2022

Smart Greenhouse

IOT PROJECT

BRYAN AERTGEERTS

Contents

Introductie

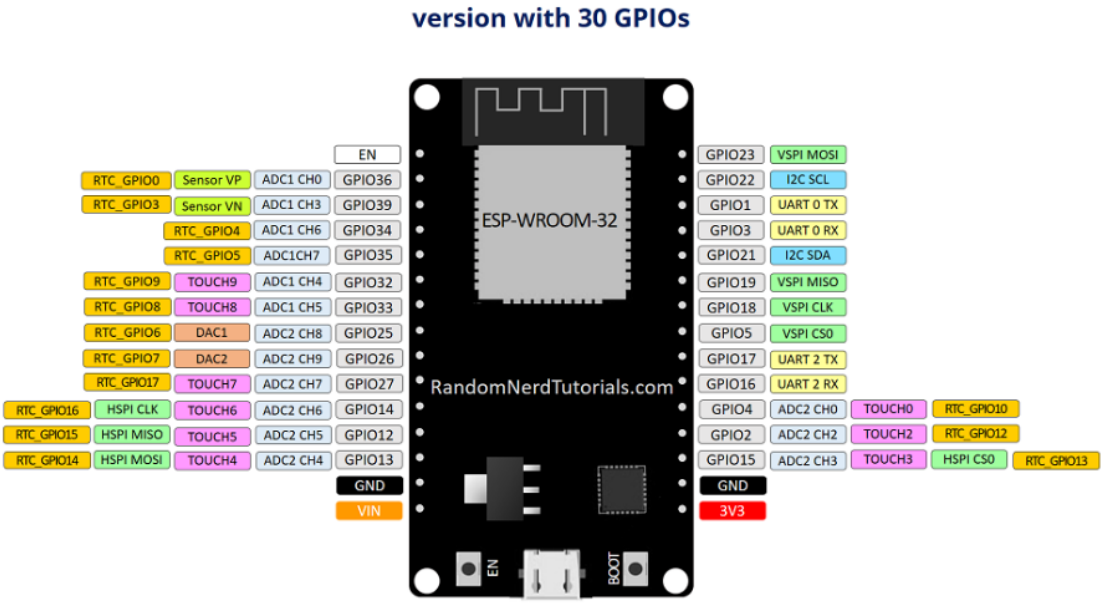
In dit project heb ik een smart greenhouse gemaakt. In dit document vertel ik over alles wat ik heb gebruikt om dit project tot stand te brengen. De eerste week ben ik begonnen met een schema en ontwerp te maken, daarna heb ik al het nodige materiaal besteld. Hierna ben ik aan de code begonnen. Na een paar errors weg te werken was dit gelukt. Na dat men componenten waren aangekomen ben ik begonnen met alles aan te sluiten en zo te testen, hier ondervond ik nog een paar errors die ik wist weg te werken niks groots. Toen dat werkte heb ik het aan mqtt verbonden en naar grafana laten versturen. In begin had ik hier moete mee maar na wat uitleg te vragen was me dit gelukt. Hierna was ik begonnen met het solderen van de printplaten, vermits ik niet goed ben in solderen was dit een hele bevalling voor mij maar uiteindelijk was dit gelukt. Vervolgens heb ik de serre in 1 gestoken en alles in elkaar gepast. Tijdens dit proces op de avond voor de presentatie is dit helemaal mis gelopen meer hier over in het onderdeel “esp32 error”. Met volgende link kan u mijn PowerPoint bezichtigen https://view.genial.ly/629618b55a233c0018759b00/dossier-reporting-smart-greenhouse.

Gebruikte hardware

ESP32

ESP32 is de naam van de chip die ontwikkeld werd door Espressif Systems. Deze biedt Wi-Fi- (en in sommige modellen) dual-mode Bluetooth-connectiviteit aan voor embedded apparaten. Hoewel ESP32 technisch gezien gewoon de chip is, worden modules en ontwikkelingsborden die deze chip bevatten door de fabrikant vaak ook "ESP32" genoemd. Er zijn verschilende moddelen op de afbeelding hieronder kunt u enkele zien, wij gebruiken de DOIT DEVKIT V1.

Hieronder ziet u een afbeelding van de pins van dit board

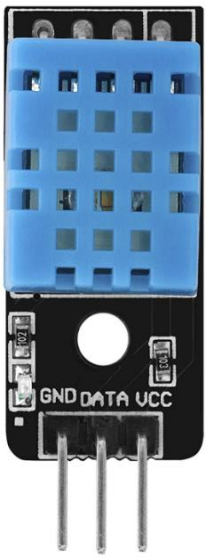


Hier is een tabel met de gegevens van dit board

Wi-Fi	2.4 GHz tot 150 Mbits/s
Bluetooth	BLE (Bluetooth Low Energy) en legacy Bluetooth
Architecture	32 bits
Clock frequency	tot 240 MHz
RAM	512 KB
Pins	30 or 36 (hangt af van het model)
Peripherals	Capacitieve aanraking, ADC (analoog naar digitaal omzetter), DAC (digitaal naar analoog omzetter), I2C (Inter-Integrated Circuit), UART (universele asynchrone ontvanger/zender), CAN 2.0 (Controller Area Netwokr), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulsbreedtemodulatie), en meer.

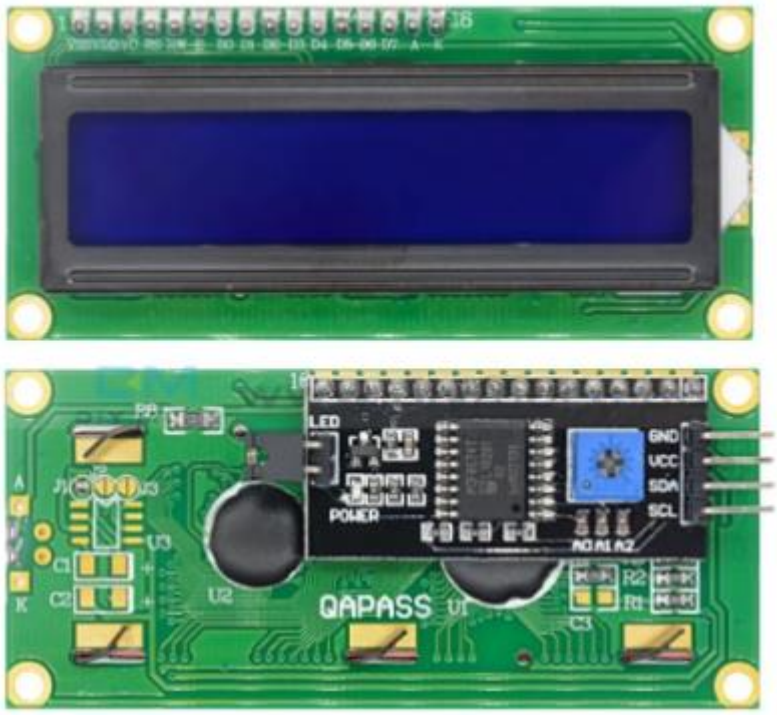
DHT11

De DHT11 is een eenvoudige, ultra-laaggeprijsde digitale temperatuur- en vochtigheidssensor. Hij gebruikt een capacitieve vochtigheidssensor en een thermistor om de omgevingslucht te meten en stuurt een digitaal signaal uit op de datapin (geen analoge ingangspinnen nodig).



I2C 16×2 LCD Screen

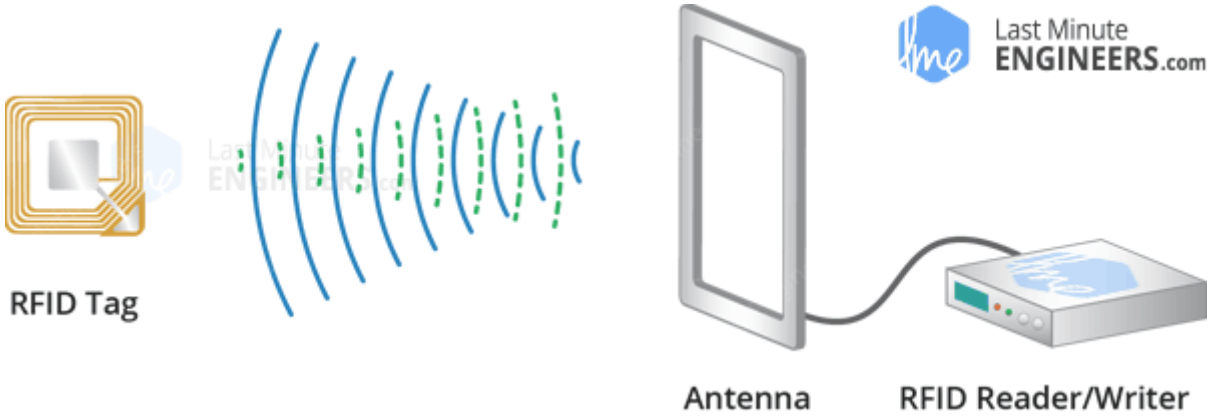
Het I2C 16×2 Arduino LCD-scherm maakt gebruik van een I2C-communicatie-interface. Het kan 16×2 karakters op 2 lijnen weergeven, witte karakters op blauwe achtergrond.



Dit display overwint het nadeel van de LCD 1602 Parallel LCD Display, waarbij je ongeveer 8 pinnen op je Arduino verspilt om het display werkend te krijgen. Gelukkig is in dit product een I2C-adapter direct op de pinnen van het display gesoldeerd. I2C is een door Philips ontwikkelde seriële bus, die gebruik maakt van twee tweerichtingslijnen, SDA (Serial Data Line) en SCL (Serial Clock Line) genaamd. Beide moeten worden aangesloten via weerstanden met een pull-up. De gebruiksspanningen zijn standaard 5V en 3,3V. Het LCD-display werkt met 5 Volt.

RFID reader

RFID of Radio Frequency Identification (radiofrequentie-identificatiesysteem) bestaat uit twee hoofdcomponenten, een transponder/tag die aan een te identificeren voorwerp wordt bevestigd, en een zendontvanger die ook bekend staat als ondervrager/lezer.



Hoe RFID-technologie werkt Lezer Schrijver Systeem Tag Communicatie

Een lezer bestaat uit een radiofrequentiemodule en een antenne die een hoogfrequent elektromagnetisch veld genereert. De tag is daarentegen meestal een passief apparaat, dat wil zeggen dat het geen batterij bevat. In plaats daarvan bevat hij een microchip die informatie opslaat en verwerkt, en een antenne om een signaal te ontvangen en uit te zenden.

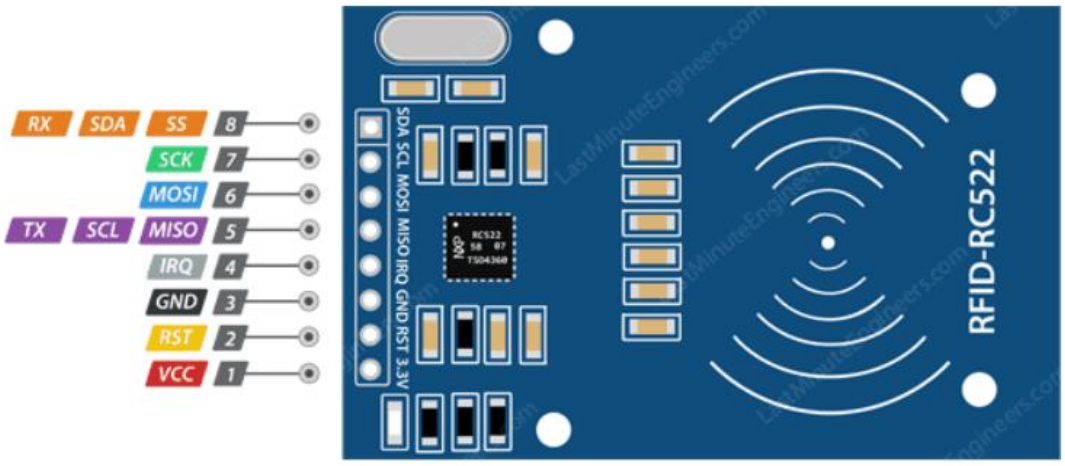
Om de op een tag gecodeerde informatie te lezen, wordt de tag in de onmiddellijke nabijheid van de lezer geplaatst (hoeft niet binnen het directe gezichtsveld van de lezer te zijn). Een lezer genereert een elektromagnetisch veld dat elektronen door de antenne van de tag doet bewegen en vervolgens de chip van energie voorziet.

De van stroom voorziene chip in de tag reageert vervolgens door zijn opgeslagen informatie terug te zenden naar de lezer in de vorm van een ander radiosignaal. Dit wordt backscatter genoemd. De backscatter, of verandering in de elektromagnetische/RF-golf, wordt gedetecteerd en geïnterpreteerd door de lezer, die de gegevens vervolgens doorstuurt naar een computer of microcontroller.

Hieronder is een tabel met de gegevens van de RFID reader

Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26mA
Min. Current(Power down)	10µA
Logic Inputs	5V Tolerant
Read Range	5 cm

Hieronder ziet u een afbeelding van de pins van dit board



VCC levert de voeding voor de module. Dit kan overal tussen 2,5 en 3,3 volt zijn. U kunt het met 3.3V output van uw Arduino verbinden. Vergeet niet dat het verbinden met 5V pin waarschijnlijk je module zal vernietigen!

RST is een input voor het Terugstellen en macht-verslaat. Wanneer deze speld laag gaat, wordt harde macht-versla toegelaten. Dit zet alle interne huidige putten met inbegrip van de oscillator uit en de inputspelden worden losgekoppeld van de buitenwereld. Bij de opgaande flank wordt de module gereset.

GND is de Grondspeld en moet met GND speld op Arduino worden verbonden.

IRQ is een interrupt pin die de microcontroller kan waarschuwen wanneer de RFID tag in zijn nabijheid komt.

MISO / SCL / Tx pin fungeert als Master-In-Slave-Out wanneer SPI interface is ingeschakeld, fungeert als seriële klok wanneer I2C interface is ingeschakeld en fungeert als seriële data-uitgang wanneer UART interface is ingeschakeld.

MOSI (Master Out Slave In) is de SPI-ingang van de RC522-module.

SCK (Seriële Klok) aanvaardt klokpulsen geleverd door de SPI bus Master i.e. Arduino.

SS / SDA / Rx pin fungeert als signaalingang wanneer SPI interface is ingeschakeld, fungeert als seriële data wanneer I2C interface is ingeschakeld en fungeert als seriële data ingang wanneer UART interface is ingeschakeld. Deze speld wordt gewoonlijk gemarkeerd door de speld in een vierkant in te sluiten zodat het kan worden gebruikt als referentie voor het identificeren van de andere spelden.

Groei LED strip

Om gewas te kweken is er naast water ook voldoende licht nodig. Niet al het licht is hiervoor zomaar geschikt. Het is nodig om de juiste balans te vinden tussen verschillende kleur en wit tinten. Hierbij wordt altijd gebruik gemaakt van de kleuren rood en blauw of een mengeling van deze 2. zodat deze ledstrips het perfecte licht afgeven voor al uw gewas. De led kweekstrip heeft veel voordelen ten op zichte van de de normale groeilamp. Een van de belangrijkste voordelen is het feit dat kweek ledstrips zeer energiezuinig zijn. Hierdoor heeft u veel minder onkosten. Deze groeilampen zijn speciaal ontworpen om natuurlijk zonlicht te vervangen, de fotosynthese te stimuleren en het juiste kleurenspectrum te bieden waarin de plant kan groeien en bloeien. Daarnaast zijn kweek ledstrips veel ruimtebesparender dan normale groeilampen, zodat u de kweekruimte veel praktischer kan inrichten. Daar komt bij dat ledstrips nauwelijks warmte afgeven, waardoor de kweekstrips dichterbij het gewas geplaatst kunnen worden.

Heater

PTC verwarmingselement wordt ook wel PTC thermistor genoemd, het is een soort halfgeleiderfunctioneel keramiek met een positieve temperatuurcoëfficiënt. Vóór de PTC-overgangstemperatuur neemt de weerstand af met de temperatuurstijging. Tussen de temperatuurovergangstemperatuur en de thermische op hol geslagen temperatuur neemt de weerstand aanzienlijk toe met de temperatuurstijging. PTC componenten hebben constante temperatuur verwarming, lange natuurlijke levensduur, energiebesparing, en geen open vlam, Goede veiligheidsprestaties, eenvoudige aanpassing van warmteopwekking en weinig invloed door de voeding spanning, enz.

Fan

Een ventilator is een draaiende machine met een draaiend onderdeel die ervoor zorgt dat lucht of een gas in beweging wordt gebracht. Een ventilator wordt gebruikt om bijvoorbeeld een ruimte te voorzien van verse lucht of om vervuilde lucht af te zuigen.

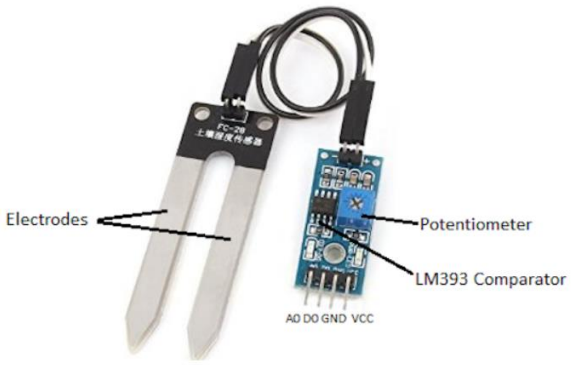
Ground moist sensor

De bodemvochtsensor bestaat uit twee sondes die worden gebruikt om het volumetrisch gehalte aan water te meten. De twee sondes laten de stroom door de bodem lopen, wat de weerstandswaarde oplevert om de vochtwaarde te meten. Als er water in de grond zit, zal deze meer elektriciteit geleiden, waardoor er minder weerstand zal zijn. Droge grond geleidt elektriciteit slecht, dus als er minder water is, zal de grond minder elektriciteit geleiden, wat betekent dat er meer weerstand zal zijn. Deze sensor kan analoog en digitaal worden aangesloten.

Hieronder is een tabel met de gegevens van de Ground moist sensor

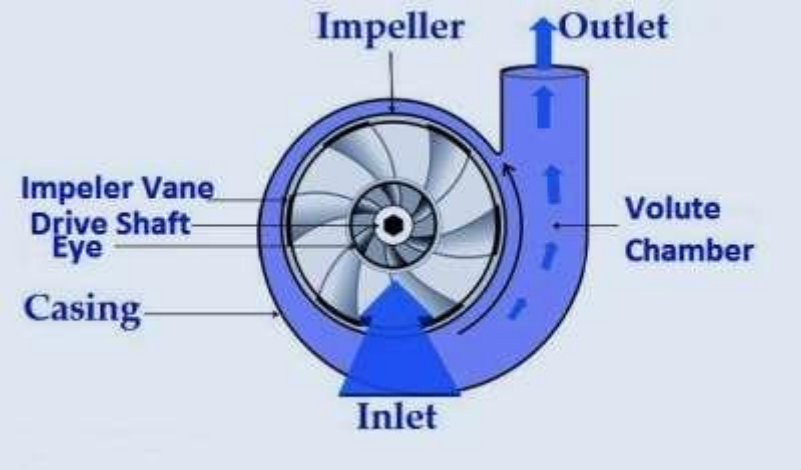
Input Voltage	3.3–5V
Output Voltage	0–4.2V
Input Current	35mA
Output Signal	analog en/of digital

Hieronder ziet u een afbeelding van de pins van de sensor



Pump

Een pomp is een mechanisch apparaat dat wordt gebruikt om water op te pompen van een laag naar een hoog drukniveau. In principe verandert de pomp de energiestroom van mechanisch naar vloeibaar. Dit kan worden gebruikt in processen waar een grote hydraulische kracht nodig is.

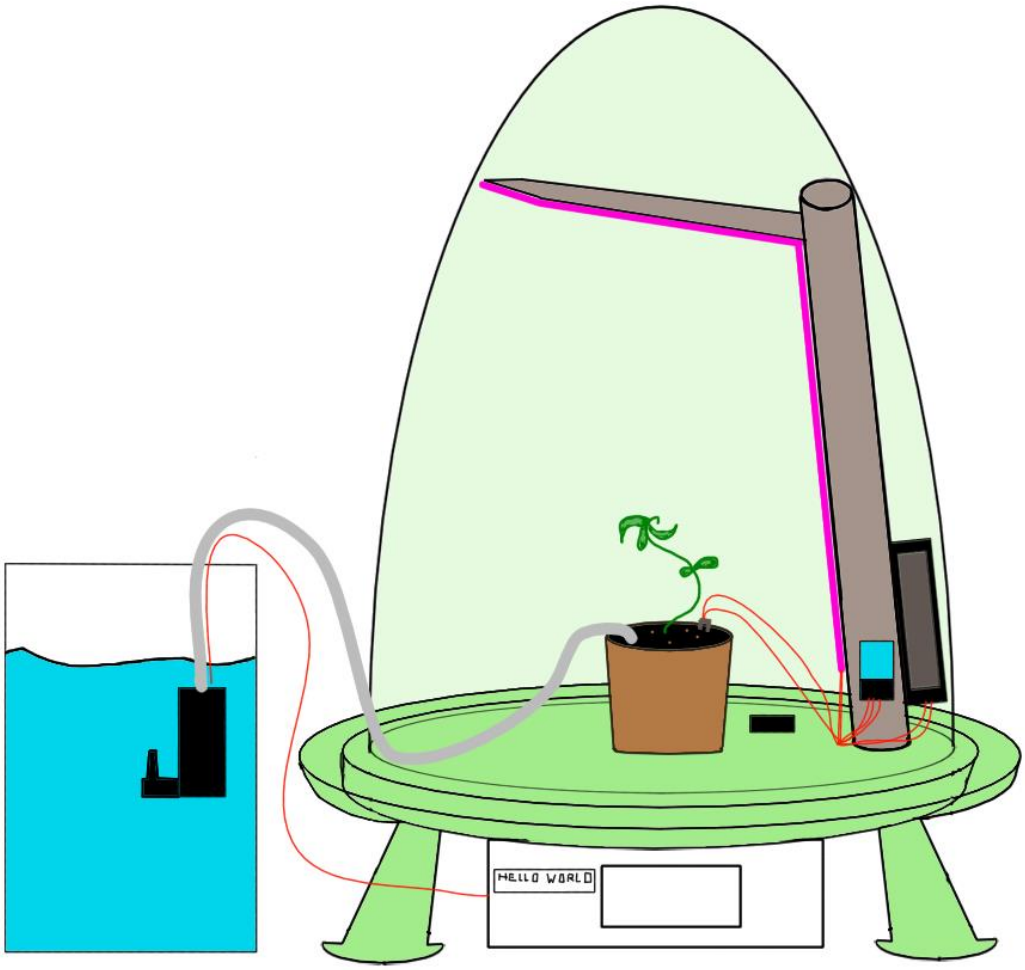


Het werkingsprincipe van een pomp is dat hij de druk van de vloeistof verhoogt om de stuwkracht te leveren die nodig is voor de doorstroming. Gewoonlijk is de drukfilterpomp een centrifugaalpomp, en het werkingsprincipe is dat de dunne modder de pomp binnendringt tijdens het roterende oog van de waaier die een cirkelvormige beweging informeert

Relais

Met een relais kan een relatief laag voltage gemakkelijk circuits met een hoger vermogen aansturen. Een relais bereikt dit door gebruik te maken van de 5V output van een Arduino pin om de elektromagneet te bekrachtigen die op zijn beurt een interne, fysieke schakelaar sluit om een circuit met hoger vermogen aan of uit te zetten.

Serre ontwerp



De bak bestaat uit meerdere samengevoegde onderdelen.

- De dome zelf is eigenlijk een drinksilo van de aveve https://shop.aveve.be/products/aveve-gerecycleerde-drinksilo-op-poten-10-l
- het waterreservoir is een plastieken bak
- een plastieken bak voor de componenten in met een uitsteksel voor de rfid reader
- houten stok voor componenten in dome op te bevestigen

ESP32 code

```
//toegevoegde libaries

#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <DHT.h>
#include <TaskScheduler.h>
#include <WiFi.h>
#include <PubSubClient.h>

//-------------------------------------------------------------------------------------------------------------------

const char* ssid = "LAPTOMIUM";                 //wifi naam
const char* password = "123456789";             //wifi wachtwoord
const char* mqttServer = "192.168.137.66";      //IP addres van de raspberry
const int mqttPort = 1883;                      //de gebruikte port
const char* mqttUser = "username";              //uw gebruikers naam van MQTT
const char* mqttPassword = "Admin";             //uw wachtwoord van MQTT
const char* clientID = "MQTTInfluxDBBridge";    // MQTT client ID

//-------------------------------------------------------------------------------------------------------------------

#define RST_PIN 2
#define SS_PIN 5

#define DHTPIN 4
#define DHTTYPE DHT11

#define GROWled 15
#define MOISTsensor 14
#define Pump 25
#define Heater 13
#define Cooler 12

//-------------------------------------------------------------------------------------------------------------------

//versympelde tijds aanduiding voor makelijker gebruik
const unsigned long SECOND = 1000;
const unsigned long MINUTE = 60 * SECOND;
const unsigned long HOUR = 60 * MINUTE;

//-------------------------------------------------------------------------------------------------------------------

int temp = 0;
int MOISTvalue;

//-------------------------------------------------------------------------------------------------------------------
```

```
//toegevoegde tasks voor de taskscedualer
void GROWledrucola();
Task ledrucola(24 * HOUR, TASK_FOREVER, &GROWledrucola);

void GROWledPlant2();
Task ledplant2(24 * HOUR, TASK_FOREVER, &GROWledPlant2);

void TEMPdisplay();
Task TEMPdis(10 * SECOND, TASK_FOREVER, &TEMPdisplay);

void GROUNDmoist();
Task moist(1 * HOUR, TASK_FOREVER, &GROUNDmoist);

void verwarming();
Task TEMPcontrol(10 * SECOND, TASK_FOREVER, &verwarming);

//--------------------------------------------------------------------------------------------------------------------

byte readCard[4];
String rucola = "93375299";  // REPLACE this Tag ID with your Tag ID!!!
String Plant2 = "B3B896D";
String tagID = "";

//--------------------------------------------------------------------------------------------------------------------

// Create instances
MFRC522 mfrc522(SS_PIN, RST_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(DHTPIN, DHTTYPE);
Scheduler runner;
WiFiClient espClient;
PubSubClient client(espClient);

//----------------------------------------------------------------------gdg-----------------------------------------

void setup()
{
  pinMode(GROWled, OUTPUT);
  digitalWrite(GROWled, LOW);

  pinMode(Pump, OUTPUT);
  digitalWrite(Pump, LOW);

  pinMode(Heater, OUTPUT);
  digitalWrite(Heater, LOW);

  pinMode(Cooler, OUTPUT);
  digitalWrite(Cooler, LOW);

  //--------------------------------------------------------------------------------------------------------------------
```

```
//eneble de toegevoegde tasks
  runner.init();
  runner.addTask(ledrucola);
  ledrucola.enable();

  runner.addTask(ledplant2);
  ledplant2.enable();

  runner.addTask(TEMPdis);
  TEMPdis.enable();

  runner.addTask(moist);
  moist.enable();

  runner.addTask(TEMPcontrol);
  TEMPcontrol.enable();

  //--------------------------------------------------------------------------------------------------------------------

  // Initiating
  Serial.begin(9600);
  SPI.begin(); // SPI bus
  mfrc522.PCD_Init(); // MFRC522
  WiFi.begin(ssid, password);

  //--------------------------------------------------------------------------------------------------------------------

  dht.begin();
  lcd.begin(); // LCD screen
  lcd.backlight();

  lcd.clear();
  lcd.print(" Access Control ");
  lcd.setCursor(0, 1);
  lcd.print("Scan Your Card>>");


  //--------------------------------------------------------------------------------------------------------------------

  //laat zien dat de wifi verbind, verbonden is of gefaald is met verbinden
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println("Connected to the WiFi network");
  client.setServer(mqttServer, mqttPort);
  dht.begin();
}

//--------------------------------------------------------------------------------------------------------------------
```

```
void loop()
{


 //-------------------------------------------------------------------------------------------------------------------------

 //laat zien dat de ESP32 verbind, verbonden is of gefaald is met verbinden met de MQTT server
 while (!client.connected()) {
  Serial.println("Connecting to MQTT...");
  if (client.connect("ESP32Client", mqttUser, mqttPassword ))
  {
   Serial.println("connected");
  } else
  {
   Serial.print("failed with state ");
   Serial.println(client.state());
   delay(2000);
  }
 }

 //-------------------------------------------------------------------------------------------------------------------------


 //Wait until new tag is available
 while (getID())
 {
  lcd.clear();
  lcd.setCursor(0, 0);

  if (tagID == rucola)
  {
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("RUCOLA");
   Serial.println("---------  nieuwe plant ingesteld ---------");
   Serial.println("rucola");
   Serial.println(tagID);
   String rucla = "100";
   client.publish("greenhouse/bak1/cardruc", rucla.c_str());
   String plant2 = "1";
   client.publish("greenhouse/bak2/cardplant2", plant2.c_str());
  }
  else if (tagID == Plant2)
  {
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Plant2");
   Serial.println("---------  nieuwe plant ingesteld ---------");
   Serial.println("Plant2");
   Serial.println(tagID);
   String rucla = "1";
   client.publish("greenhouse/bak1/cardruc", rucla.c_str());
```

```
    String plant2 = "100";
    client.publish("greenhouse/bak2/cardplant2", plant2.c_str());
   }
   else
   {
    lcd.print(" Access Denied!");
    Serial.println("---------  toegang gevraagd  ---------");
    Serial.println("nice try but nop");
    Serial.println(tagID);
    delay(10 * SECOND);
    lcd.clear();
    lcd.print(" Access Control ");
    lcd.setCursor(0, 1);
    lcd.print("Scan Your Card>>");
   }
  }
  runner.execute();
}

//-----------------------------------------------------------------------------------------------------------------------

//Read new tag if available
boolean getID()
{
  // Getting ready for Reading PICCs
  if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader continue
    return false;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) { //Since a PICC placed get Serial and continue
    return false;
  }
  tagID = "";
  for ( uint8_t i = 0; i < 4; i++) { // The MIFARE PICCs that we use have 4 byte UID
    //readCard[i] = mfrc522.uid.uidByte[i];
    tagID.concat(String(mfrc522.uid.uidByte[i], HEX)); // Adds the 4 bytes in a single String variable
  }
  tagID.toUpperCase();
  mfrc522.PICC_HaltA(); // Stop reading
  return true;
}

//-----------------------------------------------------------------------------------------------------------------------

void GROWledrucola()
{
  if (digitalRead(GROWled) == LOW && tagID == rucola)
  {
    digitalWrite(GROWled, HIGH);
    Serial.println("rucola   aan");
    String lichtONruc = "100";
    client.publish("greenhouse/bak1/licht", lichtONruc.c_str());
```

```
    delay(12 * HOUR);
  } else
  {
    digitalWrite(GROWled, LOW);
    Serial.println("rucola   uit");
    String lichtOFFruc = "1";
    client.publish("greenhouse/bak1/licht", lichtOFFruc.c_str());
  }
}
//--------------------------------------------------------------------------------------------------------------------

void GROWledPlant2()
{
  if (digitalRead(GROWled) == LOW && tagID == Plant2)
  {
    digitalWrite(GROWled, HIGH);
    Serial.println("Plant 2  aan");
    String lichtON = "100";
    client.publish("greenhouse/bak2/licht", lichtON.c_str());
    delay(10 * HOUR);
  } else
  {
    digitalWrite(GROWled, LOW);
    Serial.println("plant 2  uit");
    String lichtOFF = "1";
    client.publish("greenhouse/bak2/licht", lichtOFF.c_str());
  }
}

//--------------------------------------------------------------------------------------------------------------------

void TEMPdisplay()
{
  if (tagID == rucola || tagID == Plant2)
  {
    float t = dht.readTemperature();
    float h = dht.readHumidity();
    temp = t;
    lcd.setCursor(0, 1);
    lcd.print("Temp: ");
    lcd.setCursor(7, 1);
    lcd.print(t);
    Serial.println(t);
    lcd.setCursor(13, 1);
    lcd.print((char)223);
    lcd.setCursor(14, 1);
    lcd.print("C ");
    String humi = String(h);
    String tempi = String(t);

    if (tagID == rucola)
    {
```

```
    client.publish("greenhouse/bak1/humi", humi.c_str());
    client.publish("greenhouse/bak1/tempi", tempi.c_str());
   }
   else if (tagID == Plant2)
   {
    client.publish("greenhouse/bak2/humi", humi.c_str());
    client.publish("greenhouse/bak2/tempi", tempi.c_str());
   }
  }
}

//-----------------------------------------------------------------------------------------------------------------------

void GROUNDmoist()
{
  MOISTvalue = analogRead(MOISTsensor);
  MOISTvalue = map(MOISTvalue, 1024, 4095, 100, 0);
  Serial.print(MOISTvalue);
  Serial.println("%");
  String MOISTstring = String(MOISTvalue);
  if (tagID == rucola)
  {
   client.publish("greenhouse/bak1/moistsensor", MOISTstring.c_str());
  }
  else if (tagID == Plant2)
  {
   client.publish("greenhouse/bak2/moistsensor", MOISTstring.c_str());
  }
  if (MOISTvalue <= 50)
  {
   digitalWrite(Pump, HIGH);
   Serial.println("PUMP    aan");
   String PUMPaan = "100";
   if (tagID == rucola)
   {
    client.publish("greenhouse/bak1/pomp", PUMPaan.c_str());
   }
   else if (tagID == Plant2)
   {
    client.publish("greenhouse/bak2/pomp", PUMPaan.c_str());
   }
   delay(250);
   digitalWrite(Pump, LOW);
   String PUMPoff = "1";
   if (tagID == rucola)
   {
    client.publish("greenhouse/bak1/pomp", PUMPoff.c_str());
   }
   else if (tagID == Plant2)
   {
    client.publish("greenhouse/bak2/pomp", PUMPoff.c_str());
   }
```

```
  }
}

//-----------------------------------------------------------------------------------------------------------------------

void verwarming()
{
 if (tagID == rucola || tagID == Plant2)
 {
  if (temp < 15)
  {
   digitalWrite(Cooler, LOW);
   digitalWrite(Heater, HIGH);
   String HEATON = "100";
   if (tagID == rucola)
   {
    client.publish("greenhouse/bak1/heater", HEATON.c_str());
   }
   else if (tagID == Plant2)
   {
    client.publish("greenhouse/bak2/heater", HEATON.c_str());
   }
  }
  if (temp >= 15)
  {
   digitalWrite(Cooler, LOW);
   digitalWrite(Heater, LOW);
   String HEATOFF = "1";
   if (tagID == rucola)
   {
    client.publish("greenhouse/bak1/heater", HEATOFF.c_str());
   }
   else if (tagID == Plant2)
   {
    client.publish("greenhouse/bak2/heater", HEATOFF.c_str());
   }
  }
  if (temp <= 17)
  {
   digitalWrite(Cooler, LOW);
   digitalWrite(Heater, LOW);
   String COOLEROFF = "1";
   if (tagID == rucola)
   {
    client.publish("greenhouse/bak1/cooler", COOLEROFF.c_str());
   }
   else if (tagID == Plant2)
   {
    client.publish("greenhouse/bak2/cooler", COOLEROFF.c_str());
   }
  }
  if (temp > 17)
```

```
  {
    digitalWrite(Cooler, HIGH);
    digitalWrite(Heater, LOW);
    String COOLERON = "100";
    if (tagID == rucola)
    {
      client.publish("greenhouse/bak1/cooler", COOLERON.c_str());
    }
    else if (tagID == Plant2)
    {
      client.publish("greenhouse/bak2/cooler", COOLERON.c_str());
    }
  }
 }
}

//-----------------------------------------------------------------------------------------------------------------------
```

Raspberry code

De raspberry maakt gebruik van het bridge script en influxdb

```
import re
from typing import NamedTuple
import paho.mqtt.client as mqtt
from influxdb import InfluxDBClient
INFLUXDB_ADDRESS = '192.168.0.8'
INFLUXDB_USER = 'mqtt'
INFLUXDB_PASSWORD = 'mqtt'
INFLUXDB_DATABASE = 'weather_stations'
MQTT_ADDRESS = '192.168.0.8'
MQTT_USER = 'cdavid'
MQTT_PASSWORD = 'cdavid'
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': {
                'location': sensor_data.location
            },
            'fields': {
                'value': sensor_data.value
            }
        }
```

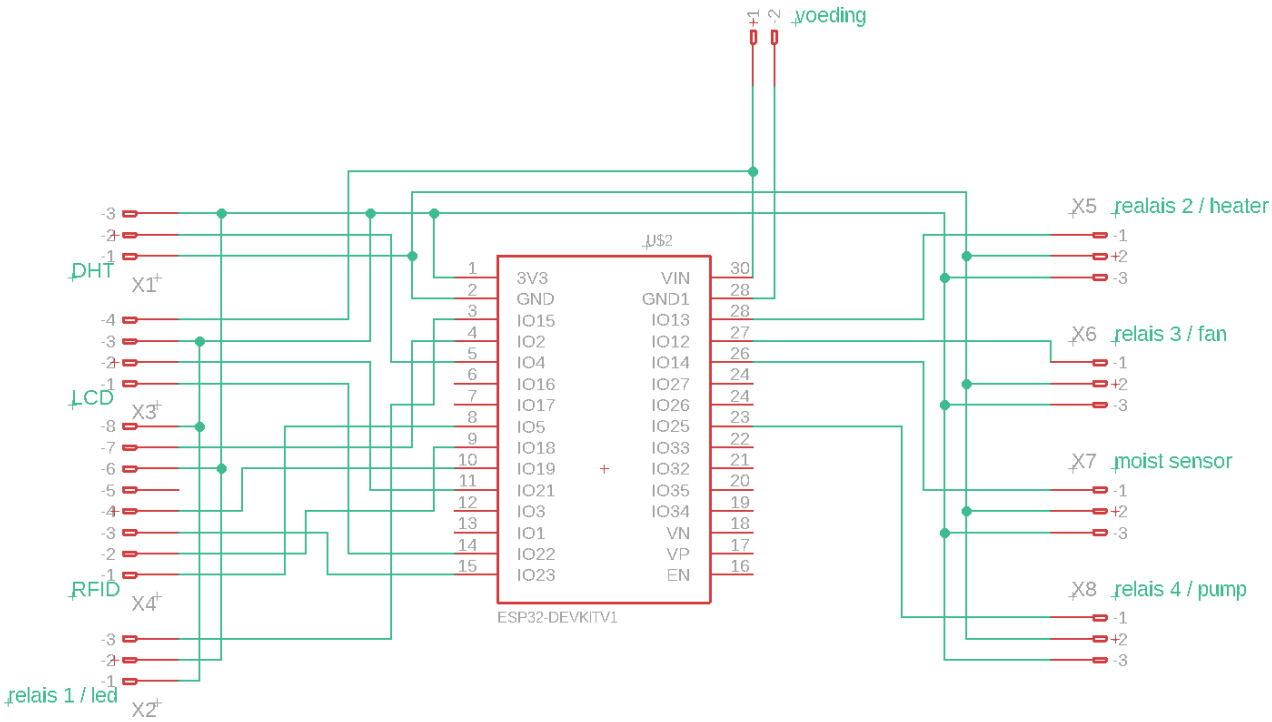
```
    ]
    influxdb_client.write_points(json_body)
def on_message(client, userdata, msg):
    """The callback for when a PUBLISH message is received from
the server."""

    print(msg.topic + ' ' + str(msg.payload))
    sensor_data = _parse_mqtt_message(msg.topic,
msg.payload.decode('utf-8'))
    if sensor_data is not None:
        _send_sensor_data_to_influxdb(sensor_data)
def _init_influxdb_database():
    databases = influxdb_client.get_list_database()
    if len(list(filter(lambda x: x['name'] == INFLUXDB_DATABASE,
databases))) == 0:
        influxdb_client.create_database(INFLUXDB_DATABASE)
    influxdb_client.switch_database(INFLUXDB_DATABASE)
def main():
    _init_influxdb_database()
    mqtt_client = mqtt.Client(MQTT_CLIENT_ID)
    mqtt_client.username_pw_set(MQTT_USER, MQTT_PASSWORD)
    mqtt_client.on_connect = on_connect
    mqtt_client.on_message = on_message
    mqtt_client.connect(MQTT_ADDRESS, 1883)
    mqtt_client.loop_forever()
if __name__ == '__main__':
    print('MQTT to InfluxDB bridge')
    main()
```

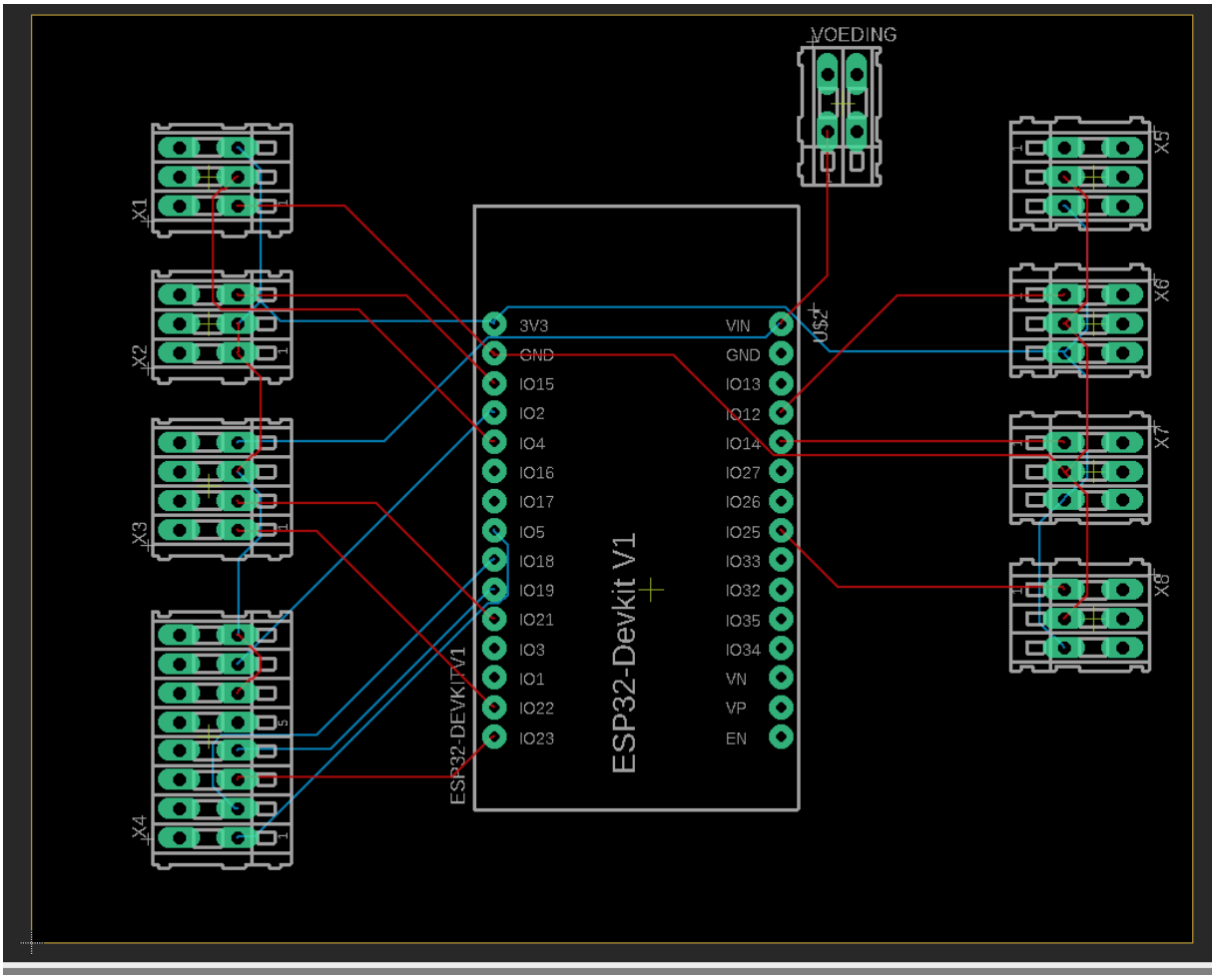
Grafana

Met Grafana kunt u uw statistieken opvragen, visualiseren, waarschuwen en begrijpen, ongeacht waar ze zijn opgeslagen. Creëer, verken en deel prachtige dashboards met uw team en bevorder een datagestuurde cultuur.

Hier zijn afbeeldingen van de grafana van dit project

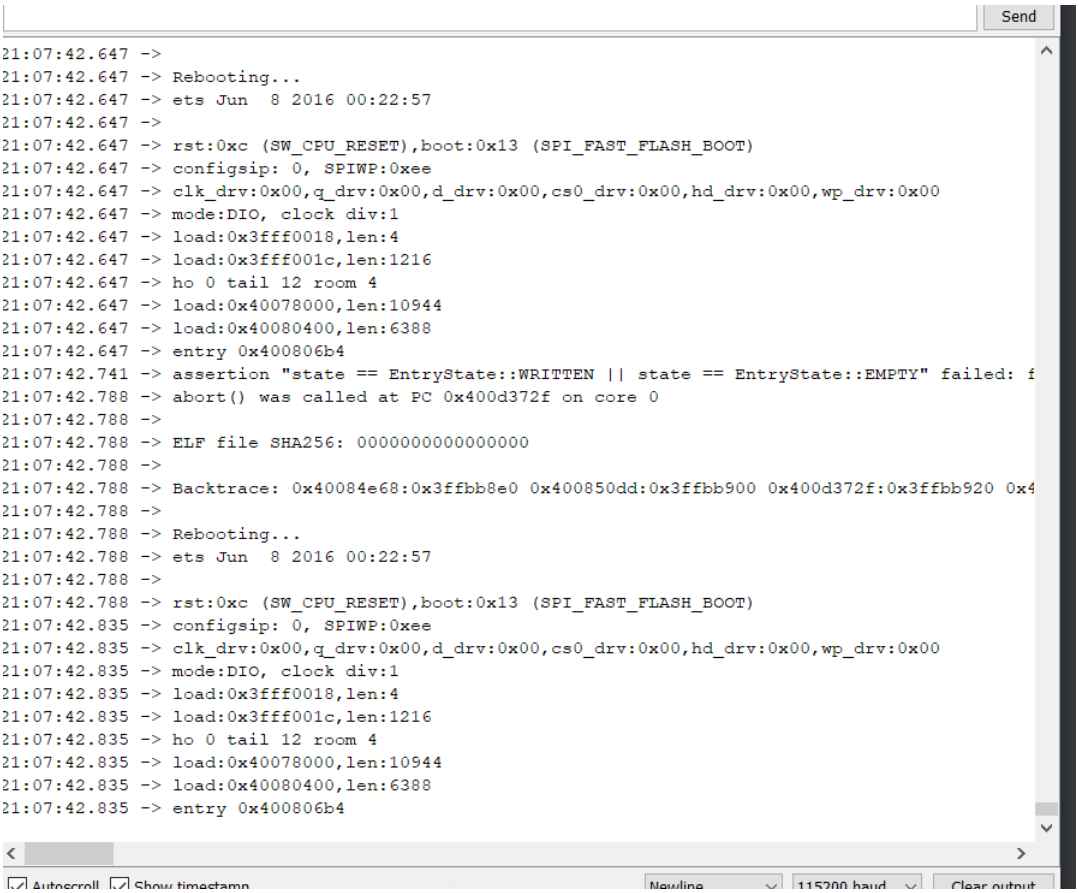
Schema & PCB


voeding
DHT
X1
LCD
X3
RFID
X4
relais 1 / led
X2
U$2
3V3
GND
IO15
IO2
IO4
IO16
IO17
IO5
IO18
IO19
IO21
IO3
IO1
IO22
IO23
VIN
GND1
IO13
IO12
IO14
IO27
IO26
IO25
IO33
IO32
IO35
IO34
VN
VP
EN
ESP32-DEVKITV1
X5 realais 2 / heater
X6 relais 3 / fan
X7 moist sensor
X8 relais 4 / pump

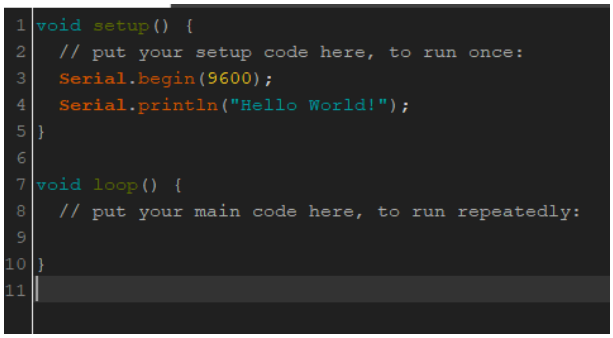

VOEDING
X1
X2
X3
X4
X5
X6
X7
X8
U$2
ESP32-Devkit V1
ESP32-DEVKITV1
3V3
GND
IO15
IO2
IO4
IO16
IO17
IO5
IO18
IO19
IO21
IO3
IO1
IO22
IO23
VIN
GND
IO13
IO12
IO14
IO27
IO26
IO25
IO33
IO32
IO35
IO34
VN
VP
EN

Esp32 error

Op de laatste dag is de esp32 in pannen gevallen. Na dat ik een programma uplouden begon hij in een eeuwige rebootloop te gaan. Hier onder kunt u de error zien die hij in serial monitor printen.



Hierna dacht ik een andere code te uplouden dus had ik snel een hello world programma geuploud.



```
1  void setup() {
2    // put your setup code here, to run once:
3    Serial.begin(9600);
4    Serial.println("Hello World!");
5  }
6
7  void loop() {
8    // put your main code here, to run repeatedly:
9
10 }
11
```

Dit echter gaf dezelfde error. Hierna dacht ik dat het misschien aan de micro usb kabel lag, dus heb ik een andere kabel gebruikt. Kreeg ik het zelfde probleem. Toen begon ik met dit probleem op te zoeken, ik kwam al snel op forums gelijk aardige problemen tegen. Na enkele van die oplossingen te proberen werkte het echter nog altijd niet. Na de resterende tijd naar oplossingen hebben gezochten en hulp aan 2 mensen hebben gevraagd heb ik niet een oplossing hiervoor kunnen vinden.