Eindwerk IoT4
Schooljaar 2024/2025

Lesgever: Filip Descheerder
Cursist: Sam Deleu

Onderwerp

Basis van een telemetrie framework gebaseerd op espnow en wifi.

Doelstellingen

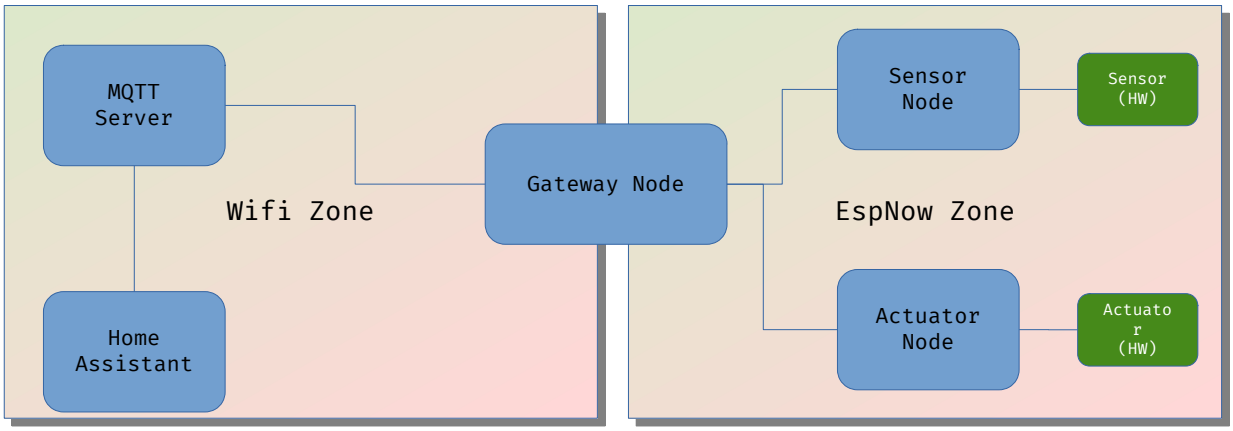
Het idee is dat meetpunten in een open veld niet altijd de toegang tot wifi hebben om te communiceren met de back-office voor het doorgeven van de data. Een vorm van peer-to-peer netwerk waarbij de gegevens van de meetsondes worden doorgegeven aan buren die wel toegang hebben kan dit probleem gedeeltelijk verhelpen.

Nodes die wel toegang hebben tot de back-office kunnen op hun beurt de data doorspelen naar de backoffice.

Om dit te realiseren op een herhaalbare wijze, waarbij nieuwe telemetrie nodes op een eenvoudige wijze kunnen gedefiniëerd worden in micropython python code, wordt tevens een eerste aanzet gemaakt om een framework voor telemetrie voor te stellen.

Opmerking: Momenteel wordt in het framework alleen gekeken naar telemetrie (lezen van sensor data) en nog niet naar aansturen van actuatoren.

Werkingsprincipe



Een Node kan ofwel access hebben tot het Wifi network of niet. Naargelang deze eigenschap kan de node een bepaalde rol vervullen.

Access tot Wifi: *Gateway Nodes*

Geen access tot Wifi: *Sensor Nodes en Actuator Nodes*

De bedoeling is dat elke node elke rol kan vervullen om een dynamische opbouw van het telemetrie mogelijk te maken. In de eerste versie is de rol vastgelegd in een configuratie file (config.py) die op de node aanwezig moet zijn.

Hier ziet men ook direct een mogelijke uitbreiding waarbij nodes zelf beslissen welke rol ze vervullen al naar gelang het feit of ze wifi kunnen bereiken of niet.

Verantwoordelijkheden per rol

SensorNode

- Zich kenbaar maken op het netwerk
- Uitlezen en verzamelen van data van gekoppelde sensoren
- Doorsturen naar een of meerdere gateway nodes

GatewayNode

- Zich kenbaar maken op het netwerk
- Ontvangen en verzamelen van data van gekoppelde SensorNodes
- Dynamisch bijhouden welke SensorNodes er in het netwerk aanwezig zijn.

ActuatorNode

- Zich kenbaar maken op het netwerk
- Ontvangen en interpreteren van specifieke commando’s
- Uitvoeren van de actie van het commando op de actuator
- Melden van de slagen of falen van de actie.

Opstart Sequentie

Als een node opgestart wordt, zal het in zijn configuratie kijken welke rol het moet vervullen.

Vervolgens zal de node een espnow broadcast uitvoeren met een “ANNOUNCE” message. Deze geeft aan dat er een nieuw toestel in het netwerk zit.

Een GateWayNode zal hierop reageren door deze node op zijn beurt een “GATEWAY” message te sturen met zijn eigen MAC-Address zodat de ontvanger vanaf nu direct data kan sturen naar de gateway.

De bedoeling is dat deze opstart sequentie steeds wordt uigevoerd vanaf dat er geen data direct naar de gateway kan gestuurd worden. Op deze wijze kan de sensor/actuator node steeds dynamisch wisselen van gateway bij een instabliel network (wat mag verwacht worden bij een opstelling “in the field”).

Data verwerking – Sensor Node

De sensor node leest de data uit van de geconfigureerde hardware sensors. Deze data wordt gedurende een bepaalde tijd bijgehouden totdat er genoeg data is om door te sturen. Deze data wordt doorgestuurd naar de verschillende gekende gateways, via een “DATA” message.

De SensorNode houdt een maximum aantal “ontdekte” gateways bij, een maximum bepaald door de *espnow* limieten.

Data verwerking – Gateway Node

De gateway node ontvangt data via de DATA message en de bijhorende sensor identificatie (MAC Address).

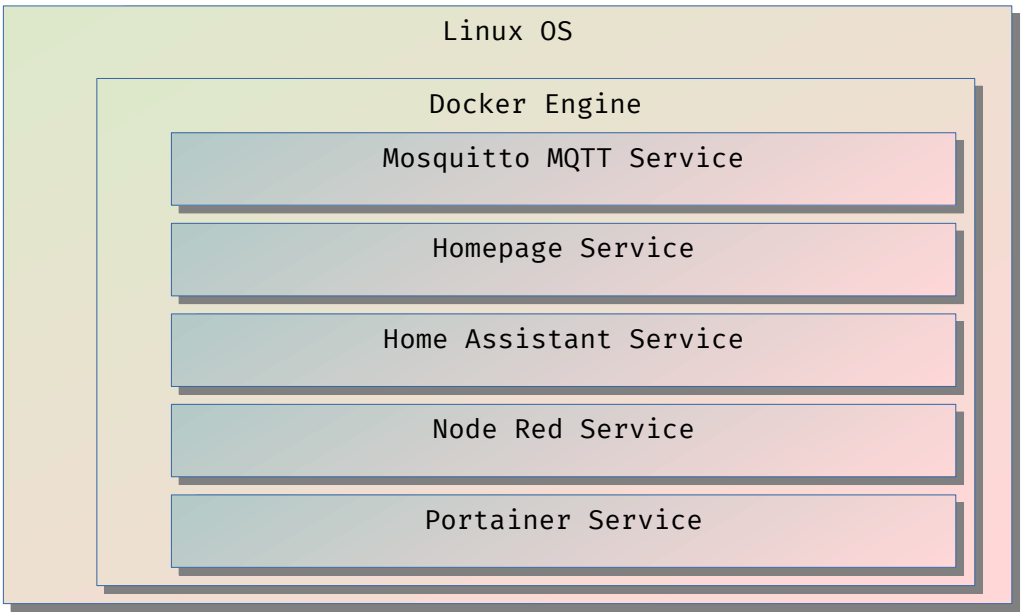
Deze data wordt direct doorgespeeld via een publish naar (1 of meerdere) MQTT server(s).

De bijhorende sensor nodes worden verwittigd dat de data goed is ontvangen zodat de sensor nodes in een nieuwe cyclus data kunnen capteren.

Data verwerking – back-end

In de huidige opstelling wordt de ontvangen data gevisualiseerd m.b.v. *Home Assistant*

Opstelling



De volledige opstelling is gebaseerd op Docker en docker compose om de services voor de verschillende functionaliteiten te starten.

- Mosquitto MQTT server (https://mosquitto.org/)
  De MQTT server die de data onvangt en bijhoudt van de verschillende gateway nodes.
- Homepage (https://gethomepage.dev/)
  Een eenvoudig dashboard.
- Home Assistant (https://www.home-assistant.io/)
  Integratie platform voor home-automation.
- Node Red (https://nodered.org/)
  Low-code automatisatie.
- Portainer (https://www.portainer.io/)
  Grafisch Docker container managed.

Uiteraard is deze opstelling nog beperkt, maar de verschillende services kunnen op aparte hardware gedeployed worden. Zo kan er een bijvoorbeeld een raspberry-pi of een lichtere laptop worden gebruikt voor home-assistant en homepage deployment.

De lokale mosquitto server kan vervangen worden door een cloud-based (commerciele) oplossing zoals HiveMQ (https://www.hivemq.com/)

Docker compose

De services worden opgestart mits een docker-compose.yml file per service. Deze kunnen gemakkelijk gebundeld worden to 1 enkele file, maar tijdens de ontwikkeling leek het beter om voor elke service een aparte file te voorzien.

Deze zorgt ervoor dat:

- de juiste image wordt gebruikt
- de nodige volumes met configuratie en werkingsdata worden aangemaakt, lokaal op de server (hardware).
- Eventueel een definitie van de netwerken zodat de verschillende services netwerk-gewijs kunnen gescheiden worden.

Voorbeeld “Mosquitto compose file”

```
name: mosquitto

services:
  mosquitto:
    container_name: mosquitto
    image: eclipse-mosquitto:2.0.18
    restart: no
    deploy:
      resources:
        limits:
          memory: 256M
    ports:
      - "1884:1883"
      - "9001:9001"
    volumes:
      - /home/sam/HomeAssistant/docker-config/mosquitto_data/config/mosquitto.conf:/mosquitto/config/mosquitto.conf
      - /home/sam/HomeAssistant/docker-config/mosquitto_data/config/pwfile:/mosquitto/config/pwfile
      - /home/sam/HomeAssistant/docker-config/mosquitto_data/data:/mosquitto/data
      - /home/sam/HomeAssistant/docker-config/mosquitto_data/log:/mosquitto/log
    security_opt:
      - no-new-privileges:true
```

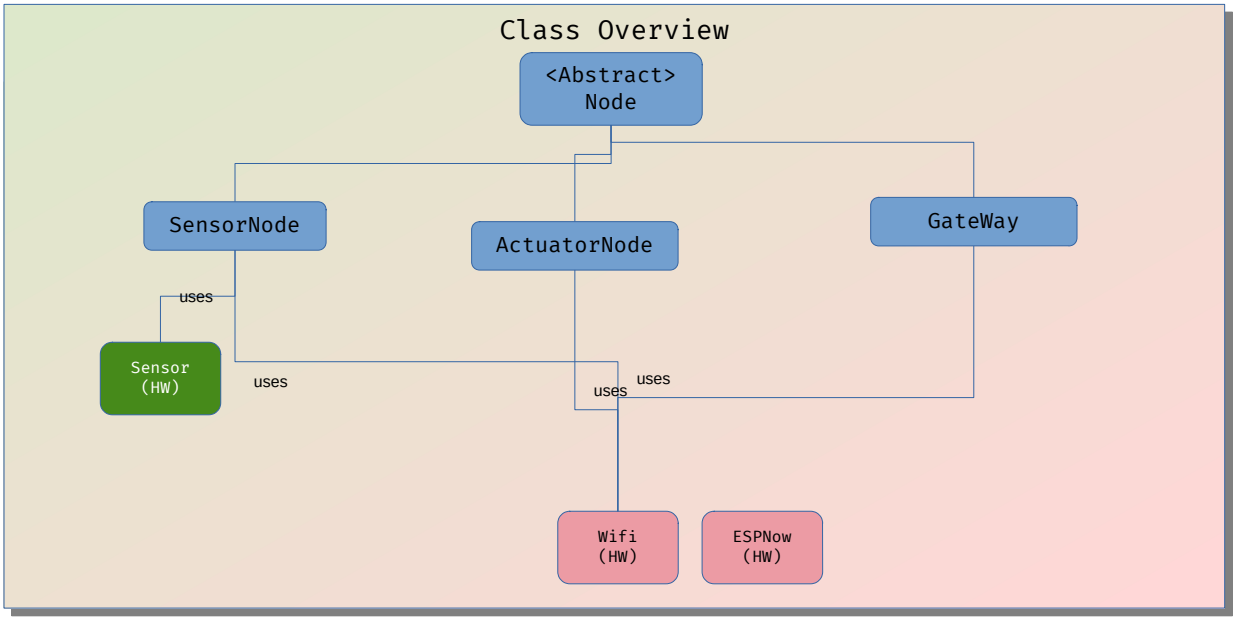
De andere compose files zijn aan het project toegevoegd.

Docker build home-assistant

Om het werken met Home-Assistant wat eenvoudiger te maken, heb ik een custom image gecreëerd, die de HACS integration reeds bevat. Het activeren moet nog steeds gebeuren zoals beschreven in *https://www.hacs.xyz/docs/use/configuration/basic/#to-set-up-the-hacs-integration*.

Framework implementatie

Classes



De basis is de Node class die dient als een “protocol guardian” zodat alle subclasses dezelfde api volgen. Dit houdt in dat de verschillende messagea zoals “*ANNOUNCE*” elk in een aparte functie van de Node class worden geimpelementeerd, zodat de logica en het protocol in de Node class wordt gedefiniëeerd.

Elke subclass overschrijft de acties die ondernomen moeten worden volgens zijn eigen speficieke taken door de corresponderende functie te herdefiniëren.

Communicatie

Zowel *espnow* (https://docs.micropython.org/en/latest/library/espnow.html#) als *wifi* (https://docs.micropython.org/en/latest/esp32/quickref.html#networking) worden gebruikt, beide protocollen zijn beschikbaar op de ESP32 types microcontrollers.

Reactiviteit

Er is gekozen voor een implementatie volledig gebaseerd op de asyncio (https://docs.micropython.org/en/latest/library/asyncio.html#) faciliteiten van micropython, waardoor mijns insziens een zeer reactief framework kan opgebouwd worden.

Er werd tevens gebruik gemaakt van de aioespnow versie van de library om ook het espnow protocol in het async process volledig te verwerken. Voor de wifi word gebruik gemaakt, voorlopig van de wifi class die gezien is in de les, maar het plan bestaat om ook deze volledig async te maken.

De coordinatie tussen de verschillende taken wordt via *Events* van de asyncio library geregeld.
Door het *setten* en *clearen* van events kunnen de verschillende taken beslissen om effectief hun taak uit te voeren of te wachten tot een bepaalde toestand is bereikt die nodig is voor de werking. Deze toestand wordt a.d.h.v. deze events doorgegeven.

Conclusies

Het gebruik van micro-python op low-cost microcontrollers opent een nieuwe kijk op home-automation.

Door het verder uitwerken en bijeenzetten van het materiaal gezien tijdens de lessenreeks in deze oefening en met de hulp van de lesgever ben ik nu in staat om zelf meer ingewikkelde automatisaties te bouwen en bruikbaar te maken.

Dit werk is door omstandigheden nog niet volledig met betrekking tot verdere automatisering via node-red.

Sam Deleu