetzelfde** staat



```
SERRE@SERRE:~  
GNU nano 7.2 /etc/mosquitto/mosquitto.conf  
# Place your local configuration in /etc/mosquitto/conf.d/  
#  
# A full description of the configuration file is at  
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example  
  
pid_file /run/mosquitto/mosquitto.pid  
  
persistence true  
persistence_location /var/lib/mosquitto/  
  
log_dest file /var/log/mosquitto/mosquitto.log  
  
#include_dir /etc/mosquitto/conf.d  
  
allow_anonymous false  
password_file /etc/mosquitto/pwfile  
listener 1883
```

The terminal window shows the configuration file for Mosquitto. The file path is /etc/mosquitto/mosquitto.conf. The configuration includes settings for persistence, log files, and a password file. The terminal window has a menu bar with icons for Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, and Copy. The status bar at the bottom shows 'Read 18 lines'.

Nu moeten we nog een naam en wachtwoord aan onze mqtt toevoegen:

```
SERRE@SERRE:~ $ sudo mosquitto_passwd -c /etc/mosquitto/pwfile SERRE  
Password:  
Reenter password:
```

Achteraan staat de **naam** en vervolgens geef je weer blindelings een gekozen **wachtwoord** in.
!!TIP!! schrijf dit allemaal ergens op.

Dit is de koptekst in stijl 'Koptekst'

Vervolgens gaan we mosquitto(-clients) **starten**, **status** controleren en gaan we deze **ENABLEN** (zodat deze bij opstart werkt).

Typ enkel in wat er rechtstreeks na SERRE@SERRE komt: “**sudo ...**”

```
SERRE@SERRE:~ $ sudo systemctl start mosquitto
SERRE@SERRE:~ $ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
    Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
    Active: active (running) since Wed 2024-05-29 09:15:27 CEST; 5min ago
      Docs: man:mosquitto.conf(5)
             man:mosquitto(8)
   Process: 30148 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 30149 ExecStartPre=/bin/chown mosquitto /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 30150 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, status=0/SUCCESS)
   Process: 30151 ExecStartPre=/bin/chown mosquitto /run/mosquitto (code=exited, status=0/SUCCESS)
 Main PID: 30152 (mosquitto)
    Tasks: 1 (limit: 8732)
      CPU: 122ms
     CGroup: /system.slice/mosquitto.service
             └─30152 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

May 29 09:15:27 SERRE systemd[1]: Starting mosquitto.service - Mosquitto MQTT Broker...
May 29 09:15:27 SERRE systemd[1]: Started mosquitto.service - Mosquitto MQTT Broker.
SERRE@SERRE:~ $ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
SERRE@SERRE:~ $ sudo systemctl enable mosquitto.service
Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
```

Nu kunnen we testen of alles werkt. Open hiervoor een 2^{de} opdrachtprompt en log hier ook via SSH op in.

Opdrachtprompt 1:

```
SERRE@SERRE:~ $ mosquitto_sub -v -t test/message -u SERRE -P SERRE
test/message hello World
test/message hello World
test/message hello World
```

Opdrachtprompt 2: Wijzig achter –u (je gebruikersnaam) en –P (je wachtwoord)

```
SERRE@SERRE:~ $ mosquitto_pub -h localhost -t test/message -m 'hello World' -u SERRE -P SERRE
SERRE@SERRE:~ $ mosquitto_pub -h localhost -t test/message -m 'hello World' -u SERRE -P SERRE
SERRE@SERRE:~ $ mosquitto_pub -h localhost -t test/message -m 'hello World' -u SERRE -P SERRE
SERRE@SERRE:~ $
```

Zolang je de 2^{de} opdrachtprompt ingeef en enterd komt er in de 1^{ste} “test/message hello World” te staan.

Nu werkt MQTT.

Dit is de koptekst in stijl 'Koptekst'

9.5. Influxdb

Om gebruik te maken van influxdb(-clients) moeten we deze 1st nog installeren:

```
SERRE@SERRE:~/serre $ sudo apt-get install influxdb
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libraspberrypi0 libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  influxdb
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,483 kB of archives.
After this operation, 18.0 MB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main arm64 influxdb arm64 1.6.7~rc0-1+b13 [4,483 kB]
Fetched 4,483 kB in 2s (1,993 kB/s)
Selecting previously unselected package influxdb.
(Reading database ... 147733 files and directories currently installed.)
Preparing to unpack .../influxdb_1.6.7~rc0-1+b13_arm64.deb ...
Unpacking influxdb (1.6.7~rc0-1+b13) ...
Setting up influxdb (1.6.7~rc0-1+b13) ...
Adding system user `influxdb` (UID 113) ...
Adding new user `influxdb` (UID 113) with group `nogroup` ...
Not creating home directory `/var/lib/influxdb'.
Adding group `influxdb` (GID 123) ...
Done.
Adding user `influxdb` to group `influxdb` ...
Done.
Created symlink /etc/systemd/system/influxd.service → /lib/systemd/system/influxdb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/influxdb.service → /lib/systemd/system/influxdb.service.
Processing triggers for man-db (2.11.2-2) ...
SERRE@SERRE:~/serre $ sudo apt-get install influxdb-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libraspberrypi0 libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  influxdb-client
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,034 kB of archives.
After this operation, 6,828 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main arm64 influxdb-client arm64 1.6.7~rc0-1+b13 [2,034 kB]
Fetched 2,034 kB in 1s (2,076 kB/s)
Selecting previously unselected package influxdb-client.
(Reading database ... 147748 files and directories currently installed.)
Preparing to unpack .../influxdb-client_1.6.7~rc0-1+b13_arm64.deb ...
Unpacking influxdb-client (1.6.7~rc0-1+b13) ...
Setting up influxdb-client (1.6.7~rc0-1+b13) ...
Processing triggers for man-db (2.11.2-2) ...
```

Vervolgens moeten we controleren of de status actief is.

Indien dit niet het geval is, pas dan in onderstaande lijn het volgende aan: status→enable. En controleer daarna nogmaals.

```
SERRE@SERRE:~/serre $ sudo systemctl status influxdb
● influxdb.service - InfluxDB is an open-source, distributed, time series database
  Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; preset: enabled)
  Active: active (running) since Wed 2024-05-29 10:07:16 CEST; 3min 50s ago
    Docs: man:influxd(1)
   Main PID: 31041 (influxd)
      Tasks: 9 (limit: 8732)
        CPU: 266ms
```

Dit is de koptekst in stijl 'Koptekst'

Als influxdb actief is dan gaan we volgend bestand wijzigen.

```
SERRE@SERRE:~/serre $ sudo nano /etc/influxdb/influxdb.conf
```

Haal de # voor enabled = true bij http weg. Om deze lijn te activeren.

```
[http]
# Determines whether HTTP endpoint is enabled.
enabled = true

# The bind address used by the HTTP service.
```

Vervolgens kan je je eigen database aanmaken.

```
SERRE@SERRE:~/serre $ influx
Connected to http://localhost:8086 version 1.6.7~rc0
InfluxDB shell version: 1.6.7~rc0
> CREATE DATABASE GreenHouse
> CREATE USER SERRE with password 'SERRE'
> GRANT ALL ON GreenHouse TO SERRE
> exit
```

- CREATE DATABASE “naam_database” → kies je een naam voor de database
- CREATE USER “gebruikersnaam” with password “wachtwoord” → kies gebruiker met wachtwoord voor database. **Belangrijk voor grafana.**
- GRANT ALL ON “naam_database” TO “gebruikersnaam”
- Exit → Verlaat influx

Dit is de koptekst in stijl 'Koptekst'

Vervolgens maken we een brug aan.

```
SERRE@SERRE:~/serre $ nano bridge.py
```

En vul je onderstaande code in:

PAS AAN: influxdb_(adress, user, password, database) en MQTT_(adress, user, password). Bij MQTT_(topci, regex) pas je het 1^{ste} woord aan.

```
C:\ SERRE@SERRE: ~/serre
GNU nano 7.2
import re
from typing import NamedTuple

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

INFLUXDB_ADDRESS = '192.168.1.34'
INFLUXDB_USER = 'SERRE'
INFLUXDB_PASSWORD = 'SERRE'
INFLUXDB_DATABASE = 'GreenHouse'

MQTT_ADDRESS = '192.168.1.34'
MQTT_USER = 'SERRE'
MQTT_PASSWORD = 'SERRE'
MQTT_TOPIC = 'serre/+/+'
MQTT_REGEX = 'serre/([^/]+)/([^\+]+)'
MQTT_CLIENT_ID = 'MQTTInfluxDBBridge'

influxdb_client = InfluxDBClient(INFLUXDB_ADDRESS, 8086, INFLUXDB_USER,
INFLUXDB_PASSWORD, None)

class SensorData(NamedTuple):
    location: str
    measurement: str
    value: float

def on_connect(client, userdata, flags, rc):
    """ The callback for when the client receives a CONNACK response from the
server."""
    print('Connected with result code ' + str(rc))
    client.subscribe(MQTT_TOPIC)

def _parse_mqtt_message(topic, payload):
    match = re.match(MQTT_REGEX, topic)
    if match:
        location = match.group(1)
        measurement = match.group(2)
        if measurement == 'status':
            return None
        return SensorData(location, measurement, float(payload))
    else:
        return None

def _send_sensor_data_to_influxdb(sensor_data):
    json_body = [
        {
            'measurement': sensor_data.measurement,
            'tags': {
                'location': sensor_data.location
            },
            'fields': {
                'value': sensor_data.value
            }
        }
    ]
    influxdb_client.write_points(json_body)

def on_message(client, userdata, msg):
    """The callback for when a PUBLISH message is received from the server."""
    print(msg.topic + ' ' + str(msg.payload))
    sensor_data = _parse_mqtt_message(msg.topic, msg.payload.decode('utf-8'))
    if sensor data is not None:
```

Dit is de koptekst in stijl 'Koptekst'

```
_send_sensor_data_to_influxdb(sensor_data)

def _init_influxdb_database():
    databases = influxdb_client.get_list_database()
    if len(list(filter(lambda x: x['name'] == INFLUXDB_DATABASE, databases))) == 0:
        influxdb_client.create_database(INFLUXDB_DATABASE)
    influxdb_client.switch_database(INFLUXDB_DATABASE)

def main():
    _init_influxdb_database()

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

    mqtt_client.connect(MQTT_ADDRESS, 1883)
    mqtt_client.loop_forever()

if __name__ == '__main__':
    print('MQTT to InfluxDB bridge')
    main()
```

^G Help **^O** Write Out **^W** Where Is **^K** Cut **^T** Execute
^X Exit **^R** Read File **^V** Replace **^U** Paste **^J** Justify **^C** Location
 ^/ Go To Line M-U Undo
 M-E Redo

Nu kan je deze code runnen: (Merk op dat de ESP ook aan moet staan)

```
SERRE@SERRE:~/serre $ python bridge.py
MQTT to InfluxDB bridge
Connected with result code 0
serre/serre/humdht11 b'48.00'
serre/serre/tempdht11 b'25.00'
serre/serre/ds18b20 b'24.63'
serre/serre/bodemvochtsensor b'0.00'
serre/serre/ultrasone b'96.82'
serre/serre/ldr b'0.00'
```

9.6. GRAFANA

Volg het stappenplan op onderstaande website stap voor stap bij installatie GRAFANA:
<https://grafana.com/tutorials/install-grafana-on-raspberry-pi/>

Als dit volledig is gelukt, kan je browsen naar `http://<ip address>:3000` inloggen met gebruikersnaam/wachtwoord admin. Vervolgens kan je deze aanpassen naar eigen keuze.

9.7. Opstarten bij boot:

In onderstaande video wordt dit uitgebreid uitgelegd hoe je dit moet verwezenlijken.
<https://www.youtube.com/watch?v=meeKv3DY9ps>

Dit is de koptekst in stijl 'Koptekst'

