

# ESP 32

## Koppelen van de ESP met verschillende platformen



## Maken van de toepassing: waste manager

**Doel:** de afval ophaalkalender opladen in een node-red applicatie: waste manager. Deze afvalmanager gaat na of er de komende dag afval wordt opgehaald, indien dit het geval is wordt een boodschap gestuurd naar een applicatie op de ESP32, dit via het protocol MQTT.

Er zijn een aantal LEDs gekoppeld aan de ESP, afhankelijk van de boodschap gaat een bepaalde LED(s) branden, zie hieronder de tabel:

Boodschap	LED
Huisvuil	rood
PMD	blauw
GFT	groen
Glas	wit
Papier	geel
Kerstboom	groen aan/uit + geel aan/uit
None	Geen

Als de gebruiker de LED ziet branden kan hij de gewenste afval buitenzetten en vervolgens op een knop drukken om het systeem duidelijk te maken dat de boodschap is ontvangen en de taak is uitgevoerd.

Er is ook nog een 2<sup>de</sup> drukknop met de ESP verbonden. Deze heeft een Reset functie. Eenmaal er op deze knop wordt gedrukt, wordt de reset boodschap doorgegeven aan de node-red applicatie zodat de gebruiker een nieuwe afvalkalender kan invoegen

## Maken van de node-red applicatie: waste manager

De applicatie gaat uit 2 panelen bestaan, nl. een File Manager om het gewenste afval ophaalkalender op te laden en dan de waste manager zelf die op een periodiek tijdstip gaat kijken naar de kalender om dan de juiste boodschap door te sturen.

### Opdracht: Van start met de File Manager

1. Voor de File Manager moet men eerst de node: node-red-contrib-fs. Met deze node kan je bestanden, mappen raadplegen in een bepaalde lokale map.
2. Na het installeren van deze node kan je op het internet surfen naar de volgende site: <https://flows.nodered.org/flow/44bc7ad491aacb4253dd8a5f757b5407> (of zoeken op node-red browse local folders and files).
3. Op deze website vind je de code voor een volledige File Manager in node-red, zie figuur hieronder:

Flow ID: 44bc7ad491aacb4253dd8a5f757b5407

Note: some third-party nodes may appear with blank styling, and not as they appear in the Node-RED Editor.

```
[{"id":"4fa73dd9.83cca4","type":"comment","z":"74f191ff.db063","name":"File Browser","info":"1"}]
the default folder in the Init node(n2) Update the default folder in the Reset node as well(n3) You
can duplicate the Reset nodes and use them as saved shortcuts(n4) Check the Convert Timestamps
function node if you want to see your dates in a different format. I just used the Javascript
toISOString format.(n5) The Graph button can read any CSV file which have any number of values, but
the first column always contains a timestamp column with javascript timestamp of the data
point.", "x":110,"y":2940,"wires":[]},{ "id":"993d7272.843ae","type":"fs-file-
lister","z":"74f191ff.db063","name":"Files","start":"/home/pi","pattern":"*.*","folders":"","hidden":
false,"lstype":"files","path":true,"single":true,"depth":0,"stat":true,"showWarnings":false,"x":510,"y
```

Copy

- ui\_tab (x1)
- ui\_template (x1)
- ui\_text (x1)
- ui\_toast (x1)

Tags

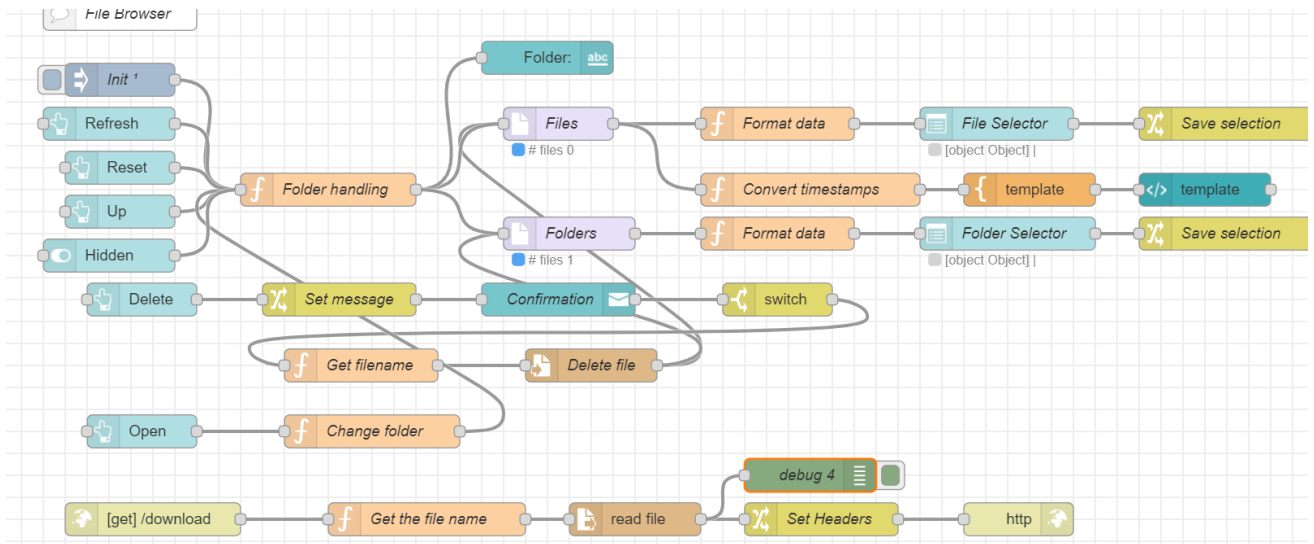
browser dashboard

Copy this flow JSON to your clipboard and then import into

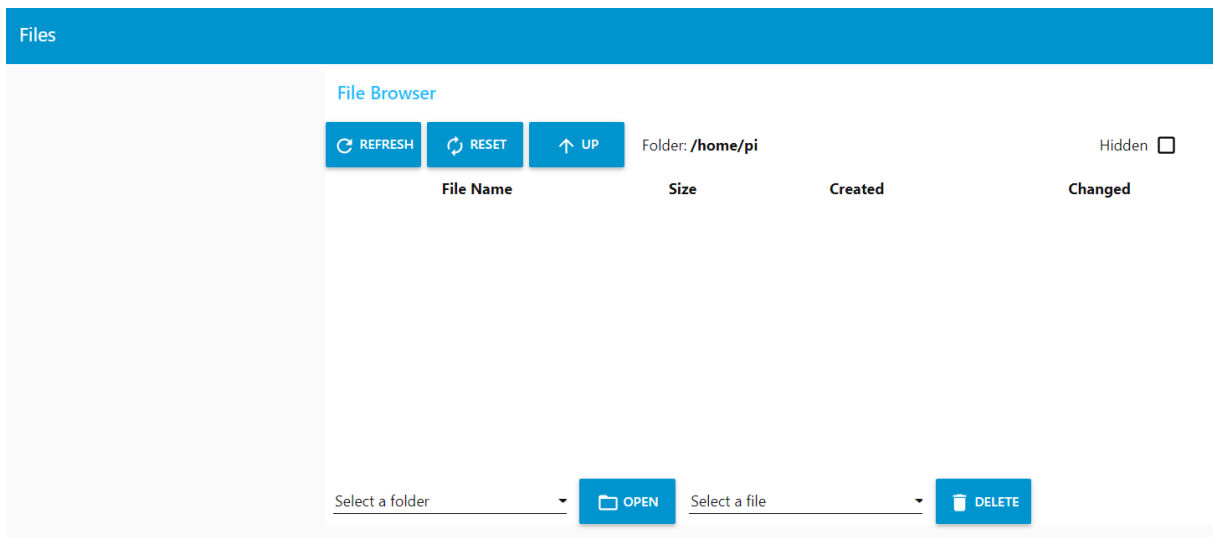
Tik op Copy, rechtsboven het zwarte scherm. Open klad blok en plak deze code.

4. Sla het bestand op als: filemanager.json in je persoonlijke map onder Documenten.

5. Start node-red (open console en voer commando: node-red in).
6. Importeer deze filemanager.json in een lege flow (tip: slepen van verkenner naar de flow).
7. Geef aan de flow de naam: FileManager.
8. Verwijder de nodes die gekoppeld zijn aan Graph (hebben we niet nodig want we gaan geen grafiek maken van onze kalender).
9. Plaats reeds een debug node achter de node: Get the file name.
10. De flow ziet er als volgt uit:



11. Open het node-red ui gedeelte, het scherm ziet er als volgt uit:



12. **Bijkomende opdracht:** zorg bij het starten van de applicatie dat het veld Folder de waarde / krijgt in plaats van /home/pi.
13. Ga naar classroom: IoT deel 3, tik op data IoT deel 3 en klik vervolgens op de map: data. Download het bestand: ILvA-OPHALING\_2024\_begin\_2025.ics. Plaats dit in je persoonlijke map in Documents.
14. Maak gebruik van de zopas gemaakte file manager in node-red om het bestand op te halen, je zou in het debugvenster de verwijzing moeten zien naar het bestand.

15. Voeg achter inject (Init) node een functie node toe en vervolledig deze als volgt:

**Edit function node**

Delete Cancel Done

**Properties**

Name: init kalender file

Setup On Start **On Message** On Stop

```
1 global.set("KALENDER", "");
2 return msg;
```

Op deze manier maken we een globale variabele: KALENDER die we in andere flows kunnen gebruiken.

16. De bedoeling is om de globale variabele KALENDER te vullen met het pad en de naam van de afvalkalender, probeer dit in orde te krijgen. Tip: functie node op dezelfde plaats van de zelf toegevoegde debug node.

Opdracht: maken van de waste messenger

1. Maak een nieuwe flow: WasteManager.
2. We gaan de flow: WasteManager laten opstarten door de flow FileManager, dit door gebruik te maken van de link nodes.
3. Voeg in de flow: FileManager een link out node toe en verbind deze met de uitgang van de node: Get the filename. Geef aan de link out node de naam: link out flow filemanager.

**Edit link out node**

Delete Cancel Done

**Properties**

Name: link out flow filemanager

Mode: Send to all connected link nodes

4. Voeg een link in node toe aan de nieuwe flow, laat deze verwijzen naar de link out node: link out flow filemanager.

**Edit link in node**

Delete Cancel Done

**Properties**

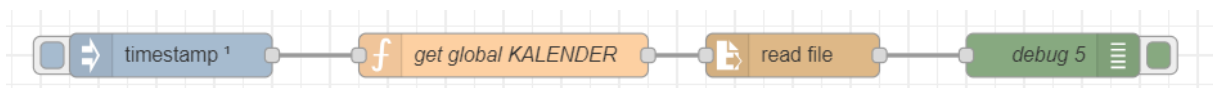
Name link in 1

link out flow filemanager

- Voeg vervolgens een functie-node toe die in wacht staat totdat de globale variabele: KALENDER is ingevuld, zou je als volgt kunnen maken (vertaal de pseudocode):

***zolang de globale variabele KALENDER == "" blijf in de lus.  
zet msg.payload = inhoud globale variabele KALENDER  
retourneer msg***

- Voeg daarna een read file node toe. Je kan de inhoud van het bestand checken door daarachter een debug node te plaatsen.



- De bedoeling is om uit het bestand de afhaaldatum en soort afval dat wordt opgehaald eruit te filteren. Ieder afhaalmoment heeft het volgende formaat:

```

BEGIN:VEVENT
UID:sLVyafXxyR8eP1HKumsdz
SUMMARY:ILvA OPHALING: Huisvuil
DTSTAMP:20240623T163537Z
DTSTART;VALUE=DATE:20250319
END:VEVENT
  
```

De lijnen die beginnen met SUMMARY: ILvA OPHALING: en DTSTART;VALUE=DATE: bevatten de gewenste data. Het is de bedoeling om deze data in een rij bij te houden. We gaan daarvoor een nieuwe functie-node toevoegen waarin de gewenste gegevens in een rij wordt gestopt.

8. Dit kan men bekomen met het volgend stuk JavaScript code:

```
1  let data = msg.payload;
2  const arr = data.split("\n");
3  let dates_wastes = [];
4  let month_now = new Date().getMonth();
5  month_now+=1;
6  let year_now = new Date().getFullYear();
7  let waste = "";
8  let date = "";
9  for(let j = 0;j<arr.length;j++){
10     if (arr[j].includes("SUMMARY:ILvA OPHALING:")){
11         let parts = arr[j].split(":");
12         waste = parts[parts.length-1];
13     }
14     if (arr[j].includes("DTSTART;VALUE=DATE:")){
15         let parts = arr[j].split(":");
16         date = parts[parts.length-1];
17     }
18     if (waste !="" && date != ""){
19         let y_date = Number(date.substring(0,4));
20         let m_date = Number(date.substring(4,6));
21         if (m_date>=month_now || y_date>year_now){
22             dates_wastes.push({"date":date, "waste":waste.trim()});
23             waste = "";
24             date = "";
25             continue;
26         }
27     }
28 }
29 msg.payload = dates_wastes;
30 return msg;
```

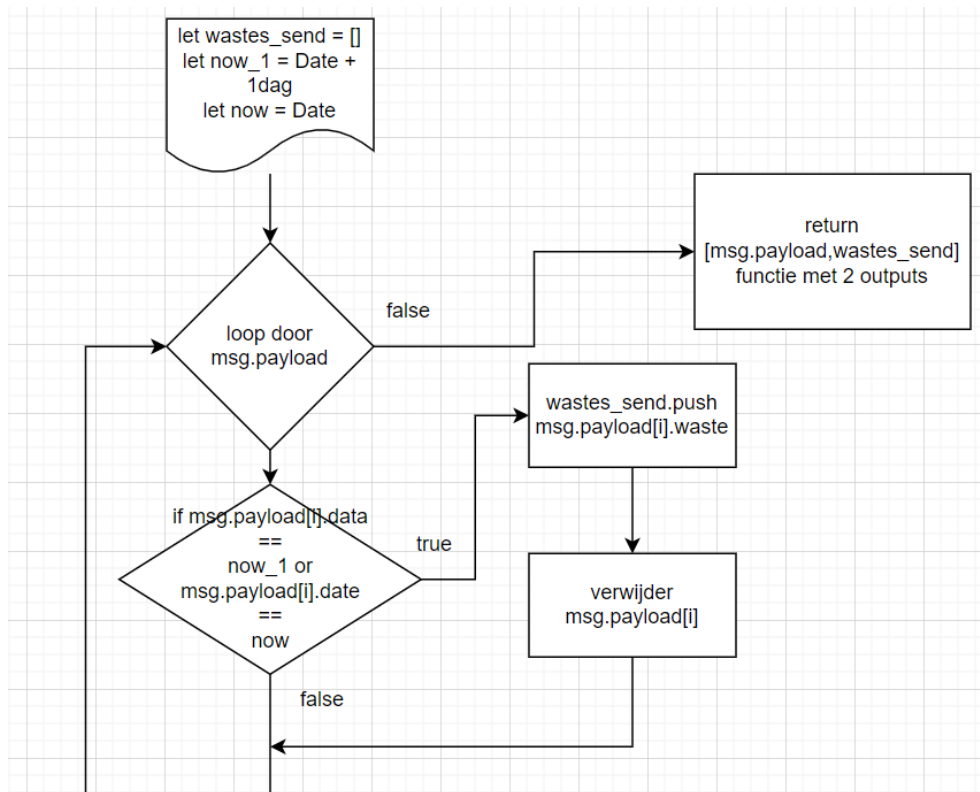
9. De bovenstaande JavaScript code kan je downloaden vanuit classroom: IoT deel 3 → data IoT deel 3 → data. Naam van het bestand: get\_dates\_and\_wastes.js. Open dit bestand kopieer de code en plak dit in een nieuwe functie-node. Geef aan de functie-node de naam: get dates & wastes.
10. Plaats deze functie node achter de read file node.

#### Wat uitleg over het JavaScript:

- ✓ De inhoud van het bestand wordt in de variabele: data gestopt.
- ✓ Iedere lijn in het bestand wordt een element in de rij: arr.
- ✓ Er worden bij bijkomende help variabelen gemaakt, nl:

- ✓ dates\_wastes: om de ophaalmomenten en corresponderende soorten afval in op te slaan.
- ✓ Vervolgens worden de huidige maand en jaar opgehaald om te gebruiken als filter zodat enkel deze van huidige maand en volgende maanden wordt opgeslagen in de rij: dates\_wastes.
- ✓ Vervolgens wordt door de rij arr gegaan om de soort op te halen afval en bijhorende datum op te halen en op te slaan in de rij: dates\_wastes. Daarbij worden enkel de data van de huidige maand en volgende maanden opgehaald. De rij: dates\_wastes zal bestaan uit json objecten = een lijst van koppels. Hier een koppel: datum en een koppel: soort af te halen afval.

11. Nu ga je zelf een functie-node maken die de dag van vandaag + 1 dag vergelijkt met de datum in de verschillende objecten. Als de dag overeenkomt wordt deze uit de rij geplukt en de soort afval doorgegeven aan de ESP32 via mqtt.
12. Voeg een nieuwe functie-node: waste\_msg toe aan de flow.
  - a. Probeer aan de hand van de volgende flowchart de JavaScript code te schrijven:



- b. Enkele tips:

- ✓ Je zal de functie van 2 outputs moeten voorzien, nl. een output van de bewerkte rij = inkomende rij met eventueel geschrapte elementen en de soorten afval die (morgen) worden opgehaald. Functie node van 2 outputs voorzien:

Open de functie node, tik op de knop Setup en wijzig outputs naar 2.

Edit function node

Delete

Cancel

Done

⚙️ Properties

⚙️ 📄 🖨️

📌 Name

function 1

📄 ▼

⚙️ Setup

On Start

On Message

On Stop

🔗 Outputs

2

⬆️ ⬇️ ⬆️

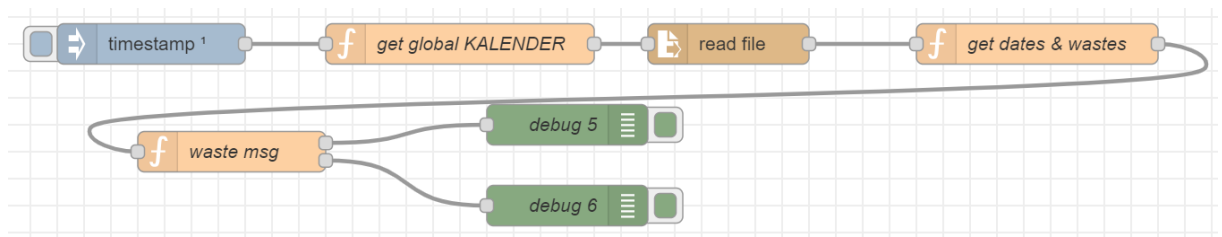
📦 Modules

Module name	Import as

Als je een bepaalde JavaScript method of functie niet kent, zoek deze op.

- ✓ Maak gebruik van ChatGPT.

13. Verbind beide uitgangen met een “eigen” debug node, zie figuur hieronder met mogelijk resultaat:



25-6-2024, 22:11:51 node: debug 5

msg.payload : array[92]

```

▶ [ object, object, object, object,
  object, object, object, object, object,
  object ... ]

```

25-6-2024, 22:11:51 node: debug 6

msg.payload : array[1]

```

▶ [ "Huisvuil" ]

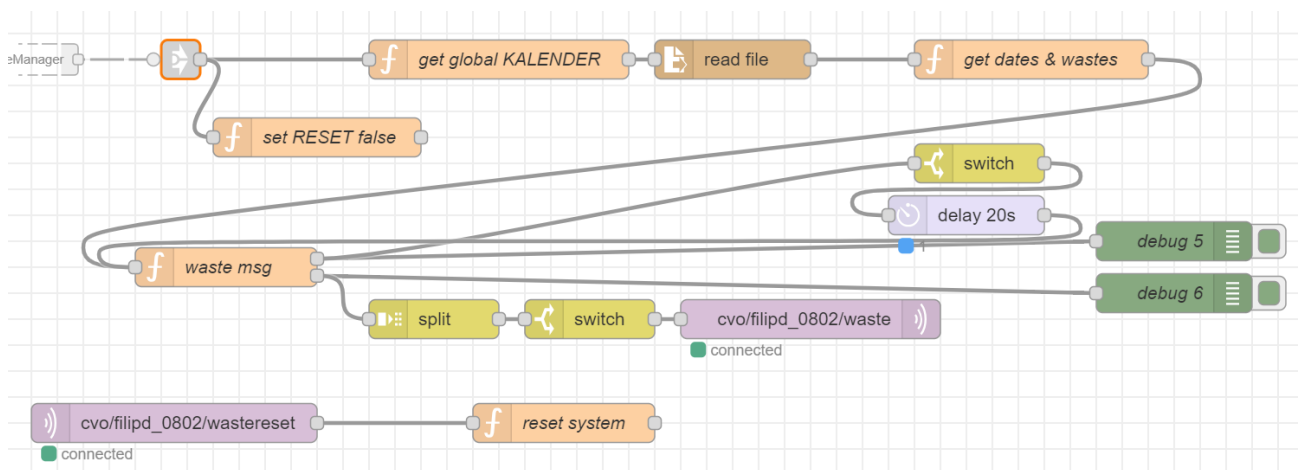
```

14. Nu moeten we de 1<sup>ste</sup> output van de functie terugsturen naar de input en de 2<sup>de</sup> output splitsen en doorsturen via MQTT naar de broker (hier hivemq). Je kan als topic: cvo/je voornaam\_enkele cijfers/waste gebruiken.

15. Maak eerst dat de 2<sup>de</sup> output werkt. Je zal een split moeten toevoegen aan de 2<sup>de</sup> output en vervolgens kijken dat het binnenkomende resultaat niet leeg (empty) is (met een switch node). Tenslotte sluit je deze tak af met een MQTT out node, gebruik de bovenvermelde broker en topic.



16. Ga naar de site: hivemq en zie dat de boodschap binnenkomt (als er (natuurlijk) een afval afhaling wordt verwacht).
17. Het is de bedoeling om via de ESP32 een reset te kunnen doorvoeren zodat er een nieuw afvalbestand kan worden opgeladen. De boodschap van de ESP32 kan worden ontvangen door een MQTT in node. Sleep zulk een node op het werkveld en laat deze verbinden met de broker: hivemq en laat de node luisteren naar de topic: cvo/je voornaam\_enkele willekeurige cijfers/reset.
18. Verbind deze met een functie-node waarin je de flow variabele: RESET op true zet en de globale variabele: KALENDER gelijkstelt aan een lege string.
19. Best hang je ook een functie-node aan de link node waarin je de flow variabele: RESET gelijkstelt aan false.
20. Verbind de 1<sup>ste</sup> output van de functie-node: waste msg met een switch die nagaat dat de flow variabele: RESET gelijk is aan false, indien het geval dan kan de output van de functie: waste msg terug doorgegeven worden aan de input van deze functie.
21. Plaats eventueel een delay tussen de switch en functie-node: waste msg, zodat de functie niet te “snel” terug wordt aangeroepen.
22. Hieronder de volledige flow:

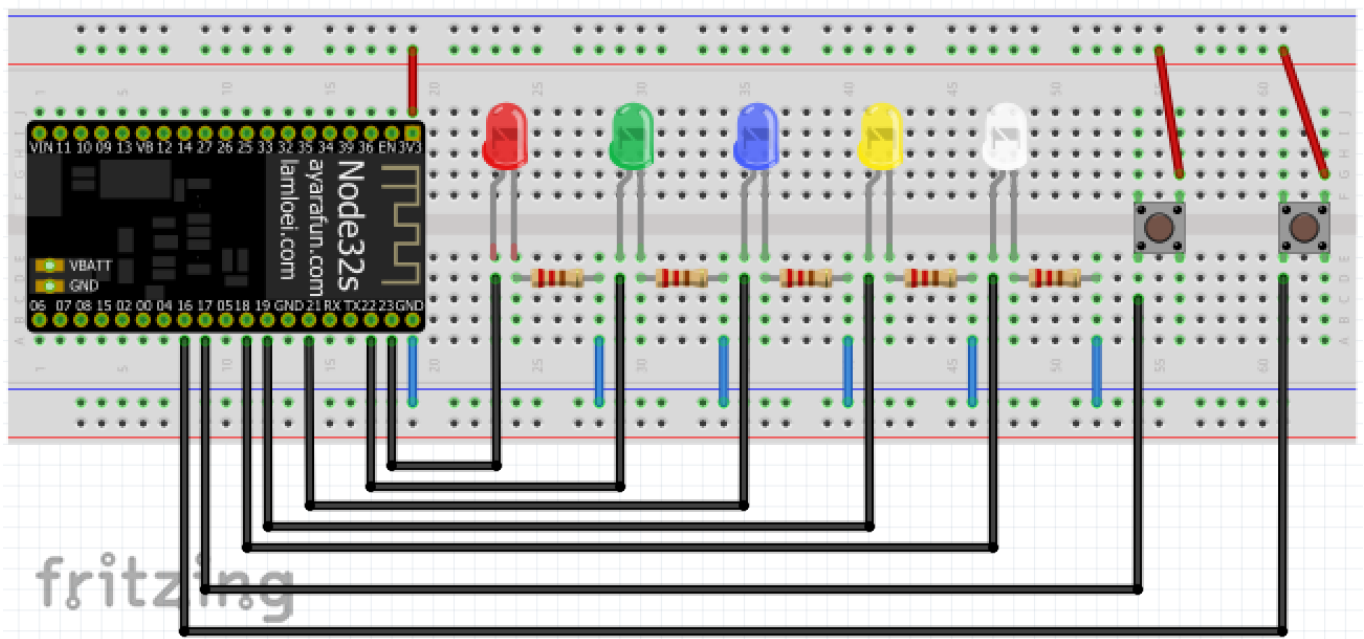


23. Test het geheel uit door gebruik te maken van de hivemq client.

Het ESP32 gedeelte

Maken van de schakeling

Maak de volgende schakeling:



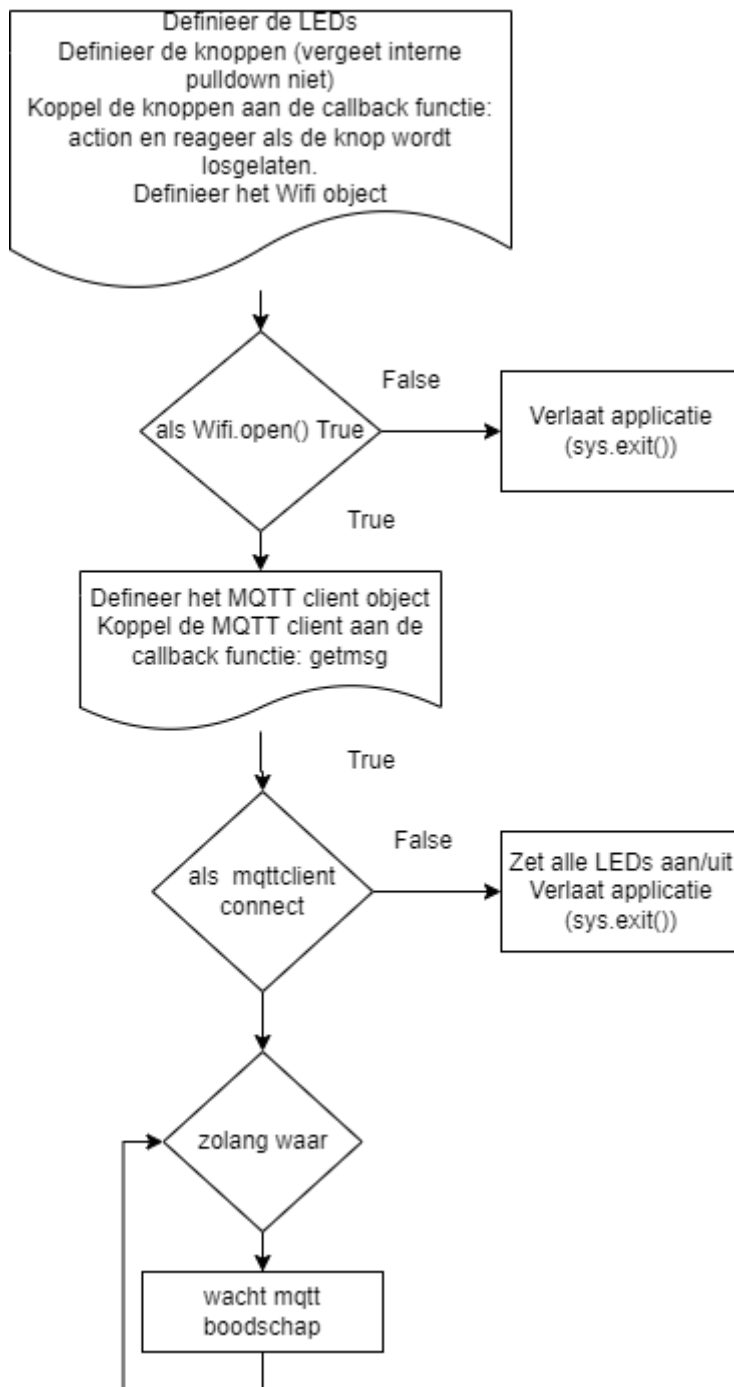
✓ De LEDs worden aangesloten op de volgende pinnen:

Boodschap	LED	GPIO Pin
Huisvuil	Rood	23
PMD	Blauw	21
GFT	Groen	22
Glas	Wit	18
Papier	Geel	19
Kerstboom	groen aan/uit + geel aan/uit	
None	Geen	

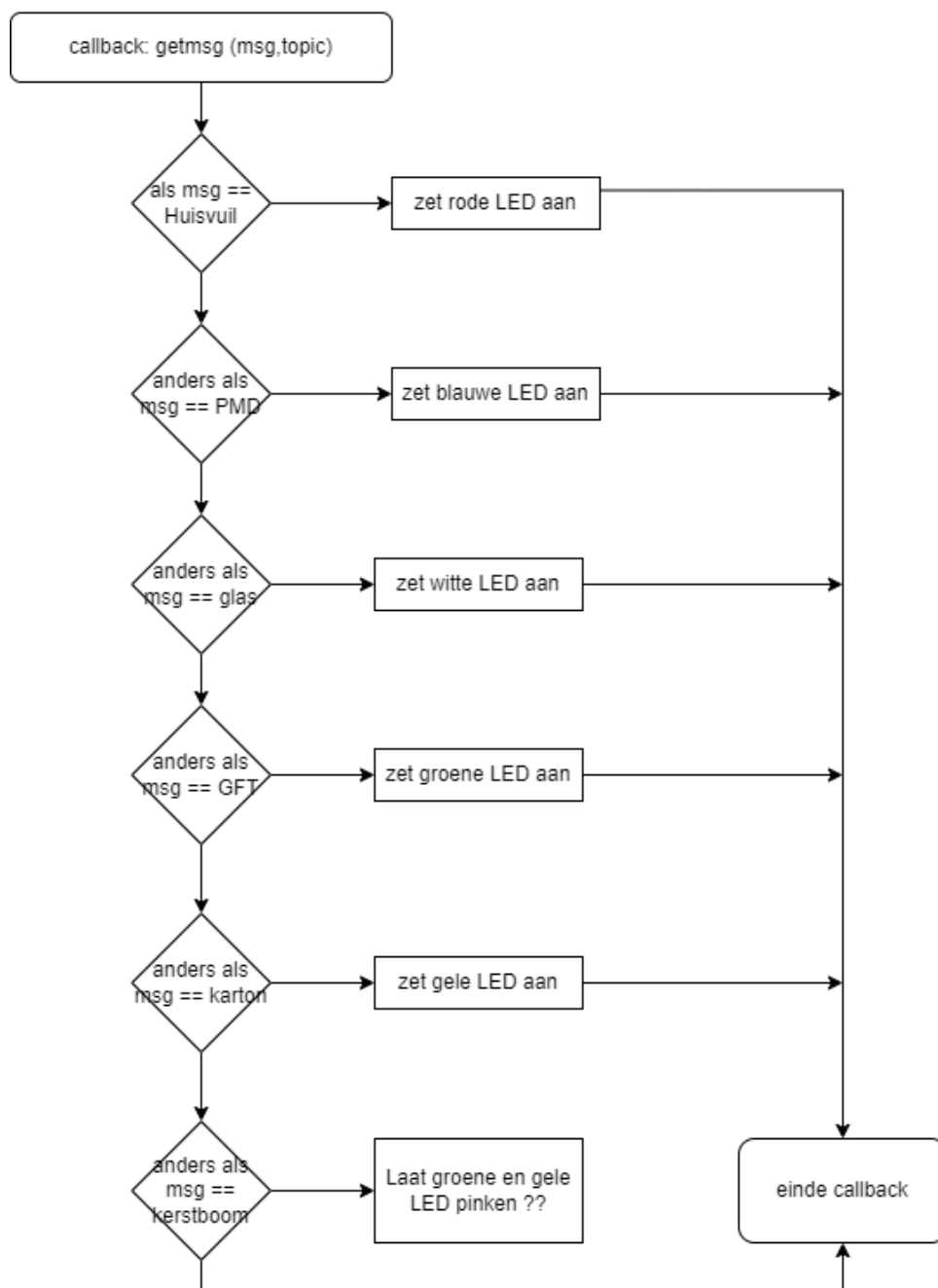
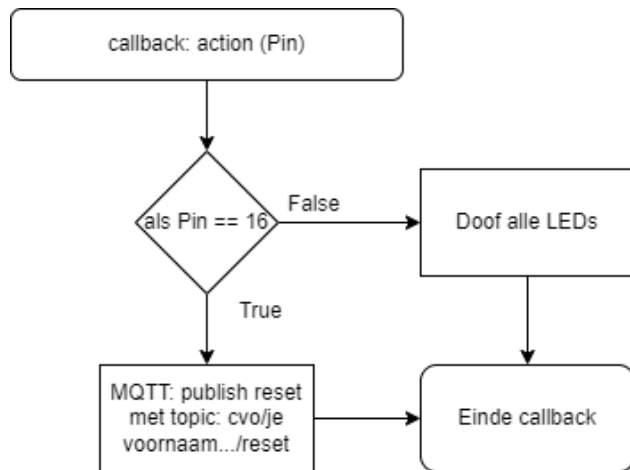
- ✓ Knop om de LED(s) terug op nul te zetten, is verbonden met pin: 17.
- ✓ De RESET knop is verbonden met pin: 16.
- ✓ Beide knoppen zijn met een interne pull-down weerstand verbonden met de GND.

Opdracht de nodige code schrijven voor de ESP32

1. Zorg dat de juiste firmware is geflasht op de ESP32 (indien dit niet het geval is, flash de firmware).
2. Zorg dat de modules voor simple Wifi V2 op het bord staan.
3. Vertaal de volgende flowchart naar micropython code.



4. Natuurlijk moeten de callback functies: action en getmsg verder worden uitgeschreven, zie flowcharts hieronder:



Bij de voorwaarde: `msg == kerstboom` moeten de groene en gele LEDs pinken. Dus in principe moeten we daar van start gaan met een oneindige lus waarin de toestand van de LEDs van uit naar aan gaan en omgekeerd. Maar deze oneindige while-lus zou hier opgestart worden in een callback functie en de volledige applicatie blokkeren en zo kunnen nieuwe MQTT boodschappen niet meer binnenkomen. We zullen hiervoor een nieuwe techniek moeten invoeren, nl. een timer object. Dit object kan periodiek een callback functie oproepen.

5. Dus importeer eerst de Timer module:

```
from machine import Timer
```

6. Maak een callback functie: flicker die de gele en groene LEDs uit of aanzet.

```
def flicker(timer):
```

```
    geel(not geel())
```

```
    groen(not groen())
```

7. Definieer bovenaan een Timer object:

```
timer_flicker = Timer(-1)
```

De parameter -1 is ingevoerd om het systeem duidelijk te maken dat het hier gaat om een virtuele timer.

8. Als in de callback functie: `getmsg` de voorwaarde `msg == kerstboom` voldaan is, initialiseer de timer: `timer_flicker`:

```
timer_flicker.init(mode=Timer.PERIODIC, callback=flicker, period=2000)
```

Dit zal om de 2 seconden de callback functie flicker oproepen. Deze functie zal de LEDs aanleggen als ze uit zijn en omgekeerd.

9. Natuurlijk zal men ook het geflicker moeten afleggen eenmaal er op de stop knop wordt gedrukt. Je zal de volgende code moeten toevoegen aan de callback functie: action:

```
try:
```

```
    timer_flicker.deinit()
```

```
except:
```

```
    pass
```

Voor het doven van alle LEDs, probeert men het timer object: `timer_flicker` te deinitialiseren. Lukt dit niet, omdat bijv. het timer object niet geïnitieerd is, dan wordt er naar de except gesprongen, maar met behulp van `pass` gaan we verder in de code.

10. Plaats het bovenstaande stukje code op de goede plaats.

## Opdracht: maken van een App met IoT MQTT Panel

Indien deze App nog niet op de Smartphone of iPhone staat, kan je deze installeren via respectievelijk de Play of App store.

---

Hieronder een kleine uiteenzetting van hoe je een connectie maakt met een Broker en hoe een paneel wordt gemaakt

### Een connectie maken met een broker

1. Tik linksboven op de menuknop (3 lijntjes) en kies voor All Connections
2. Tik vervolgens op + rechtsonder.
3. Vul de velden in. We gaan gebruik maken van de broker hivemq. Zie figuur hieronder:

← Edit Connection

Connection name  
testA

Client ID  
abc\_1234567\_hetcvo\_fdes08

Broker Web/IP address  
broker.hivemq.com

Port number  
1883

Network protocol  
TCP

Add Dashboard

mijndash

Additional options

CANCEL SAVE

4. Druk op de + naast Add Dashboard en geef een naam aan het dashboard.
5. Druk tenslotte op Save.

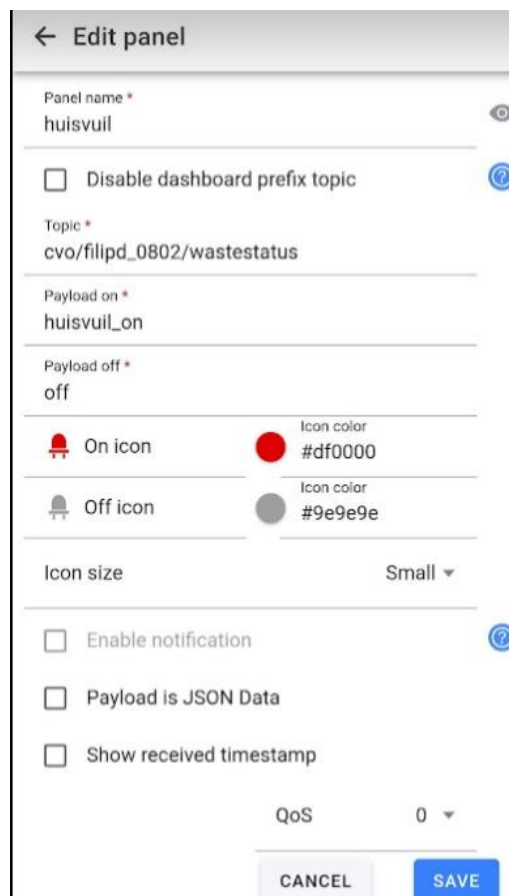
### Elementen toevoegen aan het dashboard

1. Open het dashboard (door te tikken op de connectie in de lijst).
2. Tik op de + en kies het gewenste element.

1. Maak in de App het volgende dashboard, hieronder een stukje:



2. Voeg voor elk afvaltype een LED indicator toe. Voor de kerstboom kan je ook één LED indicator gebruiken die donkergroen zal oplichten als deze wordt opgepikt.
  - a. Als topic kan je cvo/je voornaam\_met enkele cijfers/wastestatus gebruiken.
  - b. De payloads:
    - i. Om de indicator aan te zetten is: afvaltype\_on (bijv. huisvuil\_on).
    - ii. On de indicator af te zetten: off
  - c. Geef ook een bepaalde kleur aan de indicator, bijv. voor huisvuil: rood, pmd: blauw ....
  - d. Hieronder het voorbeeld van de huisvuil indicator:

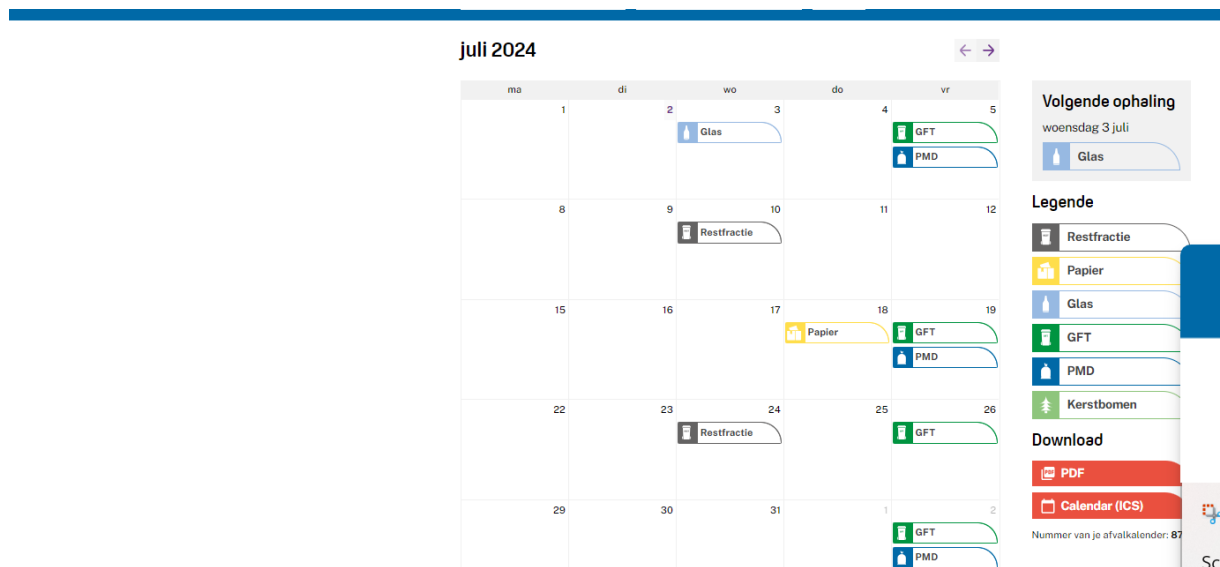


3. Wijzig vervolgens het MicroPython programma op de ESP32 zodat bij het aanzetten van een LED de corresponderende topic, payload wordt gepubliceerd naar de Broker via MQTT, bijv. als de LED voor huisvuil wordt geactiveerd, stuur naar de Broker de boodschap: huisvuil\_on gekoppeld aan de topic: cvo/je voornaam\_enkel cijfers/wastestatus.
4. Als er op de stopknop wordt gedrukt, moet via MQTT de boodschap off, gekoppeld aan de topic: cvo/je voornaam\_enkele cijfers/wastestatus worden gestuurd naar de Broker.
5. Zorg dat het ESP32 script automatisch wordt opgestart bij start van de ESP32.

### Opdracht: eigen huisafval kalender integreren

Momenteel maak je gebruik van de afvalkalender van de leraar, maar de afvalkalender van jouw woonplaats zal er normaliter anders uitzien.

1. Voor de meeste mensen die in Oost-Vlaanderen wonen zullen deze kunnen raadplegen via de Ilva site, [nl.ilva.be/afvalkalender](https://nl.ilva.be/afvalkalender). Daar vul je in de velden bovenaan jouw adres in en bekom je een volgend resultaat:



2. Je kan dan tikken op de knop: Calendar (ICS) en zo krijg je de kalender in het gewenste formaat
3. Het kan gebeuren dat er bij bepaalde afvalmaatschappijen geen ICS bestand ter beschikking is, maar eerder een PDF.
  - a. Dan zal je de PDF moeten inlezen en de gewenste gegevens ophalen en eventueel zelf een ICS maken, zodat een groot stuk van de bestaande code verder kan gebruikt worden.
  - b. Bij mij lukt het niet om een PDF met verschillende pagina's binnen te halen via node-red.
    - i. Wel kan je een PDF bestand lezen in Python met behulp van de module: pypdf en de nodige data uit het PDF bestand filteren.
    - ii. Je zou zelf een soort van ICS bestand kunnen maken met als entry:

**BEGIN:VEVENT**  
**SUMMARY:ILvA OPHALING: PMD**  
**DTSTART;VALUE=DATE:20240105**



## **END:VEVENT**

- iii. Op deze manier kan je de code behouden in node-red, want je maakt een ICS die door de bestaande code kan gelezen worden.
  - iv. Je kan de file manager in node-red verder uitbreiden met een knop: van PDF naar ICS en het Python script uitvoeren in node-red.
  - v. Een Python script kan via de exec node worden uitgevoerd ....
4. Eenmaal dat jouw afvalkalender is ingevoerd, test jouw afval manager of indicator uit.

## De servo en stepper motor

### De servo (als klep)

Een servomotor is een elektrische motor die wordt gebruikt voor precisiebewegingen en positiecontrole. Het bevat een motor, een positie-sensor (zoals een potentiometer of encoder), en een besturingselektronica. Door een feedbacksysteem kan de servomotor nauwkeurig de gewenste positie, snelheid of hoek aanhouden. Dit maakt servomotoren ideaal voor toepassingen in robotica, drones, 3D-printers en automatisering, waar precieze en gecontroleerde bewegingen nodig zijn. Ze werken meestal op basis van een pulsbreedtemodulatie (PWM)-signaal dat de gewenste positie stuurt.

Aan de hand van de breedte (duty cycle) van de pulse wordt de servo naar een bepaalde hoek gestuurd. Hieronder het algemene principe om een servo naar 90° te laten draaien volgens chatGPT:

De duty cycle die je nodig hebt om een servomotor 90 graden te draaien hangt af van het specifieke type servomotor dat je gebruikt, aangezien verschillende servomotoren verschillende instellingen hebben. Over het algemeen worden servomotoren aangestuurd door een PWM-sigitaal (pulsbreedtemodulatie) waarbij de duur van de puls de positie van de motor bepaalt.

Hier is een algemeen overzicht:

1. **Basisinstellingen:** Veel standaard servomotoren hebben een bereik van 0 tot 180 graden en worden aangestuurd met een PWM-sigitaal met een frequentie van ongeveer 50 Hz (20 ms per cyclus). In dit geval:
  - Een puls van ongeveer 1 ms (milliseconde) komt overeen met 0 graden.
  - Een puls van ongeveer 2 ms komt overeen met 180 graden.
2. **Duty cycle voor 90 graden:** Om de servo op 90 graden te zetten, gebruik je een puls die ongeveer in het midden ligt tussen deze twee waarden, namelijk rond 1,5 ms.
  - De duty cycle voor een 1,5 ms puls bij een frequentie van 50 Hz is ongeveer 7,5% (omdat  $1,5 \text{ ms} / 20 \text{ ms} = 0,075$  of 7,5%).

Controleer altijd de specificaties van je specifieke servomotor voor nauwkeurige waarden, omdat de exacte pulsduur en duty cycle kunnen variëren tussen verschillende modellen en fabrikanten.

Dus normaliter komt 0° overeen met een pulsbreedte van 1ms en 180° met een pulsbreedte van 2ms.

Als we de frequentie (f) van het signaal zelf willen instellen, zouden we de volgende formule kunnen bedenken om de servo op een bepaalde hoek te zetten:

$$\text{duty\_cycle} = 1.023 * f * (\text{hoek} * (\text{pmax} - \text{pmin}) / 180 + \text{pmin})$$

Met:

- ✓ duty\_cycle: getal tussen 0 en 1024.
- ✓ hoek: de gewenste hoek (tussen 0° en 180°)
- ✓ pmin: breedte puls in ms voor servo op 0° te zetten.
- ✓ pmax: breedte puls in ms voor servo op 180° te zetten.
- ✓  $1.023 * f$ : omreken factor van ms naar een getal tussen 0 en 1023, wat wordt verwacht als parameter voor de PWM method: duty.

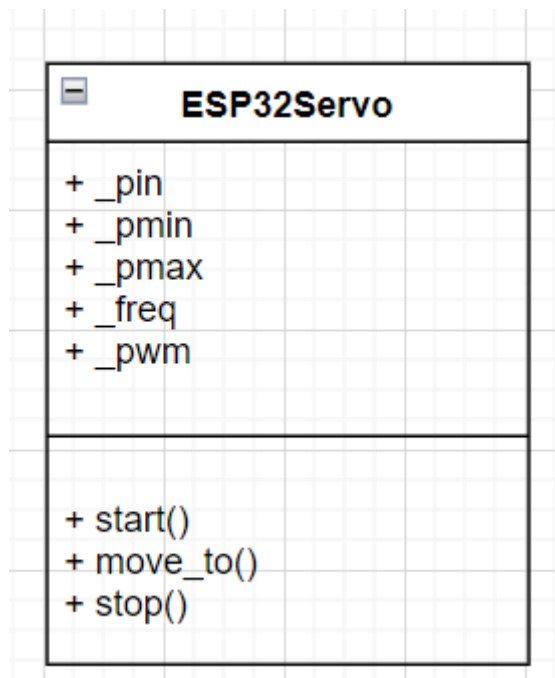
De bovenstaande formule gaan we gebruiken in onze eigen klasse: ESP32Servo.

De member, properties of eigenschappen van deze klasse zullen: pin, pwm, pmin, pmax en freq (frequentie) zijn, dit bij initialisatie kunnen worden ingesteld.

De klasse zal de volgende methods (acties) bevatten:

- ✓ `__init__`: de constructor van de klasse. (voor het maken van het object of instance).
- ✓ `start`: om de PWM te starten op de gewenste pin.
- ✓ `move_to`: om de servo naar een bepaalde hoek te laten bewegen (daar gaan we ook de formule gebruiken).
- ✓ `stop`: om het PWM object te vernietigen.

Hieronder de schematische voorstelling van de klasse: ESP32Servo:



Nu gaan we deze klasse met bijhorende methods implementeren.

1. Maak in je persoonlijke map de map `iot_modules`.
2. Open een nieuw bestand in Thonny, sla op als `servos`, in je persoonlijke map en schrijf de volgende code:

```
1 from machine import Pin,PWM
2
3 class ESP32Servo():
4
5     def __init__(self,pin,pmin=1,pmax=2,freq=50):
6
7         self._pmin = pmin
8         self._pmax = pmax
9         self._freq = freq
10        self._pin = pin
11        self._pwm = None
12
```

```

13     def start(self):
14
15         try:
16             self._pwm = PWM(Pin(self._pin))
17             self._pwm.freq(self._freq)
18             self.move_to(1)
19         except Exception as E:
20             print(f"PWM init error ({E})")
21
22     def move_to(self, angle):
23
24         try:
25             if angle < 0: angle = 0
26             if angle > 180: angle = 180
27
28             dt = 1.023*self._freq*(angle*(self._pmax - self._pmin)/180 + self._pmin)
29             print(dt)
30             self._pwm.duty(int(dt))
31         except Exception as E:
32             print(f"Error moving servo ({E})")
33
34     def stop(self):
35         try:
36             self._pwm.deinit()
37         except Exception as E:
38             print(f"PWM stop error ({E})")

```

### ***Wat uitleg:***

Wat je meteen zal opvallen is het sleutel (of keyword) `self`, hieronder wat uitleg met een bijhorend voorbeeld:

In Python betekent `self` simpelweg "deze specifieke instantie van de klasse". Het verwijst naar het object dat op dat moment is aangemaakt op basis van de klasse.

Stel je voor dat je een blauwdruk hebt voor het bouwen van auto's (dit is je klasse). Als je dan een specifieke auto bouwt op basis van die blauwdruk, is dat een object van die klasse. De variabele `self` verwijst naar die specifieke auto (het object), zodat je weet welke auto je aan het aanpassen bent of informatie van opvraagt.

```

class Auto:

    def __init__(self, kleur):

        self.kleur = kleur # self.kleur verwijst naar de kleur van dit specifieke object

    def toon_kleur(self):

        print(self.kleur) # self zorgt ervoor dat de methode de kleur van dit specifieke object gebruikt

```

### ***Nu gaan we eens de code overlopen:***

1. Met het sleutelwoord: `class` maken we de interpreter duidelijk dat hetgeen dat volgt een klasse is. Zoals hierboven reeds vermeld is een klasse een blauwdruk (zoals bijv. een plan voor een huis) waarop of waaruit een object gebaseerd is/komt (bijv. een huis). Alles wat onder deze definitie: `class ESP32Servo()`: staat en inspringt is deel van de klasse.

2. De eerste method: `__init__` is de constructor van de klasse. Als we de klasse in andere modules willen gebruiken, moeten we deze eerst importeren en dan oproepen een instance van maken door bijv. de volgende code te gebruiken:

**`motor = ESP32Servo(23,pmin=0.8,pmax=2.2,freq=60)`**

Dan wordt er een instance met de naam `motor` gemaakt. Aangezien de parameters `pmin`, `pmax` en `freq` zijn voor gedefinieerd in de `__init__` method/constructor, moeten we het bijhorende sleutelwoord vermelden als we deze willen wijzigen.

Je kan gerust een 2<sup>de</sup> instance of object maken van dezelfde klasse met volledig andere eigenschappen, bijv.

**`motor_links = ESP32Servo(22)`**

`pmin` zal dan 1 zijn, `pmax` 2 en `freq` 50, dus gelijk aan de voor gedefinieerde waarden.

3. In de `__init__` zien we bijv. `self._pmin = pmin`. De instance eigenschap: `_pmin` van de klasse wordt gelijkgesteld aan de waarde van de parameter: `pmin`. Dit kan voor ieder object die gemaakt wordt een andere waarde aannemen, deze members zijn object of instance gebonden. Het onderstreepje voor de eigenschap (kan ook voor een method) betekent dat deze members en/of methods privaat zijn, enkel te gebruiken door de klasse zelf (niet door de programmeur die de klasse gebruikt (dus instances of objecten van maakt)). We merken wel op dat dit een afspraak is en gemakkelijk kan omzeild worden. In andere programmeer talen zoals C++ en Java is een `private` member en/of method wel degelijk privaat en kan een gebruiker er geen gebruik van maken in zijn code.
4. De method start zorgt ervoor dat het Pin en PWM object worden gemaakt. Indien er iets fout gaat, wordt er een error getoond op het scherm en zal de servo motor zeker niet draaien. Een programmeur roept deze method als volgt op, als hij bijv. reeds de instance `motor` heeft gemaakt.

**`motor.start()`**

5. Je ziet dat bij het oproepen van de method van de klasse, `self` niet wordt gebruikt, `self` dient enkel voor intern gebruik en verwijst naar de instance: `motor` zelf zodat de juiste waarden worden opgehaald.
6. Met de method: `move_to` wordt de servo in de gewenste positie gebracht. Hier zie je ook dat de formule die in de inleiding werd vermeld, wordt gebruikt om de corresponderende duty cycle bij de gewenste hoek te berekenen. Als er iets fout loopt wordt er terug in except gegaan en zal de servo motor niet bewegen, bijv. als we de servo motor naar 135° willen brengen voor de instance: `motor` dan kan dit als volgt:

**`motor.move_to(135)`**

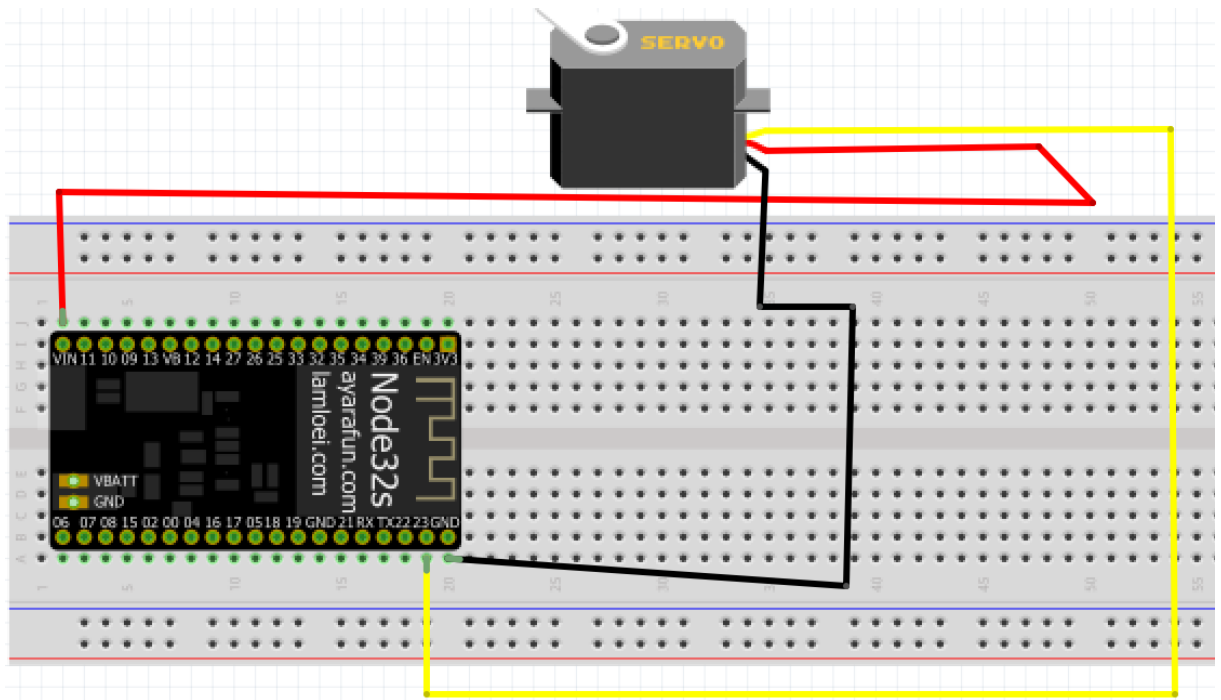
7. Met de method: `stop` wordt het “interne” PWM object vrijgegeven. De oproep in de code voor bijv. de instance: `motor` gaat als volgt:

**`motor.stop()`**

Opdracht: testen van de klasse: ESP32Servo.

Doel van deze opdracht is om de zopas gemaakte klasse te testen.

1. Maak de volgende schakeling:



Het kan gebeuren dat het bord niet genoeg vermogen heeft om de servo aan te sturen, dan zal je gebruik moeten maken van een externe bron. Vergeet dan niet de GND van het bord te koppelen met de GND van de batterij.

2. Zorg er eerst voor dat de klasse op het ESP bord komt te staan in de map: `iot_modules`.
3. Vervolgens maak je een nieuw bestand in Thonny en sla je dit op in je persoonlijke map als `testservo`.
4. Je gaat van start met het importeren van de klasse:

```
from iot_modules.servos import ESP32Servo
```

5. Schrijf vervolgens de volgende sequentie (kijk ook naar de uitleg op de vorige bladzijde):
  - a. Maak een instance: `motor` van de klasse `ESP32Servo` (gebruik eerst de voorgestelde waarden).
  - b. Roep de start method op.
  - c. Wacht 2 seconden.
  - d. Laat de motor naar  $90^\circ$  bewegen.
  - e. Wacht 2 seconden.
  - f. Laat de motor naar  $135^\circ$  bewegen.
  - g. Wacht 2 seconden.
  - h. Laat de motor naar  $180^\circ$  bewegen.
  - i. Wacht 2 seconden.
  - j. Laat de motor naar  $45^\circ$  bewegen.
  - k. Wacht 2 seconden.
  - l. Laat de motor naar  $0^\circ$  bewegen.

6. Kijk goed naar de bewegende motor en zie of deze inderdaad 90°, 180° ... bereikt. Indien niet pas in je code pmin en pmax aan.

De klep sturen via een Web Applicatie gestuurd door een webserver op de ESP32

Het doel van dit project is om via een webpagina de klep te sturen. Dus het plaatsen van de klep in de volgende toestanden:

- ✓ Gesloten
- ✓ Kwart open
- ✓ Half open
- ✓ 3 kwart open
- ✓ Volledig open

De webpagina wordt aangeboden door de ESP32 waarop de webserver draait. Deze webpagina kan er als volgt uitzien:

## Klepsturing

Gesloten
Open

- |  |
|--|
| <ul style="list-style-type: none"><li><input type="radio"/> Kwart open</li><li><input type="radio"/> Half open.</li><li><input type="radio"/> Driekwart open</li></ul> |
|--|

Status klep: Gesloten

Waar het linker kader een lijst is, het rechter kader een groep is van radio knoppen. Onderaan wordt de huidige status van de klep teruggegeven.

Voor het maken van de webserver gaan we ons baseren op de site:

<https://randomnerdtutorials.com/esp32-esp8266-micropython-web-server/> (Kan je ook terugvinden door in Google de zoektermen: ESP32 micropython webserver in te geven).

In dit voorbeeld werken ze met een knop om een LED aan en uit te doen, wij gaan werken met een lijst een een groep radioknoppen om de toestand van de servo te wijzigen. Hier wordt de volledige html pagina in de MicroPython code geplaatst, wij gaan in dit project meer algemeen tewerk gaan en een klasse structuur maken waarmee we de pagina, met behulp van Python methods, kunnen opbouwen.

Laten we eerst eens kijken naar het voorbeeld:

Het project gaat van start met het schrijven van de html pagina, die bij iedere request (aanvraag) van de client (de browser) naar de browser zal gestuurd worden.

```
def web_page():
    if led.value() == 1:
        gpio_state="ON"
    else:
        gpio_state="OFF"

    html = """<html><head> <title>ESP Web Server</title> <meta name="view"
    <link rel="icon" href="data:,"> <style>html{font-family: Helvetica;
    h1{color: #0F3376; padding: 2vh;}p{font-size: 1.5rem;}.button{display: inline-block;
    border-radius: 4px; color: white; padding: 16px 40px; text-decoration: none;
    .button2{background-color: #4286f4;}</style></head><body> <h1>ESP Web
    <p>GPIO state: <strong>"" + gpio_state + ""</strong></p><p><a href=
    <p><a href="/?led=off"><button class="button button2">OFF</button></p>
    return html
```

**Opmerking:** deze functie zullen we herschrijven met methods van de door ons nog te maken klassen.

Vervolgens zal de webserver worden opgestart en staan luisteren op de poort 80. Indien er een request (vraag ) binnenkomt, wordt deze geanalyseerd en een passende actie ondernomen.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    request = str(request)
    print('Content = %s' % request)
    led_on = request.find('/?led=on')
    led_off = request.find('/?led=off')
```

**Opmerking:** het verbinden van de ESP32 met wifi gebeurt hier in de boot.py file.

Hier wordt er gekeken of de vraag eindigt met /?led=on of /?led=off, bij ons zal het eerder zijn:

/?close, /?qrt\_open, /?half\_open, /?3qrt\_open of /?open

Vervolgens komt de actie op de vraag, hier in dit voorbeeld de LED aan of uit doen. Wordt de webpagina gemaakt en terug gestuurd naar de client , vooraf gegaan door een header die de browser vertelt wat er komt, nl. een tekst/html bestand.



```

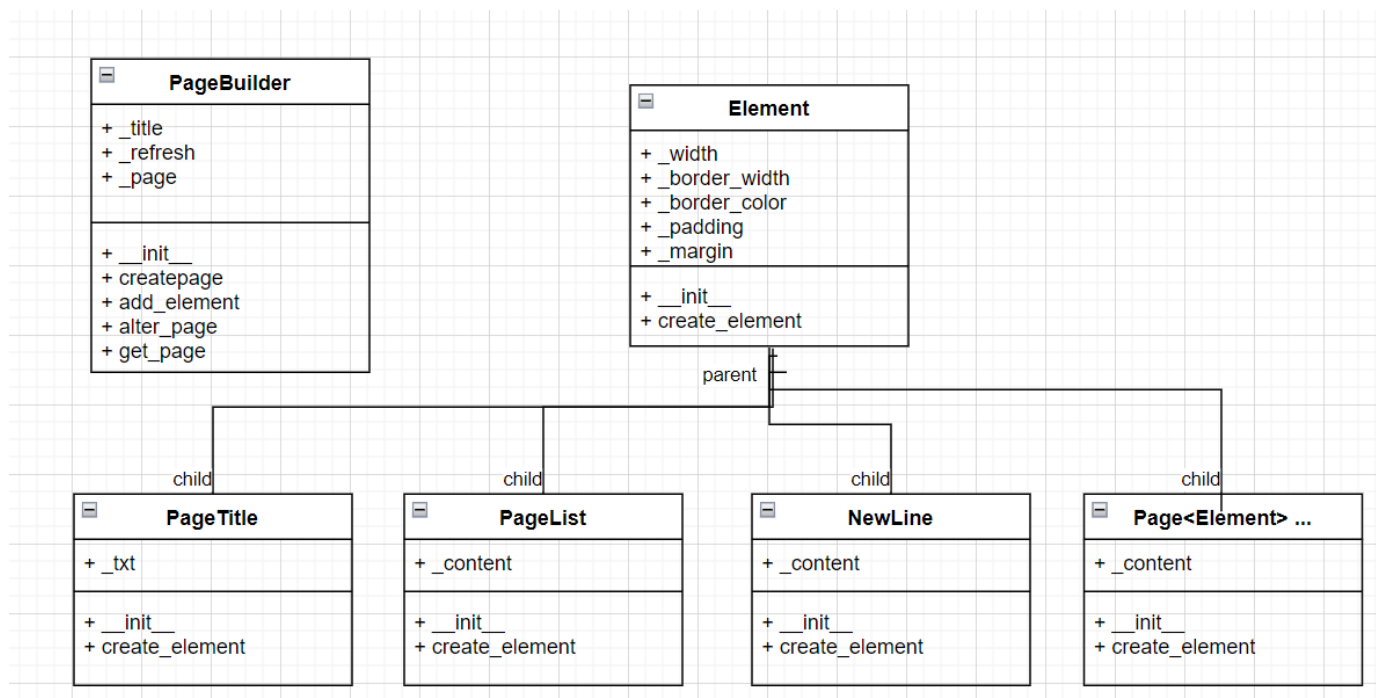
if led_on == 6:
    print('LED ON')
    led.value(1)
if led_off == 6:
    print('LED OFF')
    led.value(0)
response = web_page()
conn.send('HTTP/1.1 200 OK\n')
conn.send('Content-Type: text/html\n')
conn.send('Connection: close\n\n')
conn.sendall(response)
conn.close()

```

Het boven besproken stramien gaan we voor een groot stuk behouden, maar er zullen enkele wijzigingen worden aangebracht, nl:

- ✓ Wij zullen in de main.py de verbinding met wifi maken, met behulp van de SimpleWifi\_v2 module.
- ✓ Wij sturen geen LED maar een servo motor en zullen deze in verschillende standen kunnen zetten.
- ✓ De functie: web\_page zal geen zuivere html bevatten maar methods zoals create\_page, alter\_page, add\_element .... (Ik heb geopteerd om dit in een klassenstructuur te stoppen zodat er geen zuivere html verschijnt tussen de Python code en je “gemakkelijk” nieuwe elementen kan toevoegen aan de pagina. (Natuurlijk zal de html code in de klassen verstopt zitten).

Het grootste werk nu is om de klassenstructuur in elkaar te steken en te schrijven. (Een aantal elementen zijn al geschreven door de leerkracht).



## De klasse: PageBuilder

Is verantwoordelijk voor de opbouw van de pagina. Maakt gebruik van een bestand: template.txt (zie hieronder) die algemene html-code voor de opbouw van de pagina bevat.

Deze klasse heeft de volgende **eigenschappen**:

- ✓ `_title`: titel van de webpagina, deze wordt getoond in de titelbalk van de browser.
- ✓ `_refresh`: indien groter dan 0 wordt om de `_refresh` seconden de pagina opnieuw opgeladen (is nuttig als de pagina zichzelf moet refreshen, dus zonder interactie van de gebruiker, om resultaten van sensoren te tonen).
- ✓ `_page`: bevat de volledige html-code van de pagina.
- ✓ `_virgin_page`: zal de eerst (oorspronkelijke) opgebouwde pagina bevatten (is er achteraf bijgekomen om ervoor te zorgen dat de pagina 1x moet opgebouwd worden en dat vervolgens enkel de method: `alter_page` moet worden opgeroepen om interactie van de gebruiker op te vangen).

Deze klasse heeft de volgende **methods**:

- ✓ **De constructor: `__init__`**  
`__init__(self, title, refresh=0)`
- ✓ **`create_page`**: deze method leest het template bestand en wijzigt de titel door `{@TITLE@}` te vervangen. Kan eveneens de refresh invoeren door `<!-- -ADD REFRESH - -->` in de pagina te vervangen door de refresh html-code. (`<meta http-equiv="refresh" content="(time in seconds)">`)
- ✓ **`add_element`**: staat in om onderaan een pagina element, zoals: `PageTitle`, `PageList`, `NewLine` ... in te voeren. Dit wordt gedaan door de tekst (commentaar in html) : `<!-- - ADD ELEMENT - ->` te vervangen door het doorgegeven element.

- ✓ **alter\_page**: vervangt de inhoud/content van ieder pagina element gekoppeld aan een id met nieuwe inhoud en voegt checked toe aan de geselecteerde radioknop of checkbox(en).
- ✓ **get\_page**: retourneert de gemaakte pagina, om dan door te zenden naar de client.

Hieronder nog een print-out van de template.txt pagina (kan je ook terugvinden in classroom → drive onder data/webpagebuilder).

```

1  <html>
2  <head>
3  <!--ADD REFRESH -->
4  <title>{@TITLE@}</title>
5  <style>
6  a:link {
7      text-decoration: none;
8      color:#11AA22;
9  }
10 a:visited {
11     text-decoration: none;
12     color:#11AA22;
13 }
14 a:hover {
15     text-decoration: none;
16     color:#11AA22;
17 }
18 a:active {
19     text-decoration: none;
20     color:#11AA22;
21 }
22 </style>
23
24 <script>
25 function click_event(msg){
26     window.location.href="?/"+msg
27 }
28 </script>
29
30 </head>
31 <body>
32 <!-- ADD ELEMENT -->
33 </body>
34 </html>

```

Er is een klein stukje JavaScript code aanwezig om te kunnen “reageren” op een interactie van de gebruiker. Alle elementen die een interactie met de gebruiker verwachten moeten deze functie oproepen (dit in de href eigenschap of de onclick event, zie later).

Tussen de style tags (<style> .... </style>) staat voornamelijk de opmaak van de hyperlinks (a).

De commentaar: <! – ADD ELEMENT - - > wordt via de code vervangen door het gewenste element gevolgd door terug de commentaar: <! - - ADD ELEMENT - - >. Op deze manier is er steeds “plaats” om een nieuw element toe te voegen.

### De klasse: Element

Is de klasse die het “omhulsel” of de box van een element op de webpagina bepaalt. Alle elementen: titel, lijst, menu, ..... worden afgeleid van deze klasse.

Deze heeft de volgende **eigenschappen** of **properties**

- ✓ `_width`: breedte van het element procentueel ten opzichte van de volledige pagina breedte. (standaard: 25%)
- ✓ `_border_width`: breedte van de boord of rand van het element. (standaard geen boord = 0).
- ✓ `_border_color`: kleur van de boord of rand. (standaard geen kleur).
- ✓ `_margin`: afstand van het element ten opzichte van andere elementen. (standaard 2 pixels).
- ✓ `_padding`: afstand inhoud van het element ten opzichte van de boord van het element (standaard 2 pixels).

De **methods** van de klasse:

- ✓ **De constructor: `__init__`**  

```
def __init__(self,width=25,border_w=0,border_c="",margin=2,padding=2):
```
- ✓ **`create_element`**: maakt het element, meer bepaald een lege box met of zonder boord, een bepaalde breedte en een bepaalde marge ten opzichte van andere elementen. De inhoud van het element wordt gevuld met de tekst: {@CONTENT@} die later door het kind (= de klasse die afgeleid wordt van deze klasse) wordt ingevuld.  
Het element eindigt altijd met de commentaar: <! - - ADD ELEMENT - - >, op deze wijze wordt er plaats gemaakt om het volgende element toe te voegen.  
Standaard wordt ieder element links uitgelijnd. Dus het ene element schuift naast het andere element zolang er plaats is. (dit gebeurt met de CSS stijlregel: float: left).  
Er zal een kind of child worden gemaakt die een nieuwe lijn kan forceren door gebruik te maken van de CSS stijlregel: clear:left.

*Maken van de klassen: PageBuilder en Element.*

We gaan samen deze beide klassen voor een stuk bouwen en vervolgens de child klassen van Element bijvoegen (deze zijn reeds gemaakt door de leerkracht en bevinden zich in classroom → Link naar Drive/IoT3 → data /webpagebuilder→ PageElements.py).

Ook het bestand: template.txt bevindt zich in deze map op Drive.

We zullen van start gaan met de basis methods en dan verder aanpassen waar nodig afhankelijk van de klassen in het bestand: PageElements.py.

*Van start met de klasse: PageBuilder:*

1. Maak in je persoonlijke map de map: webpagebuilder.
2. Maak daarin het bestand: WebPageBuilder.py.

3. Je kan reeds de bestanden: template.txt en PageElements.py in deze map onderbrengen.
4. Maak vervolgens de klasse: PageBuilder:

```
class PageBuilder():
```

5. Maak daarin de volgende methods:

- a. De constructor: `__init__` (zie pagina: 26).

In deze constructor wordt niet zoveel gedaan, enkel de eigenschappen: `_title`, `_refresh`, `_page` en `_virgin_page` initialiseren. `_title` krijgt de waarde van de parameter `title`, dit is ook het geval voor `_refresh`, maar dan wel uiteraard de waarde van de parameter `refresh`.

`_page` en `_virgin_page` worden gelijkgesteld aan een lege string.

- b. De method `create_page` ziet er qua definitie als volgt uit:

```
def createpage(self):
```

Daarin gebeuren de volgende zaken:

- i. Het bestand `template.txt` wordt gelezen en gestopt in de eigenschap: `_page`. Indien dit niet lukt wordt er een foutmelding in de eigenschap: `_page` gestopt. Bijv.

```
template_page = """<html><head><title>PAGE NOT FOUND</title></head>
<body><h2>Error 404</h2><p>"""+str(E)+"""</p></body></html>"""
self._page = template_page
return False
```

- ii. Indien het niet lukt, wordt er `False` geretourneerd.
- iii. Vervolgens wordt er nagegaan of `_refresh` groter is dan 0, indien het geval wordt de html commentaar: `<!-- ADD REFRESH -->` vervangen door de html code voor refresh. Bijv.

```
template_page = template_page.replace("<!--ADD REFRESH-->",
"<meta http-equiv='refresh' content="+str(self._refresh)+">")
```

- iv. Tenslotte wordt `{@TITLE@}` vervangen door de inhoud van de eigenschap: `_title`.
- v. Op het einde wordt `True` geretourneerd om de gebruiker duidelijk te maken dat de template goed is geladen.
- c. De methods `add_element` en `alter_page` gaan we later maken, na het maken van de klasse: `Element` en na het bekijken van de child klassen van `Element`.
- d. De method: `get_page` is een “getter” om een private member (in python een eigenschap die voorop wordt gegaan met underscore (`_`) ) op te halen. In principe is het de bedoeling dat de programmeur die de code gebruikt de private members of eigenschappen enkel kan raadplegen en niet wijzigen (of er moet ook een “setter” bestaan). De “getter” ziet er als volgt uit:

```
def get_page(self):
    return self._page
```

6. Voor nu is de klasse: `PageBuilder` klaar.
7. Je kan deze klasse reeds in de gewone Python omgeving testen, dit kan als volgt:

- a. Wijzig naar Python interpreter (Tools/gereedschap → Options/Options → Interpreter).
- b. Maak in de map: webpagebuilder een bestand testbuilder.py.
- c. Voeg de volgende code in:

```

1 from WebPageBuilder import PageBuilder
2
3 page = PageBuilder("TEST DE BUILDER")
4 page.createpage()
5 print(page.get_page())

```

- i. Dus de klasse: PageBuilder wordt uit de module: WebPageBuilder geplukt.
  - ii. Vervolgens wordt de constructor opgeroepen, met de titel: TEST DE BUILDER (deze zal verschijnen in de titelbalk van de browser).
  - iii. Vervolgens wordt de method: createpage opgeroepen die voornamelijk het bestand: template.txt binnenhaalt.
  - iv. Met de method: getpage wordt de inhoud van de eigenschap: \_page opgehaald en hier geprint.
- d. Als je de html-code in de shell kopieert en plakt in een nieuw kladblok of wordpad en vervolgens opslaat als bijv. test.html (let op zet type op alles).
- e. Je kan deze test.html opslaan in de map: webpagebuilder.
- f. Ga vervolgens met de verkenner naar dit bestand, klik dubbel op dit bestand, het bestand zal geopend worden in een browser. Normaliter is dit een lege webpagina met in de titelbalk de tekst: TEST DE BUILDER.

#### Maken van de klasse: Element

1. Maak in het bestand WebPageBuilder de klasse Element, met de expressie:

```
class Element():
```

2. De eerste method is de constructor (zie ook pagina 28). In de constructor worden de volgende eigenschappen geïnitieerd:
  - a. `_width` wordt gelijkgesteld aan de waarde van de parameter `width`.
  - b. `_border_width` wordt gelijk gesteld aan de parameter `border_w`.
  - c. `_border_color` wordt gelijk gesteld aan de parameter `border_c`.
  - d. `_margin` wordt gelijk gesteld aan de waarde van de parameter `margin`.
  - e. `_padding` wordt gelijk gesteld aan de waarde van de parameter `padding`.

3. De method: `create_element`:

In deze method wordt aan de hand van Python de volgende html code geschreven:

```
<div style="width:xy%;border: wpx #XXXXXX;margin: 2px; padding:
2px;float:left">{@CONTENT@}</div><!-- ADD ELEMENT -->
```

met xy een getal tussen 0 en 100, w een getal voor de breedte van de boord rond het element, #XXXXXX een kleur in hexadecimaal formaat.

Tussen de div-tag (<div>... </div>) staat de inhoud: {@CONTENT@} die later een echte

inhoud zal krijgen via de child. Aan ieder element wordt de commentaar: <!-- ADD ELEMENT -->.die een placeholder is voor een volgend element.

Tenslotte wordt de string geretourneerd.

Het bestand: PageElements

We treffen de volgende klassen aan die afgeleid zijn van de klasse: Element, vandaar naam klasse(Element), ook wordt bovenaan een import gedaan van Element.

```
3 class PageTitle(Element):
4
5     def __init__(self, content, width=100, border_w=0, border_c="", margin=2, padding=2):
6
7         super().__init__(width=width, border_w=border_w, border_c=border_c, margin=margin, padding=padding)
8         self._txt = content
9
10    def create_element(self):
11
12        box = super().create_element()
13        box = box.replace("{@CONTENT@}", "<h2 id='__TITLE__'>" + self._txt + "</h2>")
14        return box
```

De klasse: PageTitle voegt een titel van het formaat: kop2 toe aan de pagina. We zien in de constructor dat eerst de constructor van de parent (klasse: Element) wordt opgeroepen en daarna de eigenschap \_txt wordt gelijkgesteld aan de waarde van de parameter: content.

Met de method: create\_element wordt het element gemaakt. Eerst wordt de method: create\_element van de parent opgeroepen, met behulp van de functie: super() wordt er verwezen naar de parent.

In de 2<sup>de</sup> lijn wordt {@CONTENT@} vervangen door de gewenste inhoud. Tenslotte wordt de html-code voor het element geretourneerd.

Alle volgende klassen zijn op dezelfde wijze opgebouwd, behalve de klasse NewLine. Hieronder de klassen:

```
16 class PageTxt(Element):
17
18     def __init__(self, content, width=100, border_w=0, border_c="", margin=2, padding=2):
19
20         super().__init__(width=width, border_w=border_w, border_c=border_c, margin=margin, padding=padding)
21         self._txt = content
22
23     def create_element(self):
24
25        box = super().create_element()
26        box = box.replace("{@CONTENT@}", "<span>" + self._txt + "</span>")
27        return box
28
29
30 class PageList(Element):
31
32     def __init__(self, content, width=100, border_w=0, border_c="", margin=2, padding=2):
33         super().__init__(width=width, border_w=border_w, border_c=border_c, margin=margin, padding=padding)
34         self._content = content
35
36     def create_element(self):
37         box = super().create_element()
38         vlist = ""
39         for k, v in self._content.items():
40             vlist += "<a href='\"javascript:click_event(\"'+k+'\"')\">\"'+v+\"</a><br>"
41         box = box.replace("{@CONTENT@}", vlist)
42         return box
```

```

44 class PageRadioGroup(Element):
45
46     def __init__(self, content, name, width=100, border_w=0, border_c="", margin=2, padding=2):
47         super().__init__(width=width, border_w=border_w, border_c=border_c, margin=margin, padding=padding)
48         self._content = content
49         self._name = name
50
51     def create_element(self):
52         box = super().create_element()
53         radiogroup = ""
54         for k,v in self._content.items():
55             send_grp = self._name.replace(" ", "%20")
56             input_item = "<input type='radio' id='"+k+"' name='"+self._name+"' onclick='\"javascript:click_event(\"'+k+'&"+self._name+"="+k+"')\">"+v+"</input>"
57             radiogroup+="<a href='\"javascript:click_event(\"'+k+'&"+self._name+"="+k+"')\">"+input_item+"</a><br>"
58         box = box.replace("{@CONTENT@}", radiogroup)
59         return box

```

```

70 class PageDataBox(Element):
71
72     def __init__(self, box_id, box_val, width=100, border_w=0, border_c="", margin=2, padding=2):
73
74         super().__init__(width=width, border_w=border_w, border_c=border_c, margin=margin, padding=padding)
75         self._id = box_id
76         self._value = box_val
77
78     def create_element(self):
79
80         box = super().create_element()
81         box = box.replace("{@CONTENT@}", "<span id='"+self._id+"'>"+self._value+"</span>")
82         return box

```

```

61 class NewLine(Element):
62
63     def __init__(self):
64         self._content = "<br style='clear:left'>"
65
66     def create_element(self):
67         return self._content+"<!-- ADD ELEMENT -->"
68

```

Alle klassen die interactie van de gebruiker verwachten verwijzen bij de onclick of href naar de JavaScript functie: `click_event`. Als parameter wordt de verwijzing naar het element doorgegeven die dan met behulp van de JavaScript functie geplakt wordt aan het einde van de URL om deze dan door te geven aan de webserver op de ESP32.

We merken nog op dat bij de klasse: `PageRadioGroup` de parameter wordt uitgebreid met `&naam` van de radiogroep = id van de checkbox. Op deze manier maken we de server duidelijk dat er aan dit element checked moet worden toegevoegd zodat bij het opnieuw laden van de pagina de radioknop wordt gekleurd.

In de klasse `NewLine` wordt de constructor en `create_element` van de parent niet opgeroepen.

Met de klasse: `PageDataBox` kan men waarden van de ESP32 doorgeven aan de webpagina. De constructor van deze klasse verwacht een id (om het element te localiseren) en de een waarde.

De klasse: `BuildWebPage` verder vervolledigen

De **method: `add_element`** verwacht als parameter een `Element`.

In deze method wordt dan de `Element` method: `create_element` opgeroepen en vervolgens wordt de commentaar: `<!-- ADD ELEMENT -->` vervangen door de inhoud van het element.

De **method: `alter_page`** verwacht 2 parameters:

- ✓ Een dictionary met sleutel waarde paren voor de `PageDataBox` objecten.
- ✓ Een dictionary met sleutel waarde paren voor de `PageRadioGroup` objecten. De waarde hier is checked.



In deze method wordt er eerst gecheckt of het member: `_virgin_page` reeds is ingevuld, indien niet wordt deze gelijkgesteld aan het member: `_page`. Anders wordt `_page` gelijkgesteld aan `_virgin_page`.

Vervolgens worden de dictionaries overlopen om de waarden van `PageDataBox` objecten aan te passen en/of een element in het `PageRadioGroup` te “checken”.

Het test programma verder uitbreiden.

1. Doe een import van de module: `PageElements`.

```
import PageElements as pgs
```

2. Schrijf dan de volgende code:

```
pgt = pgs.PageTitle("Dit is een test",border_w=2,border_c="#33CCFF",padding=4,margin=20)
page.add_element(pgt)
page.add_element(pgs.NewLine())
ls_content = {"valve_close":"sluit klep","valve_open":"open klep"}
pgl = pgs.PageList(ls_content,width=45,border_w=2,border_c="#22FF22",padding=4,margin=10)
page.add_element(pgl)
rd_content = {"valve_qrt_open":"klep kwart open","valve_half_open":"klep half open"}
pgrd = pgs.PageRadioGroup(rd_content,"valve",width=45,border_w=2,border_c="#22DD11",padding=4,margin=4)
page.add_element(pgrd)
page.add_element(pgs.NewLine())
page.add_element(pgs.PageTxt("Status klep:",width=8))
page.add_element(pgs.PageDataBox("__VALVE__STATE","valve is open",width=5))
page.alter_page({"__TITLE__":"Test op wijziging pagina","__VALVE__STATE":"test"},{"valve_qrt_open":"checked"})
print(page.get_page())
```

3. Eerst maken we een titel die vervolgens wordt aan de pagina.
4. Hetzelfde geldt voor een newline, een lijst, een radiogroep (hier wordt als parameter een dictionary met sleutel waarde paren. Er worden ook nog een tekst en een databox toegevoegd.
5. Met `page.alter_page` worden als test een titel en databox gewijzigd en één radioknop aangevinkt.
6. Tenslotte wordt er terug een print van de gewijzigde pagina uitgevoerd.
7. De tekst in de shell kan je terug plakken in het bestand: `test.html`. Het resultaat:

Test op wijziging pagina

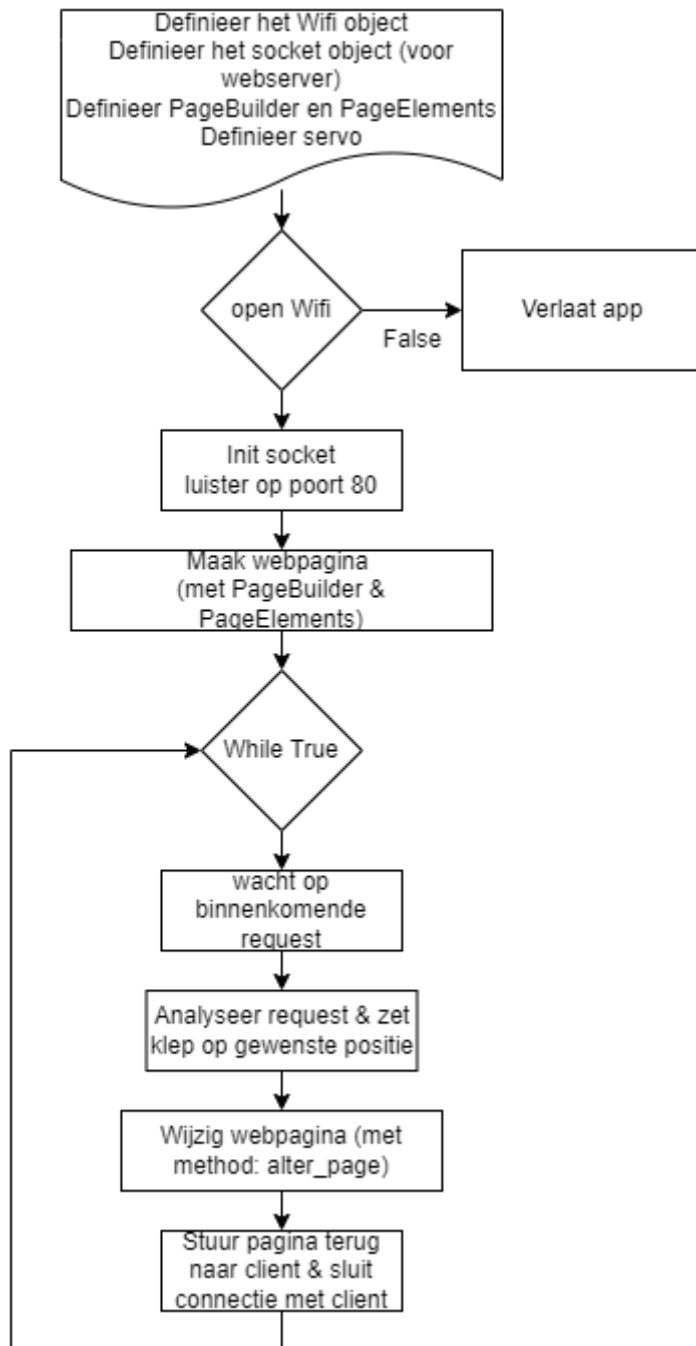
sluit klep  
open klep

Status klep: test

☒ klep kwart open  
☐ klep half open

### De ESP32 WebServer voor aansturen klep opbouwen

1. Maak op het bord een map: webpagebuilder en zorg dat de bestanden: WebPageBuilder.py, PageElements.py en template.txt in deze map komen te staan. (Uploaden).
2. Vertaal de onderstaande flowcharts naar MicroPython code voor de ESP32. Kijk ook naar het voorbeeld op de site: <https://randomnerdtutorials.com/esp32-esp8266-micropython-web-server/>



Voor het blokje: Maak webpagina zal men best een functie maken en deze dan juist na het maken van de socket oproepen. We kunnen ongeveer de code gebruiken die we in de test hebben gemaakt maar we moeten er wel voor zorgen dat alle posities van de klep kunnen worden gekozen, dus wijzig de code zodat de pagina de volgende vorm krijgt:

## Klepsturing

Gesloten  
Open

o Kwart open  
o Half open.  
o Driekwart open

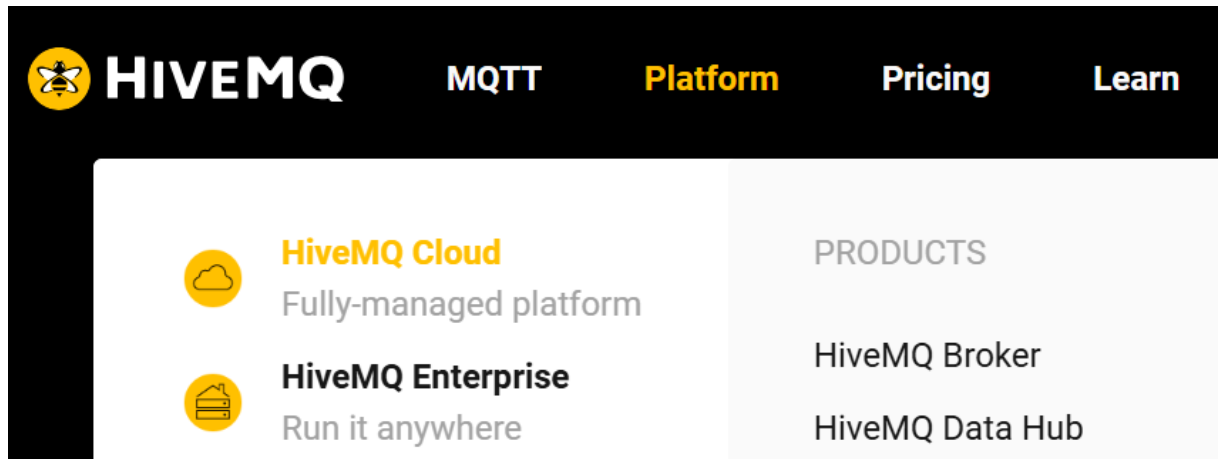
Status klep: Gesloten

3. Eenmaal het programma is gemaakt, test het uit door een browser op de PC te openen en in de adresbalk het IP-adres van de ESP32 in te geven in de adresbalk van de browser. Het bovenstaande scherm zou moeten verschijnen en de gebruiker zou via deze weg de klep (servo) moeten kunnen sturen.

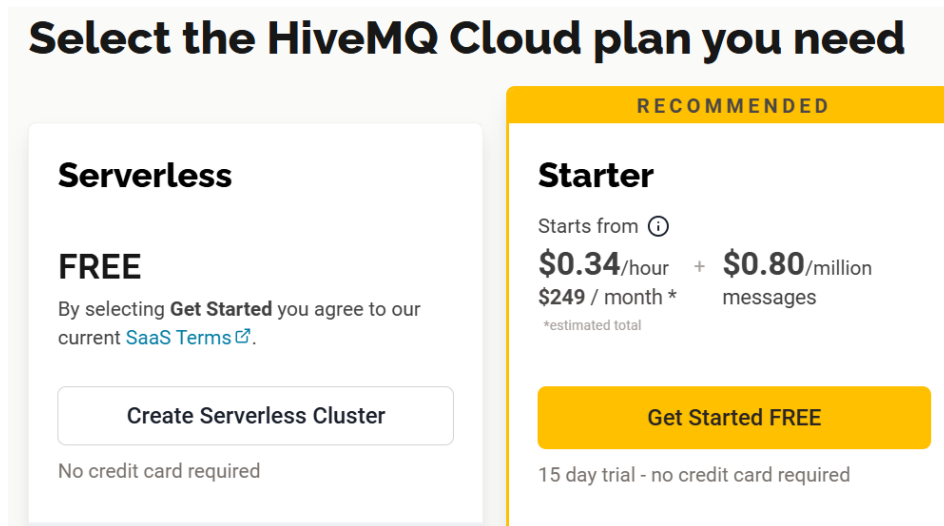
## Een eigen HiveMQ Broker in de Cloud maken en verbinden met ESP32/node-red

### Maken/registreren van een Broker in de HiveMQ Cloud

1. Ga naar de site: <https://www.hivemq.com/>.
2. Tik vervolgens op het menu: Platform en kies dan voor HiveMQ Cloud.



3. Druk op de knop: Sign up free indien je nog geen account hebt (indien je een account hebt, kan je tikken op de knop: Login).
4. Vul de nodige gegevens in. Je kan je ook registreren met een Google Account (bijv. je school account).
5. Tik vervolgens op een nieuwe Cluster maken. Kies voor het linker paneel: Create Serverless Cluster.



6. Tik vervolgens op de knop: Manage Cluster.
7. Tik bovenaan op de knop: ACCESS MANAGEMENT.
8. Tik op New Credentials en geef een gebruikersnaam en wachtwoord in. (gebruikersnaam: cvo\_voornaam\_2lettersachternaam, wachtwoord: H3tCv0.be).
9. Zorg dat deze gebruiker zowel publish als subscribe rechten krijgt.
10. Druk op de knop Save, de eigen Cloud Broker is klaar.

## Een random getal publiceren met de ESP32 via de Cloud Broker

1. Maak een nieuw MicroPython bestand: testCloudHiveMQ.py en sla op in je persoonlijke map.
2. Eerst moeten we ervoor zorgen dat er verbinding wordt gemaakt met wifi netwerk. Kijk naar vorige projecten om dit te bereiken.

We gaan de ESP32 laten connecteren met de “eigen” Broker in de Cloud en om de 2 seconden een willekeurig getal zenden naar de Broker. We zullen daarvoor nog enkele bibliotheken moeten importeren, nl.

```
2 import time
3 import random
4 from umqtt.robust import MQTTClient
5 import sys
6 import ssl
```

**random** om een willekeurig getal te genereren.

**time** om een sleep in te voeren.

**umqtt.robust** om aan MQTT te kunnen doen.

**ssl**: nodig voor de communicatie met de broker. Hieronder wat uitleg in verband met SSL (volgens ChatGPT):

SSL staat voor *Secure Sockets Layer* en is een beveiligingsprotocol dat wordt gebruikt om gegevens die via internet worden verzonden te versleutelen. Het zorgt ervoor dat gevoelige informatie, zoals wachtwoorden, creditcardgegevens en persoonlijke informatie, veilig wordt verstuurd tussen een webbrowser en een server, zodat deze niet onderschept of gewijzigd kan worden door kwaadwillende.

Hier is hoe SSL werkt in een notendop:

- **Versleuteling:** SSL versleutelt de gegevensstroom, zodat alleen de bedoelde ontvanger deze kan ontcijferen.
- **Authenticatie:** SSL zorgt ervoor dat de gebruiker communiceert met de juiste server en niet met een imitatie, wat helpt om phishing-aanvallen te voorkomen.
- **Gegevensintegriteit:** SSL detecteert veranderingen in de gegevens tijdens verzending, waardoor man-in-the-middle-aanvallen (waarbij iemand de communicatie onderschept) veel moeilijker worden.

Sinds 2015 wordt SSL officieel opgevolgd door TLS (Transport Layer Security), een verbeterde versie van hetzelfde protocol. Je herkent een website die SSL/TLS gebruikt aan het slotje naast het websiteadres in de browser en aan de "https://" in de URL.

3. Het maken van het MQTT object en connecteren met de broker is wat uitgebreider. Om het ons wat gemakkelijker te maken zullen we dit onderdeel in de functie: mqtt\_connect() onderbrengen:

```

10 def mqtt_connect():
11     context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
12     context.verify_mode = ssl.CERT_NONE
13     mqtt_cl = MQTTClient(client_id="ESP32_filipds_080270".encode(),
14     server="956356a356de4e5ea9ef8d31fb517d00.s1.eu.hivemq.cloud".encode(),
15     port=0,
16     user="cvofilipds".encode(),
17     password="H3tCv0.be".encode(),
18     keepalive=7200,
19     ssl=context
20     )
21
22     mqtt_cl.connect()
23     return mqtt_cl

```

### **Wat uitleg:**

- ✓ De eerste 2 lijnen bepalen de beveiliging van de communicatie tussen ESP32 en Broker. In het kort: alles qua beveiliging wordt overgelaten aan de broker.
  - ✓ Vervolgens wordt het MQTT Client object gemaakt, dit maal worden er meer parameter verwacht, nl:
  - ✓ Naast de id en server, wiens naam je kan terugvinden in je HiveMQ cloud configuratiescherm.
  - ✓ Nu moet je ook een port ingeven, 0 is hier default en zal dan ook de default poort voor ssl communicatie voorstellen.
  - ✓ De inloggegevens moeten ook doorgegeven worden via user en password.
  - ✓ Keepalive zorgt ervoor dat de verbinding x seconden blijft bestaan, ook als er geen boodschappen worden gezonden.
  - ✓ En ssl wordt gelijkgesteld aan hetgeen er op de 2 eerste lijnen van de functie is gedefinieerd.
  - ✓ Vervolgens wordt er een connect uitgevoerd met mqtt\_cl.connect() en wordt het MQTT Client object geretourneerd.
4. Vervolgens ga je deze functie, na de connectie met wifi, oproepen en om de 2 seconden een willekeurig getal sturen naar de Broker. Dit kan met de volgende code:

```

34 mqtt_client = None
35 try:
36     mqtt_client = mqtt_connect()
37 except Exception as E:
38     print("probleem connectie MQTT Cloud Broker:",E)
39     sys.exit()
40
41 while True:
42     rnd = random.randint(0,100)
43     mqtt_client.publish(TOPIC,str(rnd).encode())
44     print(rnd,"published")
45     time.sleep(2)

```

Opmerking: de gebruikte TOPIC hier is: cvo/randomgen.

## Ontvangen van het random gegenereerde getal door node-red

1. Start node-red.
2. Surf met de browser (chrome, firefox ...) naar localhost:1880.
3. Sleep een mqtt input node op het werkveld en stel dit als volgt in door te tikken op potlood knop naast server.

The screenshot shows the 'Edit mqtt in node > Edit mqtt-broker node' configuration window. At the top are 'Delete', 'Cancel', and 'Update' buttons. Below is a 'Properties' section with a gear icon. The 'Name' field is empty. There are three tabs: 'Connection' (selected), 'Security', and 'Messages'. Under 'Connection', the 'Server' field contains a long alphanumeric string, and the 'Port' is '8883'. There are checkboxes for 'Connect automatically' and 'Use TLS', both of which are checked. Next to 'Use TLS' is a dropdown menu labeled 'Add new tls-config...' and a pencil icon. The 'Protocol' dropdown is set to 'MQTT V5'. The 'Client ID' field is empty with the placeholder text 'Leave blank for auto generated'. The 'Keep Alive' field is set to '7200'. At the bottom, there is a 'Session' section with a checked 'Use clean start' checkbox. At the very bottom, there is a status bar showing 'Enabled', '5 nodes use this config', and 'On all flows'.

- ✓ De Server is de hostname dat je kan vinden in de HiveMQ Cloud Broker configuratiescherm.
  - ✓ Zorg dat Use TLS is aangevinkt.
  - ✓ Zet Keep Alive op 7200.
4. Tik vervolgens op Security en geef daar de gebruikersnaam en wachtwoord in.
  5. Druk dan op de knop Update (rechtsboven) en geef de topic: cvo/randomgen in.
  6. Sleep een debug node op het veld en verbind deze met de mqtt input node.
  7. Test uit door de flow te deploy'en en het MicroPython script te starten in Thonny.
  8. Na een tijdje zouden de random getallen moeten verschijnen in de debug venster van node-red.

## Downloaden en installeren van de nodige software om een operationele Home Assistant te bekomen

### Wat is Home Assistant

Home Assistant is een open-source platform voor thuisautomatisering dat ontworpen is om verschillende slimme apparaten en diensten te integreren en te beheren vanuit één centrale interface. Hier zijn de kernpunten over Home Assistant:

1. Open-Source: Het is gratis en ontwikkeld door een community van vrijwilligers. Dit betekent dat het continu verbeterd en uitgebreid wordt.
2. Flexibiliteit en Aanpassing: Home Assistant kan op een breed scala aan hardware draaien, van een Raspberry Pi tot een dedicated server. Gebruikers kunnen hun eigen automatisaties en scripts maken om apparaten en diensten op specifieke manieren te laten samenwerken.
3. Breed Apparatenbereik: Het ondersteunt duizenden apparaten en diensten, zoals verlichting, thermostaten, beveiligingssystemen, mediaplayers en meer. Integraties variëren van populaire merken als Philips Hue en Nest tot minder bekende systemen.
4. Lokale Controle: In tegenstelling tot veel andere commerciële oplossingen, kan Home Assistant lokaal draaien zonder afhankelijkheid van cloudservices, wat zorgt voor meer privacy en betrouwbaarheid.
5. Gebruiksvriendelijke Interface: Het biedt een webinterface en mobiele apps waarmee gebruikers hun slimme thuisomgeving kunnen beheren en bedienen.
6. Automatisering en Scènes: Gebruikers kunnen complexe automatiseringen instellen die reageren op verschillende triggers zoals tijd, locatie, sensorwaarden, enz. Daarnaast kunnen 'scènes' worden geconfigureerd om meerdere apparaten tegelijk te bedienen.

Door deze eigenschappen is Home Assistant een krachtige oplossing voor iedereen die volledige controle wil hebben over hun slimme thuisomgeving en graag zelf wil aanpassen en experimenteren.

Zoals in punt 2 is vermeld, kan Home Assistant draaien op verschillende platformen. Een “Home Assistant” platform, Raspberry Pi, Odroid ... maar ook op een Windows PC in een virtuele omgeving of als een Docker Container. Hier gaan we voor een Windows PC met als virtuele omgeving: VirtualBox. (Meer uitleg over de mogelijke platformen: <https://www.home-assistant.io/installation/>).

### Wat is VirtualBox?

VirtualBox is een open-source virtualisatiesoftware ontwikkeld door Oracle. Het stelt gebruikers in staat om meerdere besturingssystemen gelijktijdig op één fysieke machine te draaien. Hier zijn de belangrijkste kenmerken van VirtualBox:

1. Platformonafhankelijk: VirtualBox draait op verschillende host-besturingssystemen, zoals Windows, macOS, Linux en Solaris, en ondersteunt een breed scala aan gast-besturingssystemen, waaronder verschillende versies van Windows, Linux, BSD en andere.
2. Gebruiksvriendelijk: Het biedt een intuïtieve grafische gebruikersinterface (GUI) die het eenvoudig maakt om virtuele machines (VM's) te creëren en te beheren. Dit maakt het toegankelijk voor zowel beginners als gevorderde gebruikers.
3. Kosten: VirtualBox is gratis en open-source, wat betekent dat het geen licentiekosten heeft en de broncode beschikbaar is voor inspectie en aanpassing.
4. Snapshots: Gebruikers kunnen snapshots maken van hun VM's, waardoor ze de status van een VM op een bepaald moment kunnen vastleggen en later naar dat punt kunnen terugkeren als dat nodig is.



5. Ondersteuning voor Virtuele Schijven: VirtualBox ondersteunt verschillende soorten virtuele schijfformaten, zoals VDI (VirtualBox Disk Image), VMDK (VMware), VHD (Microsoft), en HDD (Parallels).
6. Gast Toevoegingen: Het biedt "Guest Additions" die geïnstalleerd kunnen worden op gast-besturingssystemen om prestaties te verbeteren en extra functionaliteiten te bieden, zoals gedeelde mappen, verbeterde grafische ondersteuning en naadloze muisintegratie.
7. Netwerkfunctionaliteit: VirtualBox ondersteunt verschillende netwerkmoden, waaronder NAT, bridged networking, en host-only networking, waardoor flexibele netwerktopologieën kunnen worden geconfigureerd voor VM's.
8. Extensibility: Het heeft een modulair ontwerp met een uitgebreide API en biedt ondersteuning voor scripting via verschillende programmeertalen, waardoor het gemakkelijk te integreren en uit te breiden is met andere tools en toepassingen.

VirtualBox is een populaire keuze voor zowel individuele gebruikers als organisaties vanwege zijn veelzijdigheid, gebruiksgemak en de sterke community-ondersteuning. Het wordt vaak gebruikt voor ontwikkelings-, test- en onderwijsdoeleinden.

## Downloaden en installeren van VirtualBox

1. Ga naar de site: <https://www.virtualbox.org/>



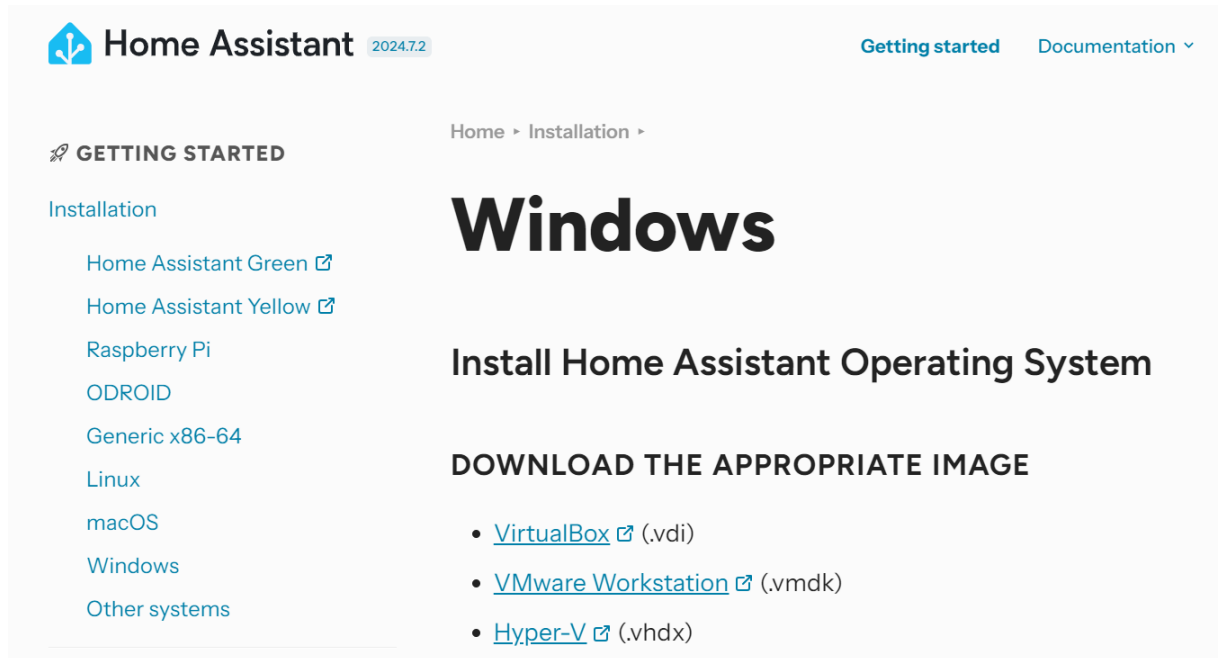
2. Tik op Downloads en tik vervolgens op Windows hosts.
3. Ga naar de map: Downloads en open het installatie pakket.
4. Volg de wizard.
5. Open vervolgens VirtualBox.



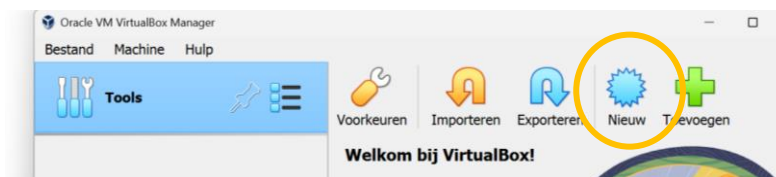
6. VirtualBox is klaar om virtuele machines te installeren.

## Downloaden van Home Assistant image voor VirtualBox en image toevoegen aan VirtualBox

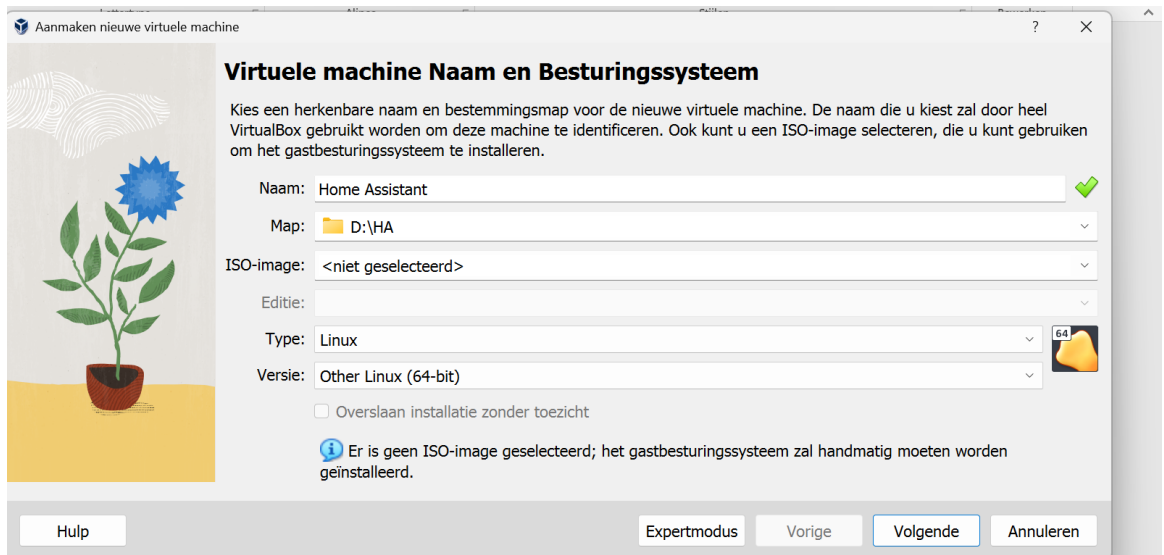
1. Voor het downloaden van de Home Assistant image voor VirtualBox ga naar de volgende pagina op het internet: <https://www.home-assistant.io/installation/windows/>.



2. Tik daar op de link: VirtualBox (.vdi).
3. Ga vervolgens op de PC naar de map: Downloads en pak het zopas gedownloade bestand uit.
4. Maak in je persoonlijke map een map: HA.
5. Verplaats het bestand: haos\_ova\_....vdi naar de map: HA.
6. Open vervolgens VirtualBox.
7. Tik op de knop: Nieuw.

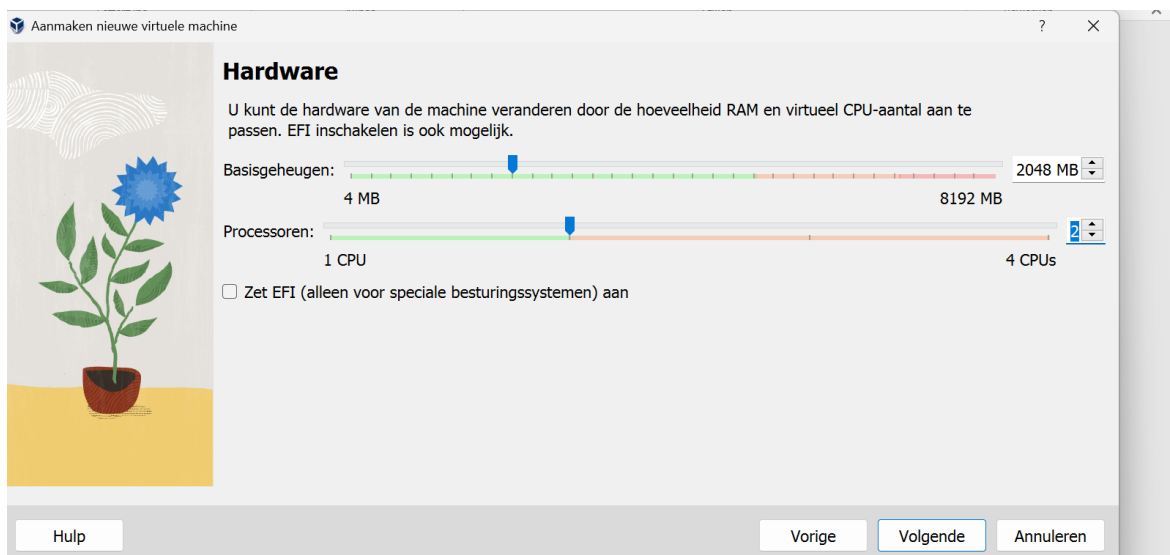


8. Vul de volgende waarden in:



Een naam, de map waarin de VirtualBox Home Assistant staat en type besturingssysteem: Linux met als versie: Other Linux (64-bit).

9. Druk op Volgende en kies in het volgende scherm voor 2GB RAM en 2 CPU's.

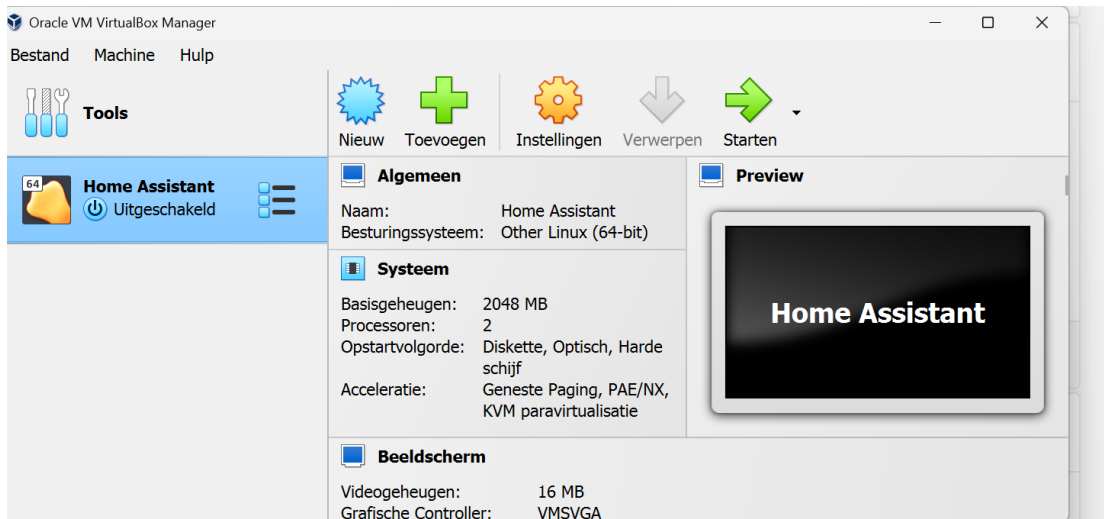


10. Druk op Volgende.
11. Tik in het volgende scherm op Bestaande virtuele harde schijf bestand gebruiken en duid het zopas gedownload vdi bestand aan in de map: HA.



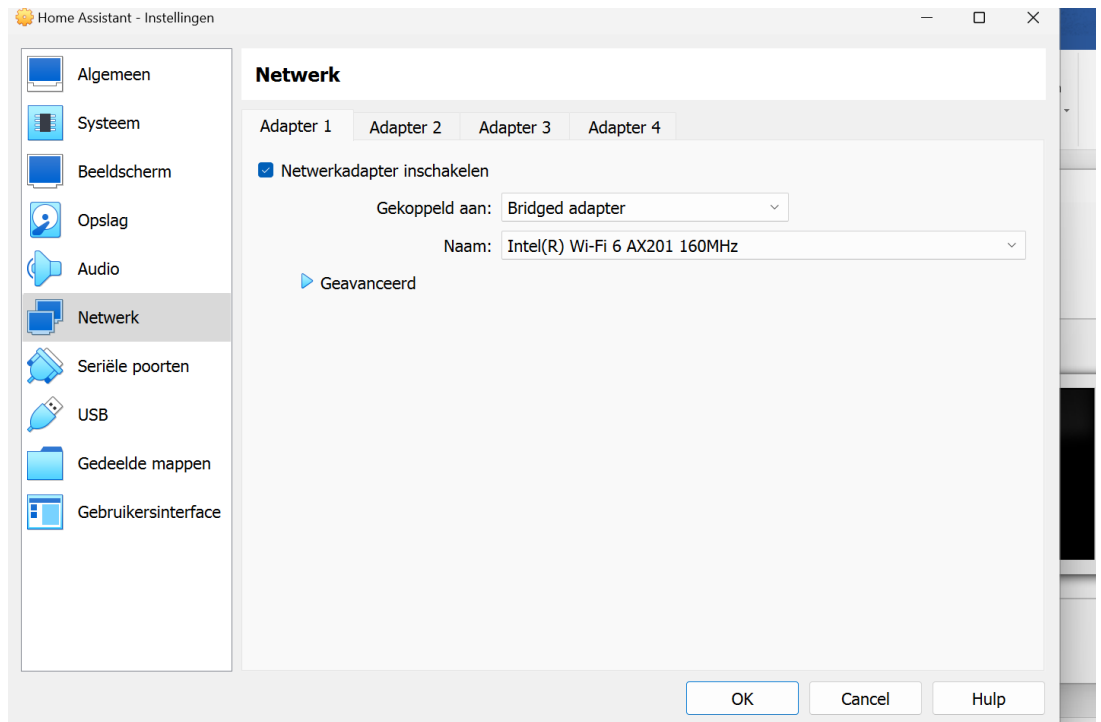
12. Tik terug op Volgende en tenslotte op de knop: Afmaken.

13. In het linker paneel zal de virtuele machine: Home Assistant verschijnen.



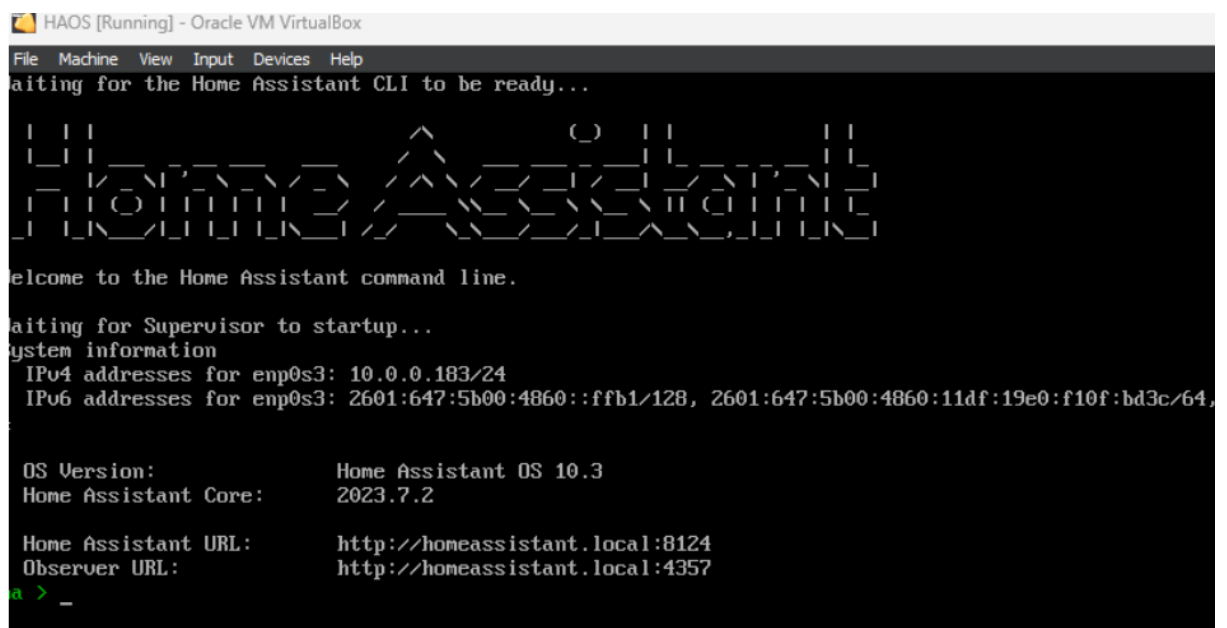
14. Vervolgens gaan we nog enkele settings aanpassen voor de Home Assistant. Tik op de knop: Instellingen.

- Ga naar "Systeem" en zet EFI (alleen voor speciale besturingssystemen). Aan.
- Ga naar Netwerk en koppel de virtuele machine aan de Bridged Adapter.

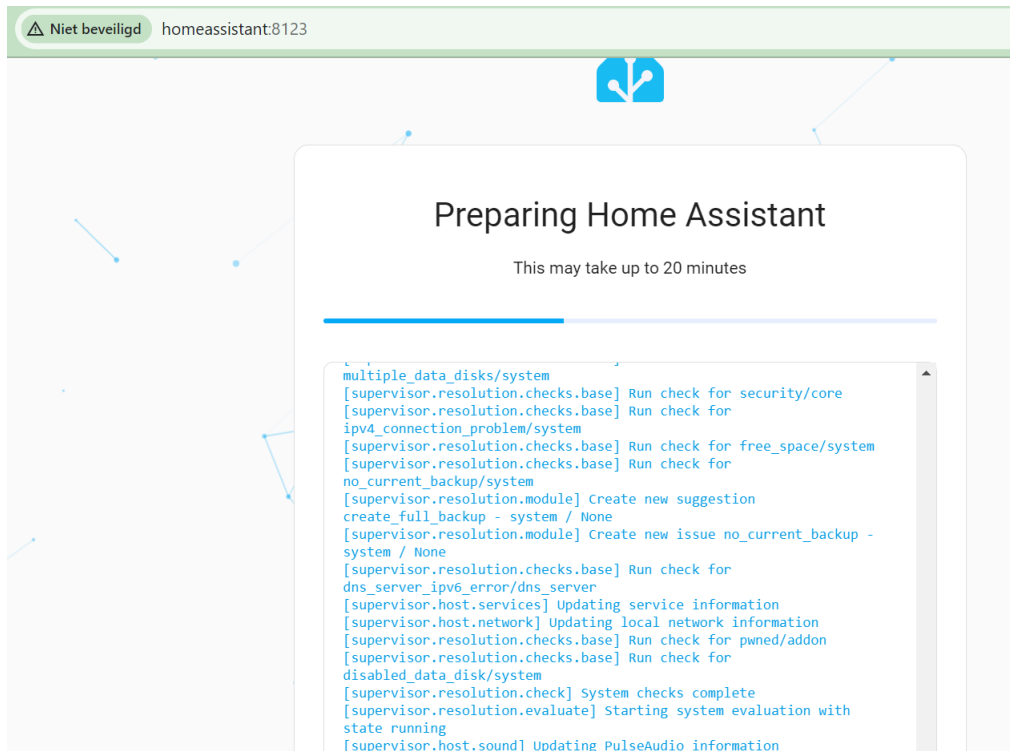


In principe wordt de fysieke netwerkkaart gedeeld met de Virtuele machine.

15. Druk op OK en start de virtuele machine door te tikken op de knop: Starten. Na een tijdje komt het volgende scherm op de voorgrond:

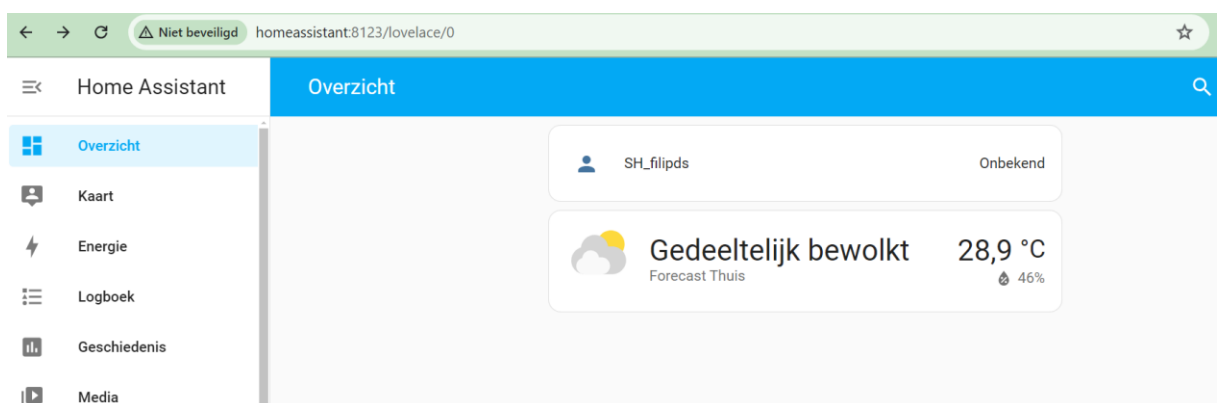


16. Open een browser en geef de volgende url in: <http://homeassistant:8123>. Het voorbereiden van Home Assistant gaat van start:



Dit kan wat tijd in beslag nemen.

17. Tik vervolgens op de knop: CREËER MIJN SMART HOME
18. In het volgende scherm moet je gebruikersnaam en wachtwoord verzinnen. Voor de les kiezen we:
  - a. Gebruikersnaam: HA\_voornaam\_2 letters achternaam
  - b. Wachtwoord: Hetcvo.be
19. Druk op de knop: Account aamaken.
20. Geef vervolgens de locatie in en tik op Volgende, tik daarna nogmaals op Volgende.
21. Het volgende scherm komt op de voorgrond:



TER INFO: opmerking in verband met een meer professionele virtuele omgeving:

#### Proxmox en waarom VirtualBox

Indien je een “oude” Intel of AMD PC met 8GB RAM en eventueel een SSD schijf op overschot hebt dan kan je op deze machine een Proxmox virtuele omgeving installeren. Daarop kan je dan verschillende “virtuele” besturingssystemen en Linux Containers toevoegen.

In principe zouden we de PC in de klas van een dualboot menu kunnen voorzien waar de gebruiker zou kunnen kiezen tussen het Windows OS en Proxmox OS maar dat zou wat complicaties met zich meebrengen, nl:

- ✓ Andere cursisten zouden in het opstart menu moeten kiezen voor Windows, wat voor veel mensen verwarrend is.
- ✓ Indien we Proxmox opstarten met daarin virtueel Home Assistant zouden we een andere PC moeten gebruiken om via de browser de Home Assistant te tonen, configureren ....

Een andere mogelijkheid is om Home Assistant als een container te laten draaien op een PC, maar dan kunnen er geen Add-ons worden geïnstalleerd.

#### Een korte uitleg over Proxmox

Proxmox is een open-source virtualisatieplatform dat ontworpen is voor het beheren van virtual machines (VM's) en containers. Hier zijn de belangrijkste kenmerken van Proxmox:

1. **Virtualisatie:** Proxmox ondersteunt zowel KVM (Kernel-based Virtual Machine) voor volledige virtualisatie als LXC (Linux Containers) voor lichtgewicht containerisatie, waardoor gebruikers meerdere gevirtualiseerde systemen kunnen draaien op één fysieke machine.
2. **Open-Source:** Het is gratis en open-source, wat betekent dat de broncode beschikbaar is voor inspectie en aanpassing door de community. Dit draagt bij aan transparantie en flexibiliteit.
3. **Webgebaseerde Beheerinterface:** Proxmox biedt een gebruiksvriendelijke webinterface waarmee gebruikers hun VM's en containers eenvoudig kunnen beheren, back-ups kunnen maken, en de status van hun systemen kunnen monitoren.
4. **Geavanceerde Functies:** Het platform ondersteunt live migratie van VM's, snapshots, high availability (HA) clustering, en software-defined storage.
5. **Ondersteuning voor Verschillende Besturingssystemen:** Proxmox kan verschillende besturingssystemen virtualiseren, waaronder Linux, Windows en BSD, waardoor het een veelzijdige oplossing is voor verschillende toepassingen.
6. **Back-up en Herstel:** Het biedt robuuste back-up- en herstelmogelijkheden, inclusief geautomatiseerde back-upschema's en incrementele back-ups.
7. **Kostenefficiënt:** Als open-source oplossing biedt Proxmox een kostenefficiënte manier om geavanceerde virtualisatiefuncties te implementeren zonder dure licentiekosten.

Proxmox is geschikt voor zowel kleine thuislabs als grote bedrijfsomgevingen, dankzij de schaalbaarheid en de uitgebreide functionaliteiten. Het is een populaire keuze onder IT-professionals die een betrouwbare en flexibele virtualisatie-oplossing nodig hebben.

[Link hoe Proxmox en Home Assistant installeren](#)

<https://smarthomescene.com/guides/how-to-install-home-assistant-on-proxmox-the-easy-way/>

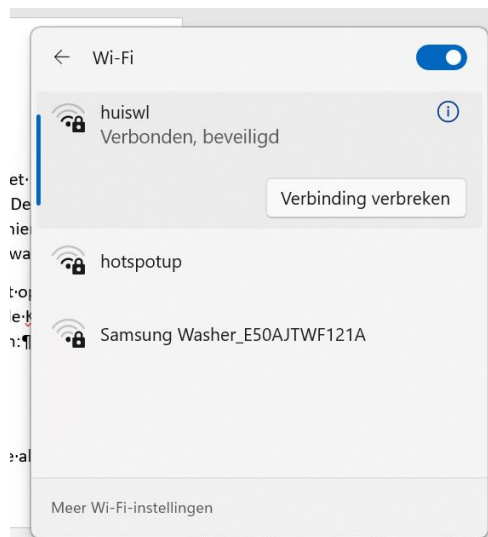
---



## Toevoegen van een oude TP-link LED bulb aan een ruimte in Home Assistant

Dit is een eerste demonstratie van het toevoegen van een smart device aan Home Assistant. Het betreft hier een oude meer kleuren LED bulb van TP-link die kan aangestuurd worden via Wifi. Deze bulb bevat een ESP32 of ESP8266 die zich met een Wifi netwerk kan connecteren. Het betreft hier de iets minder veilige variant aangezien deze geen credentials (gebruikersnaam/wachtwoord) verwacht.

1. Om dit Smart Device op het lokale wifi netwerk te krijgen, bestaat er een App: Kasa dat je eerst op jouw smartphone of iphone moet installeren. Op deze manier wordt de bulb geregistreerd in de Kasa wolk. Maar wij gaan de bulb zonder de Kasa App met het lokale netwerk verbinden, de stappen:
  - a. Voorzie de meer kleuren LED bulb van stroom. Normaliter zal deze eerst pinken.
  - b. Zorg dat je beschikt over een PC die zich kan connecteren met het Wifi netwerk.
  - c. Tik in het systeemvak op het Wifi symbool en klik de lijst bij het Wifi symbool open. Hier vind je alle beschikbare Wifi-netwerken en ook deze van de TP-link LED bulb.



Verbind je PC met deze bulb (door op te klikken en te tikken op verbinden). Na een tijdje zal je worden verbonden met de bulb.

- d. De volgende stap moet maar eenmaal worden uitgevoerd.
- e. Open Thonny en zorg dat je in de gewone Python omgeving zit (Tools → Options → Interpreter → Python).
- f. Installeer de module: python-kasa met behulp van Tools → Manage Packages.
- g. Maak vervolgens het volgende korte Python programma:

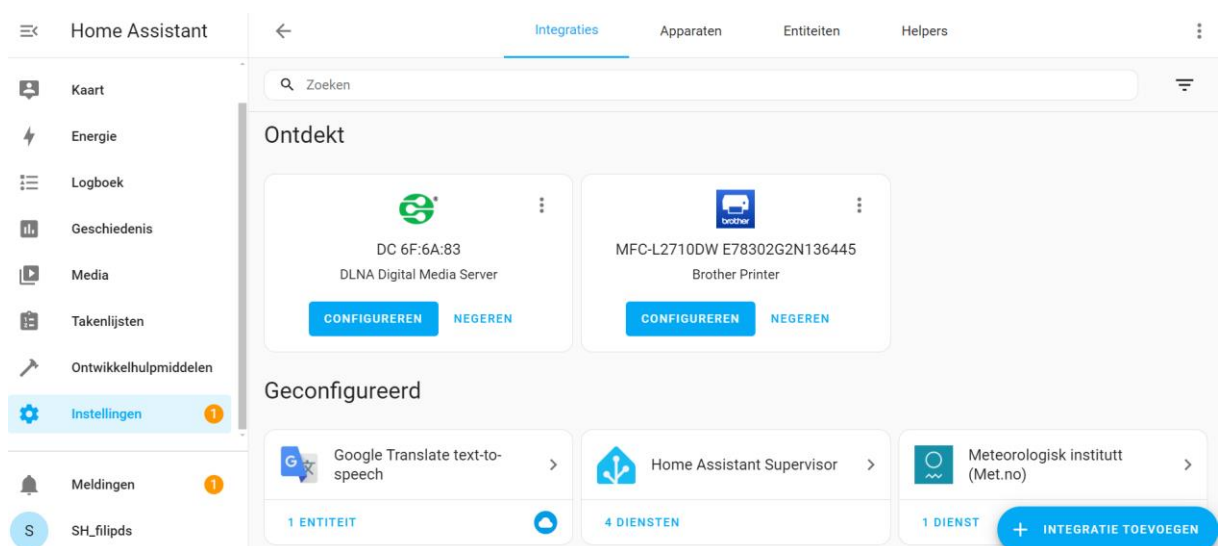
```
1 import asyncio
2 from kasa import Discover
3
4 async def main():
5     dev = await Discover.discover_single("192.168.0.1")
6     #gegevens opvragen van device
7     print(dev.mac)
8     print(dev.alias)
9     print(dev.model)
10    print(dev.device_id)
11    #device koppelen aan wifi netwerk
12    await dev.wifi_join("naam_wifi","passwd")
13
14 if __name__ == "__main__":
15     asyncio.run(main())
```

### Wat uitleg:

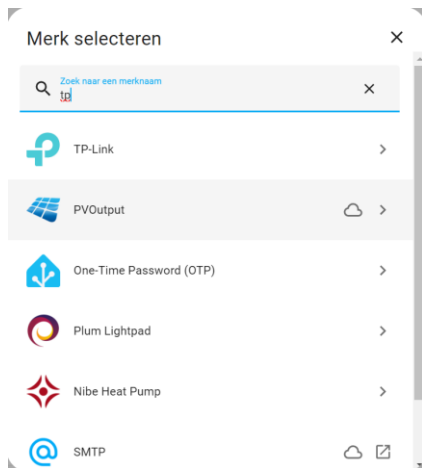
- ✓ Deze code gaat met behulp van de method: `discover_single()` het gewenste bulb object ophalen.
- ✓ Eenmaal het device is gevonden worden er enkele gegevens zoals MAC-adres, alias en model opgevraagd. Met de method: `wifi_join` wordt de bulb of andere tp-link device aan het wifi netwerk gekoppeld. We merken op dat naam\_wifi en password moeten worden vervangen door de naam van het wifi netwerk en bijhorend wachtwoord.
- ✓ Hier wordt er gebruik gemaakt van de `asyncio` module zodat methods die tijd in beslag nemen zoals: `discover` en `join_wifi` op de achtergrond verder draaien en er andere processen kunnen verder lopen.
- ✓ Sla deze code op als: `connect_tp_link_dev_wifi`.

Eenmaal dat deze verbonden is met wifi kan men verder gaan in Home Assistant.

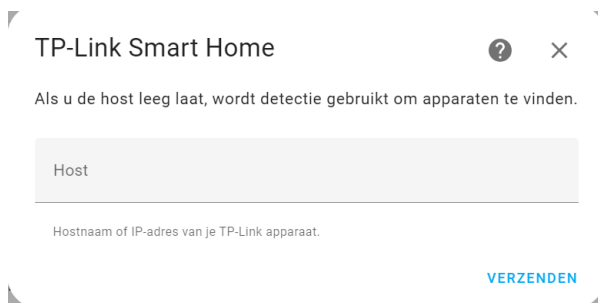
1. Leg de LED lamp aan.
2. Ga naar Home Assistant en tik in het linker paneel op Instellingen. Kies daar voor Apparaten en diensten.



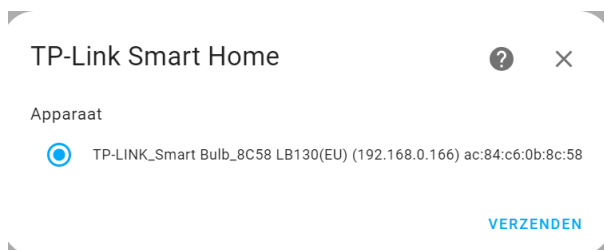
3. Tik op de knop: INTEGRATIE TOEVOEGEN.
4. Zoek vervolgens op TP-Link:



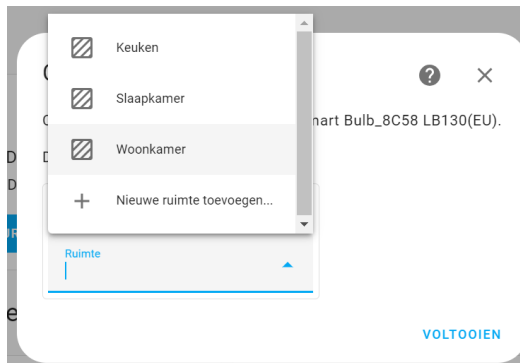
5. Tik in de lijst op TP-Link en kies vervolgens: TP-Link Smart Home.
6. Tik vervolgens op Verzenden:



7. Als de LED lamp wordt gevonden, zal je een bevestiging krijgen:

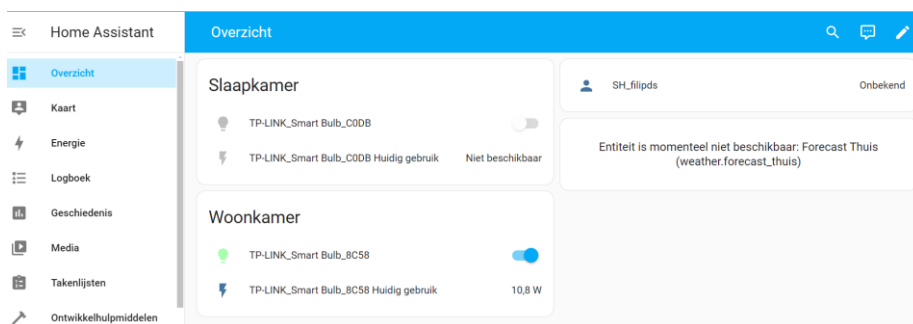


8. Tik terug op Verzenden.
9. Het volgende scherm komt op de voorgrond. Kies een voor gedefinieerde ruimte waarin de lamp zich bevindt. (Je kan hier ook een nieuwe ruimte maken).

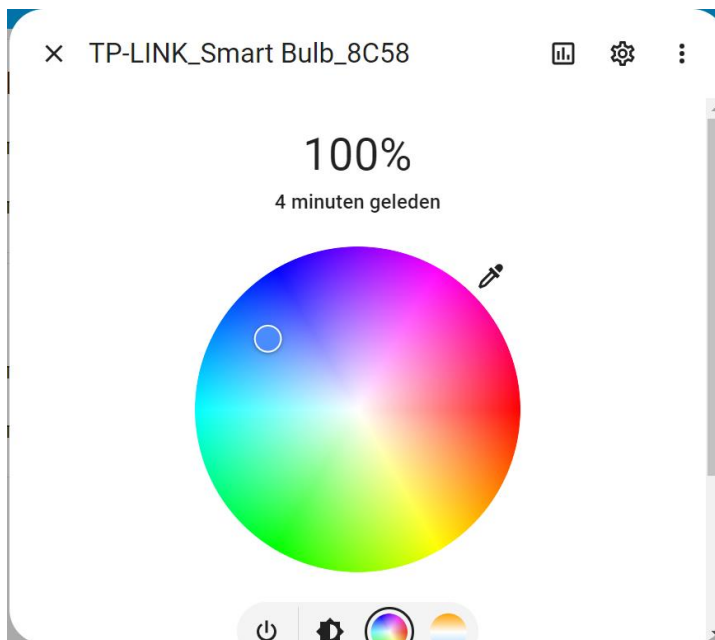


10. Tik op VOLTOOIEN.

11. Ga vervolgens naar Overzicht, de lamp zal onder de gekozen ruimte getoond worden.



12. Je kan de kleur nu instellen door in het overzicht op de gewenste lamp te tikken.



Opdracht toevoegen van één bulb aan jouw Home Assistant en daarna verwijderen.  
Eenmaal de leraar alle LED bulbs verbonden heeft met het Wifi netwerk, kan je aan de slag.

1. Kies een LED bulb om aan je Home Assistant toe te voegen. Zorg dat deze in de ruimte: klas\_je voornaam wordt toegevoegd.

2. Wijzig de kleur van de LED bulb.
3. Roep de leerkracht.
4. Probeer nu een methode te vinden om dit geïntegreerde device terug te verwijderen. Schrijf hieronder eventueel de stappen:

---

---

---

---

Opdracht: toevoegen van een TP-link plug aan jouw Home Assistant en daarna verwijderen. Eenmaal de leraar alle plugs verbonden heeft met het Wifi netwerk, kan je aan de slag.

1. Kies een TP-Link plug om aan je Home Assistant toe te voegen. Zorg dat deze in de ruimte: klas\_je voornaam wordt toegevoegd.
2. Schakel de plug aan en uit.
3. Roep de leerkracht.
4. Verwijder de plug opnieuw van je Home Assistant.

Vrije opdracht: Shelly componenten toevoegen aan Home Assistant

De leraar heeft wat shelly componenten voor deze oefening voorzien, nl.

- ✓ Smart plugs: die je in een stekker stopt en vervolgens kan in en uitschakelen. Sommige modellen houden ook het verbruik bij van het gekoppelde toestel aan het lichtnet.
- ✓ Shelly LED bulbs die je kan aansturen en eventueel wijzigen van kleur met behulp van Home Assistant.
- ✓ Indien mogelijk zal de leerkracht ook enkele Shelly relay schakelingen meebrengen waarmee kan geëxperimenteerd worden.

Veel informatie kan gevonden worden op het web om Shelly componenten te integreren in Home Assistant.

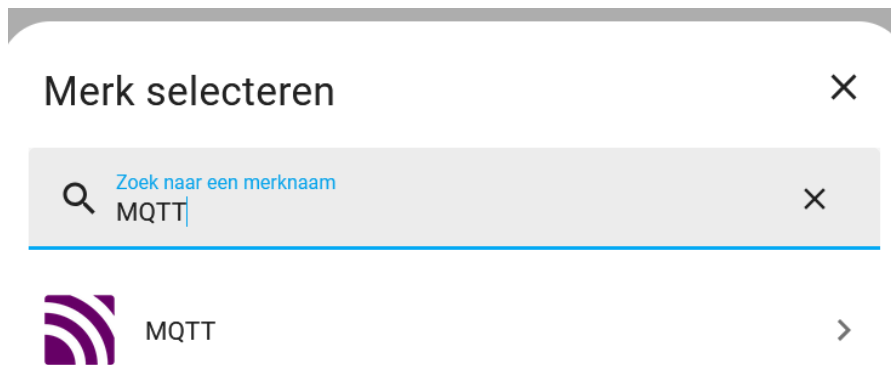
Probeer zoveel mogelijk componenten uit, combineer eventueel de componenten en werk samen met een aantal medecursisten.

## Toevoegen eigen hardware (gemaakt met ESP32) aan Home Assistant

We gaan van start met een eenvoudig project. Een LED gekoppeld aan de ESP32 aan en uit zetten via Home Assistant.

Home Assistant klaar maken om aan MQTT te doen

1. Meld je aan bij Home Assistant en ga naar instellingen.
2. Tik vervolgens op Apparaten en diensten en klik dan op de knop: INTEGRATIE TOEVOEGEN.
3. Zoek op MQTT.



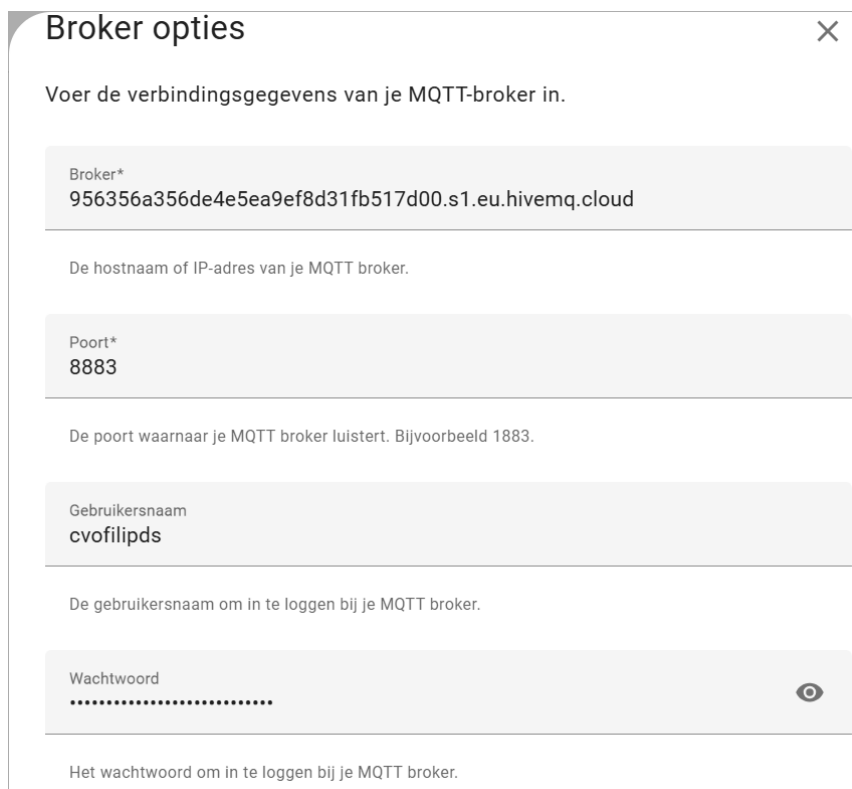
Merk selecteren

Zoek naar een merknaam

MQTT

MQTT

4. Zet onderaan de geavanceerde optie aan (aangezien we gaan gebruik maken van de HiveMQ Cloud account). (Opmerking: de geavanceerde opties worden maar zichtbaar als je onder jouw profiel de geavanceerde opties aanzet).  
En vul deze als volgt in:



Broker opties

Voer de verbindingsgegevens van je MQTT-broker in.

Broker\*

956356a356de4e5ea9ef8d31fb517d00.s1.eu.hivemq.cloud

De hostnaam of IP-adres van je MQTT broker.

Poort\*

8883

De poort waarnaar je MQTT broker luistert. Bijvoorbeeld 1883.

Gebruikersnaam

cvofilipds

De gebruikersnaam om in te loggen bij je MQTT broker.

Wachtwoord

.....

Het wachtwoord om in te loggen bij je MQTT broker.

Client ID (leeg laten voor een willekeurig ID)

Het unieke ID om de Home Assistant MQTT API te identificeren als MQTT client. Het is aanbevolen om deze optie leeg te laten.

Tijd tussen het verzenden van keep-a-live berichten

60

sec

#### Gebruik een client-certificaat

Selecteer en klik op **volgende** om een clientcertificaat en privésleutelbestand in te stellen om te authenticeren met je MQTT broker.



Broker certificaatvalidatie

Automatisch



Selecteer **Automatisch** voor automatische validatie van het certificaat van je MQTT broker of **Handmatig** in klik op **volgende** om zelf een CA certificaat in te stellen om het certificaat van je MQTT broker te valideren.



**AANGEPAST CA CERTIFICAATBESTAND UPLOADEN**

Of plaats je bestand hier

Het aangepaste CA certificaatbestand om het je MQTT broker's certificaat mee te valideren.

#### Negeer validatie van brokercertificaten

Optie om validatie van je MQTT broker's certificaat te negeren.



MQTT protocol

3.1.1



Het MQTT protocol waarmee je broker communiceert. Bijvoorbeeld 3.1.1.

MQTT transport

TCP



Dus de volgende zaken moeten als volgt worden ingevuld:

Het adres van je hivemq cloud (terug te vinden in je hivemq cloud omgeving onder Manage Clusters)

De poort: 8883 (de secure poort om aan MQTT te doen).

De aanmeld gegevens: gebruikersnaam/wachtwoord (terug te vinden in je hivemq cloud omgeving onder Manage Clusters)

Gebruik een client certificaat laat je af staan.

Zet Broker certificaat validatie op automatisch.

Laat de rest ongemoeid.

5. Tik op verzenden.

6. Testen kan je vervolgens door naar instellingen → devices te gaan en te tikken op MQTT.
  - a. Tik vervolgens op: configureren.
  - b. Vul een topic en een boodschap in:

## Publiceer een bericht

Onderwerp  
test\data

QoS  
0

☒ Vasthouden

☐ Sjabloon toestaan

Bericht

1 Dit is nog een test

PUBLICEER

- c. Zorg dat aan de andere kant (in de hivemq cloud) naar de web client wordt gegaan.
- d. Meld je daar aan met je gebruikersnaam en wachtwoord en tik op connect.
- e. Subscribe op de zopas gemaakte topic in homeassistant.

Username \*

cvofilipds

Password \*



.....

Disconnect

The WebClient is connected

### Topic Subscriptions 1

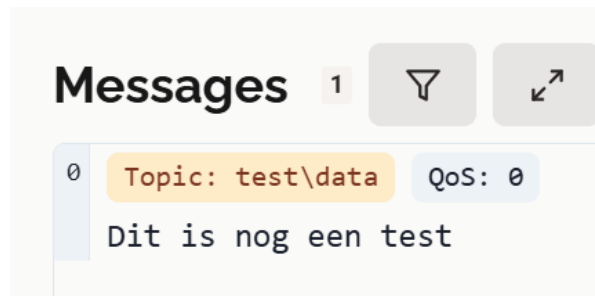
Subscribe to topics to receive messages from the HiveMQ cluster. You can also set the Quality of Service (QoS) for each topic. The higher the QoS, the more reliable the message delivery is. You can always subscribe to the ( # ) wildcard to receive all messages.

TOPIC	QOS	ACTIONS
 test\data	QoS: 0	

- f. Tik op Publiceer in homeassistant.



- g. De boodschap komt toe bij de hivemq web client, in het rechter paneel:



#### Maken MicroPython script voor ESP32

1. Start met het verbinden van een LED met GPIO poort 23. (vergeet de weerstand niet).
2. We zullen gebruik maken van MQTT om de LED gekoppeld aan de ESP32 te sturen met behulp van Home Assistant. Hieronder de voorwaarden en stappen nodig om een communicatie met Home Assistant op te zetten:
  - a. We zullen terug gebruik maken van SimpleWifiV2 om een connectie met Wifi aan te gaan. Indien geen connectie, herstart de microcontroller met behulp van: `machine.reset()`.
  - b. Standaard worden er 3 topics gebruikt om te communiceren met Home Assistant, nl:
    - i. CMD\_TOPIC: home/lab\_je\_voornaam\_eerste letters achternaam/led1/set. Het ESP32 programma zal zich abonneren op deze topic om het commando (on, off) van Home Assistant te ontvangen.
    - ii. STATE\_TOPIC: home/lab\_je\_voornaam\_eerste letters achternaam/led1. Eenmaal de LED van toestand is veranderd, wordt dit doorgegeven aan Home Assistant.
    - iii. AVAIL\_TOPIC: home/lab\_je\_voornaam\_eerste letters achternaam/available. ESP32 geeft sein aan Home Assistant als deze beschikbaar is.
  - c. Als MQTT broker maken we gebruik van je cloud hivemq broker (zie ook blz.38).
3. Maken van een LED object en MQTTClient object.
4. Koppel de callback functie: `ha_cmd` aan de MQTTClient.
5. Connecteer met de broker. Indien dit niet lukt start programma opnieuw op (`machine.reset()`).
6. Publish: online met behulp van AVAIL\_TOPIC
7. Gaat in een oneindige lus, waarin het volgende gebeurt:
  - a. Wacht op een boodschap (eenmaal boodschap toekomt, roept callback functie: `ha_cmd` op.
8. De while lus zit in een try – finally blok. In het finally blok wordt de boodschap offline gestuurd met behulp van de topic: AVAIL\_TOPIC en worden de verbindingen gesloten.
9. In de callback functie: `ha_cmd` worden de volgende zaken uitgevoerd:
  - a. De inkomende boodschap wordt gecontroleerd
    - i. Indien on, zal de LED stroom krijgen.
    - ii. Indien off, zal de LED doven.
  - b. De nieuwe toestand wordt doorgegeven aan Home Assistant. Er wordt on of off doorgegeven met de topic: STATE\_TOPIC.

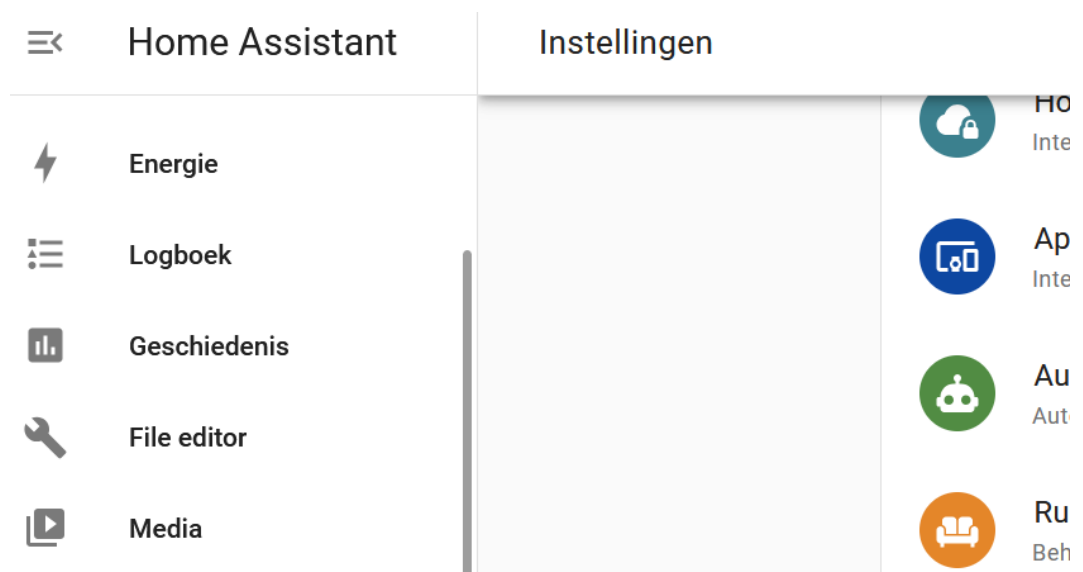
#### Opdracht: vertalen van het bovenstaande stappenplan naar code

1. Schrijf bovenstaand stappenplan uit in MicroPython code.

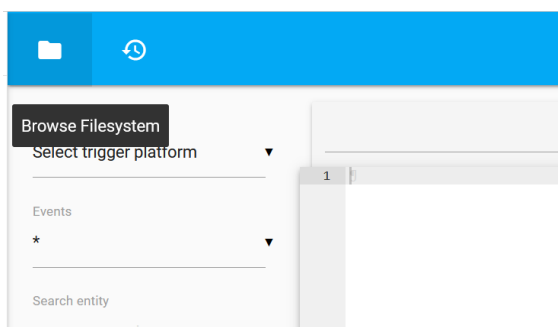
2. Test de applicatie uit met behulp van de cloud hivemq web client.
3. Sla het programma op als `basic_mqtt_HA_switch.py`.

Configureren van Home Assistant voor een eigen device (ESP met LED).

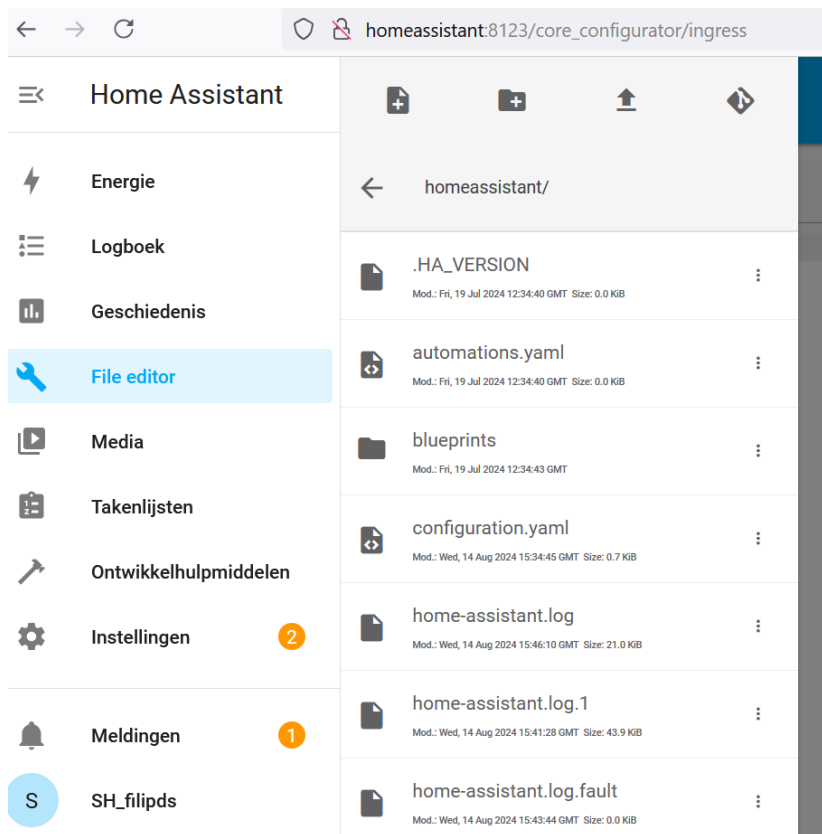
1. Start het programma op de ESP32 op.
2. We zullen een nieuwe entry moeten maken in het yaml bestand: `configuration.yaml`. Daarin zullen we een nieuwe mqtt entiteit moeten maken waaraan de nodige topics en payloads worden toegevoegd.
3. Maar voordat we het configuratie bestand: `configuration.yaml` kunnen aanpassen moeten we eerst de Add-On: File Editor installeren. (Dit hoeft maar 1x te gebeuren).
  - a. Ga naar instellingen en tik vervolgens op Add-Ons.
  - b. Zoek op File Editor en installeer deze.
  - c. De Add-On: File Editor verschijnt in het linker paneel.



4. Tik in het linker paneel op File Editor.
5. Klik links bovenaan in de File editor op het map icoon om een lijst met configuratie bestanden te bekomen.



6. De volgende lijst komt op de voorgrond:



7. Tik in de lijst op configuration.yaml.
8. En geef onderaan de volgende instellingen in voor de nieuwe mqtt entiteit. Let wel op de insprongen moeten volledig kloppen of de entiteit wordt niet gedetecteerd: (elke insprong wordt vermeerderd met 2 spaties).

**mqtt:**

**- switch:**

```

name: "led1"
unique_id: led1 lab
state_topic: "home/lab_je voornaam_2 letters achternaam/led1"
payload_on: "on"
payload_off: "off"
state_on: "on"
state_off: "off"
command_topic: "home/lab_je voornaam_2 letters achternaam/led1/set"
availability_topic: "home/lab_je voornaam_2 letters achternaam/led1/available"
payload_available: "online"
payload_not_available: "offline"
qos: 0
retain: false

```

### Wat uitleg:

- Eerst moet je onderaan het configuratie bestand een mqtt entry maken (door mqtt: in te geven).
  - Op de volgende lijn spring je vervolgens in om - switch: in te geven. Dit is het type entiteit, de meest gekende zijn: switch, light en sensor.
  - Vervolgens volgen de naam en id van de entiteit, dit is het best uniek ten opzichte van de andere entiteiten.
  - Dan worden de verschillende topics met corresponderende payloads daaronder geplaatst. De 3 topics moeten overeenkomen met de topics in het MicroPython script op de ESP32
- Zoek ook eens met Google op: homesassistant mqtt switch, dan kan je alle mogelijkheden van de switch raadplegen..

9. Druk vervolgens op het icoon met diskette.
10. Tik vervolgens in het linker paneel op Ontwikkelhulpmiddelen.

Ontwikkelhulpmiddelen

[YAML-CONFIGURATIE](#) [STATUSSEN](#) [SERVICES](#) [SJABLONEN](#) [GEBEURTENISSEN](#) [STATISTIEKEN](#) [ASSIST](#)

### Controleren en opnieuw opstarten

Een basisvalidatie van de configuratie wordt automatisch uitgevoerd voordat deze opnieuw wordt opgestart. De basisvalidatie zorgt ervoor dat de YAML-configuratie geen fouten bevat waardoor Home Assistant of een integratie niet kan worden gestart. Het is ook mogelijk om alleen de basisvalidatiecontrole uit te voeren zonder opnieuw op te starten.

[CONFIGURATIE CONTROLEREN](#) [HERSTARTEN](#)

### YAML configuratie herladen

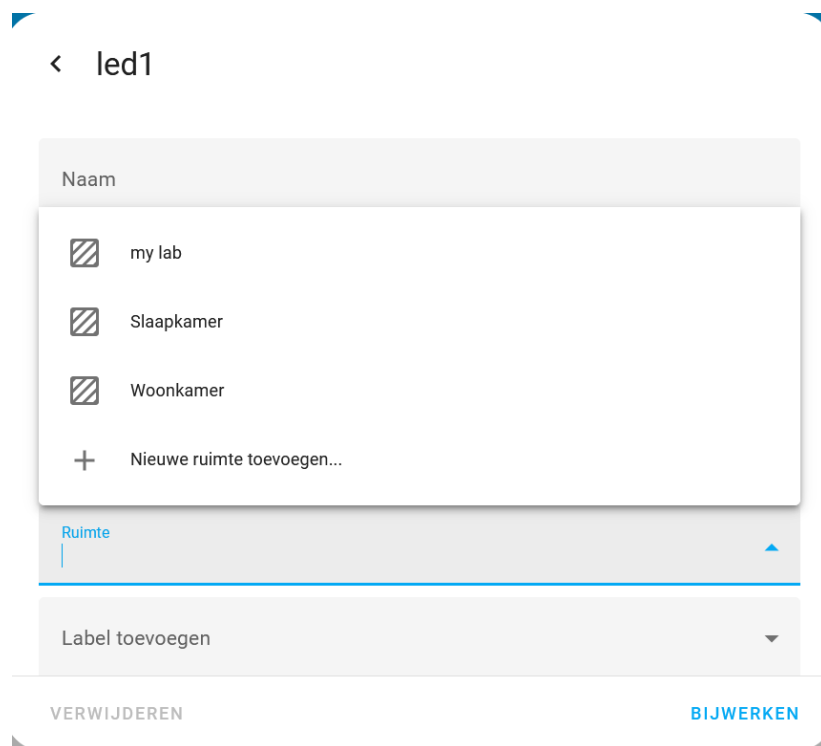
Sommige onderdelen van Home Assistant kunnen opnieuw geladen worden zonder dat een herstart nodig is. Als u op een van de onderstaande opties klikt, wordt de huidige YAML-configuratie verwijderd en de nieuwe geladen.

[ALLE YAML CONFIGURATIES](#)

[LOCATIE EN AANPASSINGEN](#)

11. Tik op CONFIGURATIE CONTROLEREN. Indien er verschijnt: Geldige configuratie, dan zijn de configuratiebestanden in orde.
12. Tik dan op HERSTARTEN. Wacht totdat Home Assistant terug heropgestart is.
13. Ga dan naar Overzicht in het linker paneel. Je zal merken dat deze entiteit nog niet zichtbaar is. Je zal ze eerst lid moeten maken van een ruimte, dit kan als volgt:
  - a. Zoek op de naam van de entiteit, nl. hier led1.
  - b. Tik in het paneel van led1 op de knop instellingen.
  - c. Schuif het item ruimte open kies voor nieuwe ruimte.
  - d. Geef aan deze ruimte de naam mijn labo.

- e. Tik op bijwerken.



14. De nieuwe ruimte met bijhorende switch komt op het overzicht te staan.
15. Om de switch: led1 volledig operatief te krijgen, zal je het python script op de esp32 nog iets moeten aanpassen.
  - a. Maak juist voor de while True: lus een lus die 10x wordt doorlopen, met daarin de volgende functionaliteit:
    - i. Publiceer online via de topic:AVAIL\_TOPIC
    - ii. Wacht na iedere publish 0.5 seconde
  - b. Om in off mode te beginnen publiceer je het best off via de topic:STATE\_TOPIC.
16. De ketting is volledig klaar, start het script op de ESP32 op. Na een tijdje zal je de led, gekoppeld aan de ESP, kunnen sturen via Home Assistant.

Opdracht: een tweede led toevoegen om aan te sturen

1. Hang een 2<sup>de</sup> led (led2) aan de ESP32, dit aan de GPIO poort: 21.
2. Zorg dat het led object voor led2 wordt gemaakt in de MicroPython code.
3. Maak voor deze led 2 nieuwe topics, nl:
  - a. STATE\_TOPIC\_2 = "home/lab\_je voornaam\_2 letters achternaam/led2"
  - b. CMD\_TOPIC\_2 = "home/lab\_filip\_ds/led2/set"
4. Pas de ha\_cmd callback functie verder aan. Dus nu zal er moeten gekeken worden of de topic led1 of led2 bevat en dan de nodige acties ondernemen.
5. Vergeet niet te abonneren op de topic: CMD\_TOPIC\_2.
6. Aan de zijde van Home Assistant zal je terug het bestand: configuration.yaml moeten openen (met behulp van de File editor).
7. Een nieuw switch blok met juiste namen, topics moeten toevoegen.
8. Het configuratie bestand moeten valideren en Home Assistant opnieuw moeten opstarten.
9. Tenslotte in overzicht zoeken op led2 en deze lid maken van de ruimte: mijn labo.
10. Start tenslotte het python script op en kijk of alles naar behoren functioneert.

## Inhoudsopgave

Maken van de toepassing: waste manager .....	2
Maken van de node-red applicatie: waste manager .....	2
Opdracht: Van start met de File Manager .....	2
Opdracht: maken van de waste messenger .....	4
Het ESP32 gedeelte .....	9
Maken van de schakeling .....	9
Opdracht de nodige code schrijven voor de ESP32 .....	10
Opdracht: maken van een App met IoT MQTT Panel .....	14
Opdracht: eigen huisafval kalender integreren .....	16
De servo en stepper motor .....	18
De servo (als klep) .....	18
Opdracht: testen van de klasse: ESP32Servo. ....	22
De klep sturen via een Web Applicatie gestuurd door een webserver op de ESP32 .....	23
Een eigen HiveMQ Broker in de Cloud maken en verbinden met ESP32/node-red .....	36
Maken/registreren van een Broker in de HiveMQ Cloud .....	36
Een random getal publiceren met de ESP32 via de Cloud Broker .....	37
Ontvangen van het random gegenereerde getal door node-red .....	39
Downloaden en installeren van de nodige software om een operationele Home Assistant te bekommen .....	40
Wat is Home Assistant .....	40
Wat is VirtualBox? .....	40
Downloaden en installeren van VirtualBox .....	41
Downloaden van Home Assistant image voor VirtualBox en image toevoegen aan VirtualBox .....	42
TER INFO: opmerking in verband met een meer professionele virtuele omgeving: Proxmox en waarom VirtualBox .....	47
Een korte uitleg over Proxmox .....	47
Link hoe Proxmox en Home Assistant installeren .....	47
Toevoegen van een oude TP-link LED bulb aan een ruimte in Home Assistant .....	49
Opdracht toevoegen van één bulb aan jouw Home Assistant en daarna verwijderen. ....	52
Opdracht: toevoegen van een TP-link plug aan jouw Home Assistant en daarna verwijderen....	53
Vrije opdracht: Shelly componenten toevoegen aan Home Assistant .....	53
Toevoegen eigen hardware (gemaakt met ESP32) aan Home Assistant .....	54
Home Assistant klaar maken om aan MQTT te doen .....	54
Maken MicroPython script voor ESP32 .....	57
Configureren van Home Assistant voor een eigen device (ESP met LED). ....	58

Opdracht: een tweede led toevoegen om aan te sturen .....	61
---	----