

# ESP 32

## ESP Projecten

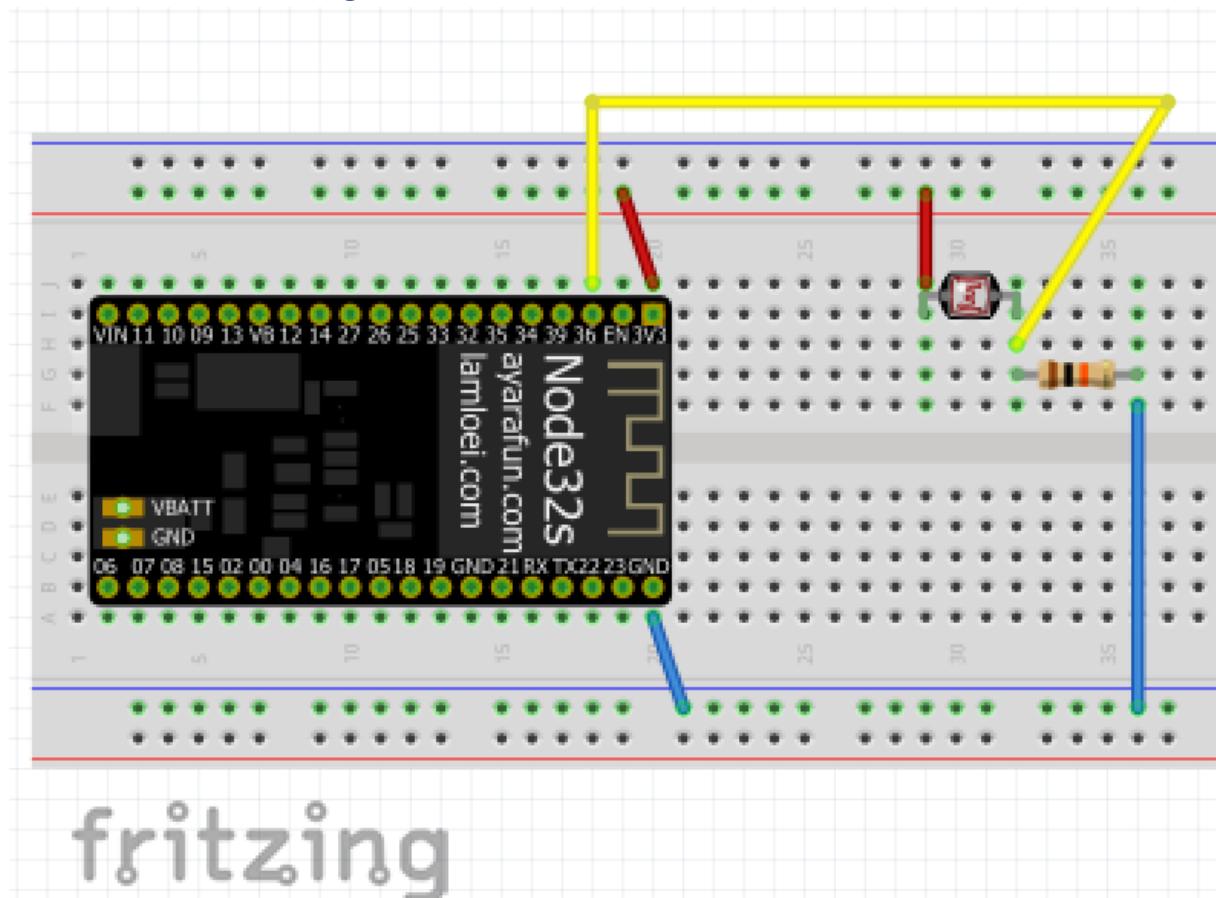


## Eigen LDR sensor koppelen aan Home Assistant

We hebben reeds een programma op de ESP32 gemaakt om LEDs aan te sturen met Home Assistant. We gaan aan de ESP een LDR koppelen en deze waarde doorgeven aan Home Assistant. Het verschil met voorgaand project is:

- ✓ Home Assistant moet automatisch de ESP met LDR detecteren.
- ✓ Als het donker wordt (dus afdekken) van LDR moet een LED bulb beginnen branden. (Dus script maken in Home Automation).

### Maken van de schakeling



### De nodige code schrijven om via MQTT ontdekt te worden door HA

We gaan terug gebruik maken van de Broker in onze HiveMQ Cloud Cluster. Omdat we vaak gebruik gaan maken van deze setting, zou het handig zijn om het connecteren met de Broker in de HiveMQ Cloud Cluster in een apart bestand onder te brengen. Meer bepaald de functie: `mqtt_connect` in het bestand: `testCloudHiveMQ.py`.

### Het connecteren met de HiveMQ Cloud Cluster onderbrengen in een nieuw bestand

1. Open Thonny (en zorg dat de interpreter ingesteld staat op MicroPython voor ESP32).
2. Maak een nieuw bestand: `ConnectHiveCloudBroker.py` en sla op in de map: `iot4_je` voornaam.
3. Open het bestand: `testCloudHiveMQ.py` en kopieer functie: `mqtt_connect` en plak deze in het nieuwe bestand.

4. En doe bovenaan de 2 volgende imports:

```
1 from umqtt.robust import MQTTClient #module voor MQTT
2 import ssl #Secure Socket Layer. Voor encrypteren van data ....
```

5. Sla de wijzigingen op.
6. Laat het runnen, indien geen foutmeldingen, is het bestand in orde.
7. Laad dit nieuwe bestand (ConnectHiveCloudBroker.py) op naar het bord.
8. Om te testen, kunnen we nu het testCloudHiveMQ.py bestand aanpassen:
  - Sla het bestand op als testCloudHiveMQ\_2.py.
  - Verwijder uit het bestand de imports: from umqtt.robust import MQTTClient en import ssl.
  - Voeg de volgende import toe:

```
from ConnectHiveCloudBroker import mqtt_connect
```

- d. Verwijder de functie: mqtt\_connect.
9. Test het geheel uit.

Het automatisch laten detecteren van onze sensor door Home Assistant

We hebben ons ESP32 en Home Assistant al kunnen connecteren met de broker in de cluster, nu is het moment aangebroken om onze sensor automatisch te laten detecteren door Home Assistant. Dus zonder dat we iets moeten vermelden in het yaml configuratie bestand.

1. We kunnen vertrekken van testCloudHiveMQ\_2.py en opslaan als esp\_Idrsensor.py.
2. Maak daarin de volgende discovery functie:

```
def publish_discovery(client,dev_name,uid):
    discovery_topic = f"homeassistant/sensor/{dev_name}/config"
    discovery_payload = {
        "name": "LDR Sensor 0",
        "state_topic": f"home/{dev_name}/state",
        "unit_of_measurement": "%",
        "unique_id":f"{dev_name}_{uid}",
        "value_template": "{{ value_json.percent }}",
        "device": {
            "identifiers": [dev_name],
            "name": dev_name,
            "model": "ESP32",
            "manufacturer": "Custom"
        }
    }
    client.publish(discovery_topic.encode(), json.dumps(discovery_payload).replace("'",'').encode(), retain=True)
    print("Home Assistant auto-discovery gepubliceerd")
```

Deze functie verwacht de 2 volgende parameters:

- client: verwijzing naar het MQTT client object.
- dev\_name: de naam dat je aan het device wil geven.
- uid: unieke identifier

Dan worden er een topic en payload gedefinieerd:

- discovery\_topic: is de topic nodig om ontdekt te worden door Home Assistant. Deze moet normaliter altijd starten met homeassistant. Zorg per sensor voor een “unieke” discover topic.
- discovery\_payload: bevat een structuur, dictionary, met meer informatie over de sensor.

Zoals:

- naam
- state\_topic: de topic die zal gebruikt worden om de status van de sensor weer te geven, hier de lichtsterkte.
- unit\_of\_measurement: hier wordt de eenheid van de gemeten waarde gedefinieerd. Wij gaan hier voor een relatieve waarde ten opzichte van het maximum (vandaar %).
- unique\_id: unieke waarde of id voor de sensor (we moeten nog eens nagaan of deze aan bepaalde criteria moet voldoen).
- device: nogmaals een samenvatting van het device.

Vervolgens wordt er een mqtt publish uitgevoerd met de voorop gestelde discovery topic.

3. Importeer bovenaan de json module.
4. Definieer ook een DEVICE\_NAME en SENSOR\_ID.  
Hieronder een voorbeeld:

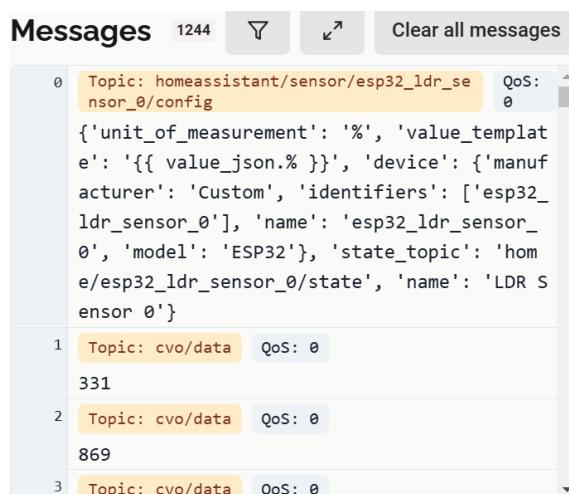
```
import json

DEVICE_NAME = "esp32_ldr_sensor_0"
SENSOR_ID = "080270"
```

5. Roep na het connecteren met de broker de zopas gemaakte functie op.

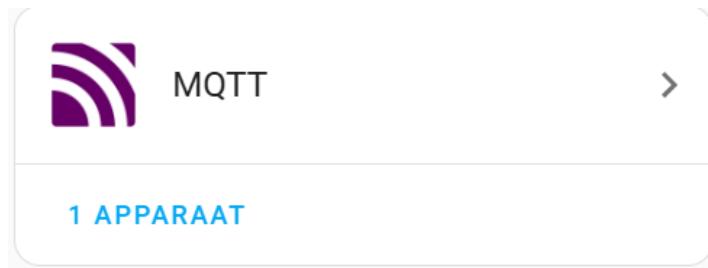
```
publish_discovery(mqtt,DEVICE_NAME,SENSOR_ID)
```

6. Start Home Assistant op.
7. Start het MicroPython script op ESP32.
8. Controleer of de boodschap toekomt bij de broker en Home Assistant:
  - a. Dus naar HiveMQ platform gaan en dan WebClient activeren en subscriben op #, normaal gezien moet je de volgende resultaten zien in het rechter paneel:



- b. Ga naar Home Assistant, meer bepaald naar Instellingen → Apparaten en diensten.

- c. Kijk naar het blokje: MQTT.



Je zou normaal gezien de vermelding moeten zien staan van 1 APPARAAT. Dit is het apparaat dat MQTT automatisch ontdekt heeft, namelijk onze sensor./

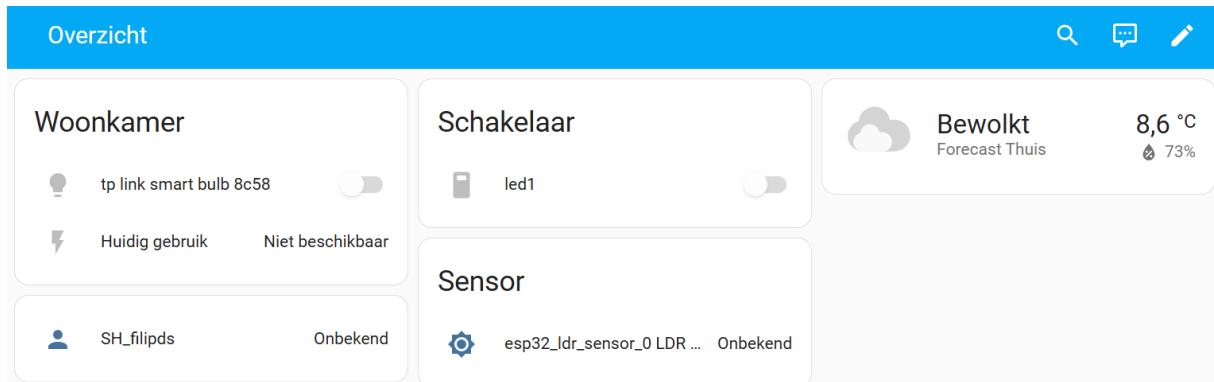
- d. Tik op MQTT dan kom je op het volgende scherm:

Je ziet dan het aantal apparaten en entiteiten die via MQTT communiceren met Home Assistant.

Er staat onbekend omdat er nog geen meetwaarden worden doorgegeven.

- e. Keer je terug en tik je op entiteiten dan zal je een lijst van de gekoppelde functies zien van die apparaten:

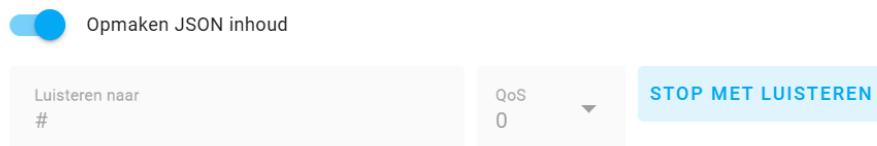
9. Ga je kijken in het overzicht dan zal de sensor ook reeds zichtbaar zijn (natuurlijk zonder waarde):



Ter info: je kan ook steeds nagaan wat voor mqtt boodschappen er binnenkomen op de broker en Home Assistant via Home Assistant, dit kan als volgt:

1. Tik terug op de MQTT in Home Assistant.
2. Druk dan op configureren.
3. Scroll naar beneden en vervolledig de topic met een #.
4. Zorg ook dat opmaken JSON inhoud aan staat.
5. En tik dan op BEGIN MET LUISTEREN, de boodschappen zullen binnenkomen (als het script op de ESP runt):

### Luisteren naar 'onderwerp'



Bericht 32 ontvangen op cvo/data om 17:01:

236

QoS: 0 - Retain: false

Bericht 31 ontvangen op cvo/data om 17:01:

170

QoS: 0 - Retain: false

Bericht 30 ontvangen op cvo/data om 17:01:

572

6. Druk vervolgens op STOP MET LUISTEREN.
7. **Opdracht: geef aan de LDR sensor een "zon" icoontje.**

Waarde van LDR naar Home Assistant sturen

1. We gaan van start met de nodige imports en definities om het analog signaal van de LDR binnen te halen.

```
from machine import Pin,ADC  
  
ldr = ADC(Pin(36))  
ldr.atten(ADC.ATTN_11DB) # Max bereik 3.3V
```

2. Vervolgens zal op geregelde tijdstippen de waarde komende van de LDR worden gepubliceerd met de bijhorende topic (hier: home/DEVICE\_NAME/state). Dit gebeurt normaliter in de while-lus, de code kan er als volgt uitzien:

```
value = ldr.read()  
value_percent = 100*(value/2**12)  
payload = {"percent":int(value_percent)}  
mqtt.publish("home/"+DEVICE_NAME+"/state",json.dumps(payload).encode())  
print(value_percent)  
time.sleep(2)
```

3. Neem eens een kijkje in Home Assistant. De procentuele waarde zou nu naast de sensor moeten prijken in het algemeen overzicht:



## Slimme lamp besturen met behulp van eigen LDR sensor

Nu de waarde van de LDR sensor binnenkomt in Home Assistant zou het leuk zijn om een slimme lamp aan te zetten als de waarde van de LDR onder de 45% duikt en terug uit gaat als de waarde van de LDR hoger wordt dan 55%.

### Opdracht:

1. Voeg een slimme lamp (Shelly, TP-link of andere) toe aan je Home Assistant.
2. Zorg dat de bulb in de ruimte: mylab komt.
3. Zorg dat mylab en bijhorende objecten (hier de Shelly bulb) zichtbaar zijn in het overzicht.
4. Zie vorige cursus: hoofdstuk Home Assistant.

### Slimme lamp doven bij een LDR waarde boven 55%

Een automatisering in Home Assistant kan je instellen via het Automatisering UI, hieronder de stappen in het kort (volgens ChatGPT):

Als je liever een visuele manier gebruikt om automatiseringen toe te voegen:

1. **Ga naar Home Assistant** en open het **dashboard**.
2. Klik in de **zijbalk (links)** op **Instellingen > Automatiseringen & Scènes**.
3. Klik rechts onderin op **+ Maak Automatisering** (of Automatisering Toevoegen)
4. Kies "**Begin met een lege automatisering**".
5. Vul de gegevens in zoals hieronder beschreven.

### Automatisering 1: LED uitschakelen bij hoge lichtintensiteit

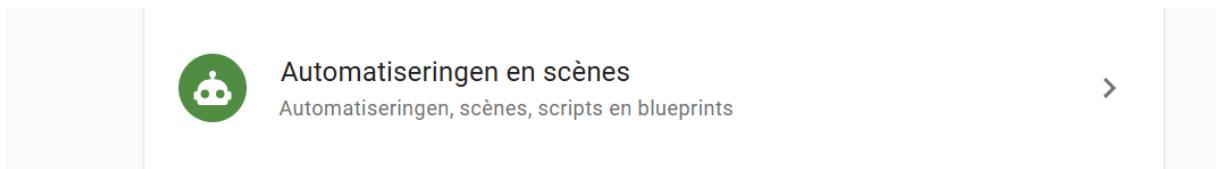
- **Trigger:**
  - Type: **Numerieke status**
  - Entiteit: sensor.ldr\_sensor (pas aan naar jouw sensor)
  - Voorwaarde: **hoger dan 55** (of een andere gewenste drempel)
- **Actie:**
  - Service: light.turn\_off
  - Doel: light.led\_lamp (pas aan naar jouw lamp)
  - Extra optie: Stel eventueel helderheid in (bijv. 255)

Klik op **Opslaan** en test de automatisering.

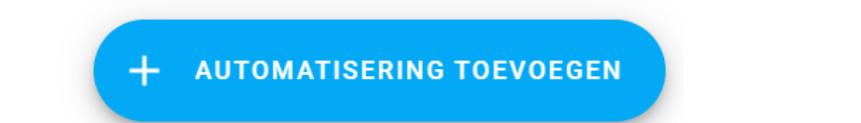
### Stap per stap uitgelegd

1. Tik in het linker paneel op Instellingen.

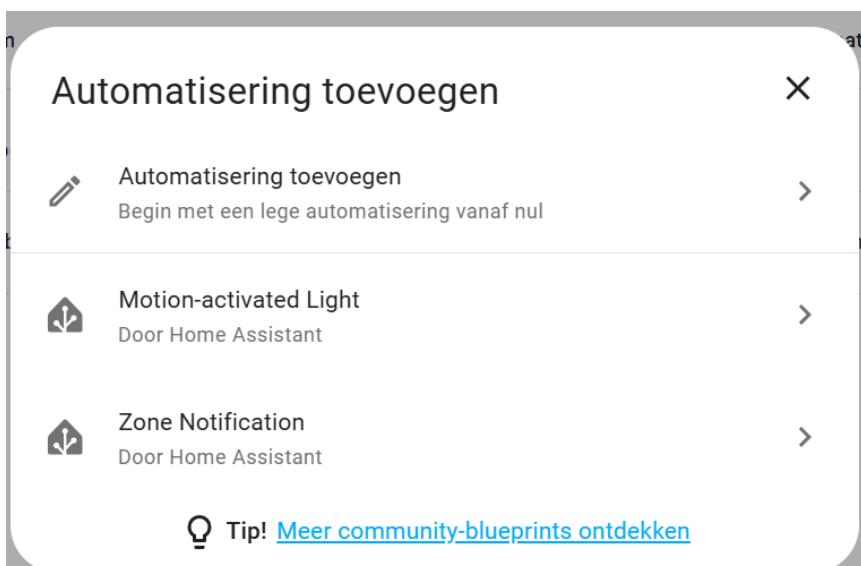
2. Tik ver volgens op Automatisering en scenes:



3. Klik op de blauwe knop onderaan op Automatisering toevoegen.



4. Kies bovenaan nogmaals voor Automatisering toevoegen



5. Je komt op het scherm: Nieuwe Automatisering.

### Wanneer



Een trigger is een specifieke gebeurtenis in of rond je huis, bijvoorbeeld: 'Als de zon ondergaat'. Elke trigger die hier wordt vermeld, start je automatisering.

+ TRIGGER TOEVOEGEN

### En als (optioneel)



Aan deze lijst met voorwaarden moet worden voldaan voordat de automatisering kan worden uitgevoerd. Aan een voorwaarde kan op elk moment wel of niet worden voldaan, bijvoorbeeld: 'Als SH\_filipds thuis is'. Je kunt bouwstenen gebruiken om complexere voorwaarden te creëren.

+ VOORWAARDE TOEVOEGEN

+ BOUWSTEEN TOEVOEGEN

### Doe dan



Deze lijst met acties wordt op volgorde uitgevoerd wanneer de automatisering wordt uitgevoerd. Een actie bestuurt meestal een van je ruimtes, apparaten of entiteiten, bijvoorbeeld: 'Doe het licht aan'. Je kunt bouwstenen gebruiken om complexere acties te creëren.

+ ACTIE TOEVOEGEN

+ VOEG BOUWSTEEN TOE

OPSLAAN

6. Tik vervolgens op TRIGGER TOEVOEGEN (bovenaan):

**X Trigger toevoegen**

Zoek trigger

- Apparaat**  
Wanneer er iets met een apparaat gebeurt.
- Entiteit**  
Wanneer er iets met een entiteit gebeurt.
- Tijd en locatie**  
Wanneer iemand een zone binnenkomt of verlaat, of op een bepaald tijdstip.

Kies hier voor Entiteit (de LDR sensor is een entiteit onder MQTT).

7. Kies dan voor numerieke status (de waarde van de LDR sensor willen we als drempel gebruiken).

The screenshot shows the Home Assistant interface with the title 'Entiteit' at the top left. Below it is a search bar labeled 'Zoeken · Entiteit'. A list of entities is displayed, starting with 'Numerieke status' and 'Status'. Each entity has a description and a '+' button to its right.

- Numerieke status**  
123 Wanneer de numerieke waarde van de status van een entiteit (of de waarde van een attribuut) een bepaalde drempel overschrijdt. +
- Status**  
Wanneer de status van een entiteit (of attribuut) verandert. +

8. Zoek in de entiteitlijst en tik op de gewenste entiteit.

A dropdown menu titled '123 Numerieke status' is open, showing a list of entities under the heading 'Entiteit'. The first item, 'activeer\_bulb', is selected and highlighted with a green border. Other items include 'esp32\_ldr\_sensor\_0 LDR Sensor 0' and 'File editor Update'.

9. Geef in de boven-modus, in het veld boven het getal 55 in.

The interface shows 'Boven-modus' selected. Under 'Vast getal', the value '55' is entered in the 'Boven' field. There is also an option 'Nummerieke waarde van een andere entiteit'.

10. Scroll naar beneden naar de rubriek Doe dan:

Doe dan

Deze lijst met acties wordt op volgorde uitgevoerd wanneer de automatisering wordt uitgevoerd. Een actie bestuurt meestal een van je ruimtes, apparaten of entiteiten, bijvoorbeeld: 'Doe het licht aan'. Je kunt bouwstenen gebruiken om complexere acties te creëren.

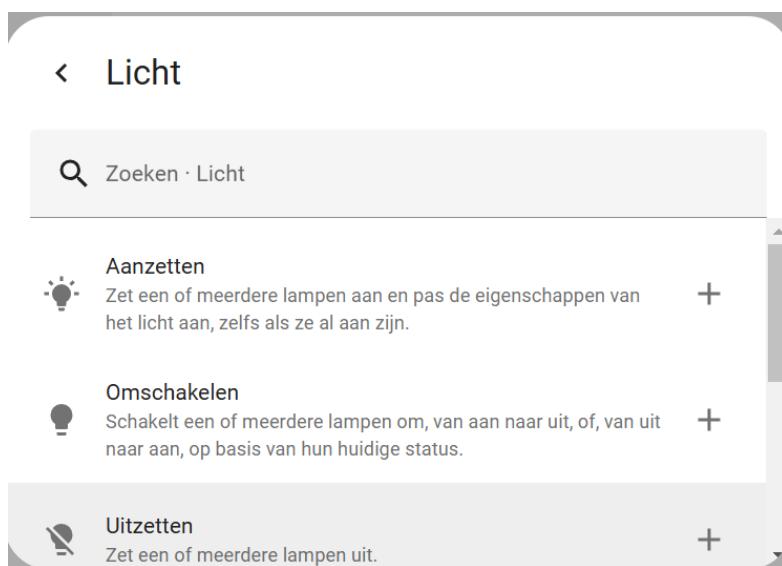
+ ACTIE TOEVOEGEN + VOEG BOUWSTEEN TOE

**OPSLAAN**

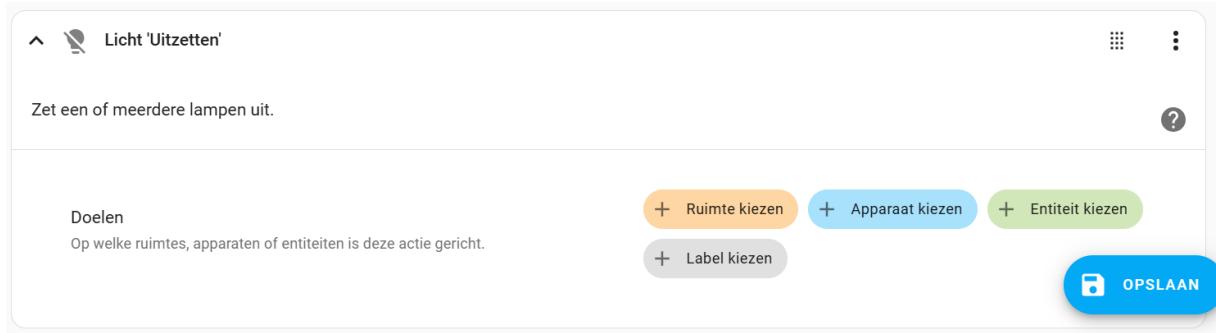
11. Klik op + ACTIE TOEVOEGEN
12. Klik in het venster Actie toevoegen op licht:



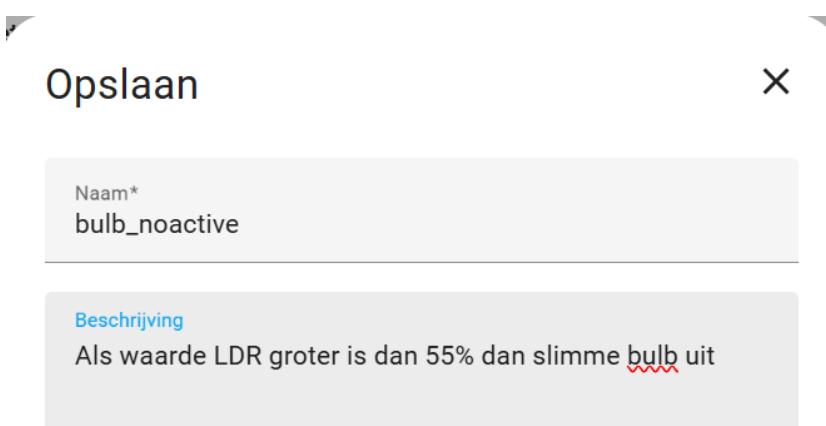
13. En kies voor uitzetten:



14. Druk op de blauwe knop: + Apparaat kiezen.



15. Duid de (gewenste) bulb aan.
16. En druk vervolgens op de knop: OPSLAAN.
17. Geef een naam aan de automatisering en eventueel een beschrijving:



18. Druk op OPSLAAN.
19. De automatisering is gemaakt.
20. Herstart Home Assistant via Ontwikkelhulpmiddelen → Herstarten.
21. Test uit door het licht te laten branden en dan voldoende licht op de LDR te laten schijnen.  
Na een tijdje zou de slimme lamp moeten doven.

## Opdrachten:

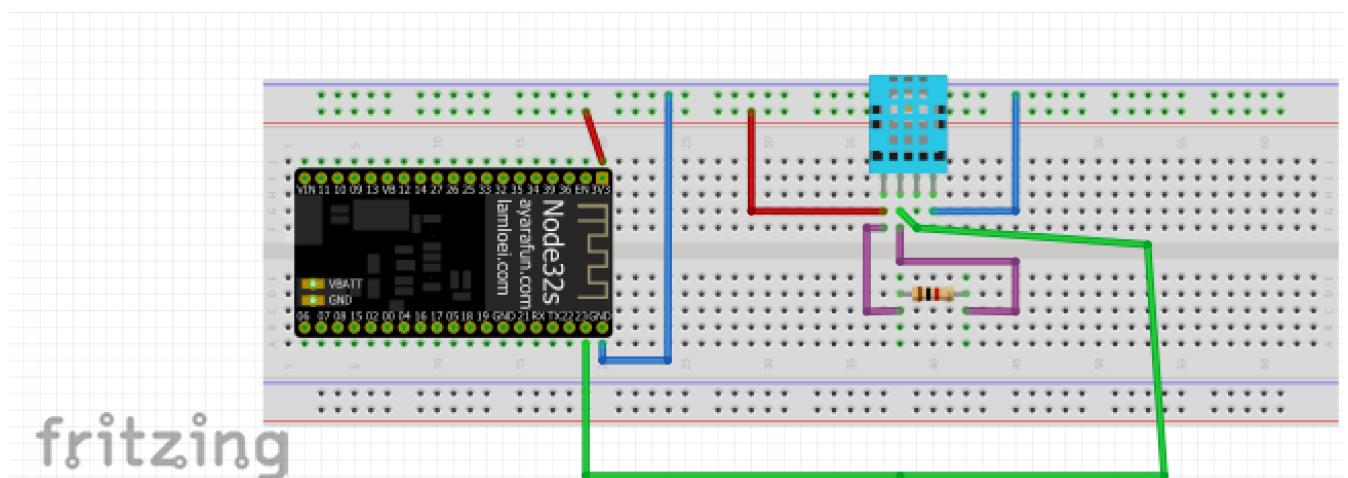
Opdracht 1: een automatisering maken om de slimme lamp te laten branden als de waarde van de LDR sensor daalt onder de 45%.

1. Je zal terug naar automatisering moeten gaan onder Instellingen.
2. Een nieuwe automatisering maken.
3. Maar nu gebruik maken van Onder een bepaalde drempel, nl. 45
4. Indien aan de voorwaarde voldaan dan laat de slimme lamp branden.
5. Sla de automatisering op als: bulb\_active.
6. Herstart Home Assistant.
7. En test uit door LDR af te dekken.

- Indien LDR afgedekt, zal de lamp branden, indien de LDR vrij en als deze voldoende licht ontvangt zal de lamp doven.

Opdracht 2: een temperatuur en vochtigheidssensor maken en verbinden met Home Assistant

- Maak de volgende schakeling:



- Open het MicroPython script: esp\_ldrsensor.py en sla op als esp\_temp\_hum\_sensor.py.

- Je gaat dit script als volgt aanpassen:

- Importeer de dht11 bibliotheek.
- Verwijder de initialisatie van de ldr sensor.
- Initialiseer de dht11 sensor, dit kan als volgt:

```
sensor = dht.DHT11(Pin(23))
```

- Wijzig en breid de functie: publish\_discovery als volgt uit:

- Wijzig: discovery\_topic = f"homeassistant/sensor/{dev\_name}/config" in

```
discovery_topic_temp =
f"homeassistant/sensor/{dev_name}_temperature/config"
```

En discovery\_payload in discovery\_payload\_temp met de volgende inhoud:

```
discovery_payload_temp = {
    "name": "Temperature",
    "state_topic": f"home/{dev_name}/state",
    "unit_of_measurement": "°C",
    "value_template": "{{ value_json.temperature }}",
    "device": {
        "identifiers": [dev_name],
```

```

    "name": dev_name,
    "model": "ESP32",
    "manufacturer": "Custom"
}

```

- ii. Wijzig publish zodanig dat er gebruik gemaakt wordt van de topic: discovery\_topic\_temp en payload: discovery\_payload\_temp.
- iii. Kopieer de bovenstaande dictionary en oproep van de functie: publish en plak deze onderaan de functie: publish\_discovery.
- iv. Vervang temperature door humidity en de eenheid naar %.
- e. Verwijder in de while lus de ldr “functionaliteit” en maak gebruik van de volgende dht functies om te meten en waarden op te halen:

```

sensor.measure()
t = sensor.temperature()
h = sensor.humidity()

```

vergeet zeker niet achter de meting een pauze in te lassen aangezien de dht11 sensor maar ongeveer om de 2 seconden een meting kan doen.

- f. Publiceer tenslotte de gemeten waarden, de door te sturen payload is:

```
{"temperature":t,"humidity":h}
```

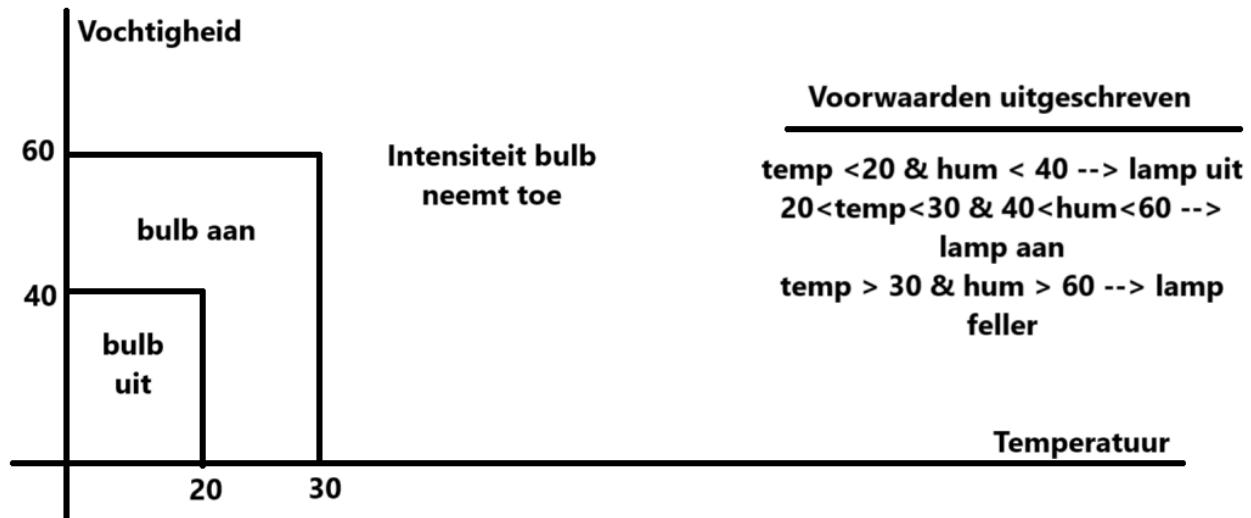
- 4. Kijk in Home Assistant dat de nieuwe sensor wordt ontdekt. Voeg deze toe aan de apparaten en kijk in het overzicht dat de waarden van de sensor worden getoond.

### Opdracht 3: maken van automatiseringen

1. Zet de voorgaande automatiseringen uit.
  - a. Instellingen → automatisering en scènes.
  - b. Zorg dat de voorgaande automatiseringen niet meer actief zijn. Hieronder een voorbeeld van 2 niet actieve automatiseringen.

	activeer_bulb	30 jan, 16:34	<input checked="" type="checkbox"/>	
	bulb_noactive	Nooit	<input type="checkbox"/>	
	moment_capture_motion_detection	21 uur geleden	<input checked="" type="checkbox"/>	
	motion_detect_bulb_on	21 uur geleden	<input checked="" type="checkbox"/>	

2. Probeer onderstaande figuur te vertalen naar een automatisering:



### Hieronder het stappenplan volgens ChatGPT:

Hier is de stap-voor-stap handleiding om deze automatisering volledig via de **UI van Home Assistant** in te stellen.

---

#### Stap 1: Open de automatiseringen UI

1. Ga naar **Home Assistant** en klik in de zijknop op **Instellingen**.
  2. Klik op **Automatiseringen & Scènes**.
  3. Klik rechts onderin op **+ Maak Automatisering**.
  4. Kies "**Begin met een lege automatisering**".
- 

#### Stap 2: Naam en omschrijving instellen

- **Naam:** Lamp aansturen op basis van temperatuur en vochtigheid
  - **Omschrijving:** Automatisering om de lamp te regelen afhankelijk van temperatuur en luchtvochtigheid.
- 

#### Stap 3: Triggers instellen

De automatisering moet worden geactiveerd bij elke verandering in **temperatuur** of **vochtigheid**.

1. Klik op **+ Trigger toevoegen**.
2. Kies **Staat** als trigger type.
3. Bij **Entiteit** kies je de **temperatuursensor** (bijv. `sensor.temperature_sensor`).
4. Laat **Van** en **Naar** leeg (zodat elke wijziging wordt gedetecteerd).
5. Klik opnieuw op **+ Trigger toevoegen** en herhaal deze stappen voor de **vochtigheidssensor** (bijv. `sensor.humidity_sensor`).

- Je hebt nu twee triggers: **temperatuur** en **vochtigheid**.
- 

Stap 4: Acties instellen met "Kiezen" (If-Else)

We gebruiken de "**Kiezen**"-functie om verschillende scenario's te maken.

1. Klik op "+ Actie toevoegen".
  2. Kies "**Kiezen**" als actie type.
  3. Klik op "+ Optie toevoegen" om verschillende voorwaarden toe te voegen.
- 

Scenario 1: Lamp UIT als temperatuur < 20°C en vochtigheid < 40%

1. Klik op "**Voorwaarden toevoegen**".
2. Kies "**Numerieke status**" als voorwaarde type.
3. Bij **Entiteit** selecteer je je **temperatuursensor** (bijv. sensor.temperature\_sensor).
4. Vul "**Onder 20**" in.
5. Klik op "+ Voorwaarde toevoegen", kies opnieuw "**Numerieke status**" en selecteer de **vochtigheidssensor** (bijv. sensor.humidity\_sensor).
6. Vul "**Onder 40**" in.
7. Klik op "+ Actie toevoegen" binnen deze keuze.
8. Kies "**Apparaat**" en selecteer je **lamp** (bijv. light.led\_lamp).
9. Kies de actie "**Uitschakelen**".

- Eerste scenario klaar:** De lamp gaat uit bij lage temperatuur en vochtigheid.
- 

Scenario 2: Lamp AAN als temperatuur tussen 20-30°C en vochtigheid tussen 40-60%

1. Klik op "+ Optie toevoegen" om een nieuwe keuze te maken.
2. Klik op "**Voorwaarden toevoegen**".
3. Voeg een "**Numerieke status**" voor **temperatuur** toe:
  - o Entiteit: sensor.temperature\_sensor
  - o **Boven 20 en Onder 30**
4. Voeg een "**Numerieke status**" voor **vochtigheid** toe:
  - o Entiteit: sensor.humidity\_sensor
  - o **Boven 40 en Onder 60**
5. Klik op "+ Actie toevoegen" binnen deze keuze.
6. Kies "**Apparaat**" en selecteer je **lamp** (light.led\_lamp).
7. Kies de actie "**Inschakelen**" en stel **helderheid** in op **150**.

- Tweede scenario klaar:** De lamp gaat aan bij gematigde temperatuur en vochtigheid.
- 

Scenario 3: Lamp feller als temperatuur > 30°C en vochtigheid > 60%

1. Klik op "+ Optie toevoegen".
2. Klik op "**Voorwaarden toevoegen**".

3. Voeg een "Numerieke status" voor **temperatuur** toe:
  - o Entiteit: sensor.temperature\_sensor
  - o **Boven 30**
4. Voeg een "Numerieke status" voor **vochtigheid** toe:
  - o Entiteit: sensor.humidity\_sensor
  - o **Boven 60**
5. Klik op "+ Actie toevoegen" binnen deze keuze.
6. Kies "**Apparaat**" en selecteer je **lamp** (light.led\_lamp).
7. Kies de actie "**Inschakelen**" en stel **helderheid** in op **255**.

**Derde scenario klaar:** De lamp wordt feller bij hoge temperatuur en vochtigheid.

---

#### Stap 5: Opslaan en testen

1. **Klik op "Opslaan"** rechts onderin.
2. **Herstart Home Assistant via Instellingen > Systeem > Herstarten.**
3. **Test de automatisering via Ontwikkelaarstools:**
  - o Handmatig de waarden van de temperatuur- en vochtigheidssensor wijzigen en kijken of de lamp correct reageert.

---

#### Samenvatting

**Triggers:** Wordt geactiveerd bij elke verandering in temperatuur of vochtigheid.  
 **Acties met kiezen:**

- **Lamp UIT** bij lage waarden.
  - **Lamp AAN** bij normale waarden.
  - **Lamp helderder** bij hoge waarden.
- Volledig ingesteld via UI** zonder handmatige code.

⌚ Je automatisering werkt nu volledig in Home Assistant! ⌚

#### **In Yaml wordt het volgens ChatGPT als volgt:**

Hier is de volledige handleiding om de automatisering **handmatig** in te stellen in Home Assistant via de configuratiebestanden. Dit omvat het bewerken van `automations.yaml`, herstarten van Home Assistant en testen van de instellingen.

---

#### Stap 1: Open het `automations.yaml` bestand

1. **Open Home Assistant** en ga naar de **File Editor** (als je deze niet hebt, installeer de **File Editor Add-on** via Supervisor).
2. Navigeer naar `config/automations.yaml` en open het bestand om te bewerken.

## Stap 2: Voeg de volgende code toe aan `automations.yaml`

```
- alias: Lamp aansturen op basis van temperatuur en vochtigheid
  description: Automatisering om de LED-lamp te regelen afhankelijk van
temperatuur en luchtvochtigheid.

  trigger:
    - platform: state
      entity_id:
        - sensor.temperature_sensor
        - sensor.humidity_sensor
  condition: []
  action:
    - choose:
        # 1. Lamp uit bij lage temperatuur en vochtigheid
        - conditions:
            - condition: numeric_state
              entity_id: sensor.temperature_sensor
              below: 20
            - condition: numeric_state
              entity_id: sensor.humidity_sensor
              below: 40
        sequence:
          - service: light.turn_off
            target:
              entity_id: light.led_lamp

        # 2. Lamp aan bij gematigde temperatuur en vochtigheid
        - conditions:
            - condition: numeric_state
              entity_id: sensor.temperature_sensor
              above: 20
              below: 30
            - condition: numeric_state
              entity_id: sensor.humidity_sensor
              above: 40
              below: 60
        sequence:
          - service: light.turn_on
            target:
              entity_id: light.led_lamp
            data:
              brightness: 150 # Gemiddelde helderheid

        # 3. Lamp feller bij hoge temperatuur en vochtigheid
        - conditions:
            - condition: numeric_state
              entity_id: sensor.temperature_sensor
              above: 30
            - condition: numeric_state
              entity_id: sensor.humidity_sensor
              above: 60
        sequence:
          - service: light.turn_on
            target:
              entity_id: light.led_lamp
            data:
              brightness: 255 # Maximale helderheid
  mode: restart
```

### Stap 3: Pas de instellingen aan

Vervang de volgende waarden in het script met jouw specifieke Home Assistant entites:

- `sensor.temperature_sensor` → Zet hier de juiste **entity\_id** van jouw temperatuurmeter.
- `sensor.humidity_sensor` → Zet hier de juiste **entity\_id** van jouw luchtvochtigheidssensor.
- `light.led_lamp` → Zet hier de juiste **entity\_id** van jouw LED-lamp.

### Hoe vind je jouw entity\_id's?

1. Ga naar **Ontwikkelaarstools** (Developer Tools) in Home Assistant.
2. Klik op **Staten** (States).
3. Zoek je temperatuur- en vochtigheidssensor en kopieer de juiste entity ID.

---

### Stap 4: Home Assistant herstarten

1. Sla het bestand op in de **File Editor**.
2. Ga naar **Instellingen > Systeem > Herstarten**.
3. Klik op **Herstarten** en wacht tot Home Assistant opnieuw is opgestart.

---

### Stap 5: Test de automatisering

1. **Ga naar Ontwikkelaarstools** in Home Assistant.
2. **Wijzig handmatig de waarde van de temperatuur- en vochtigheidssensor** door een aangepaste status in te stellen.
3. Controleer of de lamp correct in- of uitschakelt en de helderheid aanpast.

---

### Samenvatting

- Volledige automatisering** is ingesteld in `automations.yaml`.
- Home Assistant herstart** om de wijzigingen toe te passen.
- Testen via Ontwikkelaarstools** om te controleren of alles werkt.

Na deze stappen werkt de lamp automatisch op basis van de temperatuur en luchtvochtigheid.



## Een controller interface, gemaakt in node-red, toevoegen aan Home Assistant

In dit project gaan we een UI (UserInterface) maken in Node-Red die een servo gekoppeld aan een ESP32 zal aansturen. De communicatie tussen het UI en de ESP32 verloopt via sockets (de TCP/IP laag).

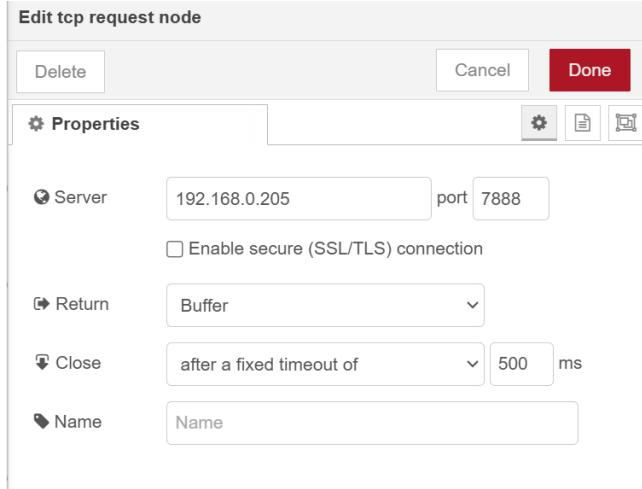
### Opdracht: maken van UI in Node-Red

1. Start Node-Red (via de console)
2. Open node-red in de browser.
3. Deactiveer alle voorgaande flows.
4. Maak een nieuwe flow: servo control.
5. Open in het rechterpaneel Dashboard en maak een nieuwe tabblad: Servo met daarin een groep: Control
6. Maak vervolgens het volgende interface:

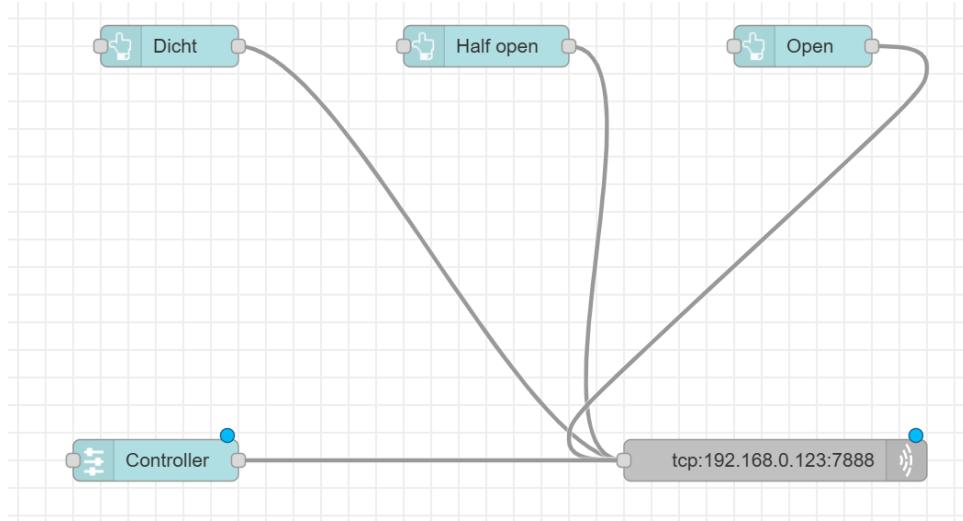


- a. Als er op de knop: DICTH wordt gedrukt, retourneert deze het getal: 0.
- b. Als er op de knop: HALF OPEN wordt gedrukt, retourneert deze het getal: 90.
- c. Als er op de knop: OPEN wordt gedrukt, retourneert deze het getal: 180.
- d. De slider retourneert enkel een getal als deze wordt los gelaten (tip: Output: only on release)
- e. **Opmerking zorg dat alle dashboard elementen lid worden van de groep: Servo - Control.**

7. Sleep vervolgens een tcp-request node op het werkveld en configureer deze als volgt:



- a. Met port de poort waarop de ESP zal luisteren.
  - b. Server: het IP-adres van de ESP (zal nog moeten worden aangepast).
  - c. En zet Close gelijk aan: after a fixed timeout of. En zet de time-out op 500 ms.
8. Verbind alle knoppen en slider met deze laatste node.
9. De flow ziet er als volgt uit:



10. We merken op dat de flow nog niet zal werken aangezien het script op de ESP nog niet klaar is.

### Opdracht maken van de socket server op de ESP

1. Zorg dat de ESP zich kan verbinden met het wifi netwerk.
  - a. Importeren van SimpleWifi\_v2.
  - b. Een Wifi instance maken.
  - c. De Wifi method open oproepen en met de method: get\_status de status opvragen en het programma sluiten als er geen connectie mogelijk is.
  - d. De IP-adres van de ESP tonen in de shell
2. Definieer bovenaan de globale variabele PORT en stel deze gelijk aan 7888.
3. Vervolledig de code met de volgende logica die een socket opstart en klaar staat om een boodschap te ontvangen:

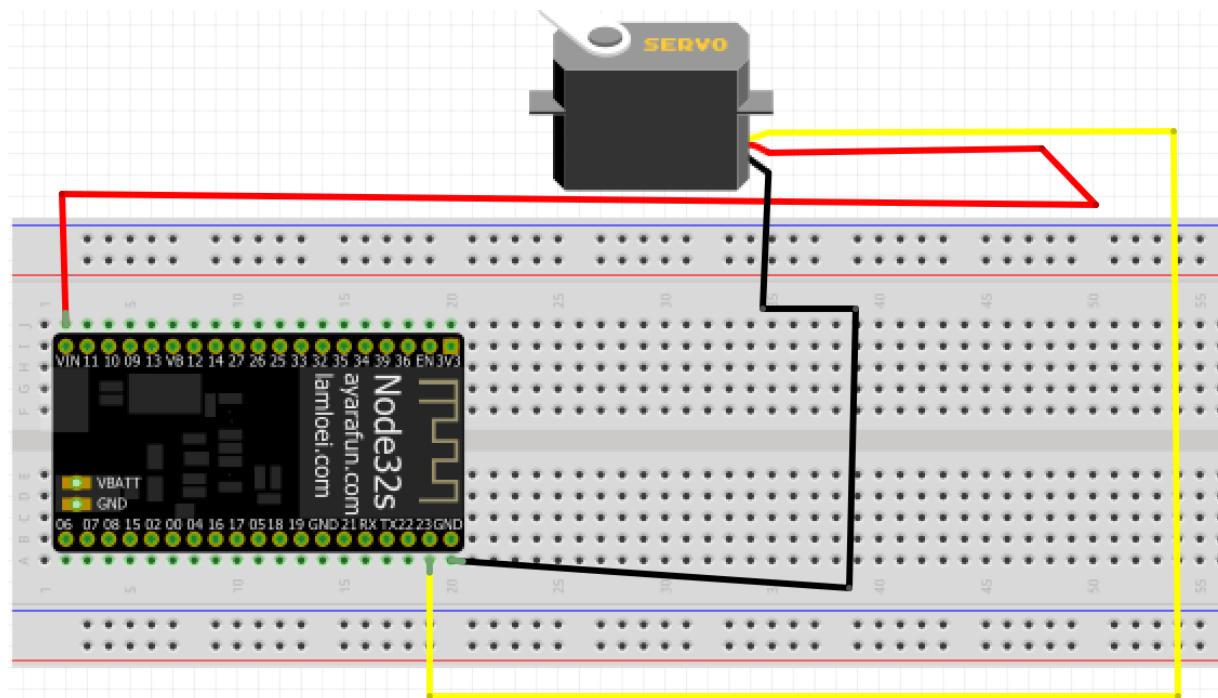
```

s = socket.socket()
s.bind(("0.0.0.0",PORT))
s.listen()

while True:
    client,addr = s.accept()
    print("verbinding met client:",addr)
    data = client.recv(32)
    data = data.decode()
    print("socket data:",data)
    #code om servo aan te sturen
    client.close()
  
```

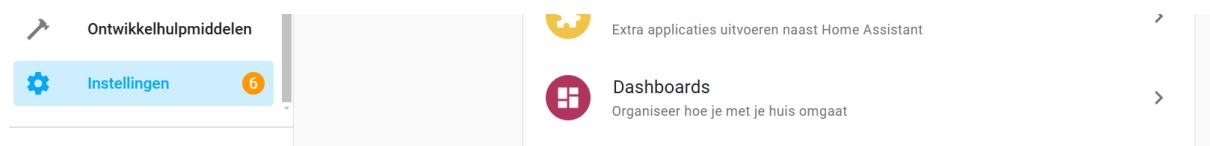
## Opdracht servo aansluiten en nodige code schrijven om servo aan te sturen

1. In de vorige cursus hebben we een klasse gemaakt om een servo aan te sturen, nl. de klasse: ESP32Servo die zich bevindt in het bestand: servos.py. Normaliter bevindt dit bestand zich al op het bord in de map: mylibs.
2. Importeer deze module.
3. Maak een instance van de klasse: ESP32Servo, juist onder de imports.
4. Start de servo (met behulp van de method: start), juist onder het maken van de instance.
5. Maak gebruik van de servo method: move\_to om de servo een bepaalde hoek te laten innemen. (Dit in de while True lus).
6. Maak de volgende schakeling:



## Toevoegen van de node-red ui aan Home Assistant

1. Open Home Assistent en zorg ervoor dat de node-red server runt.
2. Open in Home Assistent de Instellingen en kies vervolgens voor Dashboards in de rechter paneel:

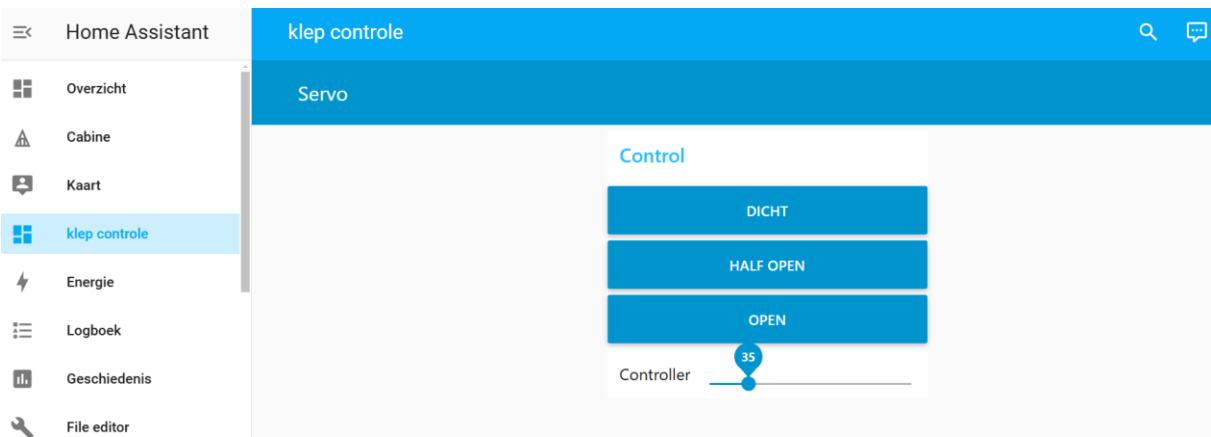


3. Tik vervolgens op blauwe knop: + Dashboard toevoegen (rechts onderaan).
4. Kies in de lijst voor Webpagina:

## Dashboard toevoegen

-  Leeg nieuw dashboard >  
Begin met een leeg dashboard
-  Default dashboard >  
Geef uw apparaten weer gegroepeerd per ruimte
-  Kaart >  
Geef personen en je apparaten weer op een kaart
-  Webpagina >  
Een webpagina integreren als dashboard

5. Vervolledig de URL als volgt: <http://IP-adres> waarop node-red draait:1880/ui
6. Geef vervolgens de titel: klep controle aan het nieuwe dashboard en sla op.
7. Je zal in het linker paneel een verwijzing: klep controle aantreffen.
8. Indien je op die verwijzing drukt, zie je de node-red user interface.

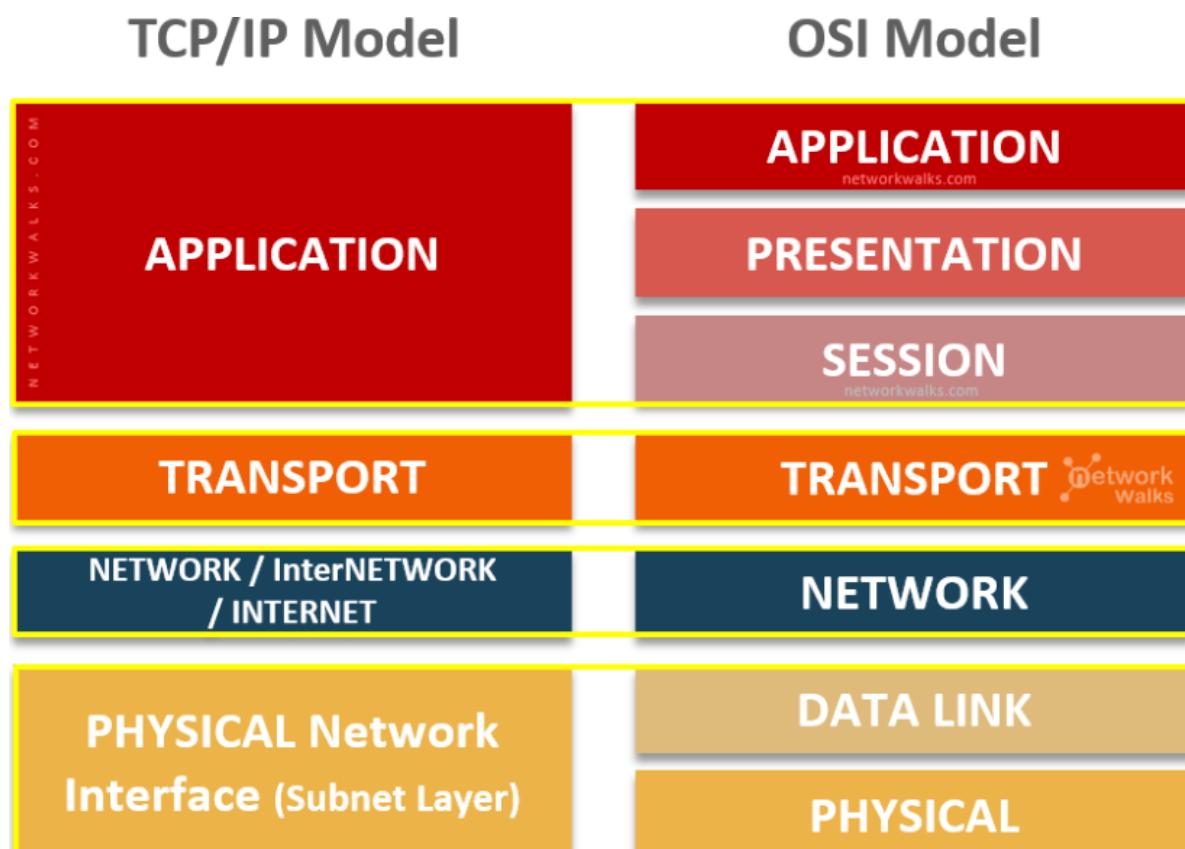


9. Start nu ook het programma op de ESP en test uit. Op deze manier kan je een servo motor vanuit Home Assistant via node-red.

## Communicatie tussen ESP's zonder Wifi (ESPNow)

### Introductie

Tot nu toe verliep de draadloze communicatie met de ESP via Wifi. Dit betekent dat de ESP een IP-adres krijgt van de DHCP server die in de meeste gevallen op de router draait. Hier wordt er gebruik gemaakt van het TCP/IP protocol (dit is in de transport en netwerklaag van het TCP/IP model en OSI model, zie figuur hieronder).



Nu gaan we ESP's met elkaar laten communiceren in een lagere laag, nl. de data link laag in het OSI model en fysische netwerk laag in het TCP/IP model). Dit betekent dat de ESP's geen IP-adres zullen krijgen maar er gebruik zal gemaakt worden van hun MAC adres. Deze methode kan handig zijn als er geen of een zwakke Wifi verbinding is. Dan kunnen verschillende ESP's data met elkaar uitwisselen en kan bijv. de ESP die wel een Wifi verbinding kan aangaan de data doorgeven aan een databank, node-red applicatie ...

### Ophalen van MAC adres ESP en doorgeven aan node-red applicatie

We gaan een klein Python programma schrijven waarin een verbinding wordt gemaakt met Wifi. Het MAC adres wordt opgevraagd en wordt via MQTT, samen met de naam van de ESP, doorgestuurd naar de node-red applicatie. Deze applicatie zal een bestand maken van alle ontvangen koppels (naam esp:MAC adres).

1. Open Thonny en maak een nieuw bestand: getMACAndSend.py

2. Terug zal er gebruik gemaakt worden van de module: SimpleWifi\_v2.py om een verbinding met Wifi te maken.
3. Indien de verbinding gelukt, wordt het MAC adres opgevraagd.
4. Vervolgens wordt een verbinding gemaakt met de broker, via MQTT.
5. Tenslotte wordt de naam en het MAC adres verzonden naar de node-red applicatie.
6. Voordat we de code kunnen schrijven zullen we eerst de module: connectMQTT, die verantwoordelijk is om een connectie te maken met de broker, verder uitbreiden met een functie om een verbinding te maken met de publieke hivemq broker. Dit omdat we de MAC adressen van de volledige klas moeten verzamelen.
  - a. Open de module: connectMQTT (staat op het bord) en voeg de volgende functie toe:

```
def mqtt_connect_public_hivemq():
    mqttcl = MQTTClient("filiplkadeb9582396", "broker.hivemq.com")
    mqttcl.connect()
    return mqttcl
```

*Let op: gebruik een andere client ID (1ste parameter van het MQTTClient object).*

7. Begin met de volgende code, verantwoordelijk voor het verbinden met Wifi te schrijven in het nieuwe bestand:

```
from simpleWifi_v2 import Wifi
from connectMQTT import mqtt_connect_public_hivemq
import sys
import ubinascii
import json
import time

ESP_NAME = "esp_filiplk"

wifi = Wifi()
wifi.open()
if wifi.get_status() <= 0:
    print("probleem verbinden netwerk")
    sys.exit()
```

8. De nodige imports worden ingevoerd. De module ubinascii zal nodig zijn om het MAC adres die in ascii formaat staat om te zetten naar hexadecimaal formaat.
9. De globale variabele: ESP\_NAME moet worden gelijkgesteld aan esp\_je voornaam.
10. Vervolgens wordt het MAC adres opgevraagd, een verbinding gemaakt met de broker en vervolgens de gegevens een 3 tal maal verstuurd met MQTT.

```

MAC = wifi.net.config('mac')
MAC = ubinascii.hexlify(MAC).decode()
print(f"MAC adres:{MAC}")

mqtt = None
mqtt = mqtt_connect_public_hivemq()
if mqtt is None:
    print("probleem connecteren broker")
    wifi.close()
    sys.exit()
payload = json.dumps({"espname":ESP_NAME,"mac":MAC}).encode()
print(payload)
for i in range(3):
    mqtt.publish("data/cvo/mac",payload)
    time.sleep(2)
wifi.close()

```

- ✓ Het MAC adres wordt opgehaald met de network method: config.
- ✓ Het MAC adres wordt van ascii code omgezet naar hexadecimale code met behulp van ubinascii.hexlify(...).
- ✓ Tenslotte wordt het MAC adres en naam van de ESP verstuurd in de JSON structuur:
- ✓ {espname: naam van de ESP,mac: MAC adres}

Een node-red flow maken om de data te accepteren en in een bestand weg te schrijven

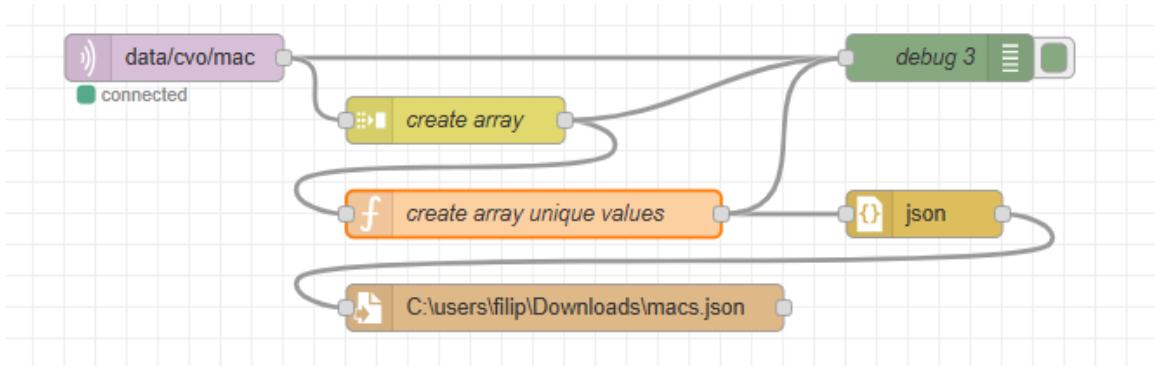
1. Open node-red (eerst een commandline het commando: node-red ingeven, dan de browser laten verwijzen naar localhost:1880).
2. Geef aan de flow de naam: Get MAC data.
3. Sleep een input mqtt node op het werkveld en laat deze verwijzen naar de broker: broker.hivemq.com.
4. Sleep vervolgens een join node op het veld en verbind deze met de mqtt node. De bedoeling is om gedurende 1 minuut data te verzamelen en het weg te schrijven in een array.
5. De kans bestaat erin dat in de rij verschillende maal dezelfde data voorkomt, deze rij gaan we opkuisen zodat iedere waarde maar eenmaal voorkomt, gebruik daarvoor een functie node, verbind deze met de voorgaande nodes en schrijf de volgende JavaScript code:

```

1  var data = msg.payload;
2  var unique = [data[0]];
3  for (var i=0;i<data.length;i++){
4      for (var j=0;j<unique.length;j++){
5          if (data[i].espname == unique[j].espname){
6              break;
7          }
8          if (j == (unique.length-1)){
9              unique.push(data[i]);
10         }
11     }
12 }
13 msg.payload = unique;
14 return msg;

```

- ✓ Er wordt een nieuwe rij unique gedefinieerd waarin al het eerste element van data wordt gestopt.
  - ✓ Dan wordt er door de rij data gegaan om iedere espname te vergelijken met de espname in de rij unique.
  - ✓ Indien aanwezig wordt de zoekactie onderbroken met een break.
  - ✓ Anders wordt het data element toegevoegd aan unique.
6. Voor de zekerheid wordt de rij komende van de functie geparst naar json met behulp van een json node.
  7. Tenslotte wordt de data in een bestand geschreven met de writefile node.
  8. De flow ziet er als volgt uit:



9. Het resultaat van het bestand kan er als volgt uitzien:

```
[{"espname": "esp_filiplk", "mac": "943cc613c8ec"}, {"espname": "esp_sidonia", "mac": "aaaabb775a5a"}]
```

## De basis: één zender en verschillende ontvangers

De ESP van de leraar is de zender. De ESP's van de cursisten zijn de ontvangers. De ESP stuurt een boodschap naar de ontvangers om een LED verbonden met GPIO 22 aan of uit te zetten.

1. Hang een rode LED aan GPIO 22 (vergeet de weerstand niet).
2. Eerst gaan we het programma schrijven voor de zender.
  - a. In principe heeft enkel de zender de MAC adressen nodig van zijn "clients" of ontvangers. Dus we zullen het bestand: macs.json, opladen naar het bord.
  - b. Dan zullen we de inhoud van het bestand inlezen in het programma.
  - c. En tenslotte met behulp van het protocol: ESPNOW een boodschap sturen naar de ontvangers, hier LED on of LED off.
3. Schrijf de volgende code voor de zender en sla op als basic\_espnow\_sender.py in je persoonlijke map.

```
1 import json
2 import network
3 import espnow
4 import time
5
6 esp_name = "esp_filiplk_01"
7
8 #lezen json bestand
9 addrs = None
10 with open("macs.json","r") as f:
11     addrs = json.load(f)
12 print("adressen:",addrs)
13
14 #Eerst het netwerk interface aanleggen omdat espnow het netwerk interface gebruikt.
15 net = network.WLAN(network.WLAN.IF_STA)
16 net.active(True)
17 net.disconnect()
18
19 #espnow object maken
20 espnet = espnow.ESPNow()
21 espnet.active(True)
22
23 #toevoegen van clients/ontvangers
24 for a in addrs:
25     x = bytearray.fromhex(a["mac"])
26     print(x)
27     espnet.add_peer(x)
28
29 while True:
30     espnet.send("f">from:{esp_name},cmd:LED=on<\n".encode())#zenden naar alle ontvangers
31     time.sleep(5)
32     espnet.send("f">from:{esp_name},cmd:LED=off<\n".encode())
33     time.sleep(5)
```

### Wat uitleg:

- Van regel 9 tem 12 wordt het json bestand geladen en de inhoud van het bestand gestopt in de variabele: addrs. De variabele addrs wordt een lijst met dictionaries.
- In de volgende regels wordt er een netwerk object gemaakt en geactiveerd. Dit netwerk interface zal gebruikt worden door het ESPNOW protocol.
- Vervolgens wordt het ESPNOW object gemaakt en gestopt in de variabele: espnet.

- In de lijnen: 24 tem 27, worden alle ontvangers (de mac adressen van de ontvangers) toegevoegd aan het espnow object (espnow.add\_peer(...)). Het mac adres dat als string is opgeslagen in het json object (of hier dictionary) wordt met de bytarray method: fromhex(...) omgezet naar bytes.
  - In de while lus wordt er om de 5 seconden een boodschap gestuurd naar alle geregistreerde ontvangers. Dit met de method: send.  
Er bestaat ook nog een method: send(mac,msg,[sync]). Met deze method kan je een boodschap sturen naar één ontvanger, de sync kan True of False zijn, indien True wordt er gewacht op een ACK van de ontvanger, anders wordt er niet gewacht.
4. Schrijf vervolgens de code voor de ontvanger, zie hieronder en sla op als basic\_espnow\_recv.py.

```

1 import network
2 import espnow
3
4 #Eerst het netwerk interface aanleggen omdat espnow het netwerk interface gebruikt.
5 net = network.WLAN(network.WLAN.IF_STA)
6 net.active(True)
7 net.disconnect()
8
9 #espnow object maken
10 espnet = espnow.ESPNow()
11 espnet.active(True)
12
13 while True:
14     host, msg = espnet.recv()
15     if msg:           # msg == None if timeout in recv()
16         print(host, msg)

```

#### ***Wat uitlegt:***

De eerste lijnen tem 11 zijn terug het activeren van het netwerk interface en het maken van een espnow object.

- In de while lus wordt met de espnow method: recv gewacht op een inkomende boodschap. Deze method retourneert 2 variabelen, nl.: host: het mac adres van de zender en msg: de boodschap.
- Indien er een boodschap is, wordt deze getoond in de shell.

#### Opdrachten

##### *Opdracht 1: LED aan/uit bij ontvanger*

Wijzig de code van de ontvanger zodat de LED aan gaat als de boodschap LED=on wordt gegeven en uit gaat als deze de boodschap LED=off ontvangt.

1. Definieer het LED object.
2. Wijzig de code in de while lus zodat de LED aan of uit gaat.

*Opdracht 2: from {esp\_name} supprimeren uit de boodschap en bij de ontvanger opvragen via macs.json.*

1. Wijzig de boodschap in de zender.
  - a. Van: >from:{esp\_name},cmd:LED=on</n
  - b. Naar:>cmd:LED=off</n
2. Doe hetzelfde voor LED=off
3. Wijzig de ontvanger als volgt:
  - a. Lees bij de opstart het macs.json bestand en stop in de variabele: addrs. (zie zender).
  - b. Maak een lijst: espnames van de namen van de esp's.
  - c. Maak een lijst: macs van de mac adressen.
  - d. Maak een dictionary: mac2name met als sleutel het MAC-adres en waarde de corresponderende naam. Dit kan je als volgt: mac2name = dict(zip(macs,espnames)).
  - e. Gebruik deze dictionary om de inhoud van de host om te zetten naar de naam van de zender. (Je zal de host moeten omzetten in string, dit kan via de volgende expressie: ubinascii.hexlifyhost).decode()
  - f. Print de ontvangen boodschap met corresponderende zender in de shell.

## Maken van een zender/ontvanger met het ESPNOW protocol

Om een wifi arme omgeving van een netwerk met ESP's te voorzien, zal elke ESP een ontvangst en zend functie moeten bezitten. Op deze manier kan de boodschap ontvangen door één ESP doorgegeven worden aan andere ESP's. Dan kan er bijvoorbeeld ook 1 ESP in het netwerk zitten die de ontvangen boodschappen via wifi doorgeeft aan een broker, node-red applicatie ...

Om "tegelijk" te kunnen ontvangen en verzenden, zullen we gebruik maken van de asyncio module. Deze zorgt voor het asynchroon uitvoeren van de code.

Hieronder een korte uitleg in verband met het asynchroon programmeren in MicroPython.

### Asyncio in het kort (uitleg chatgpt)

`asyncio` in **MicroPython** is een **lichtgewicht** implementatie van asynchrone programmering, waarmee je **niet-blokkerende** taken kunt uitvoeren. Dit betekent dat meerdere taken tegelijk kunnen worden verwerkt zonder dat de ene taak de andere ophoudt.

Basisprincipes:

1. `async def` → Definieert een asynchrone functie (coroutine).
2. `await` → Wacht op een andere coroutine zonder het programma te blokkeren.
3. `loop.create_task()` → Start een taak op de achtergrond.
4. `loop.run_forever()` → Houdt de event loop draaiende.

Voorbeeld:

```
python
KopiërenBewerken
import uasyncio as asyncio

async def taak1():
    while True:
        print("Taak 1 draait")
        await asyncio.sleep(1)
```

```

async def taak2():
    while True:
        print("Taak 2 draait")
        await asyncio.sleep(2)

loop = asyncio.get_event_loop()
loop.create_task(taak1())
loop.create_task(taak2())
loop.run_forever()

```

**💡 Uitleg:** Beide taken worden tegelijk uitgevoerd, maar zonder elkaar te blokkeren!

Van start met het maken van zender/ontvanger

1. Open het bestand: basic\_espnow\_sender.py en sla op als espnow\_sender\_receiver.py.
2. Breid de imports uit met de volgende modules:
  - a. import uasyncio as asyncio
  - b. import ubinascii

Met asyncio om functies asynchroon te kunnen uitvoeren.

Met ubinascii om het MAC-adres in byteformaat om te zetten naar string.

3. Declareer de 2 volgende variabelen:

```

CNT = 0
times_to_send = 2

```

Met CNT een variabele die een teller bijhoudt, deze teller zal een aantal maal gestuurd worden naar de peers.

times\_to\_send: het aantal maal dat CNT wordt verstuurd. Het aantal maal versturen is ingevoerd om verder liggende Esp's ook te kunnen bereiken.

4. Maak een dictionary: mac2esp voor mac-adres te kunnen vertalen naar naam van de ESP (soort van omgekeerde dns). En definieer 2 ESPNOW objecten één om te zenden: espnet\_send en één om te ontvangen: espnet\_recv:

```

#vullen van mac2esp
macs = []
esps = []
for m in esp2mac:
    macs.append(m["mac"])
    esps.append(m["espnname"])

mac2esp = dict(zip(macs,esps))
print(mac2esp)

espnet_send = espnow.ESPNow()
espnet_send.active(True)

espnet_recv = espnow.ESPNow()
espnet_recv.active(True)

```

5. Voeg de peers toe aan het espnet\_send object:

```
#toevoegen van de peers in het netwerk
for m in esp2mac:
    espnet_send.add_peer(bytarray.fromhex(m["mac"]))
```

6. Maak de zend taak:

```
async def send(times):
    print("start zender")
    global CNT
    while True:
        for i in range(times):
            espnet_send.send(">count:"+str(CNT)+"<\n").encode()
            print("boodschap:", ">count:"+str(CNT)+"<\n", "verzonden")
            await asyncio.sleep(0.05)

        CNT += 1
        await asyncio.sleep(0.05)
```

Een taak is een functie die voorafgaat met `async`. Deze functie verwacht 1 parameter, nl. `times` die bepaalt hoeveel maal dezelfde boodschap wordt verstuurd.

`CNT` is een globale variabele (hoefde eigenlijk niet globaal te zijn maar is eerder ter illustratie dat `global` ook kan gebruikt worden in een `async` functie).

In de oneindige lus wordt de teller een `times` aantal keer verzonden met een pauze van 50 ms (`await asyncio.sleep(0.05)`), die ervoor zorgt dat andere taken verder kunnen worden uitgevoerd.

Tenslotte wordt `CNT` met 1 verhoogd en wordt er weer voor 50ms ruimte gegeven aan andere taken.

7. Vervolgens schrijf je de taken voor het ontvangen van boodschappen:

```
async def recv():
    host = None;msg = None
    host,msg = espnet_recv.recv(50)
    print("(1)",host,":",msg)
    return host,msg

async def getmsg(mac2esp):
    print("start ontvanger")
    while True:
        host,msg = await recv()
        print(host,":",msg)
        if msg is not None and host is not None:
            shost = ubinascii.hexlify(host).decode()
            print("ontvangen boodschap:",msg.decode(),"van:",mac2esp[shost])
        await asyncio.sleep(0.05)
```

De taak: recv roept de espnow recv method op. De parameter 50 betekent dat deze method 50 ms wacht op inkomende boodschappen. Host en boodschap (msg) worden gereturneerd.

De taak: getmsg verwacht 1 parameter, nl. mac2esp.

In de oneindige lus wordt de taak recv opgeroepen. Men kan hier await gebruiken omdat er in de taak: recv 50ms wordt gewacht, dus terug ruimte voor andere taken.

Vervolgens wordt de ontvangen boodschap met bijhorende esp naam van de zender in de shell geprint. De naam van de esp wordt uit de dictionary: mac2esp gehaald.

Daarvoor moet de host die in byte vorm staat omgezet worden naar string met behulp van ubinascii.hexlify(...).decode().

Tenslotte wordt er terug een pauze van 50ms ingelast zodat andere taken ook ademruimte krijgen.

8. Uiteindelijk worden de taken gecreëerd en wordt een oneindige lus opgestart:

```
loop = asyncio.get_event_loop()
loop.create_task(send(times_to_send))
loop.create_task(getmsg(mac2esp))
loop.run_forever()
```

De basic ontvanger wat aanpassen om de gekregen boodschap terug te sturen naar de zender (een soort van echo)

1. Open het bestand: basic\_recv.py en sla op als basic\_recv\_echo.py.
2. Wijzig de code op het einde als volgt:

```
try:
    espnet.add_peer(host)
except:
    pass
espnet.send(("I received msg:"+msg.decode()).encode())
time.sleep(0.05)
```

3. In de try wordt er geprobeerd om de host bij de peers toe te voegen. De try .... except is nodig omdat de host maar 1x kan toegevoegd worden aan de peers.

## Opdrachten

### Opdracht 1: Gegevens DHT11 sensor doorsturen naar peers

1. Maak terug de schakeling: DHT11 sensor verbinden met de ESP32 (zie pagina 13).
2. Verstuur de boodschap met data van de DHT11 sensor naar de peers. Formaat boodschap:

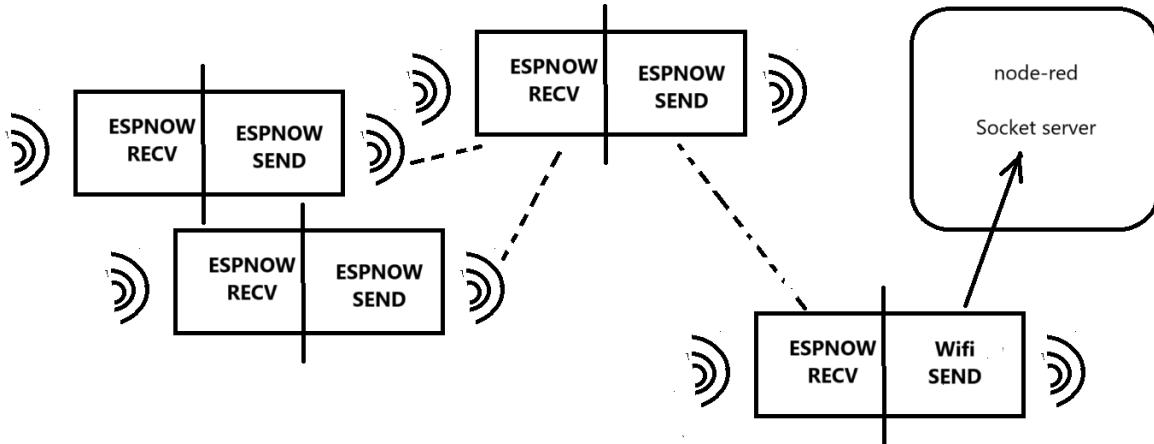
>measure\_id:x,temp:25,hum:35<\n

Maak gebruik van CNT als measure\_id, temp is de temperatuur en hum is de vochtigheid.

## Ontvangen met ESPNOW protocol, verzenden via Wifi

De volgende component die we gaan maken is een ESPNOW ontvanger en “Wifi” zender. Op deze manier kunnen er een aantal ESP sensoren die geen Wifi bereik hebben de data aan elkaar doorgeven totdat de data komt bij de ESP met ESPNOW ontvangst functionaliteit en wifi zend functionaliteit.

Hier gaan we de volgende opstelling maken:



De ESP component: ESPNOW RECV | Wifi SEND moeten we nog maken, samen met de node-red: Socket server. Het ontvangen signaal wordt via Wifi doorgegeven aan node-red. Hierbij wordt er gebruik gemaakt van het TCP/IP protocol waarbij de ESP als socket client fungiert en de node-red flow als socket server. Dus node-red wacht op inkomend signaal (= listen).

### Maken ESPNOW ontvang/ Wifi zend module

1. Open het script: basic\_espnow\_sender\_receiver.py en sla op als espnow\_recv\_and\_wifi\_send.py.
2. Wijzig de volgende zaken in de imports:
  - a. Vervang import network door from simpleWifi\_v2 import Wifi.
  - b. Importeer sys en socket.
3. Maak vervolgens een verbinding met Wifi, indien dit niet lukt stop het programma met behulp van sys.exit().
4. Definieer bovenaan de volgende globale variabelen:
  - a. SERVER: deze zal het IP-adres bevatten waarop de socket-server (en dus ook node-red) draait.
  - b. PORT: de poort om een connectie te maken met de socket server (PORT=7555).
  - c. QUEUE = []: lijst met boodschappen die toekomen en dan worden doorgegeven aan de wifi-zend taak.
5. Verwijder het maken van het esp\_send object (want nu gaan we zenden met behulp van Wifi).
6. Wijzig de taak: send(times) als volgt en geef de taak de naam: send\_socket():

```

async def send_socket():
    print("start socket zender")
    global QUEUE, SERVER, PORT
    while True:
        if len(QUEUE) == 0:
            await asyncio.sleep(0.05)
            continue
        print(QUEUE)
        s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect((SERVER,PORT))
        data = QUEUE.pop(0)
        s.send(data.encode())
        s.close()
        await asyncio.sleep(0.05)

```

### **Wat uitlegt:**

- Er wordt gekeken naar de globale variabele: QUEUE waarin de boodschappen die toekomen via ESPNOW worden opgeslagen.
  - Indien er niets in de QUEUE zit, wordt de boodschap aan de event loop doorgegeven dat er een andere taak aan de beurt is.
  - Indien er elementen in de QUEUE zitten, haal het eerste element op en stuur het naar de socket server.
  - Er wordt eerst een socket object gemaakt. Dan wordt er een verbinding gemaakt met de socket server. Vervolgens wordt de gewenste data verstuurd en wordt de socket gesloten. Terug wordt de boodschap gegeven aan de event loop dat er een andere taak aan de beurt is.
7. In de taak getmsg verwijst je naar de globale variabele: QUEUE.
- a. Voeg de ontvangen boodschap toe aan de QUEUE.
  - b. Zorg ervoor dat de boodschap maar 1x voorkomt in de QUEUE, dit kan met behulp van de set functie. Zie code hieronder.

```

QUEUE.append(mac2esp[shost]+":"+msg.decode())
QUEUE = list(set(QUEUE))

```

Maken van de socket server in node-red.

1. Start node-red (via de terminal).
2. Open de browser en ga naar het adres: 127.0.0.1:1880.
3. Maak de flow: flow\_socketserver.
4. Sleep een tcp in node op het werkveld en zet de listen port gelijk aan: 7555. Zorg ook dat de node is ingesteld om strings te ontvangen.
5. Hang een debug node achter de tcp-node.
6. We gaan de nieuwe module testen door in de klas enkele: espnow\_sender\_receiver.py scripts op te starten en op andere esp's: espnow\_recv\_and\_wifi\_send.py op te starten.
7. Als alles goed verloopt zouden er enkele boodschappen moeten toekomen in node-red.

## Loggen van data in Home Assistant in een influxdb

InfluxDB integreren met Home Assistant is een uitstekende manier om historische gegevens te loggen en te analyseren, bijvoorbeeld met Grafana. Hier is een stap-voor-stap gids om **InfluxDB te installeren en configureren in Home Assistant**:

## 1. InfluxDB installeren

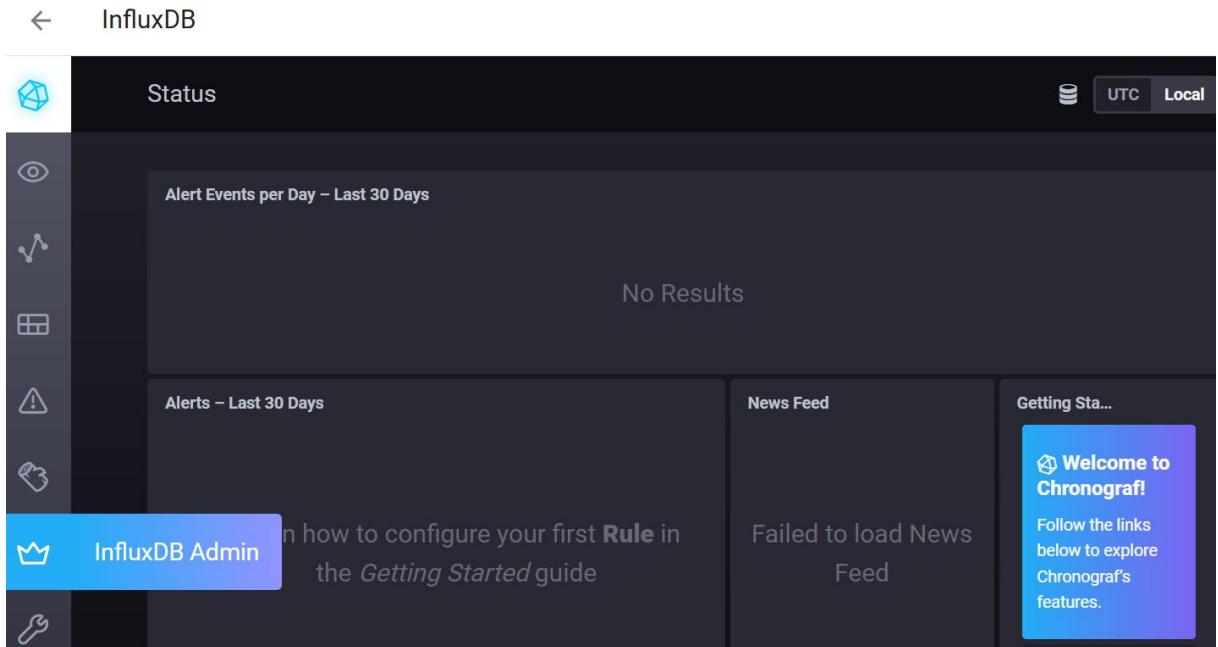
- ◊ Via Home Assistant Add-on Store

1. Ga naar **Instellingen > Add-on Store**.
  2. Zoek naar **InfluxDB** en installeer de add-on.
  3. Open de add-on en klik op **Start**.
  4. (Optioneel) Zet **Start on boot** en **Watchdog** aan.

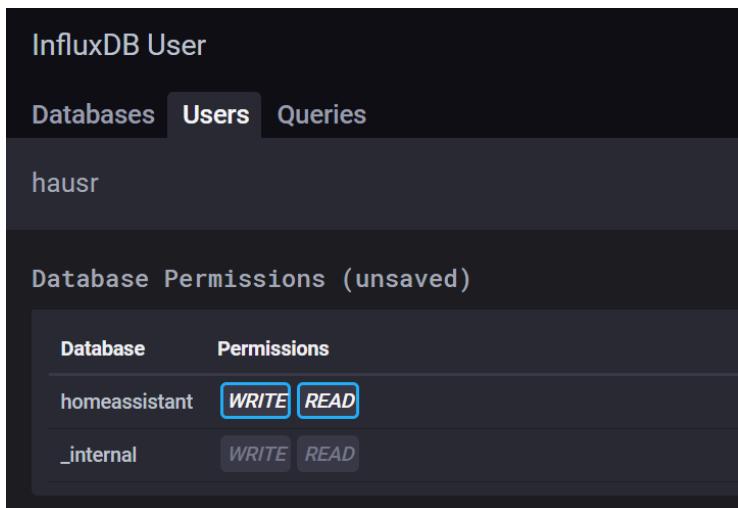
- ### 5. Kijk eens naar de Log:

## 2. InfluxDB configureren

1. Ga naar de **web UI** van InfluxDB via de knop **Open Web UI** of [http://<home\\_assistant\\_ip>:8086](http://<home_assistant_ip>:8086).
2. Druk op het kroontje (influxDB Admin) in de linker balk:



3. Maak een **database** aan, bijv. `homeassistant` door in het volgende scherm te tikken op: + CREATE DATABASE, een naam aan de databank te geven en te tikken op de groene knop met wit vinkje.
4. Maak een **gebruiker** aan met rechten op de zopas gemaakte database, door te tikken op het tabblad: USERS. Maak in het volgende blad volgende gebruiker:
  1. Gebruiker: `hausr`
  2. Wachtwoord: `Hetcvo.be`
5. Duid de rechten READ/WRITE aan voor de databank: `homeassistant`.



Database	Permissions
homeassistant	<code>WRITE</code> <code>READ</code>
_internal	<code>WRITE</code> <code>READ</code>

6. Druk op de knop: Apply Changes. De databank met bijhorende gebruiker is gemaakt.

---

### 3. Home Assistant configureren om InfluxDB te gebruiken

#### 1. Open je configuration.yaml bestand en voeg het volgende toe

```
influxdb:  
  host: 127.0.0.1 # of het IP van je InfluxDB als het extern is  
  port: 8086  
  database: homeassistant  
  username: hausr  
  password: Hetcvo.be  
  max_retries: 3  
  default_measurement: state  
  include:  
    domains:  
      - sensor  
      - switch  
      - climate
```

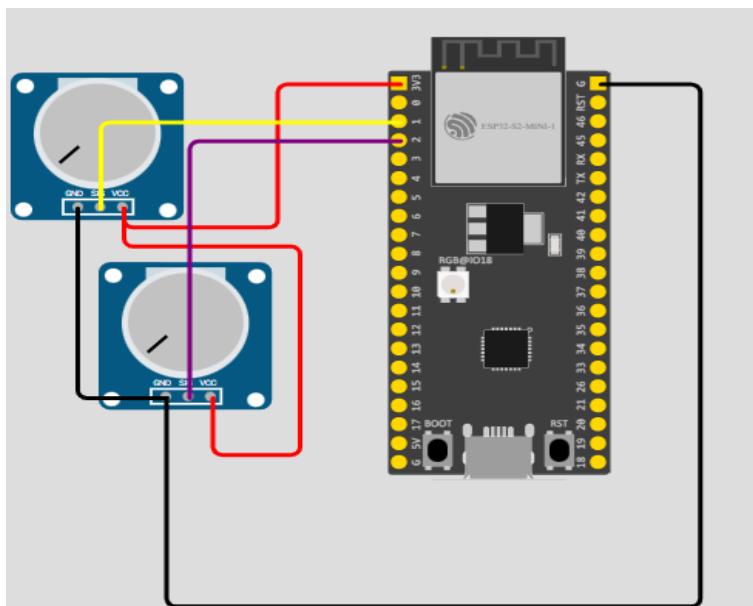
◊ Pas `include:` aan afhankelijk van welke entiteiten je wilt loggen. Je kunt ook `exclude:` gebruiken.

#### 2. Herstart Home Assistant om de configuratie toe te passen.

Opdracht: de waarden van de potentiometers wegschrijven naar de influxdb databank

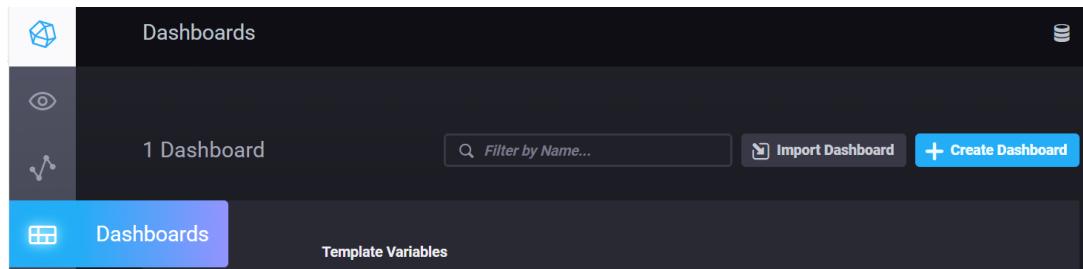
Enkele lessen geleden hebben we 2 potentiometers verbonden met de ESP32. Deze 2 potentiometers worden met behulp van autodiscover mechanisme, via MQTT, automatisch geregistreerd in Home Assistant. Ook de waarden van de potentiometers worden via MQTT doorgegeven aan Home Assistant.

#### 1. Maak terug de volgende schakeling:

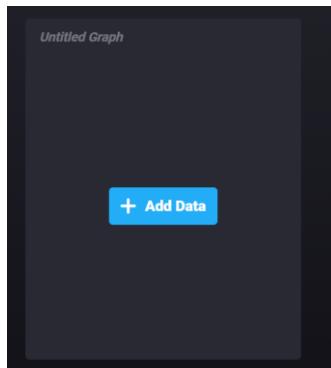


De ene potentiometer is verbonden met pin 36, de andere met pin 39.

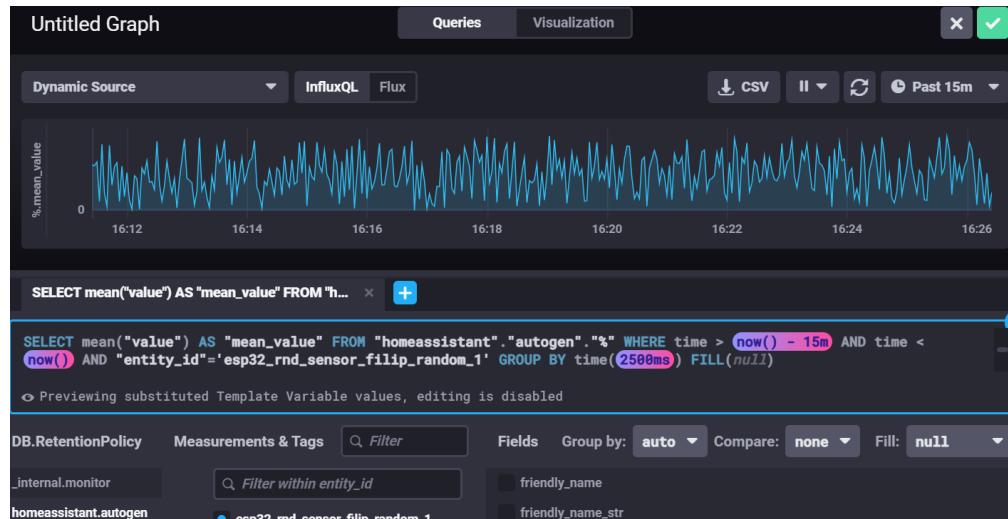
2. Open het script: esp\_2\_pot\_sensor.py en voer dit uit.
3. Ga kijken in het dashboard van Home Assistant, na een tijdje zullen er getallen verschijnen naast potentiometer 1 en potentiometer 2 (de sensoren zijn operationeel en klaar om geregistreerd te worden in de influxdb).
4. Ga terug naar het yaml configuratie bestand in Home Assistant. Zorg dat enkel influxdb domein sensor blijft staan. Herstart Home Assistant.
5. Nu is alles klaar en kan er in influxdb een dashboard worden gemaakt voor de 1<sup>ste</sup> potentiometer.
  - a. Open influxdb in Home Assistant.
  - b. Tik in de linker balk op Dashboard.



- c. Druk op de knop: + Create Dashboard
- d. Klik in het volgende paneel op de knop: Add Data



- e. Selecteer in het volgende paneel de gewenste databank (hier: homeassistant.autogen (links onderaan)).
- f. De termen: eenheden (hier %) en state komen tevoorschijn. (Opmerkingen als er verschillende sensoren zijn elk met verschillende eenheden dan zullen al deze eenheden worden getoond).
- g. Klik % open en vervolgens entity-id\_2. Selecteer daar de 1<sup>ste</sup> entiteit, nl. esp32\_pot1\_sensor ....
- h. Druk onder fields op value. Een query wordt gemaakt en de corresponderende grafiek wordt daarboven getekend.



- i. Door te tikken op Untitled Graph, kan je de naam horende bij de grafiek wijzigen. Wijzig de naam naar: potentiometer 1.
- j. Druk op de groene knop met wit vinkje. Het datapaneel is gemaakt.
- k. Klik vervolgens Name This Dashboard om een naam te geven aan het dashboard. Geef het de naam: Potentiometer sensors.
6. Volg het bovenstaande stappenplan om de grafiek voor potentiometer 2 toe te voegen. Druk op de knop: Add Cell to Dashboard (kleine knop met 3 vierkantjes en +, bovenaan) om een paneel toe te voegen.
7. Het resultaat kan er tenslotte als volgt uitzien:



## Inhoudsopgave

Eigen LDR sensor koppelen aan Home Assistant .....	2
Maken van de schakeling .....	2
De nodige code schrijven om via MQTT ontdekt te worden door HA .....	2
Het connecteren met de HiveMQ Cloud Cluster onderbrengen in een nieuw bestand .....	2
Het automatisch laten detecteren van onze sensor door Home Assistant.....	3
Waarde van LDR naar Home Assistant sturen .....	7
Slimme lamp besturen met behulp van eigen LDR sensor.....	8
Opdracht:.....	8
Slimme lamp doven bij een LDR waarde boven 55% .....	8
Stap per stap uitgelegd.....	8
Opdrachten: .....	13
Opdracht 1: een automatisering maken om de slimme lamp te laten branden als de waarde van de LDR sensor daalt onder de 45%.....	13
Opdracht 2: een temperatuur en vochtigheidssensor maken en verbinden met Home Assistant ..	14
Opdracht 3: maken van automatiseringen.....	15
Stap 1: Open de automatiseringen UI .....	16
Stap 2: Naam en omschrijving instellen .....	16
Stap 3: Triggers instellen .....	16
Stap 4: Acties instellen met "Kiezen" (If-Else) .....	17
Scenario 1: Lamp UIT als temperatuur < 20°C en vochtigheid < 40% .....	17
Scenario 2: Lamp AAN als temperatuur tussen 20-30°C en vochtigheid tussen 40-60%.....	17
Scenario 3: Lamp feller als temperatuur > 30°C en vochtigheid > 60% .....	17
Stap 5: Opslaan en testen.....	18
Samenvatting.....	18
Stap 1: Open het automations.yaml bestand.....	18
Stap 2: Voeg de volgende code toe aan automations.yaml .....	19
Stap 3: Pas de instellingen aan .....	20
Stap 4: Home Assistant herstarten.....	20
Stap 5: Test de automatisering.....	20
Samenvatting.....	20
Een controller interface, gemaakt in node-red, toevoegen aan Home Assistant.....	21
Opdracht: maken van UI in Node-Red .....	21
Opdracht maken van de socket server op de ESP.....	22
Opdracht servo aansluiten en nodige code schrijven om servo aan te sturen.....	23

Toevoegen van de node-red ui aan Home Assistant.....	23
Communicatie tussen ESP's zonder Wifi (ESPNow) .....	25
Introductie.....	25
Ophalen van MAC adres ESP en doorgeven aan node-red applicatie .....	25
Een node-red flow maken om de data te accepteren en in een bestand weg te schrijven.....	27
De basis: één zender en verschillende ontvangers .....	29
Opdrachten.....	30
Maken van een zender/ontvanger met het ESPNOW protocol .....	31
Asyncio in het kort (uitleg chatgpt).....	31
Basisprincipes: .....	31
Voorbeeld: .....	31
Van start met het maken van zender/ontvanger .....	32
De basic ontvanger wat aanpassen om de gekregen boodschap terug te sturen naar de zender (een soort van echo).....	34
Opdrachten.....	34
Opdracht 1: Gegevens DHT11 sensor doorsturen naar peers .....	34
Ontvangen met ESPNOW protocol, verzenden via Wifi.....	35
Maken ESPNOW ontvang/ Wifi zend module .....	35
Maken van de socket server in node-red.....	36
Loggen van data in Home Assistant in een influxdb .....	37
🔧 1. InfluxDB installeren.....	37
◊ Via Home Assistant Add-on Store .....	37
⚙️ 2. InfluxDB configureren .....	38
💡 3. Home Assistant configureren om InfluxDB te gebruiken .....	39
Opdracht: de waarden van de potentiometers wegschrijven naar de influxdb databank .....	39