2023 - 2024

|  |
| --- |
| Thars Van de Vel |
| Opdracht Smart Greenhouse |
| IoT project |

Inhoud

[1. Beschrijving 4](#_Toc169033945)

[1.1. Projectoverzicht 4](#_Toc169033946)

[1.2. Belang van IoT in de landbouw 4](#_Toc169033947)

[1.2.1. Precisielandbouw 4](#_Toc169033948)

[1.2.2. Automatisering en Efficiëntie 4](#_Toc169033949)

[1.2.3. Gegevensgestuurde Besluitvorming 4](#_Toc169033950)

[1.2.4. Duurzaamheid 4](#_Toc169033951)

[1.2.5. Kweeksystemen 5](#_Toc169033952)

[1.2.6. Conclusie 5](#_Toc169033953)

[2. Hardware 6](#_Toc169033954)

[2.1. ESP32 DOIT Devkit 6](#_Toc169033955)

[2.2. HC-SR04 ultrasone sensor 6](#_Toc169033956)

[2.3. I2C LCD 6](#_Toc169033957)

[2.4. Capacitive soil moisture sensor V1.2 7](#_Toc169033958)

[2.5. RFID RC522 7](#_Toc169033959)

[2.6. KY-018 LDR module 7](#_Toc169033960)

[2.7. DHT11 temperatuur en vochtigheid sensor 8](#_Toc169033961)

[2.8. Servomotor 8](#_Toc169033962)

[2.9. 5V Waterpomp 8](#_Toc169033963)

[2.10. 5V Relais 9](#_Toc169033964)

[2.11. Ledstrip 9](#_Toc169033965)

[2.12. Raspberry Pi 4 model B 9](#_Toc169033966)

[3. Schema 10](#_Toc169033967)

[4. PCB 11](#_Toc169033968)

[5. MQTT en Gafana 12](#_Toc169033969)

[6. Code : 13](#_Toc169033970)

[6.1. ESP32 code: 13](#_Toc169033971)

[6.2. Raspberry pi code: 22](#_Toc169033972)

[7. Hoe kan je het namaken? 25](#_Toc169033973)

[8. Innovaties en Aanpassingsvermogen 26](#_Toc169033974)

[8.1. Modulair Ontwerp en RFID Integratie 26](#_Toc169033975)

[8.2. Automatisering en Reactie op Omgevingsfactoren 26](#_Toc169033976)

[8.2.1. Licht 26](#_Toc169033977)

[8.2.2. Luchttemperatuur 26](#_Toc169033978)

[8.2.3. Grondtemperatuur 26](#_Toc169033979)

[8.2.4. Grondvochtigheid 26](#_Toc169033980)

[9. Conclusies en Toekomstig Werk: 27](#_Toc169033981)

[9.1. Projectevaluatie 27](#_Toc169033982)

[9.2. Aanbevelingen voor Toekomstige Verbeteringen 27](#_Toc169033983)

[10. Bibliografie: 28](#_Toc169033984)

# Beschrijving

## **Projectoverzicht**

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 basilicum andere temperatuur, vochtigheid en belichtingsperiodes hebben dan een

kweekbak voor dille.

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.

## Belang van IoT in de landbouw

IoT-technologieën zijn essentieel in moderne landbouw en kweeksystemen vanwege hun mogelijkheden voor geavanceerde monitoring en beheer. Hier zijn de belangrijkste voordelen:

### Precisielandbouw

IoT-sensoren verzamelen data over bodemvochtigheid, temperatuur en nutriëntenniveaus, waardoor boeren nauwkeurigere beslissingen kunnen nemen over irrigatie en bemesting, wat leidt tot hogere opbrengsten en minder verspilling.

### Automatisering en Efficiëntie

IoT-systemen automatiseren processen zoals irrigatie, waardoor handmatige tussenkomst wordt verminderd en gewassen optimaal worden verzorgd. Dit verhoogt de productiviteit en verlaagt de kosten.

### Gegevensgestuurde Besluitvorming

IoT-apparaten verzamelen continu gegevens, helpen bij het voorspellen van ziektes en oogstmomenten, en optimaliseren landbouwpraktijken om risico's te verminderen en opbrengsten te verbeteren.

### Duurzaamheid

Door efficiënter gebruik van water, meststoffen en pesticiden verminderen IoT-technologieën de milieubelasting en verbeteren ze de duurzaamheid van landbouwpraktijken.

### Kweeksystemen

In kassen en verticale boerderijen regelen IoT-sensoren klimaat, verlichting en voedingsstoffen automatisch, wat zorgt voor optimale groeiomstandigheden en hogere productiviteit.

### Conclusie

IoT-technologieën transformeren de landbouw door verbeterde monitoring, automatisering en gegevensgestuurde besluitvorming, wat leidt tot efficiëntere, duurzamere en milieuvriendelijkere landbouwpraktijken.

# Hardware

## ESP32 DOIT Devkit



Dit is de microcontroller dat heel het project zal besturen. Deze geeft 3.3V en 5V aan de componenten en ontvangt data van de inputs voor de outputs te besturen. Die data zal ook gestuurd worden naar een raspberry pi via MQTT

## HC-SR04 ultrasone sensor



Deze meet het waterniveau in de watertank van de waterpomp zodat als die bijna leeg is dat er een melding wordt gestuurd dat de watertank bijgevuld moet worden

## I2C LCD



Hierop worden de namen van de planten op geprint nadat er met de rfid een tag wordt gelezen. Er word ook verschillende data geprint zodat je niet op grafana moet gaan voor te kijken hoe het in de kweekbak is

## Capacitive soil moisture sensor V1.2

Afbeelding met tekst, kabel, elektronica, connector

Automatisch gegenereerde beschrijving

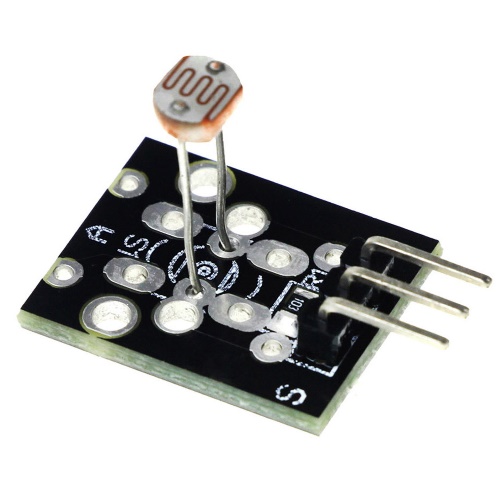
Deze meet de grondvochtigheid van de plant en als die te laag is zal de pomp aangaan zodat de plant niet zal dehydrateren.

## RFID RC522



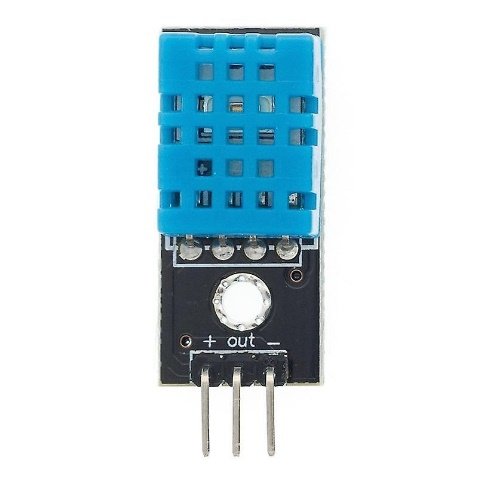
Hierop worden de rfid tags die verbonden zijn aan een plantensoort gelezen en alle parameters worden aangepast op basis van wat de plant nodig heeft.

## KY-018 LDR module



Deze meet hoeveel licht de plant raakt. Als dat te weinig is zal de groeilamp led strip aangaan

## DHT11 temperatuur en vochtigheid sensor



Deze meet hoe de temperatuur is bij de plant en als die te hoog is gaat er een raampje open zodat de plant het niet te warm kan krijgen

## Servomotor

Afbeelding met plastic

Automatisch gegenereerde beschrijving

Als de temperatuur bij de plant te hoog is zal deze een raampje opendoen en wind binnenlaten

## 5V Waterpomp

Afbeelding met cilinder, dop

Automatisch gegenereerde beschrijving

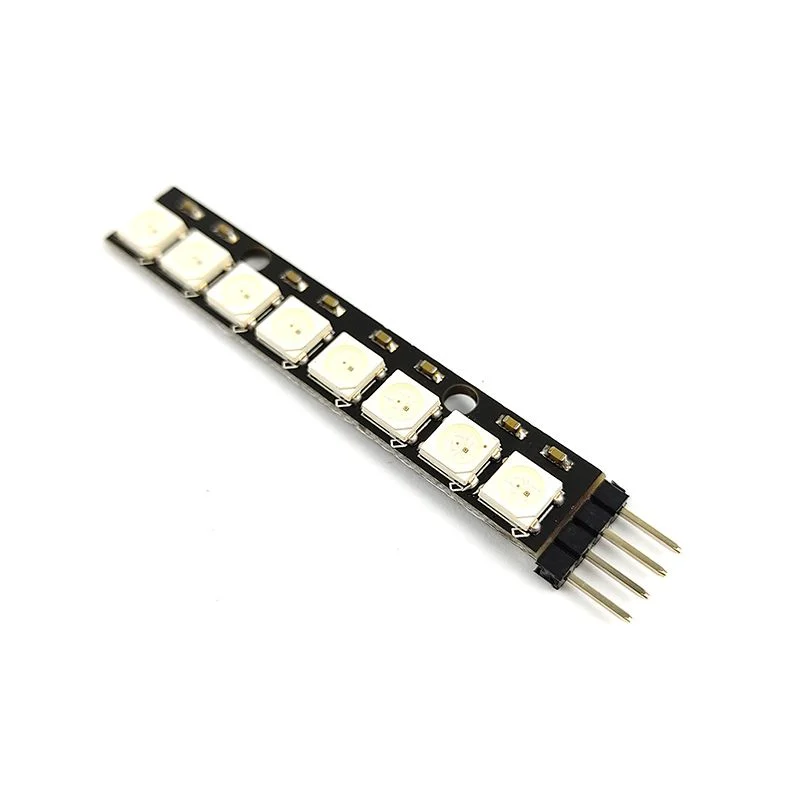
Deze pompt water in de potgrond als die niet vochtig genoeg is voor de plant.

## 5V Relais



Deze zorgt ervoor dat de esp32 kan besturen wanneer de pomp aan en uit gaat. Die pomp heeft alleen maar een + en een – draadje.

## Ledstrip



Deze wordt gebruikt als groeilamp voor de plant als de LDR niet genoeg licht ontvangt voor de plant.

## Raspberry Pi 4 model B

Afbeelding met elektronica, Elektronisch onderdeel, Stroomkringonderdeel, Passief stroomkringonderdeel

Automatisch gegenereerde beschrijving  
Deze wordt gebruikt om de data te ontvangen van de esp32 via MQTT en in een influxDB database steken die dan weergeven kan worden op grafana.

# Schema

Afbeelding met diagram, tekst, Plan, Technische tekening

Automatisch gegenereerde beschrijving

# PCB

Afbeelding met schermopname, diagram, ontwerp

Automatisch gegenereerde beschrijving

Ik heb dit design zelf niet gebruikt maar ik had beslist een prototype board te gebruiken om daarmee te leren werken en ook beter worden in solderen.

# MQTT en Gafana

De data die verzamelt wordt door de ESP32 wordt ook verstuurd naar een raspberry pi via een publish bericht van MQTT. Op die raspberry pi, die gesubscribed is op de topic van dat ESP32 MQTT bericht, staat een python script dat als service werkt en de ontvangen data omzet naar data dat in een influxdb database gestoken kan worden op de raspberry pi zelf. Dan kan je op een browser het ip adress van de raspberry typen in de zoekbalk en zo kom je op grafana waar er een kweekbak dashboard staat vol met ontvangen data zoals de temperaturen, het percentage licht, hoeveel water er nog in de tank zit en of verschillende outputs nu aan staan of uit. Daardoor hoef je niet je laptop te koppelen aan de esp32 om al de data te kunnen lezen. Ze staan in grafieken waar duidelijk te zien is hoe de gemeten data vergelijkt met de gewenste data.

# Code :

## ESP32 code:

Deze code dient voor het verzamelen van data voor grafana en de lcd en ook voor het besturen van de outputs

// Include necessary libraries

#include <Arduino.h>

#include <SPI.h>

#include <MFRC522.h>

#include <DHT.h>

#include <LiquidCrystal\_I2C.h>

#include <OneWire.h>

#include <Wire.h>

#include <DallasTemperature.h>

#include <Adafruit\_NeoPixel.h>

#include <ESP32Servo.h>

#include <WiFi.h>

#include <PubSubClient.h>

#include <ArduinoJson.h>

// define gpio pins

#define DHTPIN 4                  // Digital pin connected to the DHT sensor

#define DHTTYPE DHT11             // DHT 11

#define RST\_PIN 14                // Reset pin connected to GPIO 14 (D14)

#define SS\_PIN 21                 // Slave Select pin connected to GPIO 5 (D5)

#define SERVO\_PIN 26              // Pin to which the servo is connected

#define RELAY\_PIN 27              // Change this to the GPIO pin connected to the relay module

#define ONE\_WIRE\_BUS 22           // one wire bus to DS18B20

#define MAT\_PIN 5                 // PIN FOR heatmat

#define LED\_PIN 32                // Pin D32 on ESP32

#define LED\_COUNT 30              // Number of LEDs in your strip

const int trigPin = 12;           // Trig pin of HC-SR04

const int echoPin = 13;           // Echo pin of HC-SR04

const int soilSensorPin = 34;     // Analog pin to which the sensor is connected

const int photoresistorPin = 35;  // Digital pin 35 on ESP32

// Replace with your network credentials

const char\* ssid = "embed";

const char\* password = "weareincontrol";

// MQTT Broker data

const char\* mqtt\_server = "tharspi.local";

const char\* mqtt\_user = "thars";

const char\* mqtt\_password = "thars";

// create wifi for MQTT

WiFiClient espClient;

PubSubClient client(espClient);

Servo myServo;                                                      // Create a servo object

MFRC522 mfrc522(SS\_PIN, RST\_PIN);                                   // Create MFRC522 instance.

DHT dht(DHTPIN, DHTTYPE);                                           // create DHT object

Adafruit\_NeoPixel strip(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);  // create ledstrip object

LiquidCrystal\_I2C lcd(0x27, 16, 2);                                 // Set the LCD I2C address and the number of columns and rows

// Setup a oneWire instance to communicate with any OneWire devices

OneWire oneWire(ONE\_WIRE\_BUS);

// Pass oneWire reference to DallasTemperature library

DallasTemperature sensors(&oneWire);

// parameters voor de kweekbak

float lichtintensiteit = 0;

float bodemvochtigheid = 5000;

float bodemtemperatuur = 0;

float luchtemperatuur = 0;

float WaterAfstand = 15;

String plantname = "";

// extra parameters voor extra grafana data

int lcdState = 1;

int waterLeft = 0;

float moistpercent = 0;

float lightpercent = 0;

float pumpState = 0;

float windowState = 0;

float lightState = 0;

float hundred = 100;

float null = 0;

// Define your mappings between RFID tags and settings

const String TAG\_ID\_1 = "03 31 79 13";

const String TAG\_ID\_2 = "53 80 02 29";

// Function to compare two tag IDs

bool compareTagIDs(String tag1, String tag2) {

  return tag1.equals(tag2);

}

// Function to update settings based on the detected RFID tag

void updateSettings(String tagID) {

  if (compareTagIDs(tagID, TAG\_ID\_1)) {

    Serial.println("Settings updated for Tag 1");

    plantname = "plant1";

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(plantname);

    lichtintensiteit = 1000;

    bodemvochtigheid = 2000;

    bodemtemperatuur = 15;

    luchtemperatuur = 15;

  } else if (compareTagIDs(tagID, TAG\_ID\_2)) {

    Serial.println("Settings updated for Tag 2");

    plantname = "plant2";

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(plantname);

    lichtintensiteit = 5;

    bodemvochtigheid = 5000;

    bodemtemperatuur = 50;

    luchtemperatuur = 30;

  } else {

    Serial.println("Unknown tag ID");

  }

}

void colorWipe(uint32\_t color, int wait) {

  for (int i = 0; i < strip.numPixels(); i++) {

    strip.setPixelColor(i, color);  // Set pixel color

    strip.show();                   // Update strip

    delay(wait);                    // Pause for a moment

  }

}

void moveServo(int degrees) {

  myServo.write(degrees);  // Move the servo to the specified angle

}

void setup\_wifi() {

  delay(10);

  // Connecting to a WiFi network

  Serial.begin(115200);

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");  // print ..... if it cant connect to wifi

  }

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

}

void callback(char\* topic, byte\* payload, unsigned int length) {

  Serial.print("Message arrived [");

  Serial.print(topic);

  Serial.print("] ");

  for (int i = 0; i < length; i++) {

    Serial.print((char)payload[i]);

  }

  Serial.println();

}

void reconnect() {

  // Loop until we're reconnected

  while (!client.connected()) {

    Serial.print("Attempting MQTT connection...");

    // Attempt to connect

    if (client.connect("ESP32Client", mqtt\_user, mqtt\_password)) {

      Serial.println("connected");

    } else {

      Serial.print("failed, rc=");

      Serial.print(client.state());

      Serial.println(" try again in 5 seconds");

      // Wait 5 seconds before retrying

      delay(5000);

    }

  }

}

void setup() {

  Serial.begin(115200);  // Initialize serial communications with the PC

  while (!Serial)

    ;                                   // Wait for serial port to connect

  SPI.begin();                          // Init SPI bus

  setup\_wifi();                         //setup wifi

  client.setServer(mqtt\_server, 1883);  //set MQTT server

  client.setCallback(callback);

  mfrc522.PCD\_Init();             // Init MFRC522

  dht.begin();                    // start DHT11

  Wire.begin(25, 33);             // Start the I2C communication on pins 25 (SDA) and 33 (SCL)

  lcd.init();                     // Initialize the LCD

  lcd.backlight();                // Turn on the backlight

  lcd.clear();                    //clear the lcd

  sensors.begin();                //start the DS18B20

  pinMode(trigPin, OUTPUT);       // Set trig pin as output

  pinMode(echoPin, INPUT);        // Set echo pin as input

  pinMode(soilSensorPin, INPUT);  //set soilsensor as input

  pinMode(RELAY\_PIN, OUTPUT);     // set relay as output

  pinMode(MAT\_PIN, OUTPUT);       // set led

  myServo.attach(SERVO\_PIN);      // Attach the servo to the specified pin

  strip.begin();                  // Initialize the strip

  strip.show();                   // Initialize all pixels to 'off'

  strip.setBrightness(50);        // Set brightness (0 to 255)

}

void loop() {

  if (!client.connected()) {

    reconnect();

  }

  client.loop();

  long duration, distance;

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);  // send short ultrasonic pulse

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  // Measure the duration of the echo pulse

  duration = pulseIn(echoPin, HIGH);

  // Calculate distance in centimeters

  distance = duration \* 0.034 / 2;

  // Call sensors.requestTemperatures() to issue a global temperature

  // request to all devices on the bus

  sensors.requestTemperatures();

  // Get the temperature in Celsius

  float temperatureC = sensors.getTempCByIndex(0);

  // Check if temperature reading is valid

  if (temperatureC != DEVICE\_DISCONNECTED\_C) {

    // Print temperature to serial monitor

    if (temperatureC <= bodemtemperatuur) {

      Serial.print("ZET WARMTEMATLEDJE AAN || ");

      digitalWrite(MAT\_PIN, HIGH);

    } else {

      digitalWrite(MAT\_PIN, LOW);

    }

    Serial.print("Soil Temperature: ");

    Serial.print(temperatureC);

    Serial.println(" °C");

  } else {

    // Print an error message if temperature reading failed

    Serial.println("Error: Unable to read soil temperature data");

  }

  // Print the distance to the serial monitor

  if (distance > WaterAfstand) {

    Serial.print("VOEG WATER TOE AAN DE TANK || ");

  }

  Serial.print("Distance: ");

  Serial.print(distance);

  Serial.print(" cm || ");

  // calculate waterleft in the tank

  waterLeft = distance - 15;

  waterLeft = -waterLeft;

  Serial.print("Water Left in tank : ");

  Serial.print(waterLeft);

  Serial.println(" cm");

  // measure the soilmoisture

  float soilMoisture = analogRead(soilSensorPin);

  moistpercent = (bodemvochtigheid / soilMoisture) \* 100;

  if (soilMoisture >= bodemvochtigheid) {

    Serial.print("ZET WATERPOMPJE AAN || ");

    digitalWrite(RELAY\_PIN, HIGH);

    pumpState = 1;

  } else {

    digitalWrite(RELAY\_PIN, LOW);

    pumpState = 0;

  }

  Serial.print("Soil Moisture: ");

  Serial.print(soilMoisture);

  Serial.print(" || percent: ");

  Serial.println(moistpercent);

  float lightValue = analogRead(photoresistorPin);

  lightpercent = (lichtintensiteit / lightValue) \* 100;

  if (lightValue >= lichtintensiteit) {

    // Turn on LED strip in pink color

    colorWipe(strip.Color(255, 0, 255), 50);  // Pink color

    lightState = 1;

  } else {

    // Otherwise, turn off the LED strip

    colorWipe(strip.Color(0, 0, 0), 50);  // Off

    lightState = 0;

  }

  // Print the sensor value to the serial monitor

  Serial.print("Light Value: ");

  Serial.print(lightValue);

  Serial.print(" || percent: ");

  Serial.println(lightpercent);

  float temperature = dht.readTemperature();  // read DHT11 temperature in celsius Celsius

  if (isnan(temperature)) {

    Serial.println("Failed to read from DHT sensor!");

    return;

  }

  if (temperature >= luchtemperatuur) {

    Serial.print("ZET Raampje open AAN || ");

    moveServo(90);  // Move servo to 90 degrees

    windowState = 1;

  } else {

    moveServo(0);  // Move servo back to 0 degrees

    windowState = 0;

  }

  Serial.print("Air Temperature: ");

  Serial.print(temperature);

  Serial.println(" \*C");

  // Create JSON objects with data for grafana

  StaticJsonDocument<400> doc1;

  doc1["soilTemperature"] = static\_cast<float>(temperatureC);

  doc1["distance"] = static\_cast<float>(distance);

  doc1["soilMoisture"] = static\_cast<float>(soilMoisture);

  doc1["moistpercent"] = static\_cast<float>(moistpercent);

  doc1["lightpercent"] = static\_cast<float>(lightpercent);

  doc1["waterLeft"] = static\_cast<float>(waterLeft);

  doc1["lightValue"] = static\_cast<float>(lightValue);

  doc1["airTemperature"] = static\_cast<float>(temperature);

  // second object beacause first was too big

  StaticJsonDocument<400> doc2;

  doc2["pumpState"] = static\_cast<float>(pumpState);

  doc2["windowState"] = static\_cast<float>(windowState);

  doc2["lightState"] = static\_cast<float>(lightState);

  doc2["lichtintensiteit"] = static\_cast<float>(lichtintensiteit);

  doc2["bodemvochtigheid"] = static\_cast<float>(bodemvochtigheid);

  doc2["bodemtemperatuur"] = static\_cast<float>(bodemtemperatuur);

  doc2["luchtemperatuur"] = static\_cast<float>(luchtemperatuur);

  doc2["hundred"] = static\_cast<float>(hundred);

  doc2["WaterAfstand"] = static\_cast<float>(WaterAfstand);

  doc2["null"] = static\_cast<float>(null);

  char jsonBuffer1[400];

  char jsonBuffer2[400];

  size\_t n1 = serializeJson(doc1, jsonBuffer1);

  size\_t n2 = serializeJson(doc2, jsonBuffer2);

  Serial.print("JSON size: ");

  Serial.println(n1);

  Serial.print("JSON size: ");

  Serial.println(n2);

  serializeJson(doc1, jsonBuffer1);

  serializeJson(doc2, jsonBuffer2);

  // Publish JSON data to MQTT

  client.publish("esp32/sensors", jsonBuffer1);

  client.publish("esp32/sensors", jsonBuffer2);

  // every 5 loops lcd print shows different data

  if (lcdState > 0) {

    int procent = 0;

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(plantname);

    lcd.setCursor(0, 1);

    lcd.print(temperatureC);

    lcd.setCursor(8, 1);

    lcd.print("SoilTemp");

  }

  if (lcdState > 5) {

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(plantname);

    lcd.setCursor(0, 1);

    lcd.print(waterLeft);

    lcd.setCursor(4, 1);

    lcd.print("cm WaterLeft");

  }

  if (lcdState > 10) {

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(plantname);

    lcd.setCursor(0, 1);

    lcd.print(moistpercent);

    lcd.setCursor(6, 1);

    lcd.print("%SoilMoist");

  }

  if (lcdState > 15) {

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(plantname);

    lcd.setCursor(0, 1);

    lcd.print(lightpercent);

    lcd.setCursor(10, 1);

    lcd.print("%Light");

  }

  if (lcdState > 20) {

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(plantname);

    lcd.setCursor(0, 1);

    lcd.print(temperature);

    lcd.setCursor(9, 1);

    lcd.print("AirTemp");

  }

  if (lcdState == 25) { // when lcd state = 25 return to 0

    lcdState = 0;

  }

  lcdState++;

  delay(500);

  // Look for new cards

  if (!mfrc522.PICC\_IsNewCardPresent()) {

    return;

  }

  // Select one of the cards

  if (!mfrc522.PICC\_ReadCardSerial()) {

    return;

  }

  //Show UID on serial monitor

  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();

  Serial.print("Message : ");

  content.toUpperCase();

  Serial.println(content.substring(1));

  // Update settings based on the detected RFID tag

  updateSettings(content.substring(1));

  Serial.println(lichtintensiteit);

  Serial.println(bodemvochtigheid);

  Serial.println(bodemtemperatuur);

  Serial.println(luchtemperatuur);

  delay(1000);

}

## Raspberry pi code:

Deze code dient voor de ontvangen data in een influxdb database te steken zodat die gezien kan worden op grafana

import paho.mqtt.client as mqtt

from influxdb import InfluxDBClient

import json

# InfluxDB configurations

INFLUXDB\_ADDRESS = '192.168.0.129'

INFLUXDB\_USER = 'thars'

INFLUXDB\_PASSWORD = 'thars'

INFLUXDB\_DATABASE = 'kweekbak'

# MQTT configurations

MQTT\_BROKER = "tharspi.local"

MQTT\_PORT = 1883

MQTT\_TOPIC = "esp32/#"

MQTT\_USER = "thars"

MQTT\_PASSWORD = "thars"

# Initialize InfluxDB client

influx\_client = InfluxDBClient(

INFLUXDB\_ADDRESS,

8086,

INFLUXDB\_USER,

INFLUXDB\_PASSWORD,

INFLUXDB\_DATABASE

)

# Verify that we can connect to InfluxDB

try:

influx\_client.create\_database(INFLUXDB\_DATABASE)

print(f"Connected to InfluxDB at {INFLUXDB\_ADDRESS}")

except Exception as e:

print(f"Error connecting to InfluxDB: {e}")

def on\_connect(client, userdata, flags, rc):

print(f"Connected to MQTT Broker with result code {rc}")

if rc == 0:

print("Connection successful")

else:

print("Connection failed")

client.subscribe(MQTT\_TOPIC)

def on\_message(client, userdata, msg):

try:

payload = msg.payload.decode()

print(f"Received `{payload}` from `{msg.topic}` topic")

data = json.loads(payload)

print(f"Decoded JSON: {data}")

json\_body = [

{

"measurement": "sensor\_data",

"tags": {

"device": "esp32"

},

"fields": data

}

]

print(f"Writing data to InfluxDB: {json\_body}")

success = influx\_client.write\_points(json\_body)

if success:

print("Data written to InfluxDB successfully")

else:

print("Failed to write data to InfluxDB")

except json.JSONDecodeError as e:

print(f"Error decoding JSON: {e}")

except Exception as e:

print(f"Error: {e}")

# Initialize MQTT client

mqtt\_client = mqtt.Client()

mqtt\_client.username\_pw\_set(MQTT\_USER, MQTT\_PASSWORD)

mqtt\_client.on\_connect = on\_connect

mqtt\_client.on\_message = on\_message

# Connect to MQTT broker

mqtt\_client.connect(MQTT\_BROKER, MQTT\_PORT, 60)

# Start the MQTT client loop

mqtt\_client.loop\_forever()

# Hoe kan je het namaken?

Eerst zorg je ervoor dat je alle nodige hardware hebt:

ESP32, raspberry pi, moisture sensor, 5V relais, 5V waterpompje, LDR, ledstrip, DHT11, 5V servomotor, bodemtemperatuursensor DS18B20, led(deze dient voor te simuleren dat een warmtemat aangaat), ultrasone sensor HC-SR04. Een extra 5V voeding is ook nodig voor het pompje. Ook een doorzichtige bak en een plantenpotje voor de plant.

Test eerst alle hardware door het in elkaar te steken zoals in het schema. Je moet alleen nog de 5V pomp verbinden met de relais en de extra 5V bron. Je steekt de - van de pomp aan de - van de 5V bron en de + in de middelste com poort van de relais. de + van de 5V bron verbind je met de NO (normally open) poort van de relais.

Dan kopieer je de esp32 code en plak je die in je arduino IDE applicatie op je pc. Die verbind je met je esp32. Vergeet niet al de libraries die bij de code horen allemaal te downloaden. Verander in het bestand je wifi ssid en password zodat de esp kan verbinden. Verander ook de mqtt server naar de naam van jouw raspberry pi.

Verbind je raspberry pi met je internet via wifi of ethernet kabel en maak een python bestand en plak de raspberry pi code daarin. Maak ook in je influxdb een database aan genaamd kweekbak en een user met wachtwoord. Vergeet die niet aan te passen in de python code.

Nu mag je de code uploaden van arduino naar je esp32 en op je raspberry het python bestand runnen. Als alles juist gedaan is zou je op je raspberry te zien krijgen dat data naar een influxdb succesvol gestuurd worden. Voor te testen of het werkt kan je op influxdb op je raspberry een sql querry doen in je database voor alles te zien.

Nu kan je op grafana gaan door op je browser het ip adres van je raspberry te zetten. Voeg hier je kweekbak database toe als data source. Dan ga je naar dashboard een maak een nieuw visualisatie met dezelfde sql querry als op je influxdb en dan kan je je data zien in een grafiek.

Nu mag je van je python code een service maken zodat die werkt vanaf je de raspberry aanzet.

Als dat klaar is kan je proberen alles te solderen aan een prototype board of op het pcb bord maar die is nog nooit getest dus het kan zijn dat die niet werkt. Doe zeker genoeg tests voor te zien of het volledig werkt.

Dan mag je een behuizing maken. Wat ik heb gedaan is met dubbelzijdige tape de 3 prototype boards dat ik heb gemaakt plakken aan de binnenkant en voor watertank heb ik een plastieken fles open gesneden. Ik heb ook een aantal gaten voor kabels gemaakt voor de rfid, LCD, sonde van de waterpomp, relais, ultrasone sensor en de kabel voor de esp met de pc te verbinden voor te testen.

# Innovaties en Aanpassingsvermogen

## ****Modulair Ontwerp en RFID Integratie****

Met de integratie van een RFID lezer kan er ook met verschillende soorten kruiden gewerkt worden. Als er nu een verandering van kruid word beslist kan de plant eruit gehaald worden en een nieuwe erin gestoken. Dan pak je de RFID tag dat gemaakt is voor een specifieke plant en als je die scant zullen de vereisten van licht, vochtigheid, luchttemperatuur en grondtemperatuur aangepast worden zodat de plant de nieuwe plant de ideale groeiomgeving krijgt.

## ****A****utomatisering en Reactie op Omgevingsfactoren

De kweekbak zal zich aanpassen gebaseerd op verschillende omgevingsfactoren:

### Licht

Licht is heel belangrijk voor de groei van een plant en als die er niet is zal een plant ook niet willen groeien. Daarom wordt de lichthoeveelheid voortdurend gemeten met een LDR en zal er ook een ledstrip aangaan boven de plant om licht te geven aan de plant als er niet genoeg is. Hierdoor krijgt de plant altijd genoeg licht

### Luchttemperatuur

De temperatuur wordt ook gemeten in de kweekbak aangezien die gemaakt is zoals een klein serretje. Als de bak lang in de zon staat kan het wel zijn dat de temperatuur in de bak gaat stijgen en daarvoor hebben we een DHT11 in de bak staan. Deze meet de temperatuur in de bak en als die te hoog is zal er een raampje open gaan zodat de warme lucht kan ontsnappen voor droogte te voorkomen.

### Grondtemperatuur

De grondtemperatuur wordt gemeten met een DS18B20. Deze steek je in de potgrond en zal zo de temperatuur kunnen meten. Als deze te laag is kan er een warmtemat onder de bak aangaan zodat de temperatuur terug naar een ideale toestand kan gaan voor de plant.

### Grondvochtigheid

Water is natuurlijk ook heel belangrijk voor een plant. De plant haalt deze uit de grond via wortels en krijgt zo zijn voedingstoffen. Als de grondvochtigheid niet hoog genoeg is kan de plant niet groeien. De grondtemperatuur wordt gemeten door een capacitive moisture sensor. Zo kan er gezien worden wanneer er water in de grond moet. Als er water in de grond moet zal er een signaal gestuurd worden naar een relais dat verbonden is met een waterpompje. Dat waterpompje zal water via een buisje in de potgrond laten lopen. Het water voor dat pompje zit in een tank waar het waterniveau in gemeten wordt door een ultrasone sensor zodat er te zien is of er nog genoeg water is. Als die bijna leeg is zal er op de LCD een negatieve waterafstand staan. De watertank bijvullen is het enige wat niet geautomatiseerd is

# Conclusies en Toekomstig Werk:

## Projectevaluatie

Het project is af en meet de nodige data voor plantgroei. Het enige dat niet bestuurd wordt is de bodemtemperatuur maar voor de rest werkt alles. Het zou zeker gebruikt kunnen worden om een plant te laten groeien alleen zal er wel een krachtbron nog moeten toegevoegd worden aangezien ik als extra 5V bron de 5V pin gebruik van mijn arduino UNO verbonden met een 9V batterij.

## Aanbevelingen voor Toekomstige Verbeteringen

Het soldeerwerk kan zeker properder. De borden vast maken in de behuizing met schroeven ipv dubbelzijdige tape. Het aantal kabels ook proberen te verminderen en verstoppen in het design. Een echte warmtemat ipv het klein ledje zou ook tof zijn. Ook de ingestelde data met de rfid cards zoals bodemvochtigheid zijn allemaal placeholders voor te laten zien dat de hardware werkt en kan zeker vervangen worden naar data dat specifiek voor een plant is. Een betere spanningsbron voor de waterpomp en ook voor het hele systeem. Momenteel moet de esp vasthangen aan een computer en de raspberry in een stopcontact of ook aan een computer. Dit kan zeker vervangen worden door een andere bron

# Bibliografie:

De meest gebruikte bron is chatgpt. Ik heb bij de meeste componenten gevraagd voor een standaard testcode aangezien de meeste componenten maar 1 functie hebben en ik heb die dan proberen toe te passen bij mijn project. Als ik een error code had vroeg ik ook aan chatgpt wat het probleem was voor een error dat ik niet begreep. Voor de data naar de raspberry pi te sturen heb ik ook de suggestie van chatgpt gebruikt om de data in 2 json documenten te doen en de raspberry die documenten te laten ontcijferen en zo in een influxdb te steken.

Ik heb ook cursussen van verleden vakken gebruikt voor bijvoorbeeld grafana te laten werken of hoe ik MQTT gebruik.

Voor het aansluiten van componenten heb ik google images gebruikt en overal wat gekeken. Ik herinner niet dat ik op een afbeelding heb gedrukt voor meer details dus links naar websites heb ik niet.

Solderen op een prototype board heb ik geleerd via : <https://www.youtube.com/watch?v=l9Kbr8cPqOE>

# 

|  |
| --- |
| contact  Thars Van de Vel | Student  r0933637@student.thomasmore.be  Volg ons  www.thomasmore.be  fb.com/ThomasMoreBE  #WeAreMore |
|  |