

IOT PROJECT

# Serre met Aanpasbare parameters (RFID)

Thibau Vranken

# Inhoud

<b>1. PROJECTBESCHRIJVING</b>	<b>4</b>
<b>2. HARDWARE</b>	<b>5</b>
2.1. Microcontrollers	5
2.1.1. ESP32	5
2.1.2. RaspberryPi 4B	5
2.2. PCB Hardware	6
2.2.1. 2x 5V relay's	6
2.2.2. 3x PC817 optocouplers	6
2.2.3. 5x BC547B transistors	6
2.2.4. Schroefterminals	7
2.2.5. 5x rode Feedback LED	7
2.2.6. Weerstand	8
2.2.7. Pinheaders	8
2.3. Sensors en Modules	9
2.3.1. DHT11	9
2.3.2. LDR module	9
2.3.3. Bodemvochtigheidssensor	9
2.3.4. DS18B20 sensor	10
2.3.5. RFID module	10
2.3.6. HC SR04 Ultrasonic sensor	11
2.4. Actuatoren	11
2.4.1. 2x 12V ventilatoren	11
2.4.2. Waterpompje	12
2.4.3. Oude soldeerbout	12
2.4.4. Lamp	12
2.4.5. Oude visbak (Serre)	13
<b>3. PCB</b>	<b>13</b>
3.1. Schema	13
3.1.1. Verbindingen sensors	14
3.1.2. Schakelingen Actuatoren	14
3.2. PCB design	16
<b>4. CODE</b>	<b>17</b>
4.1. Code ESP32	17
4.1.1. Volledige code	17
4.1.2. Code uitgelegd	17
4.2. Code RaspberryPi	29
4.2.1. Volledige code	29
4.2.2. Code Uitgelegd	29
<b>5. GRAFANA SERVER</b>	<b>33</b>
5.1. MQTT communicatie	33
5.1.1. Mosquitto installeren	33

5.1.2. Configuratie bestand aanpassen	33
5.1.3. Code maken en kijken of de communicatie goed gebeurt	33
5.2. Server Setup	34
5.2.1. Installeer InfluxDB	34
5.2.2. Maak een database aan	34
5.2.3. Controleer of de data aankomt in de database	35
5.3. InfluxDB connecteren met Grafana voor visualisatie	35
5.3.1. Zoek je Grafana server	35
5.3.2. Log in in Grafana	35
5.3.3. Maak een connectie met je InfluxDB database	36
5.4. Data Visualisatie	38
<b>6. DATASHEETS</b>	<b>38</b>
6.1. ESP 32	38
6.2. RaspberryPi	38
6.3. BC547B Transistor	38
6.4. PC 817 optocoupler	38
6.5. 5V relay	39
6.6. Feedback LED's	39

# 1. Projectbeschrijving

Het project is een slimme serre die zichzelf regelt qua omgevingstemperatuur, bodem- en luchtvochtigheid en licht.

Deze waarden worden gemonitord (actuatoren worden op het juiste moment aangestuurd) door een ESP32 die de waarden doorstuurt naar een RaspberryPi waarna deze de waarden doorstuurt naar de InfluxDB database zodat we op Grafana een mooi overzicht krijgen van de geschiedenis van de waarden.

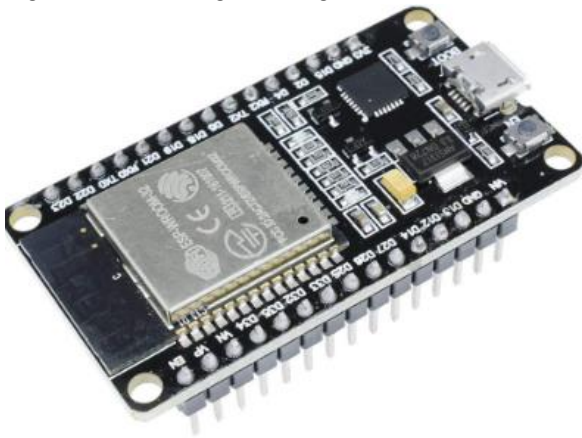
We kunnen de parameters voor alle gegevens aanpassen door RFID tags te scannen die elk vaste parameters hebben. Bijvoorbeeld: Als we de blauwe RFID tag scannen wordt de maximum temperatuur gezet op 25°C (als de temperatuur hoger is dan 25°C dan wordt er aan ventilator aangestuurd), als we dan de witte tag scannen wordt de maximale temperatuur gezet op bijvoorbeeld 20°C (zo gaat de ventilator sneller aan om de temperatuur onder 20°C te houden).

## 2. Hardware

### 2.1. Microcontrollers

#### 2.1.1. ESP32

De ESP32 regelt de hele serre. Het meet de temperatuur van de omgeving en bodem, het meet de lucht- en bodemvochtigheid, of er veel licht is in de omgeving en hoeveel water in het waterreservoir zit (alle sensoren hiervoor worden vermeldt in 'Sensors en Actuatoren'). Het houdt ook rekening met welke RFID tag voor het laatst gescand geweest is zodat alles op het juiste moment wordt aangestuurd.



#### 2.1.2. RaspberryPi 4B

De RaspberryPi ontvangt alle waarden van de ESP32 via MQTT en stuurt deze naar de InfluxDB Database. Hierdoor kunnen we mooi en overzichtelijk de geschiedenis van de waarden zien.



## 2.2. PCB Hardware

### 2.2.1. 2x 5V relay's

Deze zorgen ervoor dat je hoge voltages kan schakelen (240V). Dit is handig voor de lamp (als het te donker is) en de verwarming (als het te koud is).



### 2.2.2. 3x PC817 optocouplers

Met de PC817 kan je veilig schakelingen galvanisch scheiden doordat het een transistor is met een lichtgevoelige basis. De LED die de basis van de transistor moet schakelen wordt gecontroleerd door de ESP32. Op deze manier blijven de stroomkringen van de actuatoren en de ESP32 mooi gescheiden. (Voor toepassing, zie Schakelingen Actuatoren, Lage Voltages)



### 2.2.3. 5x BC547B transistors

Om de relay en de PC817 te kunnen schakelen heb ik de BC547B transistor gebruikt. Ze worden wel anders gebruikt naargelang de voltage die geschakeld moet worden (zie Schakelingen Actuatoren).



## 2.2.4. Schroefterminals

### 2.2.4.1. 2 Schroefterminals

Om de actuatoren en de voedingen voor de actuatoren niet definitief te monteren aan het PCB heb ik 10 keer 2 schroefterminals gebruikt. Zo kun je ook makkelijk veranderingen doen aan de voedingen/actuatoren.



### 2.2.4.2. 3 Schroefterminal

Om de DS18B20 sensor (niet definitief) te kunnen monteren heb ik een 3 schroefterminal gebruikt.



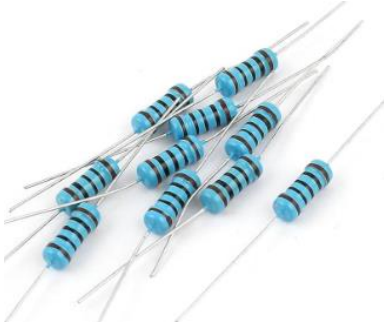
## 2.2.5. 5x rode Feedback LED

Om duidelijk te kunnen zien welke actuatoren aan zijn (welke schakeling geschakeld is), heb ik in elke schakeling een rode feedback LED geplaatst zodat je aan de LED kunt zien wat 'aan' is en wat 'uit' is.



### 2.2.6. Weerstanden

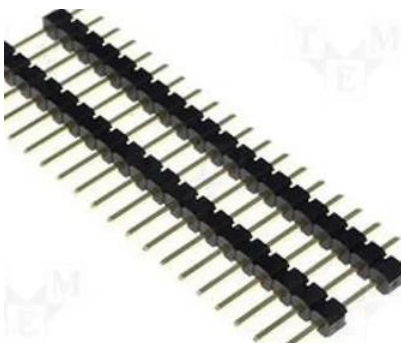
Weerstanden heb ik gebruikt om de juiste stroom/voltage vereisten te krijgen voor elke schakeling (zie PCB Design voor juiste waarden).



### 2.2.7. Pinheaders

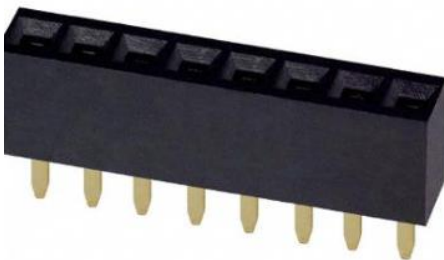
#### 2.2.7.1. Male headers

Om de sensoren te kunnen verbinden met het PCB heb ik male headers gebruikt om zo via jumpercables de sensor op de juiste plaats te krijgen.



#### 2.2.7.2. Female Headers

Om modules snel te kunnen vervangen en niet definitief aan het PCB te moeten hangen heb ik female headers gebruikt.





## 2.3. Sensors en Modules

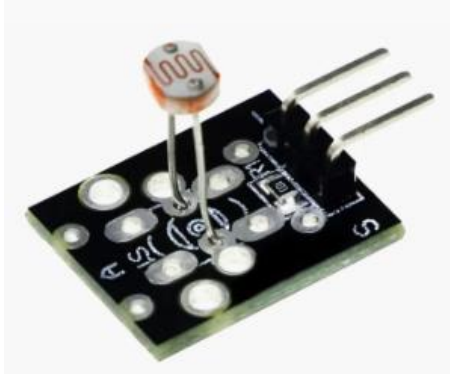
### 2.3.1. DHT11

Om de omgevingstemperatuur en luchtvochtigheid te kunnen meten en monitoren heb ik de DHT11 sensor gebruikt.



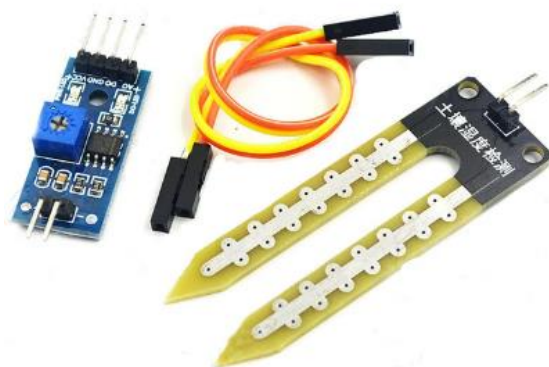
### 2.3.2. LDR module

Om het licht in de omgeving te meten heb ik de LDR module gebruikt.



### 2.3.3. Bodemvochtigheidssensor

Om de bodemvochtigheid te kunnen meten heb ik deze bodemvochtigheidssensor gebruikt.



#### 2.3.4. DS18B20 sensor

Met de DS18B20 sensor kan ik de bodemtemperatuur meten.



#### 2.3.5. RFID module

Om de parameters aan te kunnen passen naargelang het gewas/plant er in de serre staat kan je je RFID tags scannen op de RFID module. Elke tag heeft vaste vooraf ingestelde parameters (je kan deze aanpassen in je code). Zo kun je snel en makkelijk het systeem van je serre veranderen.



### 2.3.6. HC SR04 Ultrasonic sensor

De HC SR04 sensor kan zeer gedetailleerd afstanden meten. Ik heb deze boven het wateroppervlak van het waterreservoir geplaatst zodat je de inhoud van het waterreservoir kan meten en monitoren.



## 2.4. Actuatoren

### 2.4.1. 2x 12V ventilatoren

Om de luchtvochtigheid en temperatuur te regelen van de serre heb ik 2 ventilatoren gebruikt. Beide heb ik ze in het plafond van de serre geplaatst. De ene ventilator heb ik zo gemonteerd zodat deze lucht in de serre blaast voor als het te warm is, en de andere heb ik zo gemonteerd zodat deze lucht uit de serre haalt zodat de lucht kan opdrogen voor als deze te nat is.



### 2.4.2. Waterpompje

Om water te kunnen geven aan de planten als de bodem te droog is, heb je natuurlijk een waterpompje nodig. Ik heb daarvoor deze gebruikt:



### 2.4.3. Oude soldeerbout

Als verwarmingselement heb ik een oude soldeerbout gebruikt die ik nog thuis had liggen. Deze heb ik ook door het plafond naar binnen gehangen zodat de warmte mooi verdeeld word in de serre.



### 2.4.4. Lamp

Als het te donker is moet er natuurlijk licht gemaakt worden. Hiervoor heb ik een gewone lamp gebruikt.



Dit is de koptekst in stijl 'Koptekst'

#### 2.4.5. Oude visbak (Serre)

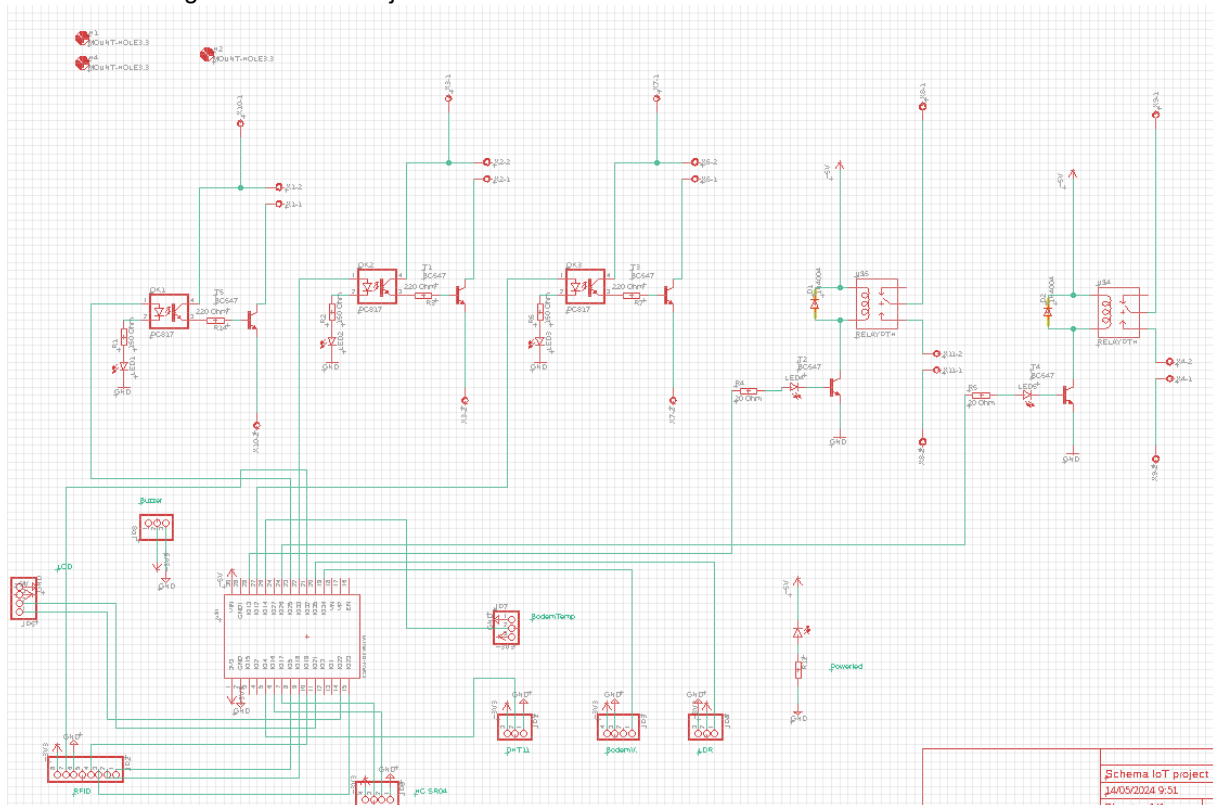
Als serre heb ik een oude visbak gebruikt die ik thuis had liggen. Ik heb er dan een plafond voor gezaagd uit hout zodat de serre echt gesloten is.



### 3. PCB

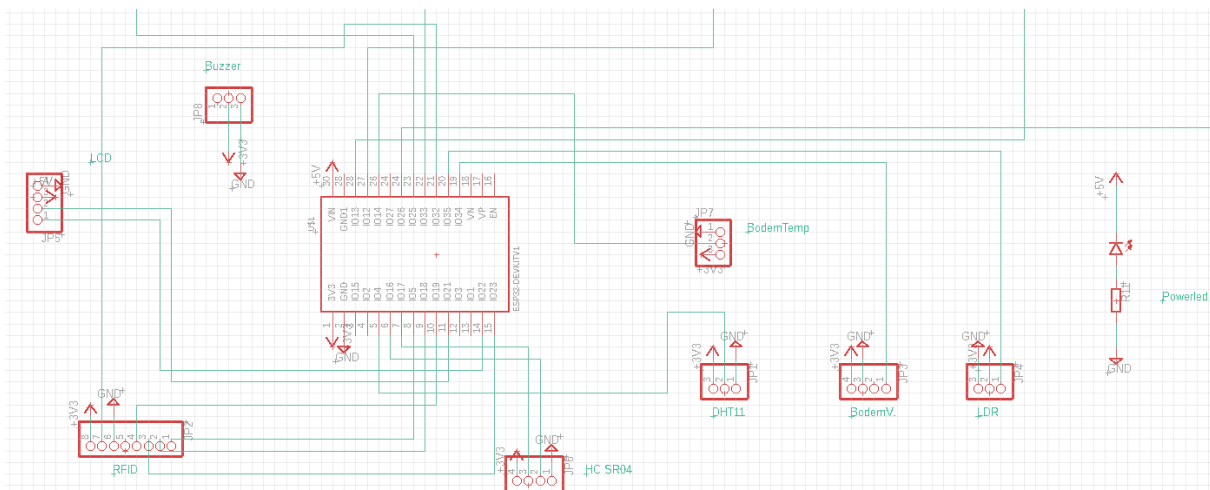
### 3.1. Schema

Dit is het volledige schema van mijn PCB.



Ik ga er wat verder op in gaan.

### 3.1.1. Verbindungen sensors



Op deze foto zie je de verbindingen die ik gemaakt heb met de sensors en de ESP32. Over het algemeen is er niets speciaal hier. Alle sensoren zijn verbonden met de 3,3V power supply van de ESP32 en zijn dan ook verbonden met de pinnen die hun zijn toegewezen (zie Code). Rechts op de foto zie je een LED staan, deze is enkel ter controle of er wel spanning staat op de relay's en de transistoren (zie Schakelingen Actuatoren).

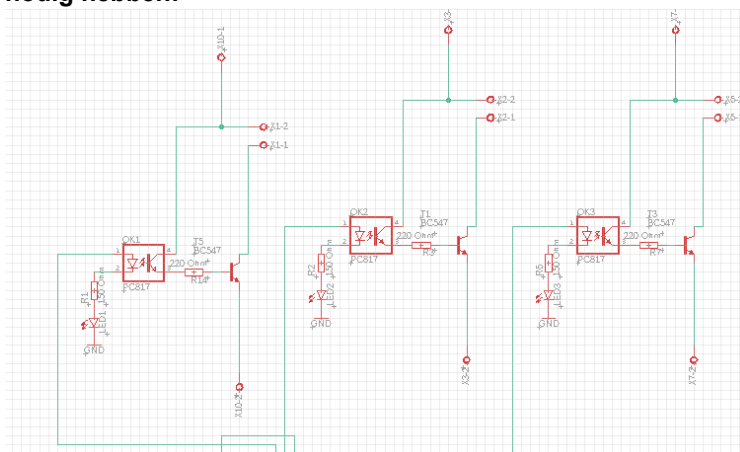
### 3.1.2. Schakelingen Actuatoren

#### 3.1.2.1. Lage voltages (Max 45V (zie datasheet BC547B transistor))

Om lage voltages te schakelen heb ik dus gekozen voor een schakeling met de BC547B transistor en de PC817 optocoupler. Deze gaan (normaal gezien) langer mee dan standaard relay's maar kunnen dus geen hoge voltages schakelen.

Uitleg schakeling:

Om de PC817 te activeren (dus de fototransistor laten geleiden) moet de bepaalde pin van de ESP32 (zie Code) 'Hoog' staan. Zo vloeit er stroom door de LED in de PC817, wordt de transistor geactiveerd en is het circuit open. (Na de LED van de PC817 heb ik zelf nog een rode LED geplaatst om te zien of de schakeling weldegelijk geschakeld is.) Aan de andere kant (kant van fototransistor) vloeit de stroom van de actuator, door dat de foto transistor 'aan' staat. Bij de aansluitingen boven en onder aan elke schakeling worden respectievelijk de positieve en de negatieve polen van de voeding aangesloten. Bij de aansluitingen in het midden van elke schakeling worden dan, ook van boven naar onder, de positieve en de negatieve kant van de actuator aangesloten. **Deze schakelingen gelden dus enkel voor actuatoren die minder dan 45V nodig hebben!**

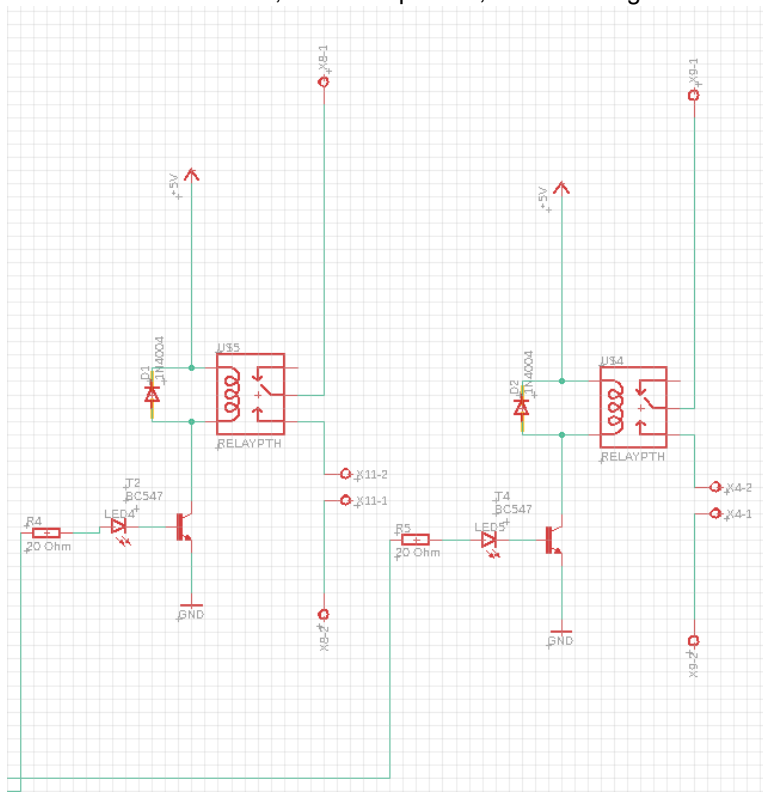


### 3.1.2.2. Hoge voltages (Max 240V AC)

Om de hoge voltages te kunnen schakelen heb ik dus standaard relay's gebruikt. Deze gaan over het algemeen minder lang mee dan transistoren en optocouplers maar kunnen dus wel grote voltages (AC en DC) schakelen. Nog een nadeel is dat ze best wel wat plek innemen op je PCB.

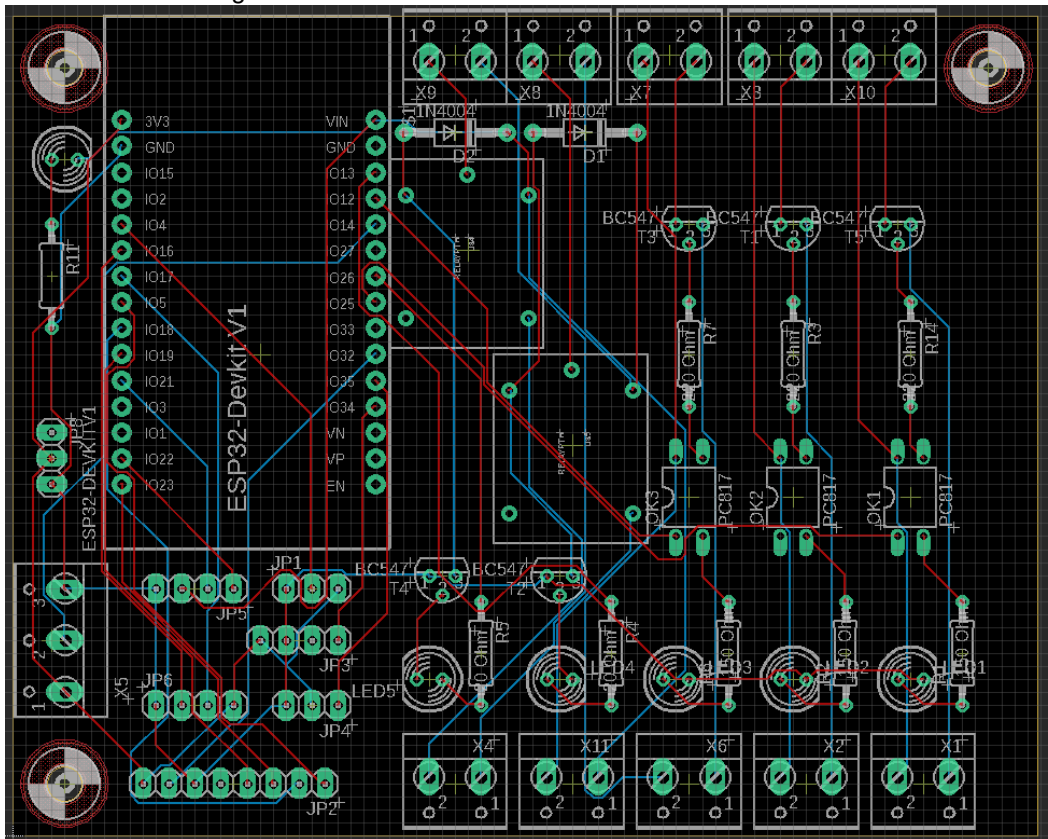
Uitleg Schakeling:

Bij deze schakeling zorgt de BC547B transistor voor de schakeling van de relay (anders dan in de vorige schakeling). De BC547B transistor heb ik geplaatst zodat deze de stroomgang van de 5V kant van de relay kan controleren. Als de bepaalde pin van de ESP32 (zie Code) 'hoog' staat dan gaat deze op de basis van de transistor en begint deze te geleiden waardoor de relay schakelt. Ook hier heb ik een LED geplaatst zodat je kan zien of de schakeling 'aan' staat (hier op de basis van de transistor). De diode die ik heb geplaatst over de spoel van de relay is een flyback diode, hierdoor wordt de spanningspiek geëlimineerd als de relay 'uit' gaat. De voeding sluit je ook hier aan zoals bij de ander schakelingen, positieve kant bovenaan, negatieve kant onderaan. De actuatoren zij verbonden met de NO pin van de relay (Normally Open) zodat deze alleen werken als de relay geschakeld is. De actuatoren sluit je hier ook aan de middelste terminals aan, bovenste positief, onderste negatief.



## 3.2. PCB design

Hier is het PCB design van het schema:



De ESP32, de RFID module en de buzzer sluit ik aan aan de printplaat via de 'female pinheaders' zodat je ze gemakkelijk kan veranderen/afhalen voor ander project.

De sensors en LCD verbindt ik via female-female jumpercables zodat ik ze op de plaatst kan leggen waar ze moeten liggen/hangen.

**Zorg ervoor dat je de aansluitingen van de actuatoren en voedingen aan de zijkant zet zodat je er gemakkelijk aan kunt.**

(Wegens een fout in de library van de ESP32 kon ik GPIO 27 niet gebruiken in mijn schema, daarom zou ik deze lijn zelf nog moeten leggen met een kabeltje.



## 4. Code

### 4.1. Code ESP32

#### 4.1.1. Volledige code

Om de volledige code te zien, zie dan bestand 'Serre.ino'.

#### 4.1.2. Code uitgelegd

Deze code doet eigenlijk bijna alles. Het meet sensorwaarden, het kijkt wanneer er een RFID tag gelezen moet worden, wanneer de parameters moeten worden aangepast, regelt het LCD scherm en stuurt al de sensorwaarden door naar de broker.

##### 4.1.2.1. Libraries

```
#include "DHT.h" // Library for DHT sensors
#include <WiFi.h> // Library for WiFi functionality
#include <PubSubClient.h> // Library for MQTT functionality
#include "LiquidCrystal_I2C.h" // Library for I2C LCD
#include "Wire.h" // Library for I2C communication
#include <WiFiClientSecure.h> // Library for secure WiFi client
#include <MFRC522.h> // Library for RFID functionality
#include <afstandssensor.h> // Custom library for distance sensor
#include <UniversalTelegramBot.h> // Library for Telegram bot
#include <ArduinoJson.h> // Library for JSON parsing
#include <SPI.h> // Library for SPI communication
#include <OneWire.h> // Library for 1-Wire communication
#include <DallasTemperature.h> // Library for Dallas Temperature
sensors
#include <Preferences.h> // Library for storing preferences
```

##### 4.1.2.2. DHT type vastzetten, pinnen voor sensoren, actuatoren

```
//Pinnen voor actuatoren en sensoren
//DHT type vastzetten
#define VentilTemp 25
#define VentilHum 33
#define Pomp 12
#define Verwarming 13
#define Lamp 26
#define ldrPin 35
#define bvPin 34
#define btPin 14
#define RST_PIN 32
#define SS_PIN 5
```

Dit is de koptekst in stijl 'Koptekst'

```
#define DHTPIN 4
#define DHTTYPE DHT11
```

#### 4.1.2.3. Wifi/MQTT Instanties en Chatbot definiëren

```
#define CHAT_ID "...
#define BOTtoken "...

WiFiClientSecure client;
WiFiClient espClient;
PubSubClient mqttClient(espClient);
Preferences preferences;
```

#### 4.1.2.4. WiFi en MQTT credentials

```
//WiFi credentials
const char* ssid = "...";
const char* password = "...";
//MQTT credentials
const char* mqtt_server = "...";
const int mqtt_port = 1883;
const char* MQTT_USER = "...";
const char* MQTT_PASSWORD = "...";
const char* MQTT_CLIENT_ID = "MQTTclient";
```

#### 4.1.2.5. MQTT topics aanmaken voor alle waarden

```
//MQTT topics aanmaken voor elke waarde
//Waarden
const char* MQTT_TOPICtemp = "serre/waarden/temperature";
const char* MQTT_TOPIChum = "serre/waarden/humidity";
const char* MQTT_TOPIClight = "serre/waarden/light";
const char* MQTT_TOPICbv = "serre/waarden/bodemvocht";
const char* MQTT_TOPICbt = "serre/waarden/bodemtemp";
const char* MQTT_TOPICWater = "serre/waarden/waterres";
//Actuatoren
const char* MQTT_TOPICVentilTempTijd = "serre/actuatoren/VentilTempDraaitijd";
const char* MQTT_TOPICVentilHumTijd = "serre/actuatoren/VentilHumDraaitijd";
const char* MQTT_TOPICPompTijd = "serre/actuatoren/PompDraaitijd";
const char* MQTT_TOPICLichtTijd = "serre/actuatoren/LampDraaitijd";
const char* MQTT_TOPICVerwTijd = "serre/actuatoren/VerwDraaitijd";
//Parameters
const char* MQTT_TOPICParameterMaxTemp = "serre/parameters/MaxTemp";
const char* MQTT_TOPICParameterMinTemp = "serre/parameters/MinTemp";
const char* MQTT_TOPICParameterMaxHum = "serre/parameters/MaxHum";
const char* MQTT_TOPICParameterMinLight = "serre/parameters/MinLight";
```

```
const char* MQTT_TOPICParameterMinBv = "serre/parameters/MinBv";

const char* MQTT_TOPICCommunication = "serre/communication/communication";
const char* MQTT_REGEX = "serre/([^\s]+)/([^\s]+)";
```

#### 4.1.2.6. Alle instanties aanmaken

```
LiquidCrystal_I2C lcd(0x27, 16, 4);
DHT dht(DHTPIN, DHTTYPE);
OneWire oneWire(btPin);
DallasTemperature sensors(&oneWire);
AfstandsSensor afstandssensor(17, 16);
UniversalTelegramBot bot(BOTtoken, client);
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

#### 4.1.2.7. Variabelen en constanten definiëren

```
unsigned long lastDisplayUpdateTime = 0;
const unsigned long displayUpdateInterval = 5000; // Interval van 5 seconden
voor het bijwerken van het display
const unsigned long waterUpdateInterval = 10000;
unsigned long lastWaardenUpdateTime = 0;
const unsigned long waardenUpdateInterval = 10000; // Interval van 5 seconden
voor het bijwerken van het display
unsigned long lastParametersUpdateTime = 0;
const unsigned long parametersUpdateInterval = 10000;
unsigned long lastActuatorsUpdateTime = 0;
const unsigned long actuatorsUpdateInterval = 3000;
unsigned long lastCommunicationUpdateTime = 0;
const unsigned long communicationUpdateInterval = 5000;
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;
bool newRFIDScan = false;
bool displaySensorValues = true;
```

#### 4.1.2.8. Standaard waarden voor parameters instellen en float maken voor waterreservoir te meten

```
int maxTemp = 24; // Maximale temperatuur standaard instelling
int minTemp = 18; // Minimale temperatuur standaard instelling
int maxHum = 60; // Maximale luchtvochtigheid standaard instelling
int minLight = 50; // Minimale lichtintensiteit standaard instelling
int minBodemvocht = 30; // Minimale bodemvochtigheid standaard instelling
float water;
TaskHandle_t Task1;
```

#### 4.1.2.9. *Funcctie om te verbinden met WiFi*

```
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Verbinding maken met WiFi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi verbonden");
    Serial.println("IP adres: ");
    Serial.println(WiFi.localIP());
}
```

#### 4.1.2.10. *Funcctie om te verbinden met MQTT broker*

```
void reconnect() {
    while (!mqttClient.connected()) {
        Serial.print("Verbinding maken met MQTT-broker...");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Verbinden met");
        lcd.setCursor(0, 1);
        lcd.print("MQTT broker...");
        if (mqttClient.connect(MQTT_CLIENT_ID, MQTT_USER, MQTT_PASSWORD)) {
            Serial.println(" verbonden");
            mqttClient.subscribe(MQTT_TOPICCommunication);
        } else {
            Serial.print(" mislukt, rc=");
            Serial.print(mqttClient.state());
            Serial.println(" opnieuw proberen in 5 seconden");
            delay(5000);
        }
    }
}
```

#### 4.1.2.11. *Funcctie die de sensorwaarden leest, de sensorwaarden en de parameters verstuurd via MQTT en het LCD scherm regelt*

```
void clientPubTask(void* pvParameters) {
    while (true) {
        if (!mqttClient.connected()) {
            reconnect();
        }
    }
}
```

```
}
sensors.requestTemperatures();
float h = dht.readHumidity();
float t = dht.readTemperature();
float l = map(analogRead(ldrPin), 0, 4096, 100, 0);
float b = map(analogRead(bvPin), 0, 4096, 100, 0);
float bt = sensors.getTempCByIndex(0);

unsigned long currentTime1 = millis();
if (currentTime1 - lastActuatorsUpdateTime >= actuatorsUpdateInterval) {
    if (t >= maxTemp) {
        digitalWrite(VentilTemp, HIGH);
        digitalWrite(Verwarming, LOW);
        mqttClient.publish(MQTT_TOPICVentilTempTijd, static_cast<const
char*>("1"));
        mqttClient.publish(MQTT_TOPICVerwTijd, static_cast<const char*>("0"));
    } else if (t >= minTemp && t < maxTemp) {
        digitalWrite(VentilTemp, LOW);
        digitalWrite(Verwarming, LOW);
        mqttClient.publish(MQTT_TOPICVentilTempTijd, static_cast<const
char*>("0"));
        mqttClient.publish(MQTT_TOPICVerwTijd, static_cast<const char*>("0"));

    } else if (t < minTemp) {
        digitalWrite(Verwarming, HIGH);
        digitalWrite(VentilTemp, LOW);
        mqttClient.publish(MQTT_TOPICVerwTijd, static_cast<const char*>("1"));
        mqttClient.publish(MQTT_TOPICVentilTempTijd, static_cast<const
char*>("0"));
    }
    if (h >= maxHum) {
        digitalWrite(VentilHum, HIGH);
        mqttClient.publish(MQTT_TOPICVentilHumTijd, static_cast<const
char*>("1"));
    } else {
        digitalWrite(VentilHum, LOW);
        mqttClient.publish(MQTT_TOPICVentilHumTijd, static_cast<const
char*>("0"));
    }
    if (l <= minLight) {
        digitalWrite(Lamp, HIGH);
        mqttClient.publish(MQTT_TOPICLichtTijd, static_cast<const
char*>("1"));
    } else {
        digitalWrite(Lamp, LOW);
        mqttClient.publish(MQTT_TOPICLichtTijd, static_cast<const
char*>("0"));
    }
}
```

```
}
if (b <= minBodemvocht) {
    digitalWrite(Pomp, HIGH);
    mqttClient.publish(MQTT_TOPICPompTijd, static_cast<const char*>("1"));
} else {
    digitalWrite(Pomp, LOW);
    mqttClient.publish(MQTT_TOPICPompTijd, static_cast<const char*>("0"));
}
Serial.println("status verzonden");
lastActuatorsUpdateTime = currentTime1;
}
unsigned long currentTime2 = millis();
if (currentTime2 - lastWaardenUpdateTime >= waardenUpdateInterval) {
    String ts = String(t);
    String hs = String(h);
    String ls = String(l);
    String bs = String(b);
    String bts = String(bt);
    mqttClient.publish(MQTT_TOPICtemp, ts.c_str());
    mqttClient.publish(MQTT_TOPICHum, hs.c_str());
    mqttClient.publish(MQTT_TOPIClight, ls.c_str());
    mqttClient.publish(MQTT_TOPICbv, bs.c_str());
    mqttClient.publish(MQTT_TOPICbt, bts.c_str());
    //Serial.println("waarden verzonden");
    lastWaardenUpdateTime = currentTime2;
}

unsigned long currentTime3 = millis();
if (currentTime3 - lastParametersUpdateTime >= parametersUpdateInterval) {
    sendParameters();
    lastParametersUpdateTime = currentTime3;
}

if (displaySensorValues) {
    lcd.setCursor(0, 0);
    lcd.print("Omgevingsinfo:");
    lcd.setCursor(0, 1);
    lcd.print("Temp.:");
    lcd.print(t);
    lcd.print(" C");
    lcd.setCursor(-4, 2);
    lcd.print("Hum. :");
    lcd.print(h);
    lcd.print("%");
    lcd.setCursor(-4, 3);
    lcd.print("Licht:");
    lcd.print(l);
}
```

```
    lcd.print("%");
} else {
    lcd.setCursor(0, 0);
    lcd.print("Bodeminfo:");
    lcd.setCursor(0, 1);
    lcd.print("BodemV.:");
    lcd.print(b);
    lcd.print("%");
    lcd.setCursor(-4, 2);
    lcd.print("BodemT.:");
    lcd.print(bt);
    lcd.print(" C");
    lcd.setCursor(-4, 3);
}

unsigned long currentTime4 = millis();
if (currentTime4 - lastDisplayUpdateTime >= displayUpdateInterval) {
    lastDisplayUpdateTime = currentTime4;
    displaySensorValues = !displaySensorValues;
    lcd.clear();
}

if (newRFIDScan) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("New Parameters ");
    lcd.setCursor(0, 1);
    lcd.print("MaxTemp = ");
    lcd.print(maxTemp);
    lcd.print(" C");
    lcd.setCursor(-4, 2);
    lcd.print("MinTemp = ");
    lcd.print(minTemp);
    lcd.print(" C");
    lcd.setCursor(-4, 3);
    lcd.print("MaxHum = ");
    lcd.print(maxHum);
    lcd.print(" %");
    delay(5000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("New Parameters ");
    lcd.setCursor(0, 1);
    lcd.print("MinLicht = ");
    lcd.print(minLight);
    lcd.print("%");
    lcd.setCursor(-4, 2);
```

```
        lcd.print("MinBodemV. = ");
        lcd.print(minBodemvocht);
        lcd.print("%");
        delay(5000);
        newRFIDScan = false; // Reset de nieuwe RFID-scan status
    }
}
}
```

#### 4.1.2.12. Functie die wordt aangeroepen als de ESP32 een MQTT bericht ontvangt

```
void callback(char* topic, byte* payload, unsigned int length) {
    unsigned long currentTime1 = millis();
    if (currentTime1 - lastCommunicationUpdateTime >=
communicationUpdateInterval) {
        Serial.print("Message arrived [");
        Serial.print(topic);
        Serial.print("] ");
        Serial.print("Message: ");
        String bericht;
        for (int i = 0; i < length; i++) {
            bericht += ((char)payload[i]);
        }
        if (String(topic) == MQTT_TOPICCommunication) {
            if (bericht == "hallo") {
                Serial.println(bericht);
            }
        }
        lastCommunicationUpdateTime = currentTime1;
    }
}
```

#### 4.1.2.13. Setup functie

```
void setup() {
    Serial.begin(9600);
    pinMode(25, OUTPUT);
    pinMode(33, OUTPUT);
    pinMode(12, OUTPUT);
    pinMode(13, OUTPUT);
    pinMode(26, OUTPUT);
    pinMode(27, OUTPUT);
    pinMode(ldrPin, INPUT);
    pinMode(bvPin, INPUT);
    pinMode(btPin, INPUT);
}
```



```
dht.begin();
setup_wifi();
mqttClient.setServer(mqtt_server, mqtt_port);
mqttClient.setCallback(callback);
reconnect();

lcd.init();
lcd.backlight();
Wire.begin(SDA, SCL);
SPI.begin();
mfrc522.PCD_Init();
sensors.begin();
client.setCACert(TELEGRAM_CERTIFICATE_ROOT);

preferences.begin("serre", false); // Beginnen met de voorkeuren met de
naam "serre"
// Lezen van opgeslagen waarden of standaardwaarden gebruiken als er geen
opgeslagen waarden zijn
maxTemp = preferences.getInt("maxTemp", 24);
minTemp = preferences.getInt("minTemp", 18);
maxHum = preferences.getInt("maxHum", 60);
minLight = preferences.getInt("minLight", 50);
minBodemvocht = preferences.getInt("minBodemvocht", 30);

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Verbonden met");
lcd.setCursor(0, 1);
lcd.print("MQTT broker");
delay(3000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Used Parameters ");
lcd.setCursor(0, 1);
lcd.print("MaxTemp = ");
lcd.print(maxTemp);
lcd.print(" C");
lcd.setCursor(-4, 2);
lcd.print("MinTemp = ");
lcd.print(minTemp);
lcd.print(" C");
lcd.setCursor(-4, 3);
lcd.print("MaxHum = ");
lcd.print(maxHum);
lcd.print(" %");
delay(5000);
lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("Used Parameters ");
lcd.setCursor(0, 1);
lcd.print("MinLicht   = ");
lcd.print(minLight);
lcd.print("%");
lcd.setCursor(-4, 2);
lcd.print("MinBodemV. = ");
lcd.print(minBodemvocht);
lcd.print("%");
delay(5000);

xTaskCreatePinnedToCore(
    clientPubTask,
    "clientPubTask",
    10000,
    NULL,
    1,
    NULL,
    0);
}
```

#### 4.1.2.14. Functie die de RFID tags leest

```
void RFIDtagLezen() {
    static String Tag = ""; // Variabele om de vorige RFID-tag op te slaan
    if (!mfr522.PICC_IsNewCardPresent()) {
        return;
    }
    // Select one of the cards
    if (!mfr522.PICC_ReadCardSerial()) {
        return;
    }
    Serial.print("UID tag :");
    String content = "";
    for (byte i = 0; i < mfr522.uid.size; i++) {
        Serial.print(mfr522.uid.uidByte[i] < 0x10 ? " 0" : " ");
        Serial.print(mfr522.uid.uidByte[i], HEX);
        content.concat(String(mfr522.uid.uidByte[i] < 0x10 ? "0" : ""));
        content.concat(String(mfr522.uid.uidByte[i], HEX));
    }
    Serial.println();
    Serial.print("Message : ");
    content.toUpperCase();
    Serial.println(content.substring(1));

    Tag = content.substring(1);
}
```

```
if (Tag == "...") {
    // Parameters voor blauwe RFID-tag
    maxTemp = 26;
    minTemp = 20;
    maxHum = 75;
    minLight = 60;
    minBodemvocht = 35;
} else if (Tag == "...") {
    // Parameters voor witte RFID-tag
    maxTemp = 22;
    minTemp = 16;
    maxHum = 65;
    minLight = 40;
    minBodemvocht = 30;
}
/* Nieuwe parameters direct opslaan zodat
deze opnieuw worden a-ingesteld als de ESP reset*/
preferences.putInt("maxTemp", maxTemp);
preferences.putInt("minTemp", minTemp);
preferences.putInt("maxHum", maxHum);
preferences.putInt("minLight", minLight);
preferences.putInt("minBodemvocht", minBodemvocht);
sendParameters();
newRFIDScan = true;
tone(27, 1000);
delay(1000);
noTone(27);
}
```

#### 4.1.2.15. 3 functies om de Telegram chatbot te laten werken

```
String getWater() {
    water = afstandssensor.afstandCM();
    Serial.print("Afstand: ");
    Serial.print(water);
    Serial.println("cm");
    String message = "Afstand: " + String(water) + "cm \n";
    return message;
    bot.sendMessage(CHAT_ID, message, "");
}

void handleNewMessages(int numNewMessages) {
    Serial.println("handleNewMessages");
    Serial.println(String(numNewMessages));
    for (int i = 0; i < numNewMessages; i++) {
        String chat_id = String(bot.messages[i].chat_id);
```

```
if (chat_id != CHAT_ID) {
    bot.sendMessage(chat_id, "Unauthorized user", "");
    continue;
}
String text = bot.messages[i].text;
Serial.println(text);
String from_name = bot.messages[i].from_name;
if (text == "/start") {
    String welcome = "Welcome, " + from_name + ".\n";
    welcome += "Use the following command to get current readings.\n\n";
    welcome += "/readings \n";
    bot.sendMessage(chat_id, welcome, "");
}
if (text == "/readings") {
    String readings = getWater();
    bot.sendMessage(chat_id, readings, "");
}
}
}

void waterReservoir() {
    static unsigned long lastWaterUpdate = 0;
    unsigned long currentTime = millis();
    // Controleer of er 10 seconden verstreken zijn sinds de laatste keer dat de
    waarde 'water' is verzonden
    if (currentTime - lastWaterUpdate >= waterUpdateInterval) {
        float water = afstandssensor.afstandCM();
        // Verstuur de waarde 'water' als deze meer dan 10 cm is
        if (water > 5 && water < 400) {
            String waters = String(water);
            Serial.println(water);
            mqttClient.publish(MQTT_TOPICWater, waters.c_str());
            lastWaterUpdate = currentTime;
            /* Stuurt een waarschuwingsbericht als het
            waterreservoir bijna leeg is*/
            if (water < 10) {
                String alertMessage = "Waarschuwing: Waterreservoir is bijna leeg!";
                bot.sendMessage(CHAT_ID, alertMessage, "");
                lastWaterUpdate = currentTime; // Update de tijd van de laatste
                verzending
            }
            // Handel de bot-updates af zoals eerder
            if (millis() > lastTimeBotRan + botRequestDelay) {
                int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
                while (numNewMessages) {
                    Serial.println("got response");
                    handleNewMessages(numNewMessages);
                }
            }
        }
    }
}
```

```
        numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
}
}
}
```

#### 4.1.2.16. Functie om parameters te versturen

```
void sendParameters() {
    String maxTemps = String(maxTemp);
    String minTemps = String(minTemp);
    String maxHums = String(maxHum);
    String minLights = String(minLight);
    String minBodemvochts = String(minBodemvocht);
    mqttClient.publish(MQTT_TOPICParameterMaxTemp, maxTemps.c_str());
    mqttClient.publish(MQTT_TOPICParameterMinTemp, minTemps.c_str());
    mqttClient.publish(MQTT_TOPICParameterMaxHum, maxHums.c_str());
    mqttClient.publish(MQTT_TOPICParameterMinLight, minLights.c_str());
    mqttClient.publish(MQTT_TOPICParameterMinBv, minBodemvochts.c_str());
}
```

#### 4.1.2.17. Loop functie

```
void loop() {
    RFIDtagLezen();
    waterReservoir();
    mqttClient.loop();
}
```

## 4.2. Code RaspberryPi

Deze code zorgt alleen voor het kunnen gebruiken van een database en deze gebruiken om data te gaan visualiseren op Grafana.

### 4.2.1. Volledige code

Om de volledige code te zien, zie dan bestand 'IoTproject.py'.

### 4.2.2. Code Uitgelegd

#### 4.2.2.1. Libraries toevoegen

```
import re
from typing import NamedTuple
import threading
```

Dit is de koptekst in stijl 'Koptekst'

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

#### 4.2.2.2. Variabelen om connectie aan te geven

```
first_connection_made = False
```

#### 4.2.2.3. InfluxDB en MQTT instellingen

```
# InfluxDB instellingen
INFLUXDB_ADDRESS = 'localhost'
INFLUXDB_USER = 'thibau'
INFLUXDB_PASSWORD = 'test'
INFLUXDB_DATABASE = 'IoTproject'

# MQTT instellingen
MQTT_ADDRESS = 'localhost'
MQTT_USER = 'test'
MQTT_PASSWORD = 'test'
MQTT_TOPICwaarden = "serre/waarden/+"
MQTT_TOPICactuatoren = "serre/actuatoren/+"
MQTT_TOPICparameters = "serre/parameters/+"
MQTT_TOPICcommunication = "serre/communication/communication"
MQTT_REGEX = "serre/([^\s]+)/([^\s/]+)"
MQTT_CLIENT_ID = 'MQTTInfluxDBBridge'

influxdb_client = InfluxDBClient(INFLUXDB_ADDRESS, 8086, INFLUXDB_USER,
INFLUXDB_PASSWORD, INFLUXDB_DATABASE)

previous_sensor_values = {}
```

#### 4.2.2.4. Bepalen hoe MQTT berichten binnen komen

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

#### 4.2.2.5. Functie om te verbinden met MQTT clients en subscriben op topics

```
def on_connect(client, userdata, flags, rc):
    global first_connection_made
    if not first_connection_made:
        print('Connected with result code ' + str(rc))
        first_connection_made = True
    client.subscribe(MQTT_TOPICwaarden)
    client.subscribe(MQTT_TOPICactuatoren)
    client.subscribe(MQTT_TOPICparameters)
```

#### 4.2.2.6. Functie om te zien of de waarden die binnen komt een getal is

```
def _is_valid_number(value):  
    try:  
        float_value = float(value)  
        return not (float_value != float_value) # Checks for NaN  
    except ValueError:  
        return False
```

#### 4.2.2.7. Functie om berichten te parseren

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

#### 4.2.2.8. Functie om data door te sturen naar database

```
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)
```

#### 4.2.2.9. Functie om berichten terug te sturen via MQTT

```
def send_mqtt_signal(client, topic, message):  
    client.publish(topic, message)
```

#### 4.2.2.10. Functie om binnengekomen berichten af te handelen

```
def on_message(client, userdata, msg):  
    print(msg.topic + ' ' + str(msg.payload))  
    sensor_data = _parse_mqtt_message(msg.topic, msg.payload.decode('utf-8'))  
    if sensor_data is not None:  
        _send_sensor_data_to_influxdb(sensor_data)  
        send_mqtt_signal(client, MQTT_TOPICCommunication, "hallo")
```

#### 4.2.2.11. Functie om te bepalen naar welke database de data moet worden gestuurd

```
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)
```

#### 4.2.2.12. MQTT-loop functie

```
def mqtt_loop():
    mqtt_client =
mqtt.Client(mqtt.CallbackAPIVersion.VERSION1, MQTT_CLIENT_ID)
    mqtt_client.username_pw_set(MQTT_USER, MQTT_PASSWORD)
    mqtt_client.on_connect = on_connect
    mqtt_client.on_message = on_message
    print("-----")

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

#### 4.2.2.13. Main functie

```
def main():
    _init_influxdb_database()

    mqtt_thread = threading.Thread(target=mqtt_loop)
    mqtt_thread.daemon = True
    mqtt_thread.start()

    while True:
        try:
            sleep(1)
        except KeyboardInterrupt:
            print("Program stopped by user")
            break
```

#### 4.2.2.14. Hoofd functie

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



## 5. Grafana Server

### 5.1. MQTT communicatie

#### 5.1.1. Mosquitto installeren

Om MQTT communicatie mogelijk te maken moet je eerst 'mosquitto' installeren. Dit doe je zo:

- `sudo apt-get update`
- `sudo apt-get install mosquitto mosquitto-clients`
- `sudo systemctl start mosquitto`
- `sudo systemctl enable mosquitto`

#### 5.1.2. Configuratie bestand aanpassen

```
GNU nano 5.4 /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

listener 1883
allow_anonymous true
```

- Listener op "1883" zetten
- Allow\_anonymous "true" zetten

#### 5.1.3. Code maken en kijken of de communicatie goed gebeurt

Hierbij een voorbeeld van hoe de geslaagde communicatie er uit kan zien:

Dit is de koptekst in stijl 'Koptekst'

```
serre/actuatoeren/LampDraaitijd b'0'  
serre/actuatoeren/PompDraaitijd b'1'  
serre/actuatoeren/VentilTempDraaitijd b'0'  
serre/actuatoeren/VerwDraaitijd b'0'  
serre/actuatoeren/VentilHumDraaitijd b'0'  
serre/actuatoeren/LampDraaitijd b'0'  
serre/actuatoeren/PompDraaitijd b'1'  
serre/actuatoeren/VentilTempDraaitijd b'0'  
serre/actuatoeren/VerwDraaitijd b'0'  
serre/actuatoeren/VentilHumDraaitijd b'0'  
serre/actuatoeren/LampDraaitijd b'0'  
serre/actuatoeren/PompDraaitijd b'1'  
serre/waarden/waterres b'26.39'  
serre/actuatoeren/VentilTempDraaitijd b'0'  
serre/actuatoeren/VerwDraaitijd b'0'  
serre/actuatoeren/VentilHumDraaitijd b'0'  
serre/actuatoeren/LampDraaitijd b'0'  
serre/actuatoeren/PompDraaitijd b'1'  
serre/actuatoeren/VentilTempDraaitijd b'0'  
serre/actuatoeren/VerwDraaitijd b'0'  
serre/actuatoeren/VentilHumDraaitijd b'0'  
serre/actuatoeren/LampDraaitijd b'0'  
serre/actuatoeren/PompDraaitijd b'1'
```

Je ziet hierbij de topics waarop er ontvangen wordt en wat er ontvangen wordt. Zie 'Code' om te zien hoe je deze communicatie kunt laten lukken

## 5.2. Server Setup

Om de server te laten werken moet je ook een InfluxDB database aanmaken dat doe je op deze manier.

### 5.2.1. Installeer InfluxDB

InfluxDB installeren doe je met deze commando's:

- sudo apt-get update
- sudo apt-get install influxdb
- sudo systemctl start influxdb
- sudo systemctl enable influxdb

### 5.2.2. Maak een database aan

Een database aan maken in InfluxDB om je data te kunnen opslaan doe je op deze manier:

```
thibau@thibaupi:~ $ influx  
Connected to http://localhost:8086 version 1.6.7~rc0  
InfluxDB shell version: 1.6.7~rc0  
> CREATE DATABASE serre  
> SHOW DATABASES  
name: databases  
name  
----  
_internal  
DATA1  
IoTproject  
serre  
> |
```

### 5.2.3. Controleer of de data aankomt in de database

Controleren of de data aankomt in je database doe je zo:

```
thibau@thibaupi:~$ influx
Connected to http://localhost:8086 version 1.6.7~rc0
InfluxDB shell version: 1.6.7~rc0
> SHOW DATABASES
name: databases
name
----
_internal
DATA1
IoTproject
serre
> use IoTproject
Using database IoTproject
> SELECT * FROM temperature
name: temperature
time                location value
-----
1716977416853151540 waarden 28.8
1716977419178657697 waarden 28.9
1716977421508571809 waarden 29.1
1716977732526096498 waarden 28
1716977734909882274 waarden 27.9
1716977739519367033 waarden 27.8
1716977744303724828 waarden 27.7
1716977746633967634 waarden 27.6
1716977751281784575 waarden 27.5
1716977765438322916 waarden 27.6
1716977767757166145 waarden 27.7
1716977772412921311 waarden 27.8
1716977777178609856 waarden 27.9
```

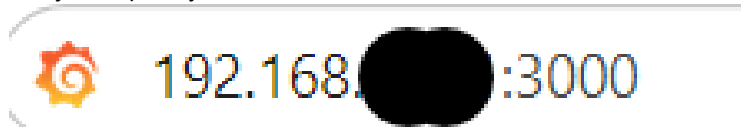
(Ik heb nu de database 'IoTproject' gebruikt en niet 'serre' omdat alle data van mijn project daar aankomen. De 'serre' database was gewoon een voorbeeld van hoe je een database moet aanmaken.)

## 5.3. InfluxDB connecteren met Grafana voor visualisatie

Om de data te visualiseren moet je je InfluxDB database koppelen aan je Grafana server. Dit doe je zo.

### 5.3.1. Zoek je Grafana server

Om op je Grafana server te geraken moet je gewoon in de zoekbalk van je browser het IP-adres in typen van je RaspberryPi met ':3000' er achteraan.



### 5.3.2. Log in in Grafana

Om in te loggen in Grafana wordt er gevraagd naar een wachtwoord en gebruikers naam. Deze zijn standaard 'admin' en 'admin'. Je kan deze aanpassen.


Dit is de koptekst in stijl 'Koptekst'

### 5.3.3. Maak een connectie met je InfluxDB database


## Add new connection

Browse and create new connections

### Data sources



InfluxDB




InfluxDB

Open source time series database

From  
Grafana Labs

Signature  
Core

Add new data source



IoTproject

Type: InfluxDB



Type: InfluxDB

Alerting  
Supported


Explore data


Build a dashboard

Settings




Name  IoTproject Default 

Query language

InfluxQL 

 Please report any issues to:  
<https://github.com/grafana/grafana/issues>

#### HTTP

URL		http://local.host:8086
Allowed cookies		New tag (enter key to add) <span>Add</span>
Timeout		Timeout in seconds

Dit is de koptekst in stijl 'Koptekst'

### Auth

Basic auth	<input checked="" type="checkbox"/>	With Credentials	<input type="checkbox"/>
TLS Client Auth	<input type="checkbox"/>	With CA Cert	<input type="checkbox"/>
Skip TLS Verify	<input type="checkbox"/>		
Forward OAuth Identity	<input type="checkbox"/>		

### Basic Auth Details

User	
Password	

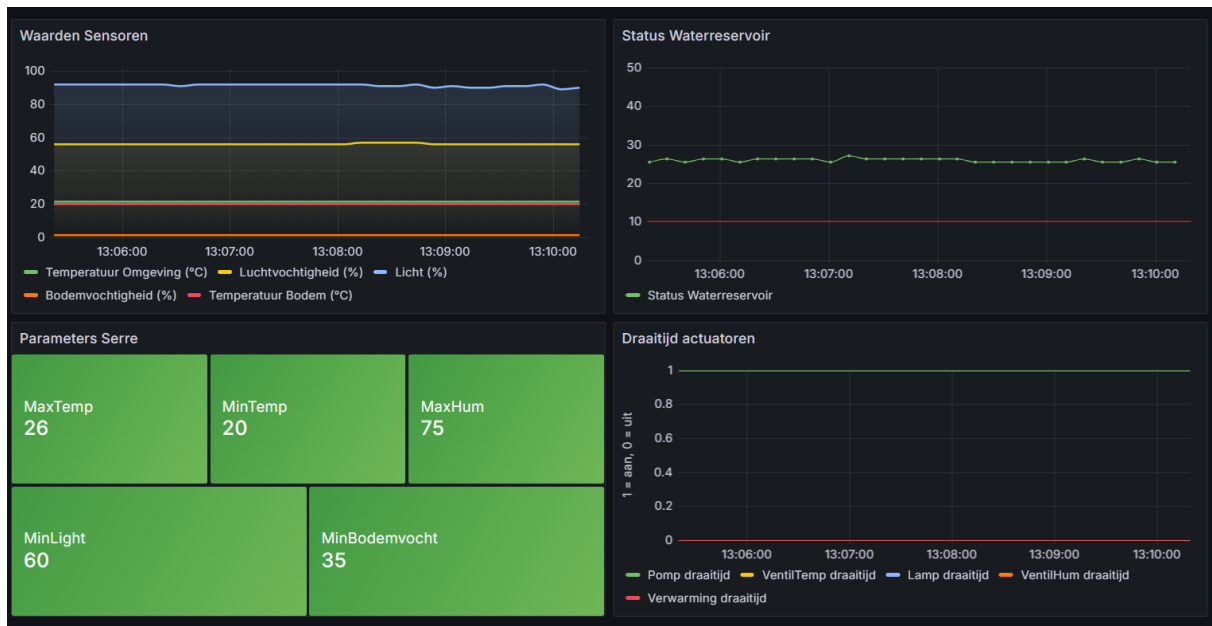
### Custom HTTP Headers

+ Add header

Database	IoTproject
User	
Password	
HTTP Method	Choose
Min time interval	10s
Max series	1000

Als na al deze stappen er onderaan het juiste aantal topics komt van je database dan heb je alles goed gedaan en kan je beginnen aan het visualiseren van je data.

## 5.4. Data Visualisatie



Zo ziet mijn dashboard eruit van de serre. Hier worden de sensor waarden, draaitijd van alle actuatoren en de ingestelde parameters op een mooie en overzichtelijke manier voorgesteld. (Alles wijst zichzelf uit qua visualiseren van je data. Kijk wat rond op de site en je zult zelf zien wat je allemaal kan doen.)

## 6. Datasheets

### 6.1. ESP 32

[https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)

### 6.2. RaspberryPi

<https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>

### 6.3. BC547B Transistor

[https://cdn-reichelt.de/documents/datenblatt/A100/BC546\\_48-CDIL.pdf](https://cdn-reichelt.de/documents/datenblatt/A100/BC546_48-CDIL.pdf)

### 6.4. PC 817 optocoupler

[https://cdn-reichelt.de/documents/datenblatt/A500/PC817XXNSZ1B\\_07OCT16\\_SPEC\\_ED-16P010.pdf](https://cdn-reichelt.de/documents/datenblatt/A500/PC817XXNSZ1B_07OCT16_SPEC_ED-16P010.pdf)

## **6.5. 5V relay**

<https://www.circuitbasics.com/wp-content/uploads/2015/11/SRD-05VDC-SL-C-Datasheet.pdf>

## **6.6. Feedback LED's**

[https://cdn-reichelt.de/documents/datenblatt/A500/NSPR510GS-E\(2047\).pdf](https://cdn-reichelt.de/documents/datenblatt/A500/NSPR510GS-E(2047).pdf)

Dit is de koptekst in stijl 'Koptekst'

## CONTACT

Thibau Vranken | Student  
r1002280@student.thomasmore.be

## VOLG ONS

[www.thomasmore.be](http://www.thomasmore.be)  
[fb.com/ThomasMoreBE](https://fb.com/ThomasMoreBE)  
#WeAreMore

THOMAS  
**MORE**