2023

Environmental Nodes

Software Architectuur

ENVIRONMENTAL NODES
TADRAŁA,PIOTR P.P.

O-PP-CMK

INHOUDSOPGAVE

1.0 INLEIDING

1.1 CONTEXT

Dit document beschrijft de software van het Environmental Nodes-systeem, een netwerk van nodes die omgevingsdata verzamelen. In dit netwerk functioneert een controller als centrale hub, waaraan meerdere nodes gekoppeld kunnen worden. Elke node verzamelt data en stuurt deze naar de centrale hub, en is daarnaast configureerbaar via de centrale hub. De centrale hub stuurt de data naar een cloudomgeving, waar deze wordt opgeslagen in een database. Gebruikers kunnen deze gegevens vervolgens bekijken via een mobiele app.

1.2 SCOPE

Het project bestaat uit de ontwikkeling 4 applicaties die een geheel vormen:

ENVIRONMENTAL NODES

Microcontrollers met sensoren verzamelen data en sturen deze naar de centrale hub. Ze moeten eenvoudig gekoppeld kunnen worden aan de hub, en het moet mogelijk zijn om meerdere microcontrollers aan het netwerk toe te voegen.

ENVIRONMENTAL HUB

Een microcontroller fungeert als hotspot voor de nodes. Deze moet door de gebruiker worden geconfigureerd om verbinding te maken met WiFi. Vervolgens ontvangt de microcontroller metingen van de sensoren en stuurt deze door naar de cloudomgeving. Daarnaast moet de hub ook configuratie-instellingen naar de nodes kunnen verzenden, zoals het aanpassen van het meetinterval van de nodes.

BACKEND

Een cloudomgeving ontvangt data van alle hubs en slaat deze op in de database. Daarnaast wordt via de API data opgehaald voor de frontend interface.

MOBIELE APP

Fungeert als een frontend waarmee gebruikers de data van hun nodes kunnen bekijken.

1.3 FUNCTIONALITEITEN

OMGEVINGSDATA VERZAMELEN

Elke node moet in staat zijn om omgevingsdata, zoals temperatuur, lichtniveau en CO2-niveau, te verzamelen en deze naar de centrale hub te sturen. Dit moet plaatsvinden in een configureerbaar interval dat via de centrale hub kan worden ingesteld.

VERBINDING MAKEN MET DE CENTRALE HUB

Elke node moet eenvoudig verbinding kunnen maken met de centrale hub om het netwerk schaalbaar te maken.

STILL ALIVE-SIGNAAL

Elke node moet in een gedefinieerd interval een ‘Still Alive-signaal’ versturen. Dit signaal zorgt ervoor dat de gebruiker in staat om te controleren welke nodes operationeel zijn, ongeacht het data-interval.

WATCHDOG TIMER

Een watchdog-timer zorgt ervoor dat een node automatisch opnieuw opstart als deze vastloopt.

CENTRALE HUB INTERACTIE

Elke node moet zowel in staat zijn om data naar de centrale hub te versturen als om gegevens te ontvangen, zoals configuratieparameters.

LED-INDICATOREN

Elke node moet de huidige status visueel kunnen weergeven met behulp van LED's. Bijvoorbeeld, een LED per sensor die aangeeft of de sensor operationeel is, en een LED voor het visualiseren van de verbinding met de centrale hub.

2.0 ARCHITECTUUR

2.1 ARCHITECTUUR TYPE

Voor dit project heb ik gekozen voor een Distributed architectuur, omdat het een relatief klein project os en het daarom niet nodig was om bijvoorbeeld een microservices-architectuur te implementeren. De huidige architectuur bestaat uit vier lagen: de Presentation, Logic, Data en Hardware componenten.

2.2 COMPONENTEN

HARDWARE

De Hardware Layer bestaat uit de nodes en de hub, die verantwoordelijk zijn voor het verzamelen van data en het doorsturen ervan naar de Logic Layer. Daarnaast heeft de Hardware Layer directe interactie met de Presentation Layer, waarbij gebruikers via BLE de hub kunnen koppelen aan de mobiele app.

LOGIC

De Logic Layer bestaat uit de API-backend, die zowel data kan ontvangen en verwerken in de Data Layer als gegevens kan ophalen uit de database voor de Presentation Layer.

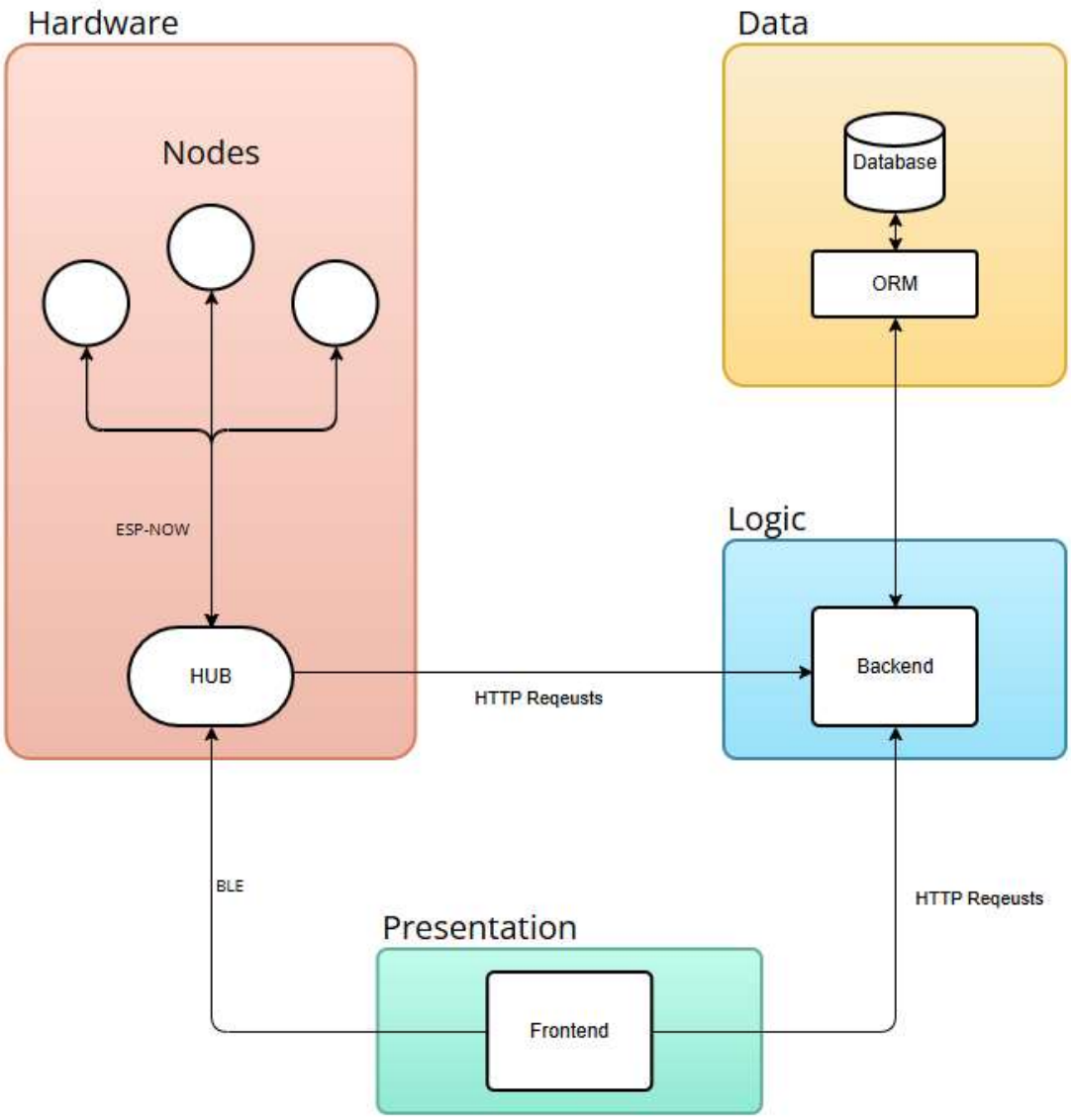
PRESENTATION

Frontend in de vorm van een mobiele app, waarmee gebruikers de gegevens van hun nodes kunnen bekijken.

DATA

ORM waarmee de logic layer met de database communiceert, en een database waar alle gegeven worden opgeslagen

2.3 CONTEXT DIAGRAM



3.0 PRESENTATION

3.1 FRAMEWORK

IONIC

Op de mobiele app te ontwikkelen zal ionic gebruikt worden. Ionic maakt het mogelijk om de frontend in een JavaScript-framework te ontwikkelen en deze vervolgens te compilen naar zowel Android als iOS.

ANGULAR

Als frontend-framework kies ik voor Angular, omdat ik hier al het meest vertrouwd mee ben. Andere potentiële opties waren React en Vite.

3.2 COMMUNICATIE MET DE HUB

De mobiele app zal in eerste instantie via Bluetooth Low Energy (BLE) communiceren met de centrale hub. BLE is gekozen omdat het de meest gebruiksvriendelijke optie is voor het koppelen van twee apparaten zonder input van de gebruiker. Het voorkomt bijvoorbeeld dat de gebruiker zijn hub uit een lijst van beschikbare apparaten moet selecteren. Een alternatieve oplossing was om de gebruiker verbinding te laten maken met de hotspot van de hub, maar de BLE-oplossing is veel gebruiksvriendelijker omdat de volledige koppeling binnen de app plaatsvindt.

3.3 COMUNICATIE MET DE BACKEND

Communicatie met de backend zal door middel van HTTP requests plaatsvinden. Hiervoor zullen services aangemaakt worden waarin ook de authenticatie plaatsvind. Voor de services zal gebruik gemaakt worden van Angular Services

- Angular, (2023) Introduction to services and dependency injection.

3.4 REQUESTS NAAR DE BACKEND

LATEST HUB DATA

Endpoint: /api/hub/latest

Params:

```
{
    "hub_mac_address": "FC:E8:C0:74:96:08"
}
```

Response:

```
{
    "nodes": [
        {
            "mac": "FC:E8:C0:74:7D:D8",
            "light": "0.00",
            "temperature": "20.00",
            "co2": "942.00",
            "humidity": "55.70"
        }
    ]
}
```

NODE DATA

Endpoint: /api/node/get

Params:

```
{
    "node_mac_address": "FC:E8:C0:74:96:08",
    "sensor": "CO2",
    "sub_hours": 6
}
```

Response

```
{
    "data": [
        "450",
        "450",
        "452",
        "473",
        "452"
    ]
}
```

4.0 HARDWARE

4.1 NODES

Een node verzamelt bij elke iteratie gegevens van elke sensor en stuurt deze door naar de hub. Om een node aan de hub te koppelen, moet de hub eerst in hotspotmodus worden gezet. Hierdoor wordt een WiFi-hotspot aangemaakt. De node probeert vervolgens verbinding te maken met deze hotspot. Zodra dit succesvol is, kan de node het MAC-adres van de hub ophalen en dit gebruiken voor de ESP-NOW-communicatie.

Daarnaast wordt de output van elke sensor doorgegeven aan de ledsController. De output van een sensor class kan als volgt zijn:

- **"-9999"**: Dit betekent dat het meten van data niet is gelukt. In dit geval gaat het bijbehorende LED-lampje rood branden.
- **"9999"**: Dit geeft aan dat er nog geen nieuwe meting is gedaan, afhankelijk van het geconfigureerde interval. In dit geval doet de ledsController niks.
- **Alles tussen "-9999" en "9999"**: Deze worden als geldige metingen gezien.

De ledsController verwerkt deze outputs en activeert de LED-indicatoren.

Door voortdurend metingen naar de backend te sturen, kan eenvoudig worden vastgesteld wanneer een node uitvalt.

DEPENDENCIES

- adafruit/DHT sensor library@^1.4.6
- adafruit/Adafruit Unified Sensor@^1.1.14
- claws/BH1750@^1.3.0
- sensirion/Sensirion I2C SCD4x@^0.4.0
- fastled/FastLED@^3.7.0

4.2 HUB

Het voornaamste doel van de hub is het verzamelen van data van de nodes en het doorsturen ervan naar de backend. Daarnaast moet de hub in staat zijn om de nodes te configureren, bijvoorbeeld door configuraties die vanuit de interface worden ontvangen door te sturen naar alle verbonden nodes.

INTERNET VERBINDING

Om de hub met het internet te verbinden, wordt de WifiManager-library gebruikt. Wanneer de hub niet eerder met het internet verbonden is geweest, creëert deze bij het opstarten een hotspot. Zodra de gebruiker verbinding maakt met de hotspot, krijgt hij toegang tot een portal waarin hij zijn thuisnetwerk kan configureren. De hub slaat de wifi-gegevens op en maakt bij de volgende opstart automatisch verbinding met het netwerk.

HOTSPOT STATE

Wanneer de hub al actief is (en verbonden met het internet), kan deze in de hotspot mode worden gezet door op de knop te drukken. In de hotspot mode wordt een wifi hotspot geactiveerd, zodat de nodes deze kunnen detecteren en het MAC-adres kunnen grabben. Daarnaast wordt in de hotspotmodus ook een BLE-service geactiveerd, waarmee de mobiele interface kan pairen.

NODES DATA

De hub ontvangt data van de nodes via ESP-NOW. Het formaat van de data is een JSON-object dat er als volgt uitziet:

```
{
  "key": "CO2",
  "value": "1384"
}
```

HTTP REQUESTS

De uitgaande data naar de backend is ook een JSON-object. Dit object bevat de inkomende data van de nodes, samen met het MAC-adres van zowel de hub als de betreffende node.

```
{
  "hub": "43:79:58:8C:04:8C",
  "node": "A2:92:1E:8D:C1:9B"
  "data": { "key": "CO2", "value": "1384" }
}
```

DEPENDENCIES

- tzapu/WiFiManager
- fastled/FastLED@^3.7.0
- h2zero/NimBLE-Arduino@^1.4.0

5.0 LOGIC

5.1 FRAMEWORK

Voor de logic layer heb ik het Laravel-framework gebruikt, voornamelijk omdat ik hier al ervaring mee had. Deze laag heeft voornamelijk twee hoofdfuncties:

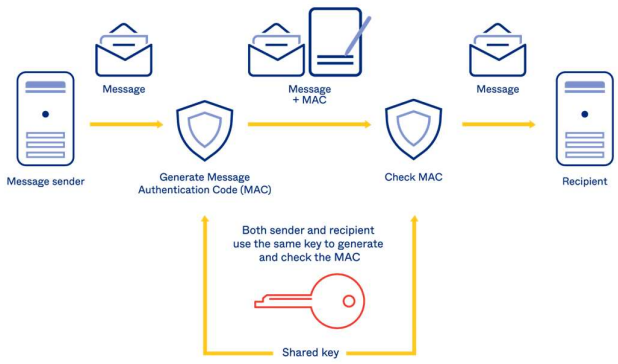
- Functioneren als een API-endpoint voor zowel de hardware layer als de mobiele interface.
- Het ondersteunen van toekomstige logica-implementaties, zoals het analyseren van inkomende data. Denk hierbij bijvoorbeeld aan het analyseren van de lux metingen en het op basis daarvan inschakelen van een samrt lamp

ENDPOINTS

Endpoint	Omschrijving
/api/hub/latest	Ophalen van de meeste recente hub data, per node
/api/node/get	Ophalen van specifieke node data, bijvoorbeeld CO2 metingen van de afgelopen 12 uur,
/api/node/submit	Opsturen van node metingen

5.2 HMAC API AUTHENTICATIE

API Authenticatie is op dit moment niet geimplementeerd vanwege tijdsebrek. Een potentiele authenticatie methode zou het HMAC (Hash-based Message Authentication Code) kunnen zijn, omdat het een veilige manier biedt om communicatie tussen de hardware, mobiele app en de backend te verifiëren zonder dat gebruikers een account hoeven aan te maken. HMAC maakt gebruik van een shared secret om een handtekening te genereren voor elke API request.



- Okta, (2024) HMAC (Hash-Based Message Authentication Codes) Definition https://www.okta.com/identity-101/hmac/

6.0 DATA

6.1 ORM

Voor de communicatie met de database heb ik in dit project gebruikgemaakt van Eloquent, de ingebouwde ORM (Object-Relational Mapping) van Laravel. Eloquent maakt het mogelijk om de database gegevens te benaderen als objecten, wat het werk met data veel eenvoudiger en overzichtelijker maakt in plaats van handmatig SQL-query's te typen.

6.2 DATABASE

De data wordt opgeslagen in een MySQL-database. Deze keuze is gemaakt vanwege mijn eerdere ervaring met MySQL. De database bevat tabellen voor de hubs, nodes en de gemeten gegevens. Elke tabel is gekoppeld aan Eloquent-modellen, wat het makkelijker maakt om gegevens op te halen en te bewerken.

7.0 BRONNEN

- Angular, (2023) Introduction to services and dependency injection. https://v17.angular.io/guide/architecture-services
- Okta, (2024) HMAC (Hash-Based Message Authentication Codes) Definition https://www.okta.com/identity-101/hmac/