WasteBin

Generated by Doxygen 1.9.4

1 Bureaubladonderzoek **1**

1.1 Hypotheses en feiten die baat hebben bij de ontwikkeling van de Waste Bin . . . 1

1.1.1 Inhoud . . . 1

1.2 Tabellen . . . 1

1.3 Achtergrond informatie . . . 1

1.4 Feiten . . . 2

1.5 Hypotheses . . . 2

1.6 Aanpak onderzoek en bronanalyse . . . 3

1.6.1 Uitwerking vragen . . . 3

1.7 Onderzoek naar koelmogelijkheden . . . 5

1.8 Wat weerhoudt veel hoogbouwwoningbewoners ervan om hun GFE-afval te scheiden? . . . 5

2 WasteBin Project: Het creëren van een innovatieve oplossing voor geurvrije en milieuvriendelijke afvalverwerking **7**

2.1 Inleiding . . . 7

2.2 Belangrijkste doelstellingen . . . 8

2.3 Kenmerken en functionaliteit van het apparaat . . . 8

2.4 Voordelen en impact . . . 9

2.5 Conclusie . . . 9

3 Het Gebruik van meerdere ESP32-Microcontrollers met Node-Red **11**

3.1 Inleiding . . . 11

3.2 Doelen . . . 11

3.2.1 Doel #1 . . . 11

3.3 Onderbouwing keuze ESP32 C3 . . . 11

3.4 Waarom geen Arduino of Raspberry Pi als computer voor de WasteBin? . . . 12

3.5 Data forwarding . . . 12

3.6 De Data . . . 12

3.7 Meldingen voor de gebruiker . . . 13

3.8 Verstuur interval . . . 13

3.8.1 In geval van fouten . . . 13

3.9 Node-Red . . . 13

3.9.1 Node-Red flow . . . 14

3.10 Database . . . 14

3.10.1 Views . . . 15

3.10.2 View 1 : view_al_testpersons_with_wastebinnr . . . 15

3.10.3 View 2: view_all_sensors_data . . . 15

3.10.4 View 3: view_all_wastebin_settings . . . 16

4 Handleiding voor het opzetten van de Mosquitto Broker **17**

4.1 Vereisten . . . 17

4.2 Configuratie van het Arduino-script . . . 17

4.3 Verbinding maken met de Mosquitto Broker . . . 18

4.4 Interactie met de Mosquitto Broker . . . 18
4.5 Conclusie . . . 18

5 Ontwerpkeuzes **19**
5.1 Functionele specificaties . . . 19
5.2 Technische specificaties . . . 19

6 PCB-ontwerp **21**
6.1 PCB V0.1 Ontwerp . . . 22

7 Testplan **25**
7.1 Inhoud . . . 25
7.2 Peltier modules . . . 25
7.3 Doelen . . . 25
7.4 Deksel positie . . . 26
7.5 Temperatuur en luchtvochtigheid . . . 26
7.5.1 Doel #1 . . . 26
7.5.2 Doel #2 . . . 26
7.5.3 Doel #3 . . . 26
7.6 Testopstelling . . . 27
7.7 Vereiste componenten . . . 27
7.8 Meetapparatuur . . . 27
7.9 Methodes . . . 27
7.10 Testresultaten . . . 28

8 File Index **29**
8.1 File List . . . 29

9 File Documentation **31**
9.1 Bureaubladonderzoek.md File Reference . . . 31
9.2 Introductie.md File Reference . . . 31
9.3 IoT oplossing.md File Reference . . . 31
9.4 Mosquittosetup.md File Reference . . . 31
9.5 Ontwerpkeuzes.md File Reference . . . 31
9.6 PCB-design.md File Reference . . . 31
9.7 Testplan.md File Reference . . . 31
9.8 waste_bin_controller.h File Reference . . . 31
9.8.1 Function Documentation . . . 32
9.8.1.1 controlPeltierModule() . . . 32
9.8.1.2 handleIncomingMessage() . . . 33
9.8.1.3 loop() . . . 33
9.8.1.4 publishMessage() . . . 33
9.8.1.5 publishSensorData() . . . 33
9.8.1.6 readSensorData() . . . 33

9.8.1.7 setup() . . . 33
9.8.1.8 setupMQTTClient() . . . 33
9.8.1.9 setupPeltierAndFan() . . . 34
9.8.1.10 setupSHT4x() . . . 34
9.8.1.11 setupWiFi() . . . 34
9.8.1.12 updateLidPosition() . . . 34
9.8.2 Variable Documentation . . . 34
9.8.2.1 BUTTON_PIN . . . 34
9.8.2.2 CLIENT_NAME . . . 34
9.8.2.3 humidity_value . . . 34
9.8.2.4 interval . . . 35
9.8.2.5 last_msg_sent . . . 35
9.8.2.6 lid_position . . . 35
9.8.2.7 mqtt_client . . . 35
9.8.2.8 MQTT_PASSWORD . . . 35
9.8.2.9 MQTT_PORT . . . 35
9.8.2.10 MQTT_SERVER . . . 35
9.8.2.11 MQTT_USERNAME . . . 35
9.8.2.12 msg_interval . . . 36
9.8.2.13 PASSWORD . . . 36
9.8.2.14 peltierandfanpin . . . 36
9.8.2.15 peltierandfanstate . . . 36
9.8.2.16 previousMillis . . . 36
9.8.2.17 pwmFreq . . . 36
9.8.2.18 SENSOR_DATA_TOPIC . . . 36
9.8.2.19 setPoint . . . 36
9.8.2.20 sht4 . . . 37
9.8.2.21 SSID . . . 37
9.8.2.22 temperature_value . . . 37
9.8.2.23 tolerance . . . 37
9.8.2.24 wifi_client . . . 37
9.9 waste_bin_controller.h . . . 37
9.10 waste_bin_controller.ino File Reference . . . 38
9.11 waste_bin_controller.ino . . . 38

Index **43**

Chapter 1

Bureaubladonderzoek

1.1 Hypotheses en feiten die baat hebben bij de ontwikkeling van de Waste Bin

1.1.1 Inhoud

- Tabellen
- Achtergrond informatie
- Feiten
- Hypotheses
- Hoofd- en deelvragen
- Aanpak onderzoek en bron analyse
- Uitwerking vragen
- Bibliografie

1.2 Tabellen

Tabel 1: Hypotheses

1.3 Achtergrond informatie

Voor het WasteBin project zijn we gevraagd om een GFE-bak te ontwerpen die GFE-afval vrijwel geurloos en hygiënisch opslaat. Om het geurloze resultaat te bereiken, moeten we het rottingsproces zoveel mogelijk vertragen. Hierbij moeten we een balans vinden tussen energieverbruik, kosten en tijd. We moeten onderzoeken hoe deze factoren invloed hebben op de werking van het product. Daarnaast moeten we vragen beantwoorden zoals: is een ozon-generator überhaupt nodig als we het rottingsproces vertragen? In dit document stellen we tests op, voeren we ze uit en bespreken we de resultaten, om vervolgens onze conclusies te implementeren in ons (technisch) ontwerp.

1.4 Feiten

- Een hoge luchtvochtigheid en temperatuur zorgen voor een sneller rottingsproces bij GFE-afval.
- Een te hoge concentratie ozon is gevaarlijk voor de gezondheid van de mens.
- Koude lucht kan een hogere luchtvochtigheid hebben dan warme lucht.

"Wanneer uurgemiddelde ozonconcentraties hoger zijn dan 180 microgram per kubieke meter lucht, is de luchtkwaliteit 'slecht'. Als deze waarde overschreden dreigt te worden, kunnen gevoelige mensen klachten krijgen en waarschuwt het RIVM. De luchtkwaliteit is 'zeer slecht' wanneer de concentraties drie uur lang hoger zijn dan 240 microgram per kubieke meter lucht. Wanneer deze waarde overschreden wordt, kan iedereen klachten krijgen. Als dit dreigt te gebeuren, zet het RIVM de waarschuwing om in een alarm." (RIVM, sd)

1.5 Hypotheses

Hypothese	Uitwerking/opmerkingen
Een gegeven materiaal houd zijn temperatuur beter vast dan lucht.	Thermoskan idee: hoe voller de thermoskan is gevuld met hete drank, hoe langer het duurt voor de inhoud is afgekoeld. Dus lucht ontnemen of de bak vullen met water om de optimale temperatuur makkelijk te behouden.
Water geleid temperatuur beter dan lucht.	Door de vuilnisbak met water te vullen en het GFE-afval in een zak hierin te koelen, zal dit een efficiëntere thermische overdracht hebben dan via de lucht. De verspreiding van de geur van het rottingsproces zal door het koelen van het GFE-afval minder snel plaatsvinden. Hierdoor is de ozon-generator overbodig.
Het invriezen van het GFE-afval stopt het rottingsproces volledig.	Het invriezen van het GFE-afval zal ervoor zorgen dat de micro-organismen die zich in het afval bevinden niet verder kunnen groeien. Sommige organismen kunnen dit alsnog doen, maar dit gebeurt langzamer en zal in de prullenbak nog niet plaatsvinden.
Het ontnemen van (een deel van) de zuurstof in combinatie met een ozon-generator remt het rottingsproces voldoende af.	De combinatie van een ozon-generator en minder zuurstof zal zorgen dat het afval dusdanig lang geremd wordt dat koeling niet nodig is.

Hoofd- en deelvragen:

1. Wat heeft de grootste invloed op het rottingsproces? a. Welk effect heeft de omgevingstemperatuur op het rottingsproces? i. Hoeveel invloed heeft een negatieve temperatuur op het rottingsproces? b. Welk effect heeft luchtvochtigheid op het rottingsproces? i. Hoeveel effect heeft een lagere luchtvochtigheid op het rottingsproces? c. Welk effect heeft zuurstof op het rottingsproces? i. Welk effect heeft de aanwezigheid van zuurstof in een ruimte op het rottingsproces? ii. Hoeveel effect heeft zuurstof op het rottingsproces? d. Wat is de maximale tijd waarop het rottingsproces kan worden uitgesteld of vertraagd rekening houdend met de drie eerder genoemde factoren? e. Is het mogelijk om het effect van ozon-generatie te compenseren met koeling (tot en op het vriespunt)?
2. Hoeveel liter GFE-afval moet de WasteBin kunnen opslaan? a. Hoeveel liter GFE-afval produceert een gemiddeld huishouden per dag? (stedelijk gebied) b. Wat is het maximale volume van de prullenbak en is dit haalbaar? c. Wat verstaan we onder een huishouden?

3. Wat weerhoudt veel bewoners van hoogbouwwoningen ervan om hun GFE-afval te scheiden? a. Welke drempels kunnen we realistisch gezien verlagen om het scheidingssproces te bevorderen?

1.6 Aanpak onderzoek en bronanalyse

Ons onderzoek, testen en documentatie zijn gebaseerd op het prototype dat ESE ontwikkelt voor Insyte. We zullen de vragen voornamelijk proberen op te lossen door gericht te zoeken op internet. We kunnen hiervoor bronnen gebruiken van Google Scholar of andere bronnen die betrouwbaar lijken. We zullen ook de bronnen gebruiken die door Insyte aan ons zijn geleverd. Deze bronnen zijn onderzoeken die onder andere zijn uitgevoerd door de HVA. Het is echter discutabel of deze bronnen betrouwbaar zijn, dus we zullen kijken naar hun werkwijze.

1.6.1 Uitwerking vragen

1. Wat heeft de grootste invloed op het rottingsproces? De grootste invloed op het rottingsprocess is temperatuur.

a. **Welk effect heeft de omgevingstemperatuur op het rottingsproces?** Volgens onderstaande grafiek neemt de afbraak van planten toe wanneer de temperatuur stijgt. Dit kan worden verklaard doordat veel bacteriën beter functioneren bij hogere temperaturen (broeien). Bij lagere temperaturen wordt het voor deze bacteriën moeilijker om te functioneren, waardoor het rottingsproces van planten vertraagt. Bij 40 graden Fahrenheit (4 graden Celsius) hebben deze groenten ongeveer 400 uur (16 dagen) nodig om het rottingsproces goed te starten. Het is gunstig om de omgevingstemperatuur laag te houden om het rottingsproces te vertragen.

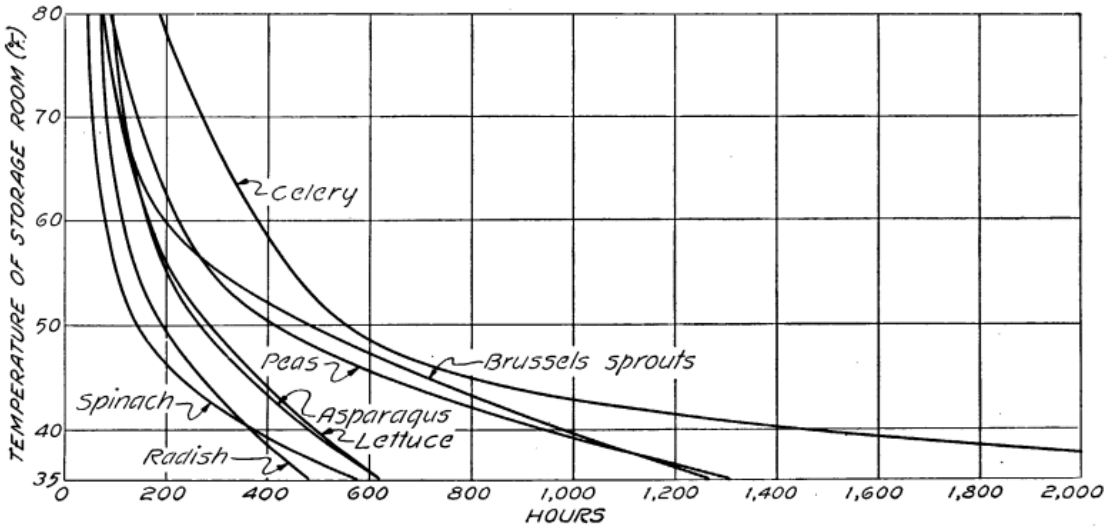


FIGURE 2.—Time-temperature curves based on the temperature of the storage rooms, indicating how long different vegetables can be held at a definite room temperature before complete deterioration occurs.

Figure 1.1 Figuur 2 Tijd-temperatuur bogen op basis van opslag kamer temperatuur(Platenius, 1939)

b. **Welk effect heeft luchtvochtigheid op het rottingsproces?** Bij een hoge luchtvochtigheid (vooral in combinatie met een hoge omgevingstemperatuur) kan er broei ontstaan in een bak, wat leidt tot de vorming van schimmels. Schimmels spelen ook een rol in het rottingsproces en kunnen onaangename geuren veroorzaken. Het is daarom belangrijk om de luchtvochtigheid zo laag mogelijk te houden.

c. **Welk effect heeft zuurstof op het rottingsproces?** Schimmels zijn aerobe organismen, maar zelfs bij zeer lage zuurstofconcentraties is groei mogelijk (Wösten, 2017). Dit betekent dat het gunstig is om een omgeving met weinig zuurstof te creëren, maar het is niet het enige middel om het rottingsproces te voorkomen.

d. **Wat is de maximale tijd waarop het rottingsproces kan worden uitgesteld of vertraagd rekening houdend met de drie eerder genoemde factoren?** Dit is afhankelijk van onder andere de inhoudsgrootte van de vuilnisbak, de staat van het GFE-afval tijdens wegwerpen en het gewenste energielabel in combinatie met de isolatie van de vuilnisbak.

e. **Is het mogelijk om het effect van ozon-generatie te compenseren met koeling (tot en met het vriespunt)?** Op internet hebben we meerdere keren gelezen dat onjuist gebruik van ozongeneratoren kan leiden tot hoge concentraties ozon (O3), wat schadelijk is voor de gezondheid van organismen. Sommige mensen zijn gevoeliger dan anderen en kunnen negatieve gezondheidseffecten ervaren. Mensen die bijzonder kwetsbaar zijn, zijn onder andere kinderen, ouderen en mensen met astma (Government, 2015).

2. Hoeveel liter GFE-afval moet de WasteBin kunnen opslaan? De WasteBin moet zoveel liter GFE-afval kunnen opslaan.

a. **Hoeveel liter GFE-afval produceert een gemiddeld huishouden per dag? (stedelijk)** Vanuit de gemeente Amsterdam is bekend dat er jaarlijks ongeveer tachtig kilo GFE-afval per persoon wordt weggegooid. Dit komt neer op ongeveer 200 gram per dag. Deze cijfers gelden specifiek voor stedelijke gebieden, waar het scheiden van dit afval moeilijker is vanwege de grote hoeveelheid hoogbouw.

Jaarlijks produceert een gemiddeld persoon ongeveer tachtig kilo GFE-afval, maar hiervan belandt ongeveer 70 kilo bij het restafval omdat het niet gescheiden wordt (Schoonvelde, sd).

b. **Wat is het maximale volume van de prullenbak en is dit realistisch?** Het optimale volume van de prullenbak moet afgestemd zijn op de grootte van de prullenbak die huishoudens al gebruiken. Op die manier zal de gebruiker, samen met het andere afval, ook het GFE-afval scheiden en de prullenbak volledig benutten. Volgens verschillende websites (Fonq.nl, brabantia.nl, prullenbak-expert.nl) is een 30-liter prullenbak geschikt voor één persoon, dus we gaan hier in dit geval van uit.

In 2020 produceerde één persoon gemiddeld 140 kilo afval per jaar (totaal). Als we rekening houden met het feit dat er jaarlijks 70 tot 80 kilo GFE-afval wordt geproduceerd (waarvan slechts een deel van ongeveer 10 kilo wordt gescheiden), kunnen we stellen dat het volume van de GFE-afvalbak maximaal de helft zou moeten zijn (Rijksoverheid, sd).

3. Wat weerhoudt veel bewoners van hoogbouwwoningen ervan om hun GFE-afval te scheiden? Er zijn verschillende drempels die veel bewoners van hoogbouwwoningen ervan weerhouden om hun GFE-afval te scheiden.

a. **Welke drempels kunnen realistisch gezien verlaagd worden zodat het scheidingsproces wel wordt uitgevoerd?** Er kunnen verschillende drempels worden verlaagd om het scheiden van GFE-afval te bevorderen. Enkele mogelijke drempels zijn de beschikbaarheid van aparte GFE-afvalbakken in hoogbouwwoningen, onduidelijkheid over wat wel en niet bij GFE-afval hoort en het gebrek aan bewustzijn en educatie over het belang van het scheiden van GFE-afval.

Om deze drempels te verlagen en het scheidingsproces te bevorderen, kunnen de volgende maatregelen worden genomen:

1. **Beschikbaarheid van aparte GFE-afvalbakken:** Het plaatsen van speciale GFE-afvalbakken in hoogbouwwoningen kan het scheiden van GFE-afval gemakkelijker maken. Deze bakken moeten voldoende capaciteit hebben en duidelijk gelabeld zijn.
2. **Duidelijke richtlijnen voor GFE-afval:** Het verstrekken van duidelijke informatie en richtlijnen over wat wel en niet bij GFE-afval hoort, kan verwarring verminderen en bewoners helpen bij het correct scheiden van hun afval. Dit kan worden gedaan door middel van educatief materiaal, zoals brochures, posters of online gidsen.
3. **Bewustmakingscampagnes:** Het uitvoeren van bewustmakingscampagnes gericht op hoogbouwwoningen kan het bewustzijn vergroten over de voordelen van het scheiden van GFE-afval. Deze campagnes kunnen informatie bevatten over de milieueffecten van GFE-afval, tips voor het verminderen van voedselverspilling en de positieve impact van afvalscheiding op het milieu.
4. **Samenwerking met afvalverwerkingsbedrijven:** Samenwerking met afvalverwerkingsbedrijven kan helpen bij het opzetten van efficiënte systemen voor het ophalen en verwerken van GFE-afval uit hoogbouwwoningen. Door het bieden van gemakkelijke en betrouwbare opties voor afvalverwijdering, kunnen bewoners worden aangemoedigd om actief deel te nemen aan het scheidingsproces.

Door deze maatregelen te implementeren, kunnen de drempels voor het scheiden van GFE-afval in hoogbouwwoningen worden verlaagd en kunnen bewoners worden gestimuleerd om actief deel te nemen aan het verminderen van voedselverspilling en het bevorderen van duurzaam afvalbeheer.

1.7 Onderzoek naar koelmogelijkheden

Verschillende categorieën, uiteindelijk is alles een warmtepomp. Verschillende categorieën kunnen worden onderscheiden op basis van techniek, volume en temperatuur. Enkele voorbeelden van koeltechnieken zijn:

- Compressor
- Peltier-effect
- Magnetische koeling

Voor ons onderzoek hebben we besloten ons te richten op Peltier-modules. We hebben helaas niet genoeg tijd en geld om alle opties te onderzoeken en de peltier modules zijn bewezen effectief met koelen en betaalbaar. Deze modules hebben de volgende voordelen ten opzichte van andere koeloplossingen:

- Ze zijn verkrijgbaar in verschillende formaten, waaronder compacte formaten.
- Ze zijn relatief goedkoop in vergelijking met andere koeltechnieken.
- Ze werken geluidloos vanwege het solid-state cooling-principe.
- Er zijn geen extra gassen of waterleidingen nodig.

Het is echter belangrijk om het te koelen gebied goed te isoleren. Op deze manier kan de Peltier-module op een efficiënte manier koelen zonder constant op vol vermogen te werken, wat ook warmte genereert vanwege het feit dat het een warmtepomp is.

1.8 Wat weerhoudt veel hoogbouwwoningbewoners ervan om hun GFE-afval te scheiden?

Uit het onderzoek "Vuilnis in de flat" van de Design Innovation Group kunnen we zien hoe de meeste mensen (22 in totaal) hun keuzes maken met betrekking tot afvalscheiding.

Op basis van dit onderzoek kunnen we concluderen dat voornamelijk de geur de uiteindelijke oorzaak lijkt te zijn voor het niet scheiden van GFE-afval. Om ervoor te zorgen dat dit scheidingsproces wel wordt uitgevoerd, kunnen we realistisch gezien de volgende drempels verlagen:

- We kunnen proberen de geur van het rottingsproces zoveel mogelijk te verminderen of te stoppen. Dit kan worden bereikt door het vertragen of stoppen van het rottingsproces, of door het verwijderen van de geuren die vrijkomen bij het rottingsproces.
- De specifieke aanpak om dit te bereiken moet nog worden vastgesteld.

"In diverse wijken in Utrecht wordt het GFT opgehaald. Eén geïnterviewde was in haar straat de enige die de GFT-bak buiten zet. Vaak werd die bak dan op de route vergeten. Twee geïnterviewden scheidden GFT maar zijn afgehaakt; de stank, vliegjes en natte zooi in huis werden als redenen gegeven." (designinnovationgroup, 2015)

Bibliografie

- Platenius, H. (1939). Effect of temperature on the rate of deterioration of fresh vegetables. Washinton, D.C.: Journal of Agricultural Research.
- Rijksoverheid. (sd). Opgehaald van Rijksoverheid.nl: https://www.rijksoverheid.↩ nl/onderwerpen/afval/huishoudelijk-afval
- RIVM. (sd). Smog Door Ozon. Opgehaald van RIVM.nl: https://www.rivm.nl/smog/smog-door-ozon
- Schoonvelde, G. (sd). Infomil.nl. Opgehaald van Infomil.nl: [https://www.infomil.↩ nl/actueel/nieuws-perspectief-1/verbetering-afvalscheiding-hoogbouw-mits/](https://www.infomil.nl/actueel/n

Chapter 2

WasteBin Project: Het creëren van een innovatieve oplossing voor geurvrije en milieuvriendelijke afvalverwerking

2.1 Inleiding

Het WasteBin Project heeft tot doel het probleem van onaangename geuren in kleine leefruimtes, zoals appartementen, aan te pakken, terwijl het efficiënte afvalscheiding en milieubehoud bevordert. Door de ontwikkeling van een uniek apparaat voorkomt dit project niet alleen onaangename geuren, maar stimuleert het ook gebruikers om verschillende soorten afval te scheiden, wat leidt tot effectievere recycling en een verminderde milieubelasting.

Op dit moment wordt afval helaas niet goed gescheden en uit onderzoek van de HvA blijkt dat hier een aantal redenen voor zijn.

Figure 2.1 redenen om afval niet te scheiden

en hiervoorafgaand is ook intrinsieke motivatie een factor. Hoe groter het gemak van scheiden, hoe groter de kans dat dit daadwerkelijk gedaan word.

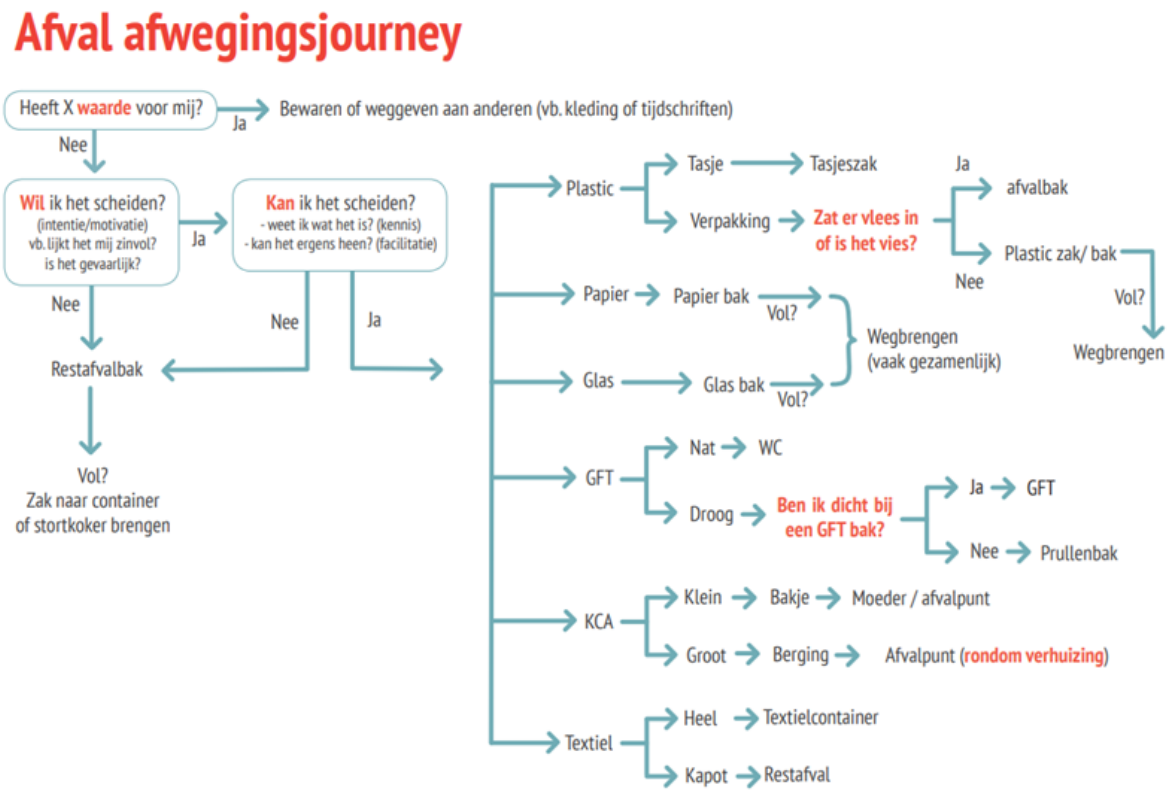


Figure 2.2 afval afwegingjourney

Om deze motivatie te vergroten hebben wij daarom de volgende doelen opgesteld.

2.2 Belangrijkste doelstellingen

1. Geurpreventie: Het primaire doel van het WasteBin Project is het creëren van een apparaat dat onaangename geuren die uit afvalbakken komen effectief opvangt en neutraliseert. Dit zal de leefomgeving aanzienlijk verbeteren voor mensen die in kleine ruimtes wonen.
2. Afvalscheiding: Door indirect het weggooien van algemeen voedselafval (GFE-afval) in reguliere afvalbakken te ontmoedigen, stimuleert het project de scheiding van verschillende soorten afval. Dit moedigt gebruikers aan om milieuvriendelijke praktijken te omarmen, wat resulteert in gemakkelijkere sortering van recyclebaar materiaal en een vermindering van afval op de vuilnisbelt.
3. Milieueffect: Het WasteBin Project draagt bij aan milieubehoud door de hoeveelheid afval die op de vuilnisbelt terechtkomt te minimaliseren. Door een betere scheiding van afval kunnen waardevolle recyclebare materialen gemakkelijker worden teruggewonnen, wat de winning van ruwe materialen vermindert en natuurlijke hulpbronnen beschermt.

2.3 Kenmerken en functionaliteit van het apparaat

Het WasteBin-apparaat heeft verschillende innovatieve kenmerken om zijn doelstellingen te bereiken:

1. Mechanisme voor geurbeheersing: Het apparaat maakt gebruik van verschillende technologieën voor geurbeheersing, zoals temperatuurregeling, geactiveerde koolstoffilters en deodorizers, om onaangename geuren die uit afvalbakken komen vast te leggen en te neutraliseren. Dit zorgt voor een frisse en geurvrije leefomgeving voor gebruikers.

2. Hulp bij afvalscheiding: Het apparaat bevat een intuïtieve interface die gebruikers informeert over afvalscheidingspraktijken. Het biedt visuele indicatoren en herinneringen voor het scheiden van recyclebaar materiaal, organisch afval en andere categorieën, waardoor het proces gemakkelijk en gebruiksvriendelijk wordt.

3. Gebruiksvriendelijk ontwerp: Het WasteBin-apparaat is ontworpen om compact, esthetisch aantrekkelijk en gemakkelijk te gebruiken te zijn. Het past naadloos in kleine leefruimtes en vormt een aanvulling op het algehele interieurontwerp. De verwijderbare compartiment van het apparaat vereenvoudigd het afvalbeheer en de schoonmaakproces.

2.4 Voordelen en impact

Het WasteBin Project biedt talrijke voordelen en positieve effecten:

1. Verbeterde levenskwaliteit: Door onaangename geuren te elimineren, verbetert het apparaat aanzienlijk de leefomgeving voor mensen die in kleine ruimtes wonen. Het bevordert comfort, welzijn en een aangename sfeer.

2. Verbeterde afvalscheiding: Het project moedigt gebruikers effectief aan om verschillende soorten afval te scheiden, wat leidt tot een hoger recyclingpercentage en een verminderde milieubelasting. Dit draagt bij aan een duurzamer afvalbeheersysteem.

3. Bewustwording en educatie: Het WasteBin-apparaat fungeert als een educatief hulpmiddel en vergroot het bewustzijn over het belang van afvalscheiding en de positieve milieu-impact ervan. Door verantwoord afvalbeheer te bevorderen, geeft het project gebruikers de mogelijkheid om milieuvriendelijke keuzes te maken.

4. Langetermijnduurzaamheid: Door het WasteBin Project wordt de adoptie van duurzame afvalbeheerpraktijken mainstream, wat een cultuur van milieubewustzijn bevordert en blijvende positieve veranderingen in afvalverwerkingsgewoonten teweegbrengt.

2.5 Conclusie

Het WasteBin Project streeft ernaar een innovatief apparaat te creëren dat niet alleen onaangename geuren in kleine leefruimtes elimineert, maar ook afvalscheiding en milieuvriendelijkheid bevordert. Door geavanceerde mechanismen voor geurbeheersing, hulp bij afvalscheiding en gebruiksvriendelijke kenmerken te integreren, stelt het apparaat individuen in staat om actief een rol te spelen in het behoud van het milieu en tegelijkertijd hun levenskwaliteit te verbeteren. Met dit project streven we ernaar bij te dragen aan een schonere, groenere en meer duurzame toekomst voor iedereen.

Chapter 3

Het Gebruik van meerdere ESP32-Microcontrollers met Node-Red

3.1 Inleiding

Meerdere ESP32-microcontrollers kunnen in combinatie met Node-Red worden gebruikt om een visuele interface te bouwen op basis van de gegevens die zijn gelogd door de ESP32. Elke ESP32 kan worden geconfigureerd om gegevens van een specifieke sensor of apparaat te loggen en deze via Wi-Fi naar Node-Red te verzenden. Node-Red kan vervolgens de gegevens ontvangen en verwerken met behulp van verschillende nodes.

3.2 Doelen

3.2.1 Doel #1

Het opzetten van een systeem waarbij meerdere ESP32-microcontrollers worden gebruikt om gegevens te loggen en deze naar Node-Red te verzenden voor visualisatie en verwerking.

3.3 Onderbouwing keuze ESP32 C3

Wij kiezen voor een ESP32 omdat deze de volgende voordelen biedt:

- Relatief goedkoop (€2 per stuk).
- Beschikbaarheid (1300 op voorraad bij Mouser, afhankelijk van het model).
- Geschikt voor de taken die moeten worden uitgevoerd.
- Compact formaat.

3.4 Waarom geen Arduino of Raspberry Pi als computer voor de WasteBin?

Raspberry Pi's zijn momenteel moeilijk verkrijgbaar, duur in aanschaf voor een testplatform en vereisen meer inspanning om op te zetten. Bovendien zijn ze vaak te krachtig voor onze doeleinden en daardoor overbodig qua rekenkracht. Arduino's zijn redelijke alternatieven, maar duurder dan losse ESP32-modules. Desalniettemin kunnen ESP32-modules via de Arduino IDE worden geprogrammeerd met behulp van de Arduino-codebibliotheken, indien gewenst.

3.5 Data forwarding

Data forwarding is de plaats waar alle data van de ESP32's (of vuilnisbakken) wordt verzameld. Een mogelijke optie is het gebruik van specifieke hardware, zoals een Raspberry Pi met een broker (zoals Mosquitto) en een Node-↩ Red-dashboard. Wij kiezen ervoor om de broker en het dashboard op onze lokale laptop te installeren in plaats van een Raspberry Pi te gebruiken. We hebben hiervoor meerdere redenen:

1. We vermijden de noodzaak van een extra apparaat (Raspberry Pi). Alles op één apparaat maakt het eenvoudiger. Bovendien zijn Raspberry Pi's moeilijk verkrijgbaar en vormen ze een extra bron van potentiële hardwarefouten (zoals SD-kaartcorruptie) en softwareproblemen (Linux-problemen) enzovoort.
2. Als we fysiek ergens anders willen werken, zouden we steeds de Raspberry Pi moeten verbinden met het netwerk op die locatie via een beeldscherm en toetsenbord.
3. De Raspberry Pi kan een beveiligingsrisico vormen.

Stappen:

1. ESP32-gegevens laten versturen (verbinding maken met internet).
2. Opzetten van een MQTT-broker.
3. De MQTT-broker accepteert gegevens van de ESP32.
4. De MQTT-broker stuurt gegevens door naar Node-Red (lokaal).
5. Inkomende gegevens van de MQTT-broker worden omgezet naar een dashboardweergave.
6. Data van Node-Red doorsturen naar een database.

3.6 De Data

De volgende gegevens willen we versturen vanaf de ESP32:

- Temperatuur (dubbel)
- Luchtvochtigheid (dubbel)
- Klepstand (boolean)
- Foutcodes (integer)
- WasteBin-ID (macadres)

Deze gegevens worden via MQTT naar Node-Red verzonden. Van daaruit worden de gegevens doorgestuurd naar een SQLite-database. We hebben gekozen voor SQLite vanwege de handige SQLiteStudio-omgeving waarin we de database gemakkelijk kunnen ontwerpen, opbouwen en testen.

3.7 Meldingen voor de gebruiker

Voor onze opstelling willen we een luidspreker gebruiken als indicator die verschillende waarschuwingen kan afgeven wanneer dat nodig is. Deze waarschuwingen zijn onder andere:

- De klep staat te lang open (mogelijk)
- De binnenkant van de bak is te warm (mogelijk klep open of slechte koeling)
- Algemene fout (ander probleem)

We hebben ervoor gekozen om deze waarschuwingen ook naar ons dashboard te sturen als ze zich voordoen. Deze waarschuwingen worden weergegeven als foutcodes.

3.8 Verstuur interval

In principe worden alle gegevens elke 15 minuten verstuurd. We denken dat dit voldoende informatie is voor elke vuilnisbak om eventuele problemen te detecteren (bijvoorbeeld te lage temperatuur). Bovendien voorkomt dit dat de database overvol raakt met gegevens, vooral wanneer er 30 vuilnisbakken zijn.

3.8.1 In geval van fouten

In geval van een fout wordt dit interval genegeerd. Het interval wordt dan 2 minuten, indien er zich één of meerdere fouten blijven voordoen.

3.9 Node-Red

We zullen Node-Red gebruiken om alle gegevens van MQTT te ontvangen, te verwerken en weer te geven. Alle ontvangen informatie wordt weergegeven via een Node-Red UI-tabblad. Dit UI-tabblad, ook wel "dashboard" genoemd in Node-Red, is uitbreidbaar indien gewenst. Alle prototype vuilnisbakken kunnen in real-time worden weergegeven op het dashboard als dat gewenst is.

De Node-Red UI ziet er als volgt uit voor twee vuilnisbakken:

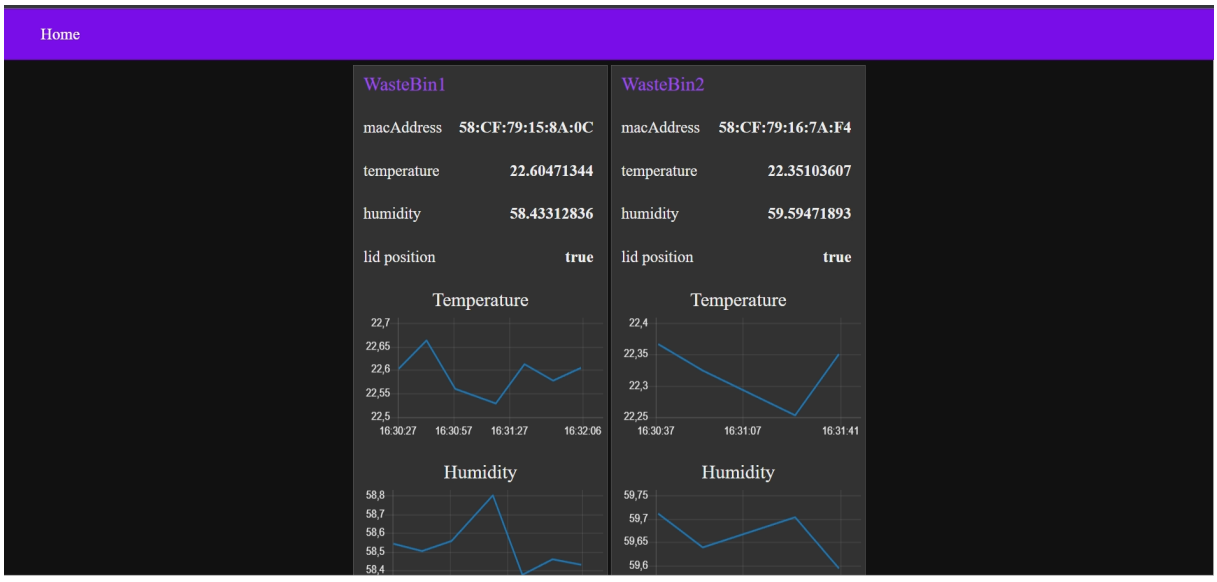


Figure 3.1 Node-Red UI

3.9.1 Node-Red flow

Om dit alles te laten werken, hebben we een Node-Red flow gemaakt. De Node-Red flow ziet er als volgt uit:

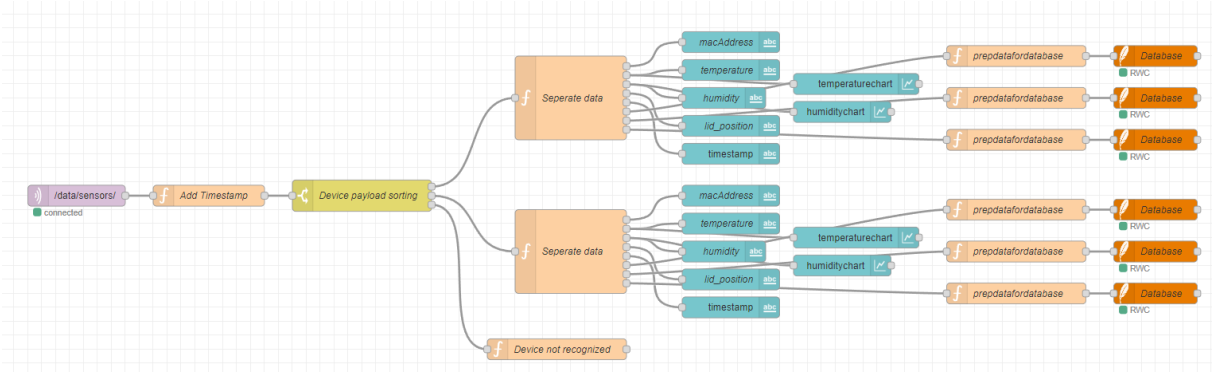


Figure 3.2 Node-Red setup

In deze flow ontvangt de eerste node (paars) de MQTT-gegevens. Vervolgens wordt er een tijdstempel aan het bericht toegevoegd en worden alle inkomende berichten gescheiden met behulp van een schakelaar (switch). De gegevens worden gescheiden op basis van het macadres (uniek) van de vuilnisbak.

Nadat de berichten zijn gescheiden, komen ze in een grote functieblok (oranje) dat het bericht analyseert en verdeelt over de uit puts van het functieblok. Ik heb hiervoor gekozen, zodat het functieblok modulair blijft en de uitvoer flexibel is. De uitvoer wordt gestuurd naar dashboardblokken (blauw) en naar nieuwe functieblokken die extra gegevens toevoegen (zoals sensor-ID) en deze als query naar een SQLite-database sturen.

3.10 Database

Alle informatie wordt via Node-Red doorgestuurd naar een SQLite-database. Het doel van de database is om een doorzoekbare geschiedenis van alle ontvangen gegevens van de vuilnisbakken te creëren en een overzicht te bieden van alle gebruikers (testpersonen) en hun gegevens.

De database is als volgt opgebouwd:

Hierbij zijn alle onderstreepte woorden sleutelwaarden. Helaas toont het Database Entity Relationship-diagram (DBER) niet alle details, maar het geeft wel een goed overzicht van de databasestructuur.

De tabellen en relaties in de database zijn als volgt:

Tabelnaam	Veldnaam	Type	Relatie
Users	UserID	INTEGER	PRIMARY KEY
	Name	TEXT	
	Email	TEXT	
	Phone	TEXT	
WasteBins	WasteBinID	INTEGER	PRIMARY KEY
	Location	TEXT	
	SensorID	INTEGER	
	UserID	INTEGER	FOREIGN KEY
SensorData	SensorDataID	INTEGER	PRIMARY KEY
	WasteBinID	INTEGER	FOREIGN KEY
	Timestamp	TIMESTAMP	

Tabelnaam	Veldnaam	Type	Relatie
	Temperature	REAL	
	Humidity	REAL	
	ValveStatus	INTEGER	
	ErrorCode	INTEGER	
SensorStatus	SensorID	INTEGER	PRIMARY KEY
	Status	TEXT	
	LastUpdated	TIMESTAMP	

Dit is de tabelstructuur voor de database, waarbij elke tabel de bijbehorende velden en datatypes bevat. De tabelrelaties zijn aangegeven met "PRIMARY KEY" en "FOREIGN KEY" om de verbanden tussen de tabellen weer te geven.

De database kan worden doorzocht met behulp van "views" in SQLiteStudio. Deze kunnen door de gebruiker worden gemaakt en aangepast.

Dit is een overzicht van de opzet voor het gebruik van meerdere ESP32-microcontrollers met Node-Red. Met deze configuratie kunnen we gegevens loggen van verschillende sensoren of apparaten, deze visualiseren op een dashboard en ze opslaan in een database voor verdere analyse en beheer.

3.10.1 Views

Er zijn 3 views beschikbaar gemaakt voor de database.

Deze views zijn gemaakt om een sneller en beter inzicht te krijgen in bepaalde data combinaties.

Hier is een Markdown representatie van de drie views:

3.10.2 View 1 : view_al_testpersons_with_wastebinnr

```
SELECT tp.*,
       wb.wastebinnr
  FROM testperson AS tp
       JOIN
       wastebin AS wb ON tp.idnr = wb.testperson
```

Deze query selecteert alle kolommen (*) uit de `testperson`-tabel en voegt de kolom `wastebinnr` uit de `wastebin`-tabel toe. De JOIN-operatie wordt uitgevoerd op basis van de overeenkomstige `idnr`-kolommen tussen de `testperson`- en `wastebin`-tabellen.

3.10.3 View 2: view_all_sensors_data

```
SELECT s.sensor_type,
       CASE WHEN sensor_type = "lidposition" AND
                 sd.sensor_value = 1 THEN 'closed' WHEN sensor_type = "lidposition" AND
                                                        sd.sensor_value = 0 THEN 'open' ELSE sd.sensor_value
      END AS sensorvalue,
       sd.timestamp,
       sd.mac_address
  FROM sensor AS s
       JOIN
       sensordata AS sd ON s.sensor_idnr = sd.sensor_idnr
```

Deze query selecteert de kolommen `sensor_type`, `sensorvalue`, `timestamp` en `mac_address`. De `CASE`-expressie wordt gebruikt om de waarde van `sensorvalue` te bepalen op basis van voorwaarden. De JOIN-operatie wordt uitgevoerd op basis van de overeenkomstige `sensor_idnr`-kolommen tussen de `sensor`- en `sensordata`-tabellen.

3.10.4 View 3: view_all_wastebin_settings

```
SELECT w.wastebinnr,
       w.mac_adress,
       s.SSID,
       s.password,
       s.mqtt_server,
       s.mqtt_username,
       s.mqtt_password,
       s.mqtt_port,
       s.sensor_data_topic,
       w.testperson
  FROM wastebin AS w
       JOIN
       settings AS s ON w.mac_adress = s.mac_adress
```

Deze query selecteert verschillende kolommen uit de `wastebin`- en `settings`-tabellen. De JOIN-operatie wordt uitgevoerd op basis van de overeenkomstige `mac_address`-kolommen tussen de twee tabellen.

De bovenstaande weergave is bedoeld om de query's in een leesbare vorm te presenteren.

Er word aangemoedigd om de views uit te breiden of aan te passen naar wens van de gebruiker.

Chapter 4

Handleiding voor het opzetten van de Mosquitto Broker

Deze handleiding biedt begeleiding bij het instellen en gebruiken van de Mosquitto broker met het meegeleverde Arduino-script. De Mosquitto broker maakt communicatie mogelijk tussen het Arduino-apparaat en andere MQTT-clients. Volg de onderstaande instructies om de benodigde instellingen te configureren voor succesvolle communicatie.

4.1 Vereisten

- Arduino-bord (bijv. ESP32)
- Arduino IDE (Integrated Development Environment)
- Geïnstalleerde en actieve Mosquitto broker

4.2 Configuratie van het Arduino-script

Open het Arduino-script (`.ino`-bestand) in de Arduino IDE en zoek het volgende gedeelte:

```
// WiFi-instellingen
const char* SSID = "WWegvanons3";
const char* PASSWORD = "JuCasSan27@#";
// MQTT Broker-instellingen
const char* MQTT_SERVER = "192.168.111.237";
const char* MQTT_USERNAME = "mqtt";
const char* MQTT_PASSWORD = "WasteBin5#";
const int MQTT_PORT = 1883;
```

1. **WiFi-instellingen**: Werk de `SSID`- en `PASSWORD`-variabelen bij met de gegevens van jouw WiFi-netwerk.
2. **MQTT Broker-instellingen**: Wijzig de `MQTT_SERVER`, `MQTT_USERNAME`, `MQTT_PASSWORD` en `MQTT_PORT`-variabelen om overeen te komen met jouw Mosquitto broker-configuratie. De `MQTT_SERVER` moet het IP-adres of de hostnaam van jouw MQTT broker bevatten.

4.3 Verbinding maken met de Mosquitto Broker

Om het Arduino-apparaat met de Mosquitto broker te verbinden, zorg ervoor dat de broker actief is en toegankelijk is vanaf het netwerk van de Arduino. Volg deze stappen:

1. **WiFi-verbinding instellen**: Het Arduino-script bevat een `setupWiFi()`-functie die de verbinding met jouw WiFi-netwerk afhandelt. Het maakt automatisch verbinding met behulp van de opgegeven `SSID` en `PASSWORD`. Zorg ervoor dat het Arduino-apparaat binnen het bereik van jouw WiFi-netwerk bevindt.
2. **MQTT-client configureren**: Het script heeft een `setupMQTTClient()`-functie die de MQTT-client configureert met de opgegeven broker-instellingen. Controleer of de `MQTT_SERVER`, `MQTT_USERNAME`, `MQTT_PASSWORD` en `MQTT_PORT`-variabelen overeenkomen met jouw Mosquitto broker-configuratie.
3. **Uploaden en uitvoeren**: Upload het aangepaste Arduino-script naar jouw Arduino-bord en bekijk de Seriële Monitor voor de verbindingsstatus en eventuele foutmeldingen.
4. **Verbinding verifiëren**: Nadat het script succesvol is geüpload, zou het Arduino-apparaat proberen verbinding te maken met de Mosquitto broker. Controleer de Seriële Monitor op het bericht " WiFi connected" gevolgd door het toegewezen IP-adres van de Arduino. Als de verbinding mislukt, zorg er dan voor dat de broker-instellingen en WiFi-inloggegevens correct zijn.

4.4 Interactie met de Mosquitto Broker

Het Arduino-script voert de volgende taken uit:

- Leest sensordata uit een SHT4x-sensor.
- Update de positie van de deksel op basis van de status van een knop.
- Bestuurt een Peltier-module op basis van temperatuurfout.
- Publiceert sensordata naar de MQTT-broker.

Om te communiceren met de Mosquitto broker en gegevens te ontvangen die door de Arduino zijn gepubliceerd, heb je een MQTT-client nodig, zoals MQTT.fx, MQTT Explorer of een aangepaste toepassing.

1. **Inschrijven op sensordata**: Abonneer je in jouw MQTT-client op het onderwerp dat is gespecificeerd in de `SENSOR_DATA_TOPIC`-variabele ("/data/sensors/"). Hiermee kun je sensordata ontvangen die door de Arduino is gepubliceerd.
2. **Berichten publiceren om Arduino te besturen**: Om berichten naar het Arduino-apparaat te sturen en het gedrag ervan te besturen, publiceer je berichten naar specifieke onderwerpen die het Arduino-script kan verwerken. Raadpleeg de code van het script voor beschikbare onderwerpen en hun overeenkomstige acties.
3. **MQTT-berichten controleren**: Bekijk in jouw MQTT-client de inkomende berichten van de Arduino op het geabonneerde onderwerp. De Arduino zal periodiek sensordata publiceren op basis van de `msg_interval`-variabele.

4.5 Conclusie

Door de instructies in deze handleiding te volgen, kun je de Mosquitto broker en het Arduino-script configureren om communicatie tot stand te brengen tussen jouw Arduino-apparaat en MQTT-clients.

Chapter 5

Ontwerpkeuzes

In dit document worden de functionele en technische specificaties besproken voor het ontwerp van het product. De functionele specificaties geven een overzicht van de gewenste functionaliteiten, terwijl de technische specificaties de gebruikte technologieën en componenten beschrijven.

5.1 Functionele specificaties

#	MoSCoW	Beschrijving
F1	M	Het product moet de verspreiding van de GFE-afvalgeur voorkomen.
F1	M	Het GFE-rottingsproces moet vier dagen worden vertraagd door het product.
F1.1	M	Het GFE-afval moet gekoeld worden.
F1.↩ 1.1	S	Het product moet zo efficiënt mogelijk de bak gekoeld krijgen en behouden.
F2	M	De prijs moet onder de €50 blijven voor de consument bij afname van 100 stuks.
F3	S	Het product moet gebruikersindicatoren hebben voor te lang openstaan.
F3.1	S	Het product moet een temperatuurindicator (te warm/te koud) hebben.
F3.2	C	Het product moet een inhoudshoeveelheidindicator (vol/leeg) hebben.
F4	M	Het product prototype moet draadloze gegevensoverdracht hebben voor testdoeleinden.
F5	C	Het product moet feedback aan de gebruiker geven.
F5.1	C	Het product moet feedback geven via audio.
F5.2	C	Het product moet feedback geven via licht/beeld.
F6	W	Het product moet een "off the grid" functie hebben.
F7	S	Het product moet simpel en intuïtief zijn in gebruik.
F8	M	De prullenbakinhoud moet 6 liter zijn, afgestemd op het gemiddelde GFE-afval van een 4-persoonshuishouden.
F9	S	Het product moet zo stil mogelijk zijn tijdens gebruik.
F10	M	Het product moet de klepstand kunnen bepalen.
F11	S	Het product moet zou energie moeten "terugwinnen" (door GFE-preservatie).
F12	C	Het product (prototype) moet een logfunctie hebben voor diagnose.

5.2 Technische specificaties

#	Beschrijving
1	Het koelen van het product wordt gedaan met behulp van een Peltier-module.
2	Het product gebruikt een Piezo-speaker om de gebruiker te waarschuwen wanneer de klep te lang openstaat.
3	De klepstand wordt gemeten d.m.v. een Hall-effectsensor of een reed switch.
4	Bij logfuncties in het eindproduct is een externe EEPROM een optie.
5	Het product krijgt een externe voeding om de repareerbaarheid te verbeteren en de complexiteit te verminderen.
6	Het prototype gebruik de ESP32 C3 Devkit module als processor en IoT module. Voor het eindproduct raden wij een Microchip ATTINY 13A aan i.v.m. de adequaatheid, lage kosten en de beschikbaarheid van de chip.
7	De SHT40 wordt gebruikt om de temperatuur en luchtvochtigheid in het systeem te meten. Er bestaan goedkopere sensoren zoals de LM35, DHT22 of LM75 (communiceert via I2C). Deze goedkopere alternatieven of derden zijn te overwegen voor het eindproduct.
8	We ontwerpen onze eigen buck converter voor de ESP32 C3 Devkit module.

Chapter 6

PCB-ontwerp

Voorlopig prototype:

Voor het voorlopige prototype maken we gebruik van twee specifieke PCB's: één voor de conversie en één voor het eindproduct. De "IoT" PCB bevat alle functionele componenten die het prototype onderscheiden van het uiteindelijke product. De overige componenten bevinden zich op de "eindproduct" PCB.

Voor het prototype moeten we de volgende gegevens kunnen uitlezen:

1. Luchtvochtigheid
2. Temperatuur
3. Klepstand (indien van toepassing)

Voor het eindproduct moeten we de volgende gegevens kunnen uitlezen:

1. Temperatuur (in graden Celsius)
2. Klepstand (open/gesloten)

Discussie prototype:

Optie 1: Gebruik van één specifieke PCB voor het prototype en één specifieke PCB voor het eindproduct heeft de volgende voordelen:

- Prototypes zijn vaak duurder. Door te besparen op productiekosten van 30 vuilnisbakken, kunnen we veel ontwerptijd besparen. Het gebruik van een wat eenvoudigere printplaat met duurdere componenten kan uiteindelijk net zo kosteneffectief zijn.
- Het eindproduct heeft geen IoT-functionaliteit, dus een nieuw en compacter PCB-ontwerp is zinvol bij grootschalige productie.

Optie 2: Een alternatieve benadering is om een coöperatieve PCB te gebruiken. Hierbij wordt één hoofd-PCB met IoT-functionaliteit fysiek verbonden met de PCB van het eindproduct. Het IoT PCB wordt verwijderd tijdens de productie. De redenen hiervoor zijn als volgt:

- Het hergebruiken van de 30 prototype-printplaten voor het uiteindelijke ontwerp kan kosteneffectiever zijn voor de prototypes.
- Het tegenargument is dat dit punt bij een grootschalige productie van bijvoorbeeld 1000 vuilnisbakken bijna verwaarloosbaar kan worden en daarmee het argument niet meer van toepassing is.
- Het kan zo zijn dat het ontdekken van fouten in de hardware makkelijker is als de printplaten los van elkaar te koppelen zijn.

Keuze voor printplaten:

Na een gesprek met de opdrachtgever is het akkoord gegeven om vanuit optie 1 te werk te gaan. Het prototype printplaat word ontworpen op de ESP32 C3 Devkit module. Deze module is gemakkelijk op de printplaat te schuiven door middel van de 2 headers. De printplaat is volledig afgestemd op het prototype en word ontworpen als een proof of concept voor het uiteindelijke resultaat. De printplaat voor het eindproduct zal immers meerdere testfases heen moeten voordat deze in productie kan. Vandaar dat wij ons hier nu niet op focussen.

6.1 PCB V0.1 Ontwerp

Het ontwerp van de eerste printplaatversie (V0.1) is een bijna volledige kopie van het prototype met enkele wijzigingen. De buck converter is vervangen door een eenvoudige LDO omdat de buck converter overgespecificeerd was. Bovendien zijn er headers toegevoegd voor extra voeding en GPIO pinnen, waardoor het prototype flexibel blijft en mogelijke tests kunnen worden uitgevoerd, zelfs als deze printplaat die functies nog niet heeft. Het schema van deze printplaat ziet er als volgt uit:

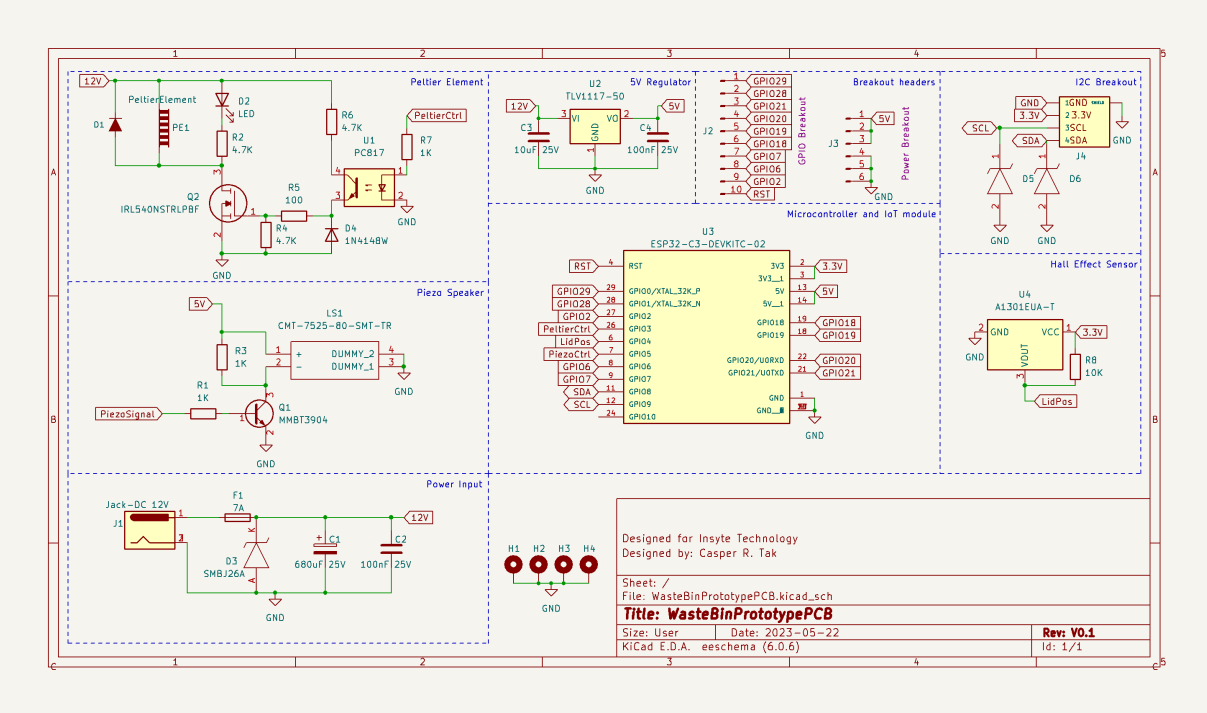


Figure 6.1 schematicv0.1

Hierbij hoort ook een 3D-ontwerp dat als .step-bestand wordt geëxporteerd naar de IPO-studenten. Bij het indelen van de componenten heb ik groepen gemaakt van alle modules. De module met hoge stroom, de Peltier en het bijbehorende schakelmateriaal bevinden zich dicht bij de voedingsjack om te voorkomen dat er hoge stroom door het hele bord loopt en datalijnen mogelijk worden verstoord. Dit maakt het ook gemakkelijker om verbindingen

te leggen. Daarnaast heb ik ervoor gezorgd dat alles zeer toegankelijk en gemakkelijk te solderen is, voor zover mogelijk. Tot slot heb ik de silkscreen geplaatst bij alle pinnen, zodat het gemakkelijker is om te zien waar welke pin zich bevindt, zonder telkens het schema of het printplaatontwerp te raadplegen. De silkscreen helpt ook bij het minimaliseren van mogelijke foutieve aansluitingen door de gebruiker.

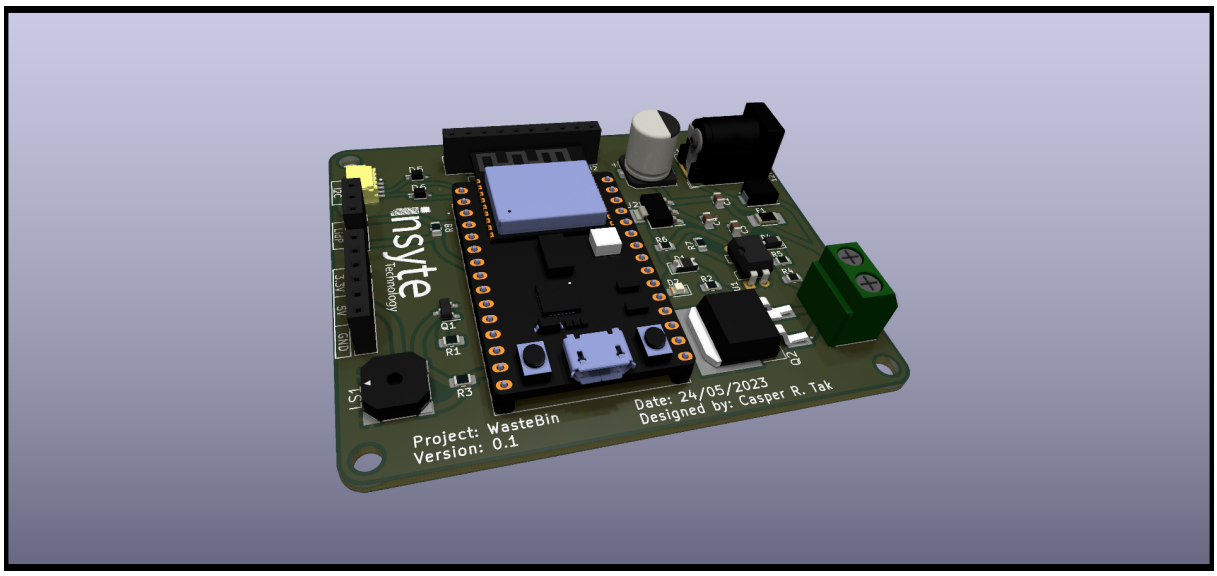


Figure 6.2 3ddesignv0.1

Chapter 7

Testplan

7.1 Inhoud

- Peltier modules
- Doelen
- Deksel positie
- Temperatuur en luchtvochtigheid
- Testopstelling
- Vereiste componenten
- Meetapparatuur
- Methodes
- Testresultaten

7.2 Peltier modules

De Peltier modules worden verder bestudeerd en getest om te kijken of deze geschikt zijn om te gebruiken voor onze koeling. We gaan gebruik maken van een 12V module. Dit omdat het de standaard is voor Peltier-modules en omdat de spanning afhankelijk is van het formaat vermoeden wij dat een 5V module niet genoeg warmte kan verplaatsen voor onze bak.

7.3 Doelen

Met dit onderzoek willen wij de volgende kwesties benaderen en uitwerken:

- Kunnen we het volledige systeem op 12V laten werken door een buck-converter te gebruiken voor componenten die 5V nodig hebben?
- Wat is de spanning die de ventilatoren vereisen en kunnen we dit regelen om zo een stilleren functionaliteit te krijgen? Hoeveel effect heeft dit op het koelen van de Peltier-module(s)?

7.4 Deksel positie

Het meten van de dekselpositie zullen wij doen d.m.v. een drukknopje of een hall effect sensor in combinatie met een magneet. De werking ervan hoeft naar onze mening niet getest te worden. Een knop is te simpel om een serieuze test voor te verzinnen en daarom gaan we voor de aanpak: krijgen we een waarde te zien wanneer we de knop indrukken? Dan werkt de knop.

7.5 Temperatuur en luchtvochtigheid

Omdat wij in de testfase informatie willen verzamelen van onze testopstelling, willen wij onder andere temperatuurmetingen doen om inzicht te krijgen in de werking en stabiliteit van de testopstelling.

7.5.1 Doel #1

Wij willen voor metingen weten wat de temperatuur en de luchtvochtigheid is in de ruimte waarin het GFE-afval zich bevindt. Hiervoor moeten wij:

- Een 2-in-1 sensor (temperatuur en luchtvochtigheid) kiezen en deze verifiëren op werking en accuraatheid (sensoren los testen).
- De sensor installeren in een gecontroleerde opstelling en de resultaten noteren.

7.5.2 Doel #2

Verder zijn wij benieuwd naar de verdeling van de koelmodules over de bak. De temperatuur kunnen we meten door middel van een thermische camera. Met zo'n camera kunnen wij de temperatuurgeleiding en -verspreiding door de bak zien. Het doel hiervan is om te bepalen waar we de Peltier-module het beste kunnen plaatsen.

7.5.3 Doel #3

Kijken of we met behulp van PWM een ventilator op een deel van zijn toeren kunnen

laat draaien om de temperatuur te regelen, maar vooral om te kijken of we de ventilator stiller kunnen krijgen zonder al te veel invloed op de temperatuur te hebben.

7.6 Testopstelling

Er wordt een uitlijning in het deksel gesneden voor de Peltier-module. Deze wordt met de "hete" kant boven hierin gelegd, zodat de heatsink boven zit. Met een thermometer meten wij de temperatuur in de bak, hiervoor wordt een gat gemaakt. De thermometer wordt aangesloten op een Raspberry Pi en hierin wordt de data gelogd en weergegeven. We meten de temperatuur in de tijd en vergelijken dit onder de 12V en 5V modules.

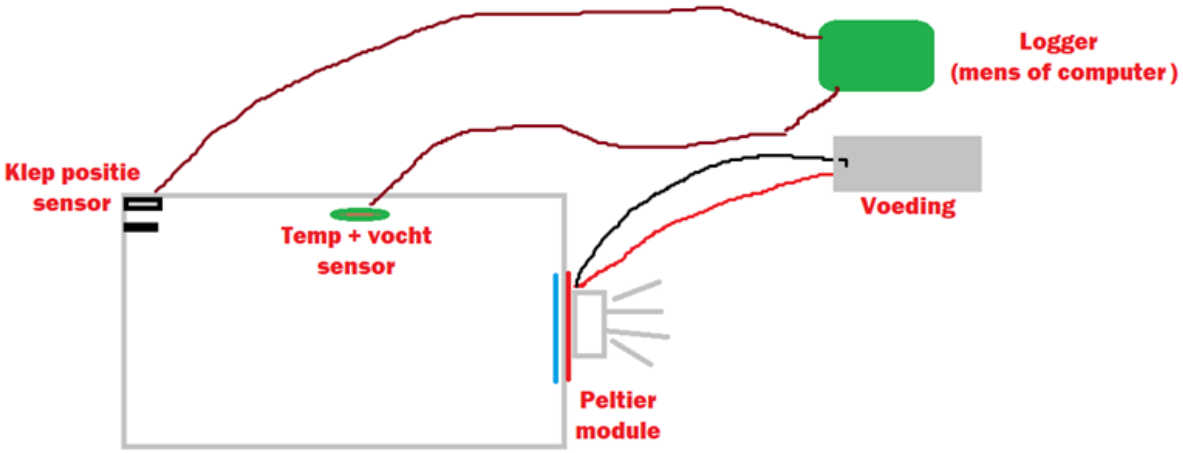


Figure 7.1 Testopstelling

7.7 Vereiste componenten

- Voeding 12V 6A (of labvoeding)
- 2x Peltier modules 12V
- Heatsinks (voor Peltier) met ventilator
- Thermal paste (voor heatsinks)
- Temperatuursensor
- Vochtsensor
- Draden 10AWG, jumper draden (male/female)

7.8 Meetapparatuur

- Multimeter
- Thermometer
- Thermische camera
- Raspberry Pi

7.9 Methodes

Eis	Methode
De temperatuur- en luchtvochtigheidsmeting moet niet meer afwijken dan de aangegeven afwijking.	Meet de temperatuur en luchtvochtigheid met een gekalibreerde meter en vergelijk deze waarden. De eis is voldaan wanneer deze waarden minder afwijken dan de aangegeven afwijking.
De Peltier-modules moeten op hun meest optimale plaats gemonteerd worden.	Door de temperatuurspreiding te meten kunnen wij achterhalen welke plek het beste is voor deze modules.
Aangetoond moet worden hoeveel effect de ventilatoren hebben op de koeling wanneer dit door middel van PWM wordt geregeld.	De koeling wordt gemeten bij een duty cycle van 100%, 87.5%, 75%, 50% en 25%. Hierdoor is duidelijk hoeveel invloed het afvoeren van lucht heeft op de te behalen koelingstemperatuur.

7.10 Testresultaten

Een tijdsopname van de data die door de vuilnisbak is opgeslagen, is in onderstaande tabel te vinden. Hierbij is sensoridnr 1 de temperatuursensor, 2 is de luchtvochtigheid en 3 is de klepstand van de vuilnisbak (knop). Aangezien de vuilnisbak hier nog maar net aanstaat, is de temperatuur nu op kamertemperatuur.

Tijd	Sensoridnr 1 (temperatuur)	Sensoridnr 2 (luchtvochtigheid)	Sensoridnr 3 (klepstand)
00:00:00	20℃	50%	Gesloten
00:05:00	18℃	48%	Open
00:10:00	16℃	45%	Open
00:00:00	20℃	50%	Gesloten
00:05:00	18℃	48%	Open
00:10:00	16℃	45%	Open
00:15:00	14℃	42%	Open
00:20:00	13℃	40%	Open
00:25:00	13℃	39%	Open
00:30:00	13℃	38%	Open
00:35:00	13℃	37%	Open

Later neemt deze temperatuur zeer sterk af tot het ingestelde punt: 13 graden Celsius.

I.v.m. energiekosten en de slechte isolatie van de testopstelling kiezen wij ervoor om de temperatuur niet verder te laten dalen. Wat ons opviel is dat zelfs bij het ingestelde punt de heatsink niet erg warm wordt. Wij vermoeden daarom dat wij de temperatuur nog verder kunnen laten dalen als we dat willen. Helaas staat de ventilator wel voluit te blazen bij het koelen en is het geluidsniveau redelijk hoog. Naar onze mening kan dit worden verholpen op 2 manieren:

1. Verander de huidige ventilator naar een ander, hoger kwaliteitsmodel, met onder andere betere lagers.
2. Vergroot de heatsink of maak de heatsink zelfs een deel van de vuilnisbakbehuizing (metalen vuilnisbak).

Dit zijn de resultaten van het testplan voor de ESE op 8 februari 2023. Met deze resultaten kunnen we beoordelen of de Peltier-modules geschikt zijn voor onze koeling, hoe we de ventilatoren kunnen regelen om het geluidsniveau te verminderen en of we de temperatuur en luchtvochtigheid nauwkeurig kunnen meten in de testopstelling.

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

waste_bin_controller.h 31
waste_bin_controller.ino 38

Chapter 9

File Documentation

9.1 Bureaubladonderzoek.md File Reference

9.2 Introductie.md File Reference

9.3 IoT oplossing.md File Reference

9.4 Mosquittosetup.md File Reference

9.5 Ontwerpkeuzes.md File Reference

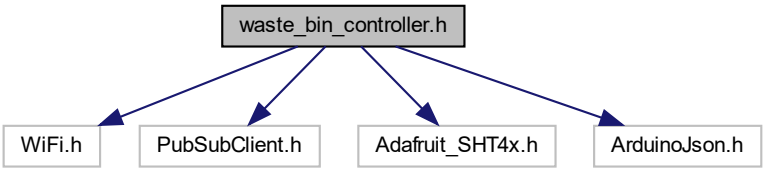
9.6 PCB-design.md File Reference

9.7 Testplan.md File Reference

9.8 waste_bin_controller.h File Reference

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_SHT4x.h>
#include <ArduinoJson.h>
```

Include dependency graph for waste_bin_controller.h:



Functions

- void setupWiFi ()
- void setupPeltierAndFan ()
- void setupSHT4x ()
- void setupMQTTClient ()
- void readSensorData ()
- void updateLidPosition ()
- void controlPeltierModule ()
- void publishSensorData ()
- void handleIncomingMessage (char *topic, byte *payload, unsigned int length)
- void publishMessage (const char *topic, const float temperature, const float humidity, const bool lid_position)
- void setup ()
- void loop ()

Variables

- const char * SSID
- const char * PASSWORD
- const char * MQTT_SERVER
- const char * MQTT_USERNAME
- const char * MQTT_PASSWORD
- const int MQTT_PORT
- const char * CLIENT_NAME
- const char * SENSOR_DATA_TOPIC
- const uint8_t peltierandfanpin
- const long interval
- int peltierandfanstate
- WiFiClient wifi_client
- PubSubClient mqtt_client
- unsigned long last_msg_sent
- const unsigned long msg_interval
- unsigned long previousMillis
- const int setPoint
- const int tolerance
- const int pwmFreq
- const int BUTTON_PIN
- bool lid_position
- float humidity_value
- float temperature_value
- Adafruit_SHT4x sht4

9.8.1 Function Documentation

9.8.1.1 controlPeltierModule()

```
void controlPeltierModule ( )
```

9.8.1.2 handleIncomingMessage()

```
void handleIncomingMessage (
            char * topic,
            byte * payload,
            unsigned int length )
```

9.8.1.3 loop()

```
void loop ( )
```

9.8.1.4 publishMessage()

```
void publishMessage (
            const char * topic,
            const float temperature,
            const float humidity,
            const bool lid_position )
```

9.8.1.5 publishSensorData()

```
void publishSensorData ( )
```

9.8.1.6 readSensorData()

```
void readSensorData ( )
```

9.8.1.7 setup()

```
void setup ( )
```

9.8.1.8 setupMQTTClient()

```
void setupMQTTClient ( )
```

9.8.1.9 setupPeltierAndFan()

```
void setupPeltierAndFan ( )
```

9.8.1.10 setupSHT4x()

```
void setupSHT4x ( )
```

9.8.1.11 setupWiFi()

```
void setupWiFi ( )
```

9.8.1.12 updateLidPosition()

```
void updateLidPosition ( )
```

9.8.2 Variable Documentation

9.8.2.1 BUTTON_PIN

```
const int BUTTON_PIN  [extern]
```

9.8.2.2 CLIENT_NAME

```
const char* CLIENT_NAME  [extern]
```

9.8.2.3 humidity_value

```
float humidity_value  [extern]
```

9.8.2.4 interval

```
const long interval  [extern]
```

9.8.2.5 last_msg_sent

```
unsigned long last_msg_sent  [extern]
```

9.8.2.6 lid_position

```
bool lid_position  [extern]
```

9.8.2.7 mqtt_client

```
PubSubClient mqtt_client  [extern]
```

9.8.2.8 MQTT_PASSWORD

```
const char* MQTT_PASSWORD  [extern]
```

9.8.2.9 MQTT_PORT

```
const int MQTT_PORT  [extern]
```

9.8.2.10 MQTT_SERVER

```
const char* MQTT_SERVER  [extern]
```

9.8.2.11 MQTT_USERNAME

```
const char* MQTT_USERNAME  [extern]
```

9.8.2.12 msg_interval

```
const unsigned long msg_interval  [extern]
```

9.8.2.13 PASSWORD

```
const char* PASSWORD  [extern]
```

9.8.2.14 peltierandfanpin

```
const uint8_t peltierandfanpin  [extern]
```

9.8.2.15 peltierandfanstate

```
int peltierandfanstate  [extern]
```

9.8.2.16 previousMillis

```
unsigned long previousMillis  [extern]
```

9.8.2.17 pwmFreq

```
const int pwmFreq  [extern]
```

9.8.2.18 SENSOR_DATA_TOPIC

```
const char* SENSOR_DATA_TOPIC  [extern]
```

9.8.2.19 setPoint

```
const int setPoint  [extern]
```

9.8.2.20 sht4

```
Adafruit_SHT4x sht4  [extern]
```

9.8.2.21 SSID

```
const char* SSID  [extern]
```

9.8.2.22 temperature_value

```
float temperature_value  [extern]
```

9.8.2.23 tolerance

```
const int tolerance  [extern]
```

9.8.2.24 wifi_client

```
WiFiClient wifi_client  [extern]
```

9.9 waste_bin_controller.h

Go to the documentation of this file.

```
00001 #ifndef WASTE_BIN_CONTROLLER_H
00002 #define WASTE_BIN_CONTROLLER_H
00003
00004 #include <WiFi.h>
00005 #include <PubSubClient.h>
00006 #include <Adafruit_SHT4x.h>
00007 #include <ArduinoJson.h>
00008
00009 // WiFi settings
00010 extern const char* SSID;
00011 extern const char* PASSWORD;
00012
00013 // MQTT Broker settings
00014 extern const char* MQTT_SERVER;
00015 extern const char* MQTT_USERNAME;
00016 extern const char* MQTT_PASSWORD;
00017 extern const int MQTT_PORT;
00018
00019 extern const char* CLIENT_NAME;
00020
00021 // MQTT topics
00022 extern const char* SENSOR_DATA_TOPIC;
00023
00024 // Device settings
00025 extern const uint8_t peltierandfanpin;
00026 extern const long interval;
```

```
00027 extern int peltierandfanstate;
00028
00029 // PubSubClient setup
00030 extern WiFiClient wifi_client;
00031 extern PubSubClient mqtt_client;
00032
00033 // Other variables
00034 extern unsigned long last_msg_sent;
00035 extern const unsigned long msg_interval;
00036 extern unsigned long previousMillis;
00037
00038 extern const int setPoint;
00039 extern const int tolerance;
00040 extern const int pwmFreq;
00041
00042 // Sensor pin
00043 extern const int BUTTON_PIN;
00044
00045 // Sensor values
00046 extern bool lid_position;
00047 extern float humidity_value;
00048 extern float temperature_value;
00049
00050 // SHT4x sensor setup
00051 extern Adafruit_SHT4x sht4;
00052
00053 void setupWiFi();
00054 void setupPeltierAndFan();
00055 void setupSHT4x();
00056 void setupMQTTClient();
00057
00058 void readSensorData();
00059 void updateLidPosition();
00060 void controlPeltierModule();
00061 void publishSensorData();
00062 void handleIncomingMessage(char* topic, byte* payload, unsigned int length);
00063 void publishMessage(const char* topic, const float temperature, const float humidity, const bool
      lid_position);
00064
00065 void setup();
00066 void loop();
00067
00068 #endif  // WASTE_BIN_CONTROLLER_H
```

9.10 waste_bin_controller.ino File Reference

9.11 waste_bin_controller.ino

Go to the documentation of this file.

```
00001 #include "waste_bin_controller.h"
00002
00003 // WiFi settings
00004 const char* SSID = "WWegvanons3";       ///< SSID of the WiFi network
00005 const char* PASSWORD = "JuCasSan27@#";  ///< Password for the WiFi network
00006
00007 // MQTT Broker settings
00008 const char* MQTT_SERVER = "192.168.111.237";  ///< MQTT broker server IP address
00009 const char* MQTT_USERNAME = "mqtt";           ///< MQTT broker username
00010 const char* MQTT_PASSWORD = "WasteBin5#";     ///< MQTT broker password
00011 const int MQTT_PORT = 1883;                   ///< MQTT broker port number
00012
00013 const char* CLIENT_NAME = "ESP32WasteBin";  ///< Name of the MQTT client
00014
00015 // MQTT topics
00016 const char* SENSOR_DATA_TOPIC = "/data/sensors/";  ///< Topic for sensor data
00017
00018 // Device settings
00019 const uint8_t peltierandfanpin = 3;  ///< Pin for Peltier module and fan
00020 const long interval = 20000;         ///< Interval for Peltier and fan control
00021 int peltierandfanstate = LOW;        ///< Current state of the Peltier module and fan
00022
00023 // PubSubClient setup
00024 WiFiClient wifi_client;                 ///< WiFi client for MQTT connection
00025 PubSubClient mqtt_client(wifi_client);  ///< MQTT client
00026
00027 // Other variables
00028 unsigned long last_msg_sent = 0;           ///< Time when the last MQTT message was sent
00029 const unsigned long msg_interval = 15000;  ///< Interval for sending MQTT messages
```

```
00030 unsigned long previousMillis = 0;          ///< Previous timestamp for interval calculations
00031
00032 const int setPoint = 15;  ///< Temperature set point in Celsius
00033 const int tolerance = 2;  ///< Temperature tolerance in Celsius
00034 const int pwmFreq = 200;  ///< PWM frequency in Hz
00035
00036 // Sensor pin
00037 const int BUTTON_PIN = 4;  ///< Pin for the lid position sensor
00038
00039 // Sensor values
00040 bool lid_position = false;      ///< Current state of the lid position
00041 float humidity_value = 0.0;     ///< Current humidity value
00042 float temperature_value = 0.0;  ///< Current temperature value
00043
00044 // SHT4x sensor setup
00045 Adafruit_SHT4x sht4 = Adafruit_SHT4x();  ///< SHT4x sensor object
00046
00047 /**
00048  * @brief Set up the WiFi connection.
00049  */
00050 void setupWiFi() {
00051   Serial.print("Connecting to ");
00052   Serial.println(SSID);
00053
00054   WiFi.mode(WIFI_STA);
00055   WiFi.begin(SSID, PASSWORD);
00056
00057   // Wait for WiFi connection
00058   while (WiFi.status() != WL_CONNECTED) {
00059     delay(500);
00060     Serial.print("trying to connect...  ");
00061   }
00062
00063   Serial.println("\nWiFi connected\nIP address:  ");
00064   Serial.println(WiFi.localIP());
00065 }
00066
00067 /**
00068  * @brief Set up the Peltier module and fan pin.
00069  */
00070 void setupPeltierAndFan() {
00071   // Peltier + fan pin
00072   pinMode(peltierandfanpin, OUTPUT);
00073   analogWriteFrequency(pwmFreq);
00074 }
00075
00076 /**
00077  * @brief Set up the SHT4x sensor.
00078  */
00079 void setupSHT4x() {
00080   // SHT4x sensor setup
00081   Serial.println("Adafruit SHT4x test");
00082   if (!sht4.begin()) {
00083     Serial.println("Couldn't find SHT4x");
00084     delay(1000);
00085   }
00086   Serial.println("Found SHT4x sensor");
00087   Serial.print("Serial number 0x");
00088   Serial.println(sht4.readSerial(), HEX);
00089   sht4.setPrecision(SHT4X_HIGH_PRECISION);
00090   sht4.setHeater(SHT4X_NO_HEATER);
00091 }
00092
00093 /**
00094  * @brief Set up the MQTT client.
00095  */
00096 void setupMQTTClient() {
00097   // MQTT client setup
00098   mqtt_client.setServer(MQTT_SERVER, MQTT_PORT);
00099   mqtt_client.setCallback(handleIncomingMessage);
00100 }
00101
00102 /**
00103  * @brief Read sensor data from the SHT4x sensor.
00104  */
00105 void readSensorData() {
00106   // Read sensor data
00107   sensors_event_t humidity, temp;
00108   uint32_t timestamp = millis();
00109   sht4.getEvent(&humidity, &temp);
00110   timestamp = millis() - timestamp;
00111
00112   // Update sensor values
00113   humidity_value = humidity.relative_humidity;
00114   temperature_value = temp.temperature;
00115 }
00116
```

```
00117 /**
00118  * @brief Update the current lid position based on the button state.
00119    @brief Whenever the lid is opened longer than threshold, sound will be played trough piezo and led
      will light up.
00120
00121  */
00122 const int led_pin = 1;       // Pin for the LED
00123 const int piezo_pin = 5;     // Pin for the piezo speaker
00124
00125 unsigned long lidOpenStartTime = 0;  // Variable to store the lid open start time
00126 bool isBeeping = false;              // Flag to track if the piezo speaker is beeping
00127
00128 void updateLidPosition() {
00129   // Get lid position
00130   bool button_state = digitalRead(BUTTON_PIN);
00131   if (button_state == LOW) {
00132     lid_position = true;  // lid is closed
00133     lidOpenStartTime = millis();  // Reset lid open start time
00134     digitalWrite(led_pin, LOW);   // Turn off the LED
00135
00136     // Stop the piezo speaker from beeping
00137     if (isBeeping) {
00138       noTone(piezo_pin);
00139       isBeeping = false;
00140     }
00141   } else {
00142     lid_position = false;  // lid is opened
00143
00144     // Check if the lid has been opened for longer than 10 seconds
00145     if (millis() - lidOpenStartTime >= 10000) {
00146       // Blink the LED every 2 seconds
00147       if ((millis() / 2000) % 2 == 0) {
00148         digitalWrite(led_pin, HIGH);  // Turn on the LED
00149
00150         // Start the piezo speaker beeping if it's not already beeping
00151         if (!isBeeping) {
00152           tone(piezo_pin, 500, 750);  // Beep for 500ms at 750Hz
00153           isBeeping = true;
00154         }
00155       } else {
00156         digitalWrite(led_pin, LOW);   // Turn off the LED
00157
00158         // Stop the piezo speaker from beeping
00159         if (isBeeping) {
00160           noTone(piezo_pin);
00161           isBeeping = false;
00162         }
00163       }
00164     } else {
00165       digitalWrite(led_pin, LOW);   // Turn off the LED
00166
00167       // Stop the piezo speaker from beeping
00168       if (isBeeping) {
00169         noTone(piezo_pin);
00170         isBeeping = false;
00171       }
00172     }
00173   }
00174 }
00175
00176
00177 /**
00178  * @brief Control the Peltier module based on the temperature error.
00179  */
00180 void controlPeltierModule() {
00181   float error = setPoint - temperature_value;  // Calculate error
00182   Serial.println(temperature_value);
00183   Serial.println(error);
00184
00185   if (error > tolerance) {
00186     // If temperature is too HIGH, turn off Peltier module
00187     analogWrite(peltierandfanpin, 0);  // Set maximum duty cycle
00188   } else if (error < -tolerance) {
00189     // If temperature is too LOW, turn on Peltier module
00190     analogWrite(peltierandfanpin, 255);  // Set minimum duty cycle
00191   }
00192   // Otherwise, maintain current state of Peltier module
00193 }
00194
00195 /**
00196  * @brief Publish sensor data to the MQTT broker.
00197  */
00198 void publishSensorData() {
00199   // Publish sensor data to MQTT broker
00200   if (mqtt_client.connected()) {
00201     if (millis() - last_msg_sent > msg_interval) {
00202       publishMessage(SENSOR_DATA_TOPIC, temperature_value, humidity_value, lid_position);
```

```
00203       last_msg_sent = millis();
00204     }
00205   } else {
00206     // Attempt to reconnect to MQTT broker
00207     if (!mqtt_client.connect(CLIENT_NAME, MQTT_USERNAME, MQTT_PASSWORD)) {
00208       Serial.println("Failed to connect to MQTT broker");
00209     }
00210   }
00211
00212   // Maintain MQTT connection
00213   mqtt_client.loop();
00214 }
00215
00216 /**
00217  * @brief Handle incoming MQTT messages.
00218  * @param topic The topic of the incoming message.
00219  * @param payload The payload of the incoming message.
00220  * @param length The length of the payload.
00221  */
00222 void handleIncomingMessage(char* topic, byte* payload, unsigned int length) {
00223   // Handle incoming MQTT messages
00224   String incoming_message = "";
00225   for (int i = 0; i < length; i++) {
00226     incoming_message += (char)payload[i];
00227   }
00228   Serial.println("Message arrived [" + String(topic) + "]:  " + incoming_message);
00229 }
00230
00231 /**
00232  * @brief Publish a message to the MQTT broker.
00233  * @param topic The topic to publish the message to.
00234  * @param temperature The temperature value to include in the message.
00235  * @param humidity The humidity value to include in the message.
00236  * @param lid_position The lid position value to include in the message.
00237  */
00238 void publishMessage(const char* topic, const float temperature, const float humidity, const bool
      lid_position) {
00239   // Create JSON object
00240   auto macAdress = WiFi.macAddress();
00241   StaticJsonDocument<200> jsonDoc;
00242
00243   jsonDoc["macAdress"] = macAdress;
00244   jsonDoc["temperature"] = temperature;
00245   jsonDoc["humidity"] = humidity;
00246   jsonDoc["lid_position"] = lid_position;
00247
00248   // Serialize JSON object to string
00249   String message;
00250   serializeJson(jsonDoc, message);
00251
00252   // Publish message to MQTT broker
00253   if (mqtt_client.publish(topic, message.c_str())) {
00254     Serial.println("Message published [" + String(topic) + "]:  " + message);
00255   } else {
00256     Serial.println("Failed to publish message [" + String(topic) + "]:  " + message);
00257   }
00258 }
00259
00260 /**
00261  * @brief Perform the initial setup of the system.
00262  */
00263 void setup() {
00264   Serial.begin(115200);
00265   Serial.print("Connecting to ");
00266   Serial.println(SSID);
00267
00268   WiFi.mode(WIFI_STA);
00269   WiFi.begin(SSID, PASSWORD);
00270
00271   // Wait for WiFi connection
00272   while (WiFi.status() != WL_CONNECTED) {
00273     delay(500);
00274     Serial.print("trying to connect...  ");
00275   }
00276
00277   Serial.println("\nWiFi connected\nIP address:  ");
00278   Serial.println(WiFi.localIP());
00279
00280   setupPeltierAndFan();
00281   setupSHT4x();
00282   setupMQTTClient();
00283 }
00284
00285 /**
00286  * @brief The main program loop.
00287  */
00288 void loop() {
```

```
00289   readSensorData();
00290   updateLidPosition();
00291   controlPeltierModule();
00292   publishSensorData();
00293 }
```

Index

Bureaubladonderzoek.md, 31
BUTTON_PIN
    waste_bin_controller.h, 34

CLIENT_NAME
    waste_bin_controller.h, 34
controlPeltierModule
    waste_bin_controller.h, 32

handleIncomingMessage
    waste_bin_controller.h, 32
humidity_value
    waste_bin_controller.h, 34

interval
    waste_bin_controller.h, 34
Introductie.md, 31
IoT oplossing.md, 31

last_msg_sent
    waste_bin_controller.h, 35
lid_position
    waste_bin_controller.h, 35
loop
    waste_bin_controller.h, 33

Mosquittosetup.md, 31
mqtt_client
    waste_bin_controller.h, 35
MQTT_PASSWORD
    waste_bin_controller.h, 35
MQTT_PORT
    waste_bin_controller.h, 35
MQTT_SERVER
    waste_bin_controller.h, 35
MQTT_USERNAME
    waste_bin_controller.h, 35
msg_interval
    waste_bin_controller.h, 35

Ontwerpkeuzes.md, 31

PASSWORD
    waste_bin_controller.h, 36
PCB-design.md, 31
peltierandfanpin
    waste_bin_controller.h, 36
peltierandfanstate
    waste_bin_controller.h, 36
previousMillis
    waste_bin_controller.h, 36
publishMessage
    waste_bin_controller.h, 33
publishSensorData
    waste_bin_controller.h, 33
pwmFreq
    waste_bin_controller.h, 36

readSensorData
    waste_bin_controller.h, 33

SENSOR_DATA_TOPIC
    waste_bin_controller.h, 36
setPoint
    waste_bin_controller.h, 36
setup
    waste_bin_controller.h, 33
setupMQTTClient
    waste_bin_controller.h, 33
setupPeltierAndFan
    waste_bin_controller.h, 33
setupSHT4x
    waste_bin_controller.h, 34
setupWiFi
    waste_bin_controller.h, 34
sht4
    waste_bin_controller.h, 36
SSID
    waste_bin_controller.h, 37

temperature_value
    waste_bin_controller.h, 37
Testplan.md, 31
tolerance
    waste_bin_controller.h, 37

updateLidPosition
    waste_bin_controller.h, 34

waste_bin_controller.h, 31, 37
    BUTTON_PIN, 34
    CLIENT_NAME, 34
    controlPeltierModule, 32
    handleIncomingMessage, 32
    humidity_value, 34
    interval, 34
    last_msg_sent, 35
    lid_position, 35
    loop, 33
    mqtt_client, 35
    MQTT_PASSWORD, 35
    MQTT_PORT, 35

MQTT_SERVER, 35
MQTT_USERNAME, 35
msg_interval, 35
PASSWORD, 36
peltierandfanpin, 36
peltierandfanstate, 36
previousMillis, 36
publishMessage, 33
publishSensorData, 33
pwmFreq, 36
readSensorData, 33
SENSOR_DATA_TOPIC, 36
setPoint, 36
setup, 33
setupMQTTClient, 33
setupPeltierAndFan, 33
setupSHT4x, 34
setupWiFi, 34
sht4, 36
SSID, 37
temperature_value, 37
tolerance, 37
updateLidPosition, 34
wifi_client, 37
waste_bin_controller.ino, 38
wifi_client
waste_bin_controller.h, 37