THOMAS MORE

KLIK HIER OM DE SUBTITEL AAN TE PASSEN

Zelf een serre maken

IOT project

2023 - 2024

Inhoud

1. Inleiding

Ik ga hier vertellen wat het project uiteindelijk gaat worden wat het belang van IOT inde landbouw.

1.1. Projectoverzicht

Dit project richt zich op het creëren van een geautomatiseerd kweeksysteem met behulp van IoT. Het doel is om de monitoring en sturing van factoren zoals temperatuur, vochtigheid, licht en bodemvochtigheid te kunnen doen. Hierdoor vind je misschien rapper en makkelijker problemen en weet je echt hoe het met je plantje gaat.

1.2. Belang van IoT in de landbouw

IoT-technologieën hebben de moderne landbouw getransformeerd door het mogelijk te maken om data te verzamelen, analyseren en te gebruiken voor beslissingen. In kweeksystemen biedt IoT waardevolle inzichten in de groeiomstandigheden en ishet in staat om te reageren op veranderende omgevingsfactoren. Hierdoor kunnen ziekten en plagen worden verminderd, het watergebruik kan worden geoptimaliseerd, en de algehele productiviteit en duurzaamheid van landbouwpraktijken kan worden verbeterd.

2. Technische beschrijving

2.1. Systeemarchitectuur

De algemene structuur van het smart greenhouse systeem omvat de volgende componenten:

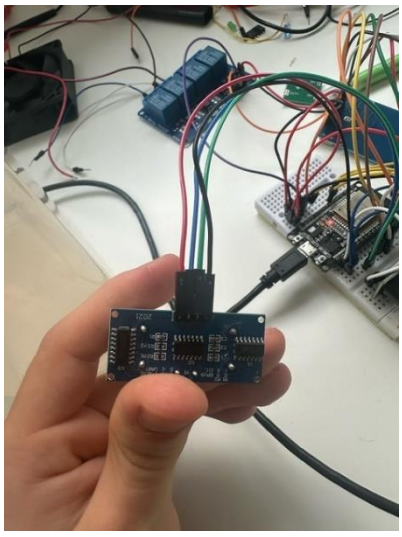
- *Sensoren: Voor het meten van temperatuur, luchtvochtigheid, lichtintensiteit, en bodemvochtigheid.*
- *Actuatoren: Voor het aansturen van ventilatoren, waterpompen, en verlichting.*
- *Microcontrollers: ESP32 als centrale eenheid die data van sensoren verzamelt en actuatoren aanstuurt.*
- *Singleboardcomputer: alle data word naar de Raspberry Pi gestuurt en zo kun je op grafana alles beter zien*

2.2. Hardware

- Esp32
- 4 5V relaismodule
- Fan
- 12V waterpomp
- Vochtigheidssensor
- Temperatuursensor
- Buzzer
- Knop
- Ldr sensor
- Spanningsregelaar
- 12V adapter
- Jumper wires
- Weerstandjes
- Rfid sensor (met ship)
- Afstandssensor

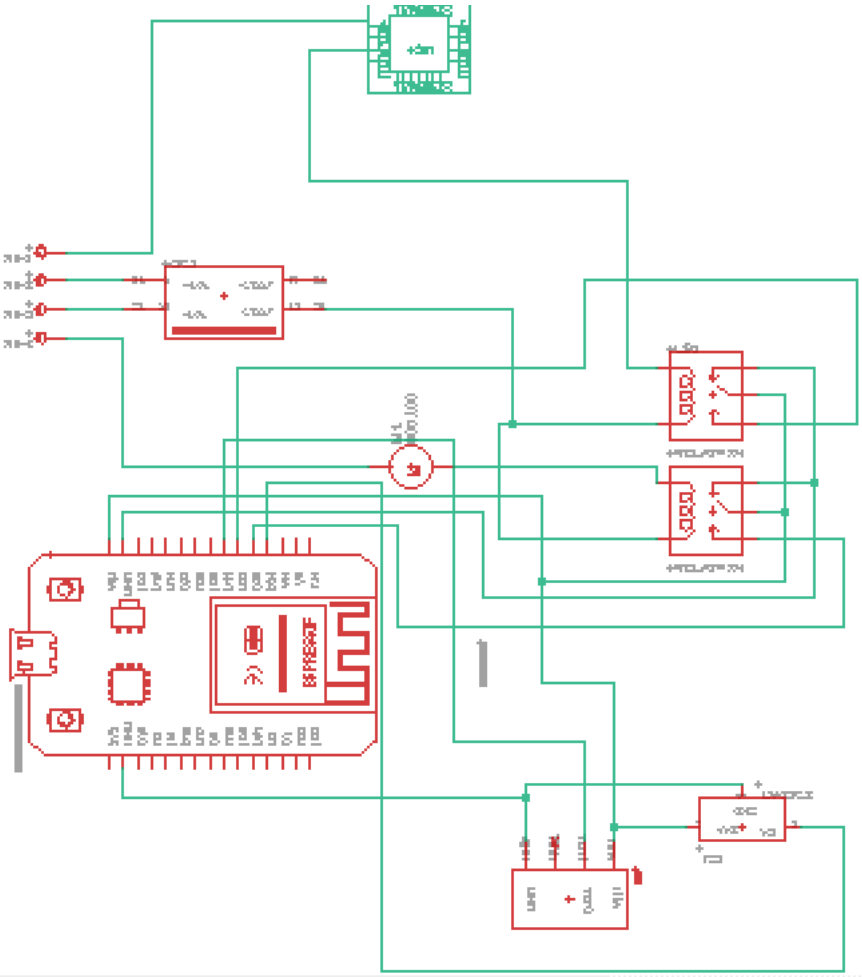
Ik heb al de sensoren aan mijn esp32 gehangen, ik vind het zelf het makkelijks om daarmee te werken, met veel sensoren had ik nog niet gewerkt maar ze hebben allemaal hun eigen functie en ik heb ze zo goed mogelijk proberen implementeren in dit project.

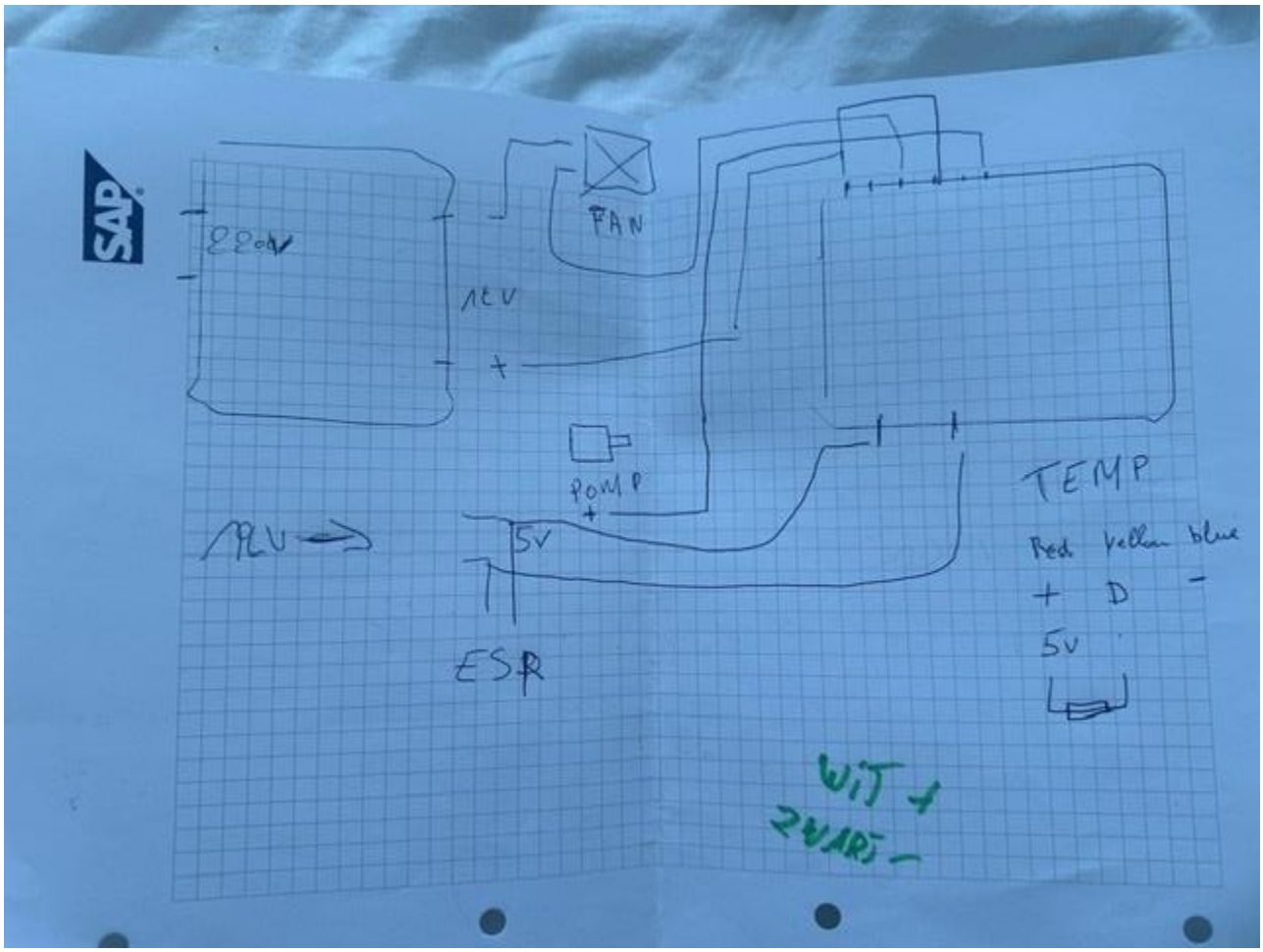
De fan en waterpomp worden zijn allebei verbonden met de adapter, aan de adapter hangt een spanningsregelaar die van 12V naar 5V gaat, de pomp hangt aan de 12V en de fan een de 5V, ze hangen ook beide aan de realais die gestuurd worden op arduino.



De afstandssensor kijkt naar hoeveel water er in de tank zit, als er te weinig in zit gaat de buzzer geluid maken. Ik een knop gemaakt die bij op het duwen van de knop signaal geeft aan de pomp dat hij moet pompen. Bijvullen moet handmatig.

De fan zal om de 10 seconden voor 10 seconden aangaan als vorm van verluchting in de serre.





2.3. Software specificaties

Code arduino:

code_eindproject.ino

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <afstandssensor.h>
#include <SPI.h>
#include <MFRC522.h>
#include <WiFi.h>  // Library for WiFi functionality
#include <PubSubClient.h>

#define RELAY_PIN_FAN 14  // Pin waarop IN1 van het 4-relaismodule is aangesloten voor de fan
#define MOISTURE_SENSOR_PIN 33
const int ldrPin = 34;
// Data wire is connected to GPIO 4
#define ONE_WIRE_BUS 27
#define RST_PIN 21  // Configurable, see typical pin layout above
#define SS_PIN 5    // Configurable, see typical pin layout above

const int buttonPin = 25;  // pin waar de knop is aangesloten
const int pompPin = 26;
bool pompState = true;  // variabele om de status van de LED bij te houden

const char* ssid = "SasHome_5GHz";
const char* password = "Quint4Willem5";

const char* mqtt_server = "192.168.0.178";
const int mqtt_port = 1883;
const char* MQTT_USER = "username";
```

code_eindproject.ino

```
27  const char* MQTT_PASSWORD = "test";
28  const char* MQTT_CLIENT_ID = "MQTTclient";
29  //Waarden
30  const char* MQTT_TOPICtemp = "home/serre/temperature";
31  const char* MQTT_TOPICvocht = "home/serre/vocht";
32  const char* MQTT_TOPICafstand = "home/serre/afstand";
33
34  const char* MQTT_REGEX = "home/([^/]+)/([^/]+)";
35
36
37  MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance
38  WiFiClient espClient;
39  PubSubClient mqttClient(espClient);
40
41  // Setup a oneWire instance to communicate with any OneWire devices
42  OneWire oneWire(ONE_WIRE_BUS);
43  AfstandsSensor afstandssensor(4, 2);  // Initialiseer de afstandssensor met triggerPin op 4 en echoPin op 2.
44
45  // Pass our oneWire reference to Dallas Temperature sensor
46  DallasTemperature sensors(&oneWire);
47
48  unsigned long previousMillisFan = 0;     // Variabele om de tijd van de vorige relaisactivering van de fan bij te houden
49  unsigned long previousMillisSensor = 0;  // Variabele om de tijd van de vorige relaisactivering van de vochtigheidssensor bij te houden
50  const long fanInterval = 10000;          // Interval van 10 seconden (in milliseconden) voor de fan
51  const long fanDuration = 10000;          // Duur van de relaisactivering van de fan (in milliseconden)
52  bool fanState = false;                   // Hui
```

code_eindproject.ino

```
53
54  void setup_wifi() {
55    delay(10);
56    Serial.println();
57    Serial.print("Verbinding maken met WiFi...");
58    WiFi.begin(ssid, password);
59    while (WiFi.status() != WL_CONNECTED) {
60      delay(500);
61      Serial.print(".");
62    }
63    Serial.println("");
64    Serial.println("WiFi verbonden");
65    Serial.println("IP adres: ");
66    Serial.println(WiFi.localIP());
67  }
68
69  void reconnect() {
70    while (!mqttClient.connected()) {
71      Serial.print("Verbinding maken met MQTT-broker...");
72      if (mqttClient.connect(MQTT_CLIENT_ID, MQTT_USER, MQTT_PASSWORD)) {
73        Serial.println(" verbonden");
74
75      } else {
76        Serial.print(" mislukt, rc=");
77        Serial.print(mqttClient.state());
78        Serial.println(" opnieuw proberen in 5 seconden");
```

code_eindproject.ino

```
      delay(5000);
    }
  }
}

void callback(char* topic, byte* payload, unsigned int length) {
  unsigned long currentTime = millis();
  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]);
  }
  Serial.println(bericht);
}


void setup() {
  // Start serial communication for debugging purposes
  Serial.begin(115200);
  pinMode(RELAY_PIN_FAN, OUTPUT);
  pinMode(ONE_WIRE_BUS, INPUT);
  pinMode(MOISTURE_SENSOR_PIN, INPUT);
  pinMode(pompPin, OUTPUT);  // zet de pin van de LED als output
```

code_eindproject.ino

```
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(22, OUTPUT);  // Zet pin 22 als uitgang (voor de buzzer)
  pinMode(34, INPUT);
  sensors.begin();
  setup_wifi();
  mqttClient.setServer(mqtt_server, mqtt_port);
  mqttClient.setCallback(callback);
  reconnect();
  while (!Serial)
    ;                 // Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)
  SPI.begin();        // Init SPI bus
  mfrc522.PCD_Init();  // Init MFRC522
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}

void pomp() {
  if (digitalRead(buttonPin) == LOW) {
    // Toggle de status van de LED
    pompState = !pompState;
    delay(250);
  }
  if (pompState) {
    digitalWrite(pompPin, HIGH);
  } else {
    // Zet de LED aan of uit afhankelijk van de pompState
    digitalWrite(pompPin, LOW);
```

code_eindproject.ino

```
    // Wacht een korte tijd om debounce te simuleren
  }
  Serial.println("pomp");
}

void loop() {
  mqttClient.loop();
  pomp();
  temperatuur();
  ventilator();
  bodemvocht();
  waterbak();
  light();
  RFID();
}

void temperatuur() {
  sensors.requestTemperatures();
  float temperatureC = sensors.getTempCByIndex(0);
  // Print the temperature to the Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println(" °C");
  String temps = String(temperatureC);
  mqttClient.publish(MQTT_TOPICtemp, temps.c_str());
  Serial.println(temps);
```

code_eindproject.ino

```
}

void ventilator() {
  unsigned long currentMillis = millis();  // Huidige tijd in milliseconden
  // Controleer of het tijd is om de fan in te schakelen
  if (currentMillis - previousMillisFan >= fanInterval) {
    previousMillisFan = currentMillis;  // Update de tijd van de laatste activering van de fan
    // Wissel de staat van het relais voor de fan
    fanState = !fanState;
    // Schakel het relais voor de fan in of uit op basis van de huidige staat
    digitalWrite(RELAY_PIN_FAN, fanState ? HIGH : LOW);
  }
}

void bodemvocht() {
  // Lees de vochtigheidswaarde van de sensor op IN2
  float moistureValue = map(analogRead(MOISTURE_SENSOR_PIN), 0, 4096, 100, 0);

  // Stuur de vochtigheidswaarde naar de seriële monitor
  Serial.print("Vochtigheid: ");
  Serial.println(moistureValue);
  String vochts = String(moistureValue);
  mqttClient.publish(MQTT_TOPICvocht, vochts.c_str());
}

void waterbak() {
```

code_eindproject.ino

```
  int afstand = afstandssensor.afstandCM();
  Serial.print("waterreservoir: ");
  Serial.println(afstand);
  String afstands = String(afstand);
  mqttClient.publish(MQTT_TOPICafstand, afstands.c_str());


  if (afstand < 10) {  // Controleer of de afstand kleiner is dan 10 cm
    tone(22, 1000);    // Laat de buzzer een toon van 1000 Hz afspelen
  } else {
    noTone(22);  // Schakel de buzzer uit
  }
}

void light() {
  int ldrValue = map(analogRead(ldrPin), 0, 4096, 100, 0);

  // Print de gemeten waarde naar de seriële monitor
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);
}

void RFID() {
  static String jeton = "";
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  // Select one of the cards
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  Serial.print("UID tag :");
  String content = "";
  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();
  Serial.print("Message : ");
  content.toUpperCase();
  Serial.println(content.substring(1));

  jeton = content.substring(1);
  if (jeton == "374F913") {
    Serial.println("jeton");
  }
}

```

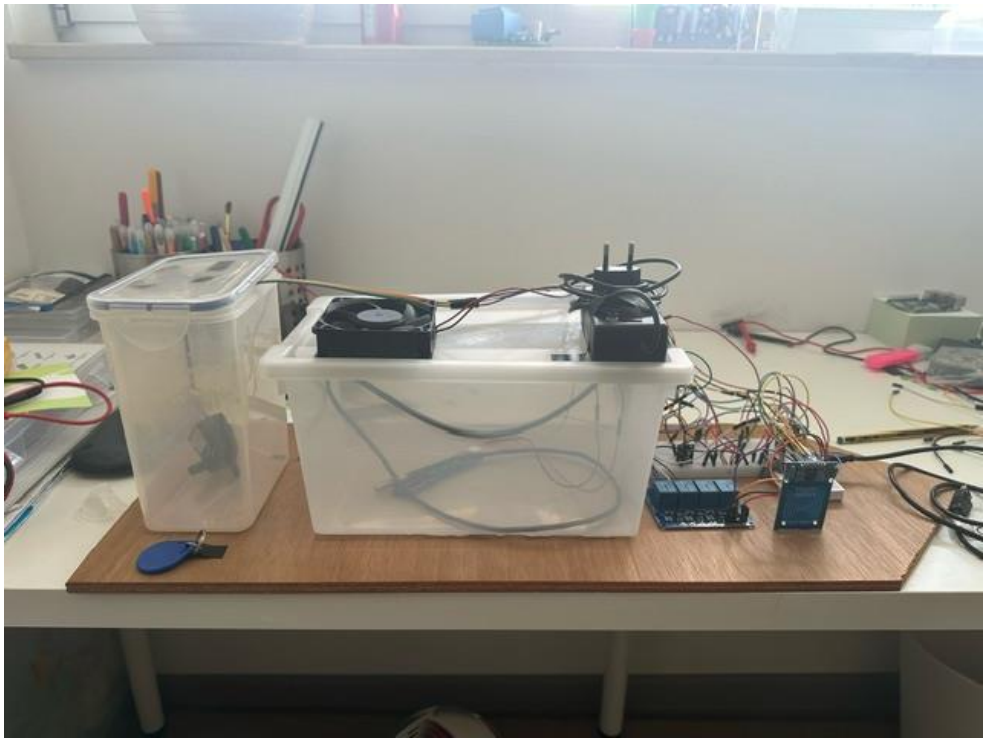
Code putty:

```
 GNU nano 7.2                                                    eindproject.py
import re
from typing import NamedTuple

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

INFLUXDB_ADDRESS = 'localhost'
INFLUXDB_USER = 'wimmel'
INFLUXDB_PASSWORD = 'comcon'
INFLUXDB_DATABASE = 'wimmel'

MQTT_ADDRESS = 'localhost'
MQTT_USER = 'username'
MQTT_PASSWORD = 'test'
MQTT_TOPIC = 'home/+/+'
MQTT_REGEX = 'home/([^/]+)/([^/]+)'
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': {
```

3. Implementatiedetails

3.1. Opzet en Configuratie

1. *Verbind alle sensoren op een breadboard aan de esp32*
2. *Soldeer een spanningsregelaar op een lcd plaatje*
3. *Hang de fan en de pomp aan de juiste kanten van de spanningsregelaar*
4. *Aan de relaismodules hang je de fan en de pomp ook aan*
5. *Maak een code op arduino zodat het werkt*
6. *Verbind de raspberry pi met je computer*
7. *Maa keen code in putty*
8. *Neem 2 dozen en hang alles aan elkaar (ik heb lijm gebruikt maar je kan dit ook op andere manieren doen)*



3.2. Testprocedures

1. **Functionele Testen**:

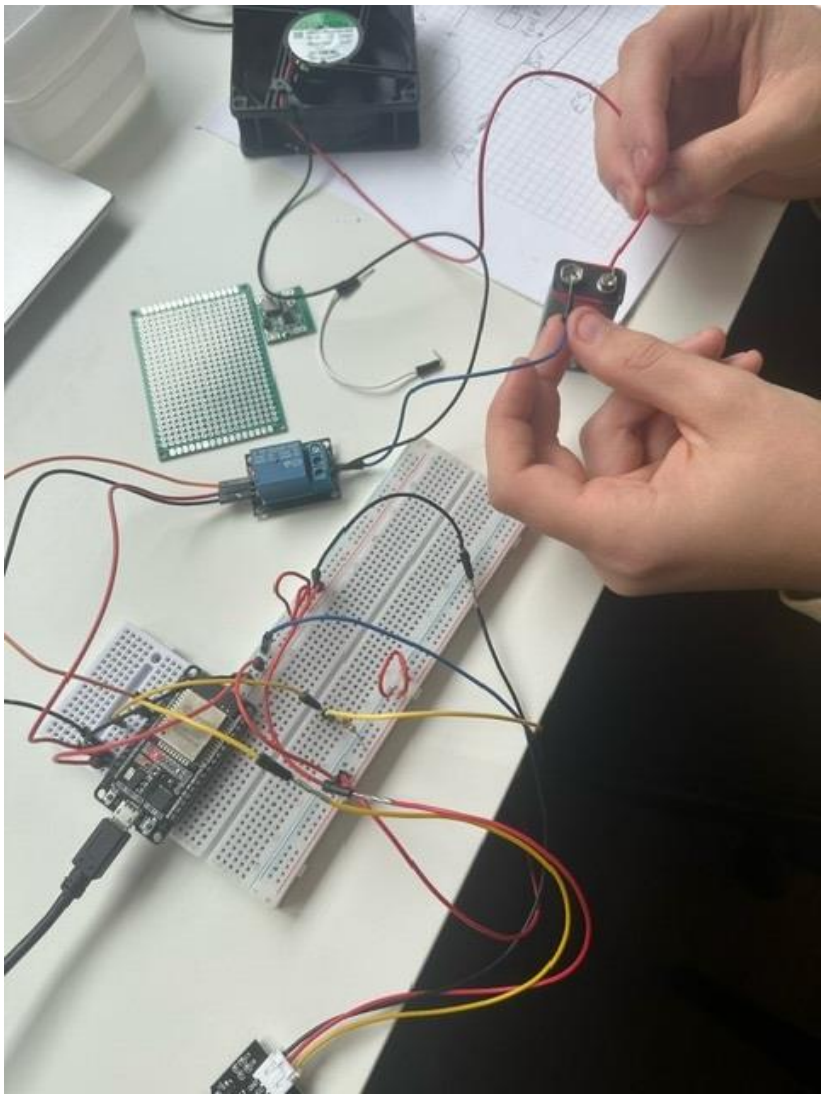
   Controleer of sensoren correct data uitlezen.
   Verifieer de werking van actuatoren door handmatige aanzetten via de ESP32.

2. **Communicatietesten**:

   Test de Wi-Fi verbinding en de dataoverdracht naar de centrale server.

3. **Integratietesten**:

   Verifieer de volledige systeemintegratie, waarbij sensoren data naar de server sturen en actuatoren automatisch reageren op veranderingen in sensorwaarden



4. Data Management en Visualisatie

4.1. Data Logging en Beheer

Sensor Data Verzameling: *De ESP32 verzamelt data van alle sensoren.*

Beheer: *Data wordt beheerd via een webinterface waar telers trends kunnen analyseren en historische data kunnen bekijken.*

4.2. Visualisatie met Grafana

Dashboard Creatie*: Grafana wordt gebruikt om dashboards te maken die de sensorwaarden in real-time weergeven.*

Analyse*: Door gebruik te maken van Grafana's visualisatietools kan ik trends identificeren en beter geïnformeerde beslissingen nemen.*

Besluitvorming*: Real-time visualisatie helpt bij het snel reageren op veranderende omstandigheden en optimaliseert het kweekproces.*

5. Innovaties en Aanpassingsvermogen

5.1. Modulair Ontwerp en RFID Integratie

De rfid zou er voor moeten zorgen dat je de sensoren allemaal kunt aansturen en hun data kunt laten veranderen door een keer met de ship voor de rfid tag ta gaan. Bij mij is dit niet zo goed gelukt, er komt op de serial monitor gewoon te staan dat je hem gescant hebt en verder past deze niets aan bij mij.

5.2. Automatisering en Reactie op Omgevingsfactoren

Als het water in de tank te laag staat merkt de afstandssensor dit op en geeft hij teken aan de buzzer dat hij geluid mag maken, ook met het duwen op een knop laat je de pomp werken.

6. Conclusies en Toekomstig Werk

6.1. Projectevaluatie

Bijna alle sensoren werken hoe ze moeten werken buiten de rfid sensor, ik heb veel moeite gehad met het begrijpen en leren werken met een raspberry pi en dit is me uiteindelijk ook nog gelukt. Ik heb leren werken met nieuwe sensoren en heb een mooi project kunnen neerzetten.

6.2. Aanbevelingen voor Toekomstige Verbeteringen

Uiteindelijk zou ik natuurlijk met elke sensor kunnen werken en wil ik dit ook voor in de toekomst onthouden om ze beter te begrijpen en meer opzoekwerk te verrichten bij het gebruiken van een nieuwe sensor. In de toekomst heb ik geen plannen om nog een project als dit te maken maar ik vind het iets leuks en nieuw om allemaal bij te leren.

7. Bibliografie

- **Online Materialen**:
  - *Arduino. (n.d.). "Arduino IDE". van* https://www.arduino.cc/en/Main/Software
  - *Grafana Labs. (n.d.). "Grafana: The open observability platform". Van* https://grafana.com/
- **Websites en Tutorials**:
  - *Random Nerd Tutorials. (n.d.). "ESP32 with DHT22 Temperature and Humidity Sensor using Arduino IDE". van* https://randomnerdtutorials.com/esp32-dht22-arduino-ide/
  - *SparkFun Electronics. (n.d.). "ESP32 Thing Hookup Guide". van* https://learn.sparkfun.com/tutorials/esp32-thing-hookup-guide/all
  - *Bosch. (n.d.). "BH1750 Digital Ambient Light Sensor". van* https://www.bosch-sensortec.com/products/environmental-sensors/ambient-light-sensors-bh1750/