THOMAS MORE

EINDPROJECT 2024

IoT-Serre

Ruben Goovaerts IoT

2023 - 2024

1. Technische aspect

In het eerste deel van deze samenvatting, wordt het technische aspect van het project samengevat. Hier wordt de gebruikte hardware besproken, hoe je de broker moet installeren, en de benodigdheden voor het project.

1.1 Hardware:

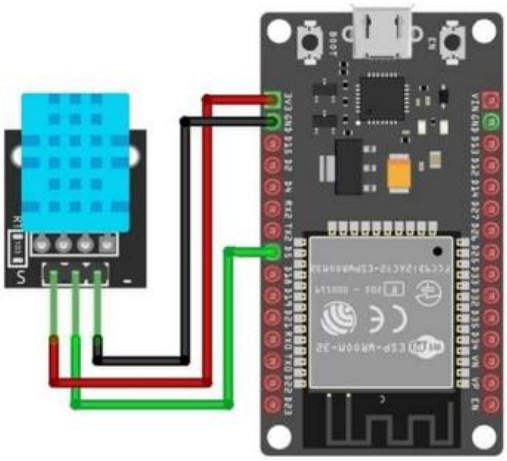
- LCD-Scherm
- DHT-11
- BD18b20
- 2x Relais
- Ventilator
- Led-Strip
- 12V voeding
- ESP32
- LED's
- Raspberry Pi
- RFID-Reader
- ID-Kaarten
- 12V pomp

1.2 Benodigdheden:

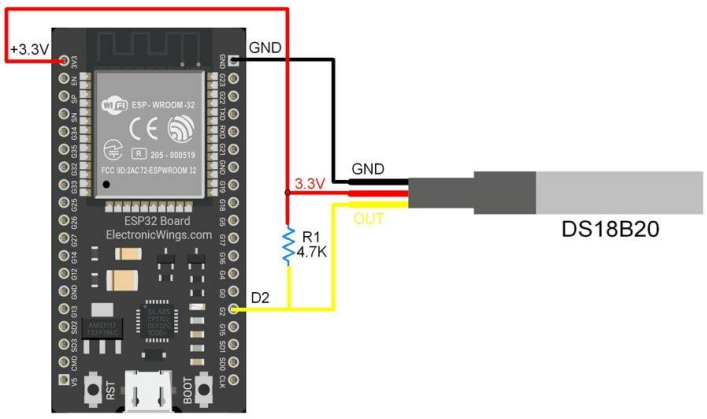
- Oude plantenbak
- Prototypingboard
- Draad (Rood, Geel en Blauw)
- Soldeerbout
- Soldeertin
- Micro-USB voedingskabel

2. Stappenplan

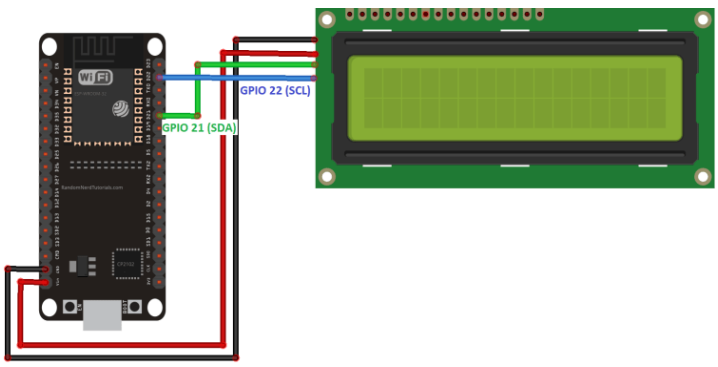
2.1 Sluit de DHT-11 aan op het prototype-bord:



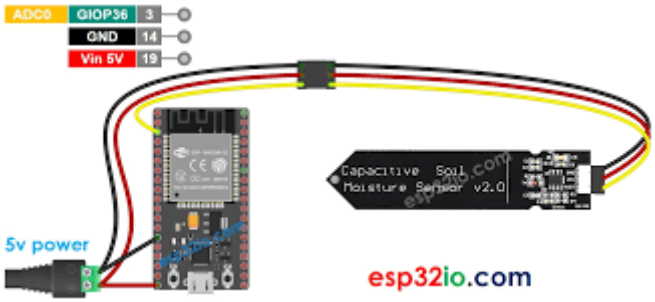
2.2 Sluit de DB18b20 sensor aan op de ESP32: (Let op de 4.7k Ohm weerstand tussen de voeding en de data)



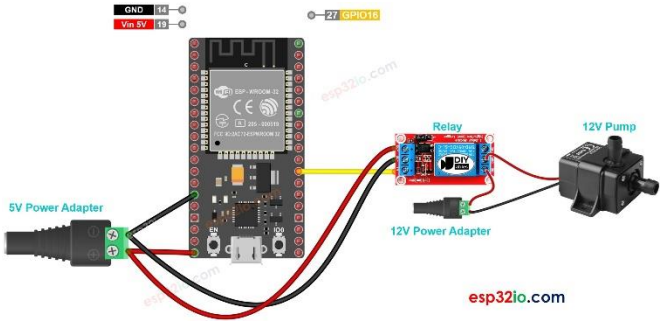
2.3 Sluit het I2C-LCD-Scherm aan op de ESP32:



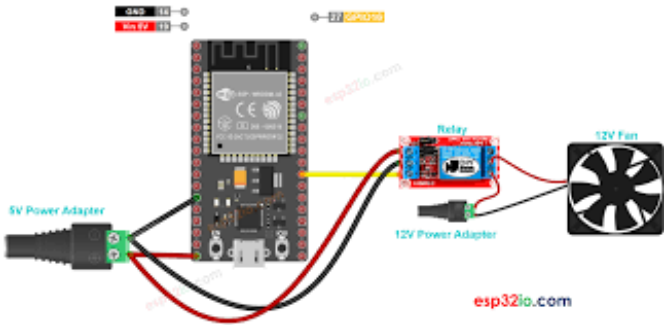
2.4 Sluit de bodemvochtigheidssensor aan op de ESP32:



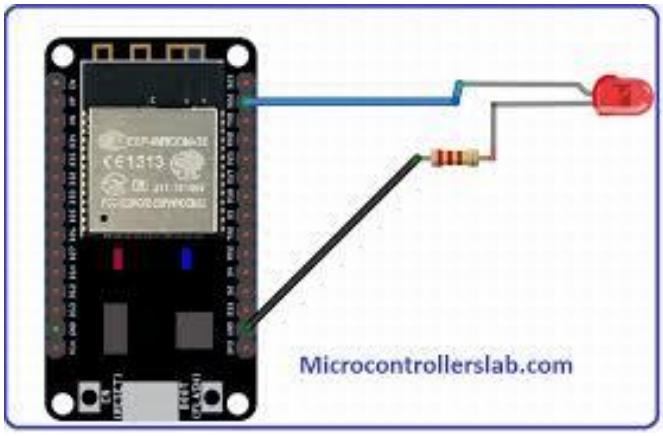
2.5 Sluit de pomp aan op de ESP-32:



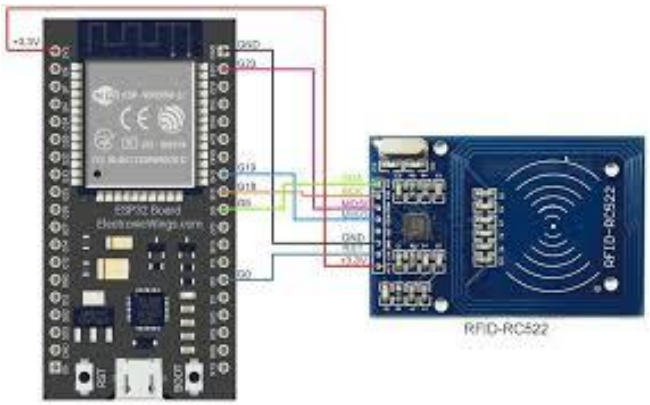
2.6 Sluit de ventilator aan op de ESP-32:



2.7 Sluit de ‘Verwarming’ (LED) aan op de ESP-32:



2.8 Sluit de RFID-Sensor aan op de ESP-32:



3. Broker configureren

3.1 Installeer Mosquitto (MQTT) op de raspberrypi:

```
ruben@rubsberrypi:~ $ sudo apt install mosquitto mosquitto-clients -y
ruben@rubsberrypi:~ $ sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

3.2 Installeer Python op de Pi:

```
ruben@rubsberrypi:~ $ sudo apt update
sudo apt install python3-venv

python3 -m venv myenv


source myenv/bin/activate

pip install paho-mqtt influxdb
```

3.3 Installeer InfluxDB op je Pi:

```
ruben@rubsberrypi:~ $ sudo apt install influxdb
ruben@rubsberrypi:~ $ sudo systemctl start influxdb
```

3.4 Enable de InfluxDB-service wanneer je de pi boot:

```
ruben@rubsberrypi:~ $ sudo systemctl enable influxdb
```

3.5 Configureer InfluxDB:

```
ruben@rubsberrypi:~ $ influx
Connected to http://localhost:8086 version 1.6.7~rc0
InfluxDB shell version: 1.6.7~rc0
>
```

3.6 Maak een database aan op InfluxDB:

```
> CREATE DATABASE Serre;
```

3.7 Druk op enter en exit:

```
> exit
```

3.8 Maak een Pythonscript aan op de pi, waarop we de MQTT topics naar influxDB gaan schrijven:

```
ruben@rubsberrypi:~ $ sudo nano Serre.py
```

3.9

Importeer MQTT, influxDB en json. Vermeld zeker het IP van de broker, en kijk na op welke poort de MQTT-berichten binnenkomen. Als je een account hebt gemaakt op influxDB (dat is het veiligste), vergeet dan niet om je gegevens van je profiel in the voeren in de Python-code. De topics van MQTT worden hier al gemaakt, deze namen zullen ook voorkomen in de code die we schrijven voor de ESP32.

```python
import paho.mqtt.client as mqtt
from influxdb import InfluxDBClient
import json

# MQTT settings
MQTT_BROKER = "192.168.1.55"
MQTT_PORT = 1883
MQTT_TOPICS = [("esp32/temperature", 0), ("esp32/humidity", 0), ("esp32/soil_moisture", 0),
               ("esp32/led", 0), ("esp32/heater", 0), ("esp32/fan", 0), ("esp32/water_pump", 0)]

# InfluxDB settings
INFLUXDB_ADDRESS = 'localhost'
INFLUXDB_PORT = 8086
INFLUXDB_USER = 'root'
INFLUXDB_PASSWORD = 'root'
INFLUXDB_DATABASE = 'Serre'

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    for topic in MQTT_TOPICS:
        client.subscribe(topic)

def on_message(client, userdata, msg):
    print(f"Topic: {msg.topic}, Message: {msg.payload.decode()}")
    try:
        payload = float(msg.payload.decode())
    except ValueError:
        payload = msg.payload.decode()

    json_body = [
        {
            "measurement": msg.topic,
            "fields": {
                "value": payload
            }
        }
    ]
    influxdb_client.write_points(json_body)

def setup_influxdb():
    client = InfluxDBClient(INFLUXDB_ADDRESS, INFLUXDB_PORT, INFLUXDB_USER, INFLUXDB_PASSWORD, None)
    databases = client.get_list_database()
    if not any(db['name'] == INFLUXDB_DATABASE for db in databases):
        client.create_database(INFLUXDB_DATABASE)
    client.switch_database(INFLUXDB_DATABASE)
    return client
```

3.10 Run het Pythonbestand:

```
ruben@rubsberrypi:~ $ sudo python Serre.py
```

3.11 Activeer de influxDB-database:

```
ruben@rubsberrypi:~ $ influx
Connected to http://localhost:8086 version 1.6.7~rc0
InfluxDB shell version: 1.6.7~rc0
> USE Serre;
Using database Serre
>
```

4. Code ESP32

4.1 Libraries

Hier worden alle nodige libraries toegevoegd:

```
#include <DHT.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFi.h> // Include the WiFi library for ESP32
#include <WiFiClient.h>
#include <Wire.h> // Include the Wire library for I2C communication
#include <LiquidCrystal_I2C.h> // Include the LiquidCrystal_I2C library for I2C LCD screen
#include <PubSubClient.h> // Include the MQTT library
#include <MFRC522.h> // Include the MFRC522 library for RFID
```

4.2 Mqtt-constants:

```
const char* ssid = "GOOVAERTSBUREAU";
const char* password = "0475876973";
const char* mqtt_server = "192.168.0.146"; // MQTT broker address
const int mqtt_port = 1883; // MQTT broker port
```

Geef hier jouw SSID in, en het wachtwoord van de wifi. Het ip-adres van de broker en de juiste poort (1883 is standaard, je kan dit aanpassen)

4.3 Pins

```
const int soilMoisturePin = 34; // Digital pin used for soil moisture sensor
const int dhtPin = 4;
const int relayPin = 2; // Pin for the relay and fan
const int heaterPin = 15; // Pin for the heater
const int waterPumpPin = 27; // Pin for the water pump
const int ledPin = 26; // Pin for the LED
const int ds18b20Pin = 14; // Pin for DS18B20 sensor
```

```
// RFID pins
#define SS_PIN 5
#define RST_PIN 13
```

(Pins voor de RFID)

Hier worden de juiste pins voor elke sensor/relay toegevoegd. Je kan ook andere namen voor de sensoren kiezen. Pas deze dan ook aan in de code.

4.4 Definieer de ID's van de ID-Kaarten: (je kan de ID's uitlezen in een example code op arduino)

```
byte card1UID[] = {0xA3, 0x81, 0x54, 0x28};
byte card2UID[] = {0xF3, 0xAE, 0x3C, 0x28};
```

4.5 Functie voor het printen van temperatuur & vochtigheid:

```
DisplayMode currentDisplay = TEMPERATURE_HUMIDITY;
unsigned long displayChangeTime = 0;
const unsigned long displayDuration = 5000; // 5 seconds
```

4.7 Wifi-setup + prints voor serial monitor:

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

4.8 Reconnect-loop voor de mqtt-broker:

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP32Client")) {
      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);
    }
  }
}
```

4.9 Initialiseren van de sensorpins & lcd-scherm:

```
void setup() {
  Serial.begin(115200); // Increase the baud rate for faster communication
  delay(10);

  setup_wifi(); // Connect to WiFi

  client.setServer(mqtt_server, mqtt_port);

  dht.begin();
  sensors.begin();
  SPI.begin(); // Initialize SPI bus
  rfid.PCD_Init(); // Initialize the RFID reader

  pinMode(relayPin, OUTPUT);
  pinMode(heaterPin, OUTPUT);
  pinMode(waterPumpPin, OUTPUT);
  pinMode(ledPin, OUTPUT); // Set LED pin as output
  pinMode(relayPin, INPUT_PULLUP); // Set pin for the fan as input with internal pull-up resistor

  lcd.init(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight

  Serial.println("Setup completed.");
}
```

4.10 Functie voor nieuwe RFID-kaart:

```
if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
  byte *uid = rfid.uid.uidByte;
```

4.11 Kaart 1 Peper-Mode:

Als de kaart overeenkomt met de ID van kaart 1, dan zal de Peper-Mode worden geactiveerd, en zal de maximumtemperatuur worden ingesteld op 25°C. Ook zal de “Peper-Mode” voor 5 seconden op het LCD-scherm geprint worden.

```
if (isSameUID(uid, card1UID)) {
  // Card 1 detected
  fanThreshold = 25.0;
  Serial.println("Card 1 detected");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Peper-Mode");
  currentDisplay = RFID_MODE;
  displayChangeTime = millis() + displayDuration;
```

4.12 Kaart 2 Tomaten-Mode:

Als de kaart overeenkomt met de ID van kaart 2, dan zal de Tomaten-Mode geactiveerd worden, en zal de maximumtemperatuur worden ingesteld op 28°C. Ook zal de “Tomaten-Mode” voor 5 seconden op het LCD-Scherm gerpint worden.

```
} else if (isSameUID(uid, card2UID)) {
  // Card 2 detected
  fanThreshold = 28.0;
  Serial.println("Card 2 detected");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Tomaten-Mode");
  currentDisplay = RFID_MODE;
  displayChangeTime = millis() + displayDuration;
```

4.13 Ongeldige kaart:

Als er een ongeldige kaart tegen de RFID-sensor gehouden wordt, dan zal er " Ongeldige kaart" geprint worden, en zal er niets veranderen.

```
    } else {
      // Invalid card detected
      Serial.println("Ongeldige Kaart");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Ongeldige Kaart");
    }
```

4.14 LED-strip

Hier wordt de status van de LED-strip geprint, en gepublished op het topic esp32/led.

```
  // Print LED state
  if (digitalRead(ledPin) == HIGH) {
    Serial.println("LED turned on");
    client.publish("esp32/led", "on");
  } else {
    Serial.println("LED turned off");
    client.publish("esp32/led", "off");
  }
}
```

4.15 Floats & Int

Hier worden de floats en int gedefinieerd voor een aantal sensoren.

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
int soilMoisture = analogRead(soilMoisturePin); // Read soil moisture
int fanState = digitalRead(relayPin); // Read the fan state
```

4.16 Publish aan broker via MQTT

Hier worden de topics van de DHT11, bodemvochtigheid en de ventilator gepublished naar de broker.

```
client.publish("esp32/temperature", String(temperature).c_str());
client.publish("esp32/humidity", String(humidity).c_str());
client.publish("esp32/soil_moisture", String(soilMoisture).c_str());
client.publish("esp32/fan", fanState == HIGH ? "on" : "off");
```

4.17 Sensordata aan serial monitor

Hier wordt de sensordata naar de serial monitor geprint.

```
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println(" °C");
Serial.print("Humidity: ");
Serial.print(humidity);
Serial.println(" %");
Serial.print("Soil Moisture: ");
Serial.println(soilMoisture);
Serial.print("Fan State: ");
Serial.println(fanState);
```

4.18 Tresholds

Hier wordt bepaald welke sensor wanneer geactiveerd wordt. Als de temperatuur kleiner is dan 21 graden, dan zal de verwarming aanspringen. Is de temperatuur gorter dan de ingestelde temperatuur (afhankelijk van welke mode) dan zal de ventilator aanspringen.

```
if (!isnan(temperature) && !isnan(humidity)) {
  if (temperature < 21.0) { // Adjusted threshold for heater
    digitalWrite(heaterPin, HIGH);
    digitalWrite(relayPin, LOW);
    Serial.println("Heater turned on");
    client.publish("esp32/heater", "on");
  } else if (temperature >= fanThreshold) {
    digitalWrite(relayPin, HIGH);
    digitalWrite(heaterPin, LOW);
    Serial.println("Fan turned on");
    client.publish("esp32/fan", "on");
  } else {
    digitalWrite(heaterPin, LOW);
    digitalWrite(relayPin, LOW);
    Serial.println("Heater turned off");
    Serial.println("Fan turned off");
    client.publish("esp32/heater", "off");
    client.publish("esp32/fan", "off");
  }
```

4.19 Soil Moisture

Hier wordt de data van de soilmoisturesensor uitgelezen. Als de vochtigheid lager is dan 50, dan zal de pomp aanspringen. Is de vochtigheid hoger dan 70, dan zal de pomp uitspringen.

```
    if (soilMoisture < 50) {
      digitalWrite(waterPumpPin, HIGH);
      delay(1000); // Run water pump for 1 second
      digitalWrite(waterPumpPin, LOW);
      Serial.println("Water pump turned on for 1 second");
      client.publish("esp32/water_pump", "on");
      delay(5000); // Delay for stability
    } else if (soilMoisture > 70) {
      digitalWrite(waterPumpPin, LOW);
      Serial.println("Water pump turned off");
      client.publish("esp32/water_pump", "off");
    }
  } else {
    Serial.println("Failed to read from DHT sensor!");
  }

  delay(5000); // Delay before next sensor reading
}
```

5. Grafana op de pi

5.1 Installeer Grafana op de Raspberry Pi:

```
ruben@rubsberrypi:~ $ wget https://dl.grafana.com/oss/release/grafana_8.3.5_armhf.deb
```

```
ruben@rubsberrypi:~ $ sudo dpkg -i grafana_8.3.5_armhf.deb
```

5.2 Start de Grafana-service:

```
ruben@rubsberrypi:~ $ sudo systemctl start grafana-server
```

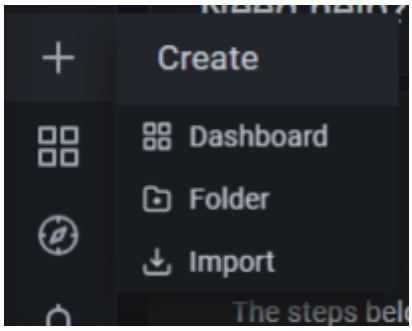
5.3 Start Grafana bij het booten van de Pi:

```
ruben@rubsberrypi:~ $ sudo systemctl enable grafana-server
```

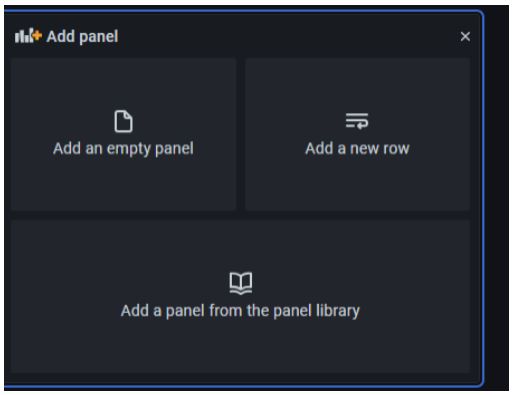
5.4 Configureer Grafana

Geef http://"jouwip":3000 in in de browser

5.5 Creëer een dashboard "Serre"

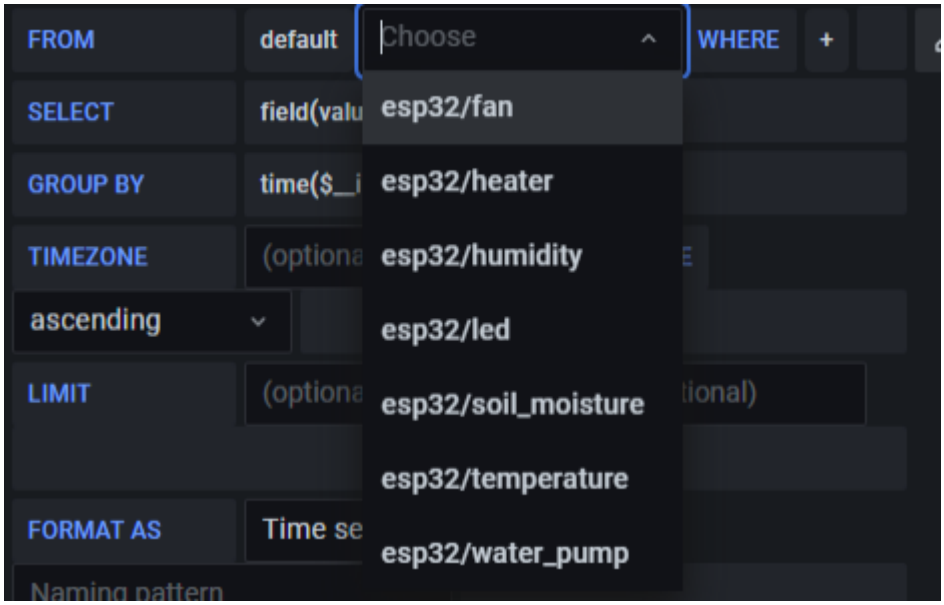


5.6 Add empty panel:

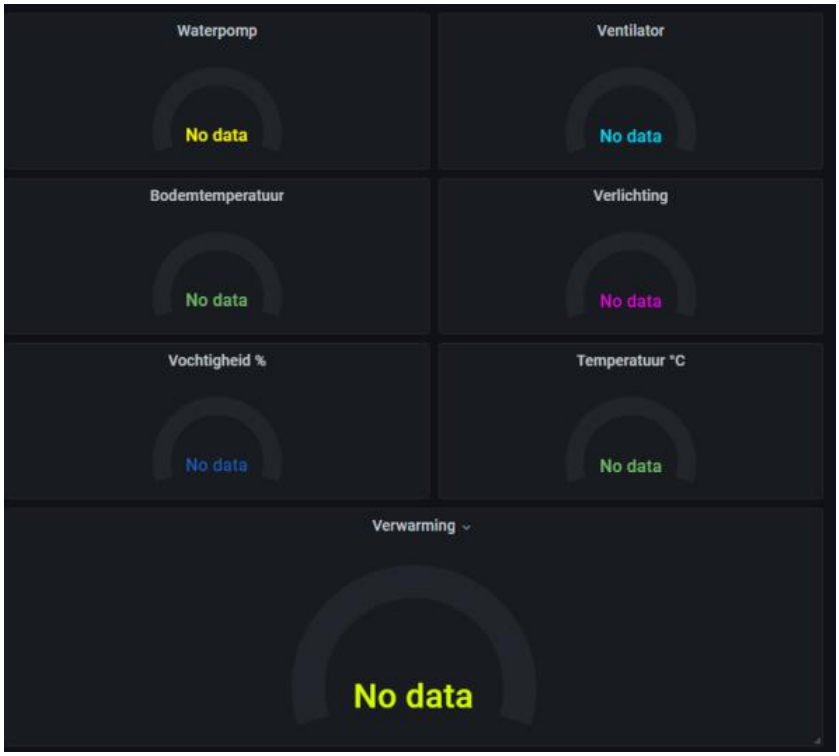


5.7 Selecteer de juiste data-source:

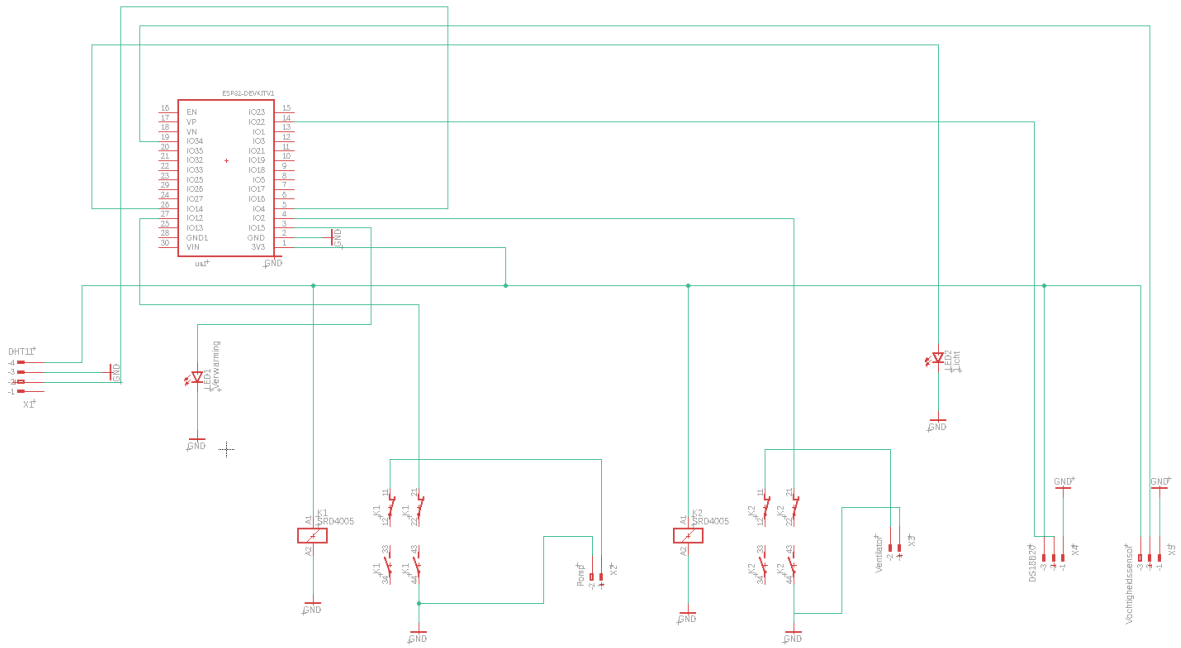
5.8 Selecteer de juiste measurement (Hier komen jouw topics te staan):



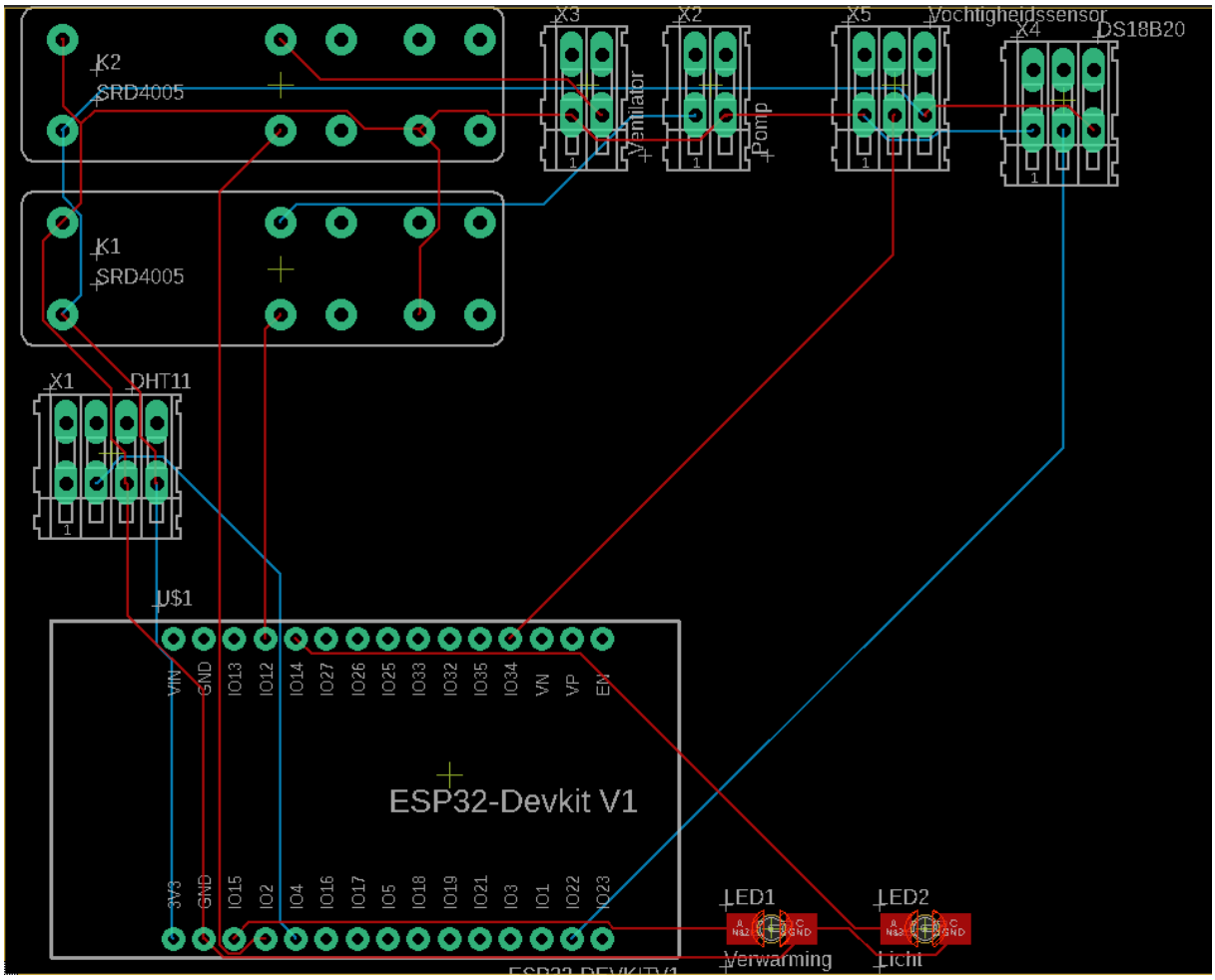
5.9 Data op Grafana:



6. Schema Eagle



7. PCB


K2
SRD4005
K1
SRD4005
X3
Ventilator
X2
Pomp
X5
Vochtigheidssensor
X4
DS18B20
X1
DHT11
U$1
VIN
GND
IO13
IO12
IO14
IO27
IO26
IO25
IO33
IO32
IO35
IO34
VN
VP
EN
ESP32-Devkit V1
3V3
GND
IO15
IO2
IO4
IO16
IO17
IO5
IO18
IO19
IO21
IO3
IO1
IO22
IO23
LED1
Verwarming
LED2
Licht

THOMAS MORE