

Naturbrand Sensor

Daniel Nybo, Emil kaare Adler Pehrson, Mikael Lau Slot, Marcus Agerlin

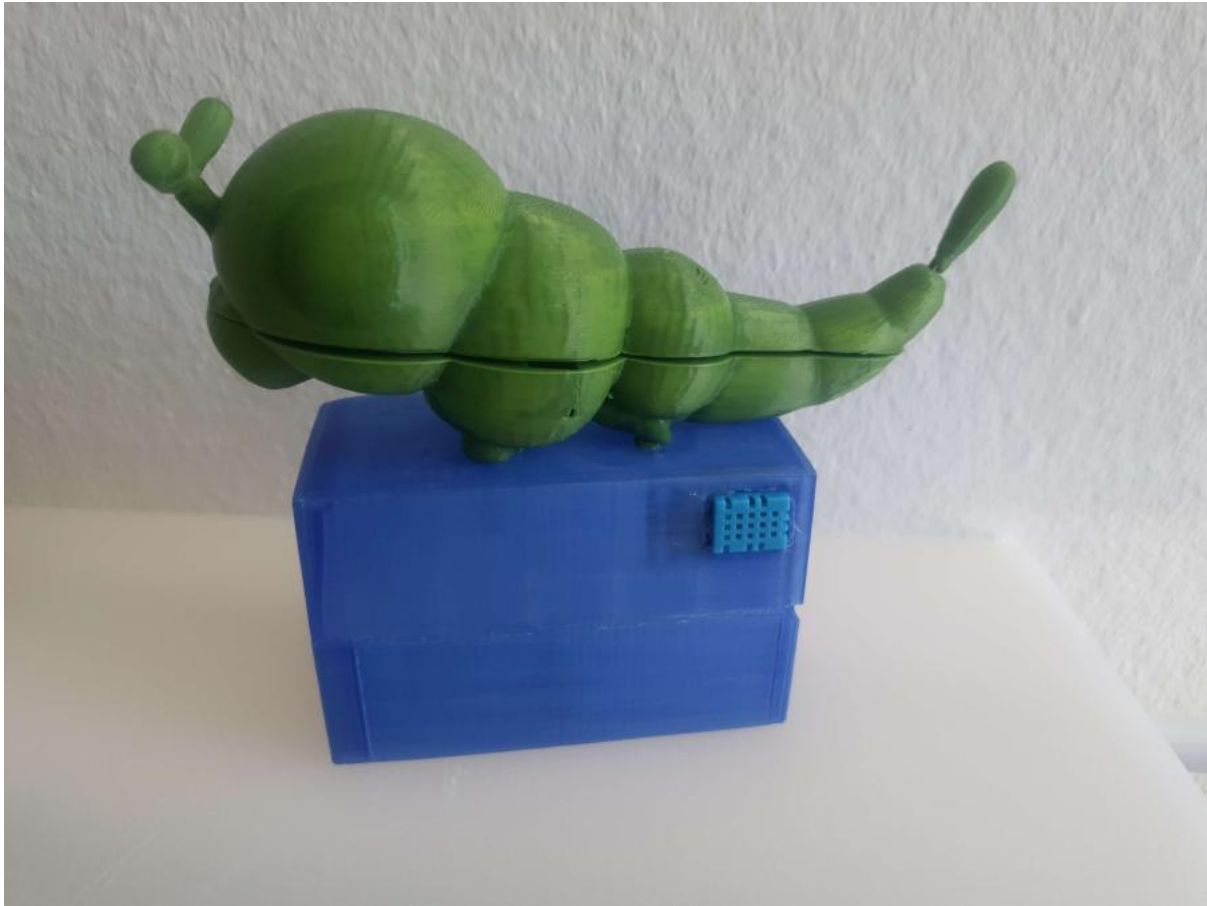
Gruppe: 2 B,

Klasse og uddannelse: IT-Teknologi, B Klassen

Projektnavn og årgang: IOT2-Projekt - F23

Vejleder: Malene Hasse. Kevin Holm. Dan Madsen. Bo Hansen.

Antal anslag: 59.404



FOREST SAVER

MAIN

GRAPH

TABLE

INFO

Naturbrande har en meget stor negativ effekt på vores klima i dag. Ikke nok med at der udledes Co2 så mister vi også skove og områder med planter som man fotosyntese og områder hvor dyr og vildt lever. Vi har lavet et produkt som kan måle Varme, Luftfugtighed og Luftkvalitet for at forbygge starten og størrelsen af naturbrande. På denne side kan man se målingerne på det forskellige parameter som vurderer risikoen for brand. For eksempel hvis et område har meget lav luftfugtighed og meget høj varme er det jo i en risikozone og det kan så skabe en større opmærksomhed omkring den zone specifikt. Ved at gøre dette kan beredskabet være mere forberedt og hvis en naturbrand skulle starte bliver de sandsynligvis opdateret hurtigere så det kan slukkes for det spredt sig til ekstreme størrelser.

Forest Saver is the way to go!

1. Resumé

I projektet arbejder vi med et af FNs 17 verdensmål og hvordan vi kan løse den med en IOT-løsning. Vi kommer til at arbejde med at forebygge naturbrande, hvordan vi kan gøre det via en IOT-løsning og om det virkelig er noget vi har brug for. I vores projekt har vi specielt fokus på at stoppe branden før den er begyndt eller før den når at vokse for meget. Det gør vi via flere forskellige komponenter som skal udmåle et områdes tilstand fx. luftfugtighed og varme. Vi kaster lys over hvor stort et problem det egentlig er og hvad beredskabsstyrelsens mening er om det.

Det er en teknisk gennemgang af vores arbejdsprocesser, hvordan vi kom til vores endelige problemformulering.

Der er de tekniske detaljer på vores IOT-løsning, som houser alle vores mobile komponenter, og en konklusion af forløbet.

2. Indholdsfortegnelse

Contents

1. Resumé	2
2. Indholdsfortegnelse	3
3. Indledning	5
3.1 Beskrivelse af virksomhed	6
3.2 Verdensmål	6
4. Problemformulering	7
5. Indledende undersøgelse	8
5.1 Ideudvikling	8
5.2 Research	8
5.3 Brugerundersøgelse / Design Thinking	9
6. Kravspecifikation og accepttest metode	14
6.1 Rationale for prioritering af krav	19
7. Analyse	20
7.1 Arkitektur af pathfinding	21
7.2 Den endelige arkitektur	23
7.3 Teori af relevante elektroniske/programmering/netværk/server arkitekturblokke	24
8. Løsningsdesign	31
8.1 Elektriske skematikker, målinger og udregninger (Indlejrede systemer)	31
8.3 Kode beskrivelse (Programmering)	32
Boot.py	32
Main.py	34

Measure.py	37
RPI-Kodning	38
SubScriber.bash	39
table.html	41
graph.html	41
style.css	41
9. Test af løsning	45
9.1 Pathfinding testresultater	45
9.2 Dokumentation af DUT	46
9.3 Udførsel af brugertest	48
9.4 Udførsel af Accepttest på DUT	50
Test af krav ID: 1, ID: 2 & ID: 3	50
10. Implementering af løsning i drift (eventuel)	55
11. Praktisk projektplanlægning og ledelse	56
11.1 WBS	56
11.2 Gantt	57
11.3 Scrum	58
11.4 Projektanalyse	59
11.5 Risikovurdering	61
12. Konklusion	63
13. Projektforløbet	64
14. Perspektivering	65
15. Litteraturliste	Fejl! Bogmærke er ikke defineret.

3. Indledning

I 1938 opdagede amatørforskeren Guy Callendar forbindelsen mellem menneskelig aktivitet og global opvarmning . Den Industrielle Revolution kom på bekostning af jordens klima og alt hvad der benyttede den, os selv inkluderet. Til trods for de tegn som allerede var begyndt at vise sig helt tilbage i 1896, er der ikke blevet taget store preventive handlinger.

Vi skriver nu 2019, og 24,3 millioner hektar brænder på grund af en ekstrem tørke, på baggrund af en massiv hedebløge.

Ikke engang et år senere i 2022 blev USA's vestkyst ramt af rekord-store vild-brande, der lagte Oregon, Californien og Washington ned.

Hvad havde begge naturkatastrofer tilfælles? Slem tørke og en ekstrem hedebløge. Naturbrande kan have katastrofale konsekvenser for miljøet, og alt, der anvender det, om det er dyr eller mennesker.

Så hvordan kan man bekæmpe naturbrande? Det er et problem i Danmark, hvor vi gennemsnitligt har 2.000 alarmerings opkald til 112 angående naturbrande. Ifølge Dansk Beredskab, er det fra sommermånederne fra Marts til September gennemsnitligt 200 udrykninger om måneden

Brande opstår normalt ikke af sig selv, selvantændelse er sjældent, da visse betingelser skal være opfyldt. Lavt fugtniveau, høje temperaturer, og sol-lys. Men i beredskabs bestyrelsens egne ord, så er alle IT anvendelser budt velkommen med åbne arme : Hurtig opdagelse af brande kan redde liv, og spare masser af penge.

Så problemet gruppe 2B har valgt at undersøge i dette IOT-projekt er en løsning til at stoppe naturbrande, specifikt hvad skal der til for at man kan informere om beredskabet om at en brand er opstået hvor der ikke er den store fæstelse af mennesker, som eksempelvis ude i naturskovene.

Vores løsning er en box af sensorer der måler fugten i miljøet, temperaturen, og kulstofsniveauet, samler den data og sender den afsted til en server. Meningen er at alarmberedskabet kan bruge den data til at sætte stoppe naturbrande, før de når brede sig ud til seriøse trusler. (Dansk Beredskabs, n.d.) (Dee, 2022)

3.1 Beskrivelse af virksomhed

Den organisation, som projektet har et samarbejde med, er Naturstyrelsen, da løsningerne skal hjælpe med at forebygge naturbrande.

Naturstyrelsen er en del af miljøministeren. Naturstyrelsen har til opgave at varetage skove, naturområder og kyster i Danmark. Naturstyrelsen har til opgave at forvalte omkring 200.000 hektar statslige skove og naturområder, dette gør de for at skabe størst værdi for det danske samfund og lave gode rammer for friluftslivet. De sørger også for at beskytte og sikre effektiv drift af styrelsen, skove og naturområder.

Nogle af de opgaver som Naturstyrelsen har, er jagt- og vildtforvaltning og konkrete friluftslivs- og naturprojekter, som for eksempel at plante nye skove nær byer.

3.2 Verdensmål

Den opgave, som der var stillet, havde til formål at lave en løsning til et af de 17 verdensmål. De 17 verdensmål omhandler 17 mål, som FN har lavet for at fremme den bæredygtige udvikling. I dette projekt arbejdes der med verdensmål 15, som er livet på land, hvor der arbejdes med at bevare skovarealer. Projekt vil omhandle naturbrande, og hvordan man ville være i stand til at forebygge naturbrande ved hjælp af en IoT løsning.

Beredskabsstyrelsen anvender 112 opkald til at reagere på naturbrande, med øjenvidner, der rapporterer lokationer og seriøsiteten over branden. Der bliver derefter skrevet en rapport som uploades til ODIN databasen.

Ved at opdage brande før de bliver en trussel, og påpeger risiko zoner med aktive målinger på miljøet, kan vores løsning være med til at reducere truslen til det 15 verdensmål om at beskytte skovarealer

4. Problemformulering

“Det er et problem, at naturbrande opdages for sent i Danmark”

Naturbrande er i højere og stigende grad blevet mere og mere hyppige. Årsagerne ligger i menneskelig skabt klimaforandringer. Tørke og hedeølger stiger både i hvor ofte de sker, men også med de kolossale og katastrofale skader som sker på konsekvens af de naturbrande, som opstår. Man skal bare se på de forfærdelige billeder som der opstod under Australiens store skovbrand i 2019, og ikke meget kort efter den enorme miljøkatastrofe, der ramte USA's vestkyst i 2020.

Naturbrande kan ske meget let på grund af mennesker (Cigaretskodder, Kul-gril eller et bål), men kan også sagtens starte af sig selv. Selvantændelse kan ske ved 200-400 grader, men nogle træer, som grantræer, kan have det ufattelig nemt ved at antænde sig selv. Tørke kort efterfulgt af en hedeølge.

Vi kan herhjemme mærke denne globale tendens, da vi i året 2018 havde over 2000 naturbrande. Landmænd mærker det på deres marker, og naturen lider også som konsekvens pga. stigende tendens til naturbrande.

IoT er allerede i brug hos beredskabsstyrelsen, der med hjælp af indsamlede data fra både 1-1-2 alarmerings opkald, også gøres brug af deres ODIN system : Online Dataregistrerings- og INdberetningssystem

Hvordan kan man med en IOT løsning forebygge mod naturbrande i danske skove?

- “Hvor er der flest naturbrande i Danmark?”
- “Hvad er den hyppigste årsag til naturbrande?”
- "Hvordan kan vi ved hjælp af dataindsamling forebygge naturbrande?”

5. Indledende undersøgelse

Naturbrande og Beredskab

Naturbrande er meget mere hyppig end man umiddelbart ville tro i Danmark. Dansk beredskab melder ud, at der er 200 rapporteringer i månederne marts - september i året 2018. Årsagen er frostvejret, der kan hive fugten ud i form af frost; En kold tørke.

Naturbrande i Danmark sker ikke ofte af sig selv, men det er muligt. Grantræer kan selvantænde under specifikke krav; tørke, hedebløge og direkte sollys på en skyfri dag men er dog meget sjældent. Langt fleste naturbrande sker på baggrund af menneskelige handlinger.

Opdagelse af naturbrande sker ofte for sent, når branden allerede har gjort skader. Dette er i form af 112 opkald der rapporter om aktive brande, og disse udmeldinger bliver så dokumenteret i beredskabsstyrelsens ODIN¹ arkiv

5.1 Ideudvikling

Projektet startede med at der blev lavet en brainstorm, hvor alle kom med nogle ideer om hvilken af de 17 verdensmål projektet skulle handle om. Der blev lavet nogle fremlæggelser om de forskellige verdensmål i undervisning. Der blev også lavet brainwriting for at videreudvikling af projektet og hvilken forskellige problemer der kunne løses. Efter dette blev der lavet en Walk'n'talk, hvor løsningen til vores problem blev dannet.

5.2 Research

Under ideudvikling blev der fundet nogle artikler som beskrev, hvordan klimaforandringerne ville påvirke naturbrande. Der blev også undersøgt, hvor naturbrandene var mest hyppige, senere blev det fundet ud af, at der var et stort antal i Danmark.

VisionTIR firma, bruger et kamera der måler termiske billeder(thermal imaging), smoke og heat sensor, som kan måle og advare lige så snart det er inde for range af et forud bestemt sæt af regler. 112 opkald bliver brugt til at spore brande, og er et uperfekt system. Derfor anvendes ODIN-data (Online Dataregistrerings- og Indberetningssystem) som en yderligere hjælp til brandfare og prævention. ODIN er en forsamling af detaljerede rapporter.

¹ Online Dataregistrerings og Indberetningssystem

5.3 Brugerundersøgelse / Design Thinking

I vores brugerundersøgelse lavede vi et spørgeskema, som vi sendte til Beredskabsstyrelsen, som de gav svar på skriftligt. Deres svar gav lidt et indblik i, hvordan deres struktur er og at det er noget der skal implementeres på kommunebasis og ikke på landsbasis. Det gav også et indblik i, hvor stort et problem det reelt er for dem og hvordan de får deres information om, hvad det er der sker, når der starter en naturbrand på dansk land. De nævner også at det kan ske at der starter en naturbrand som tager en stor indsats fra redningsberedskabet som for eksempel nævner de naturbrandene på Randbøl Hede og i Vildmose i 2018 som tog dem flere døgn at få slukket. De giver os også et perspektiv på hvilke parametre de tager højde for når det kommer til naturbrande fx. hvilken type natur, tørhed, blæst, adgang til vand, hvor ekstreme de er og hvilken indflydelse de har. De nævner deres program ODIN som er en løsning som hjælper dem med at få info om naturbrande. De nævner også kort brandfare indekset og hvordan de bruges af kommunale beredskaber til at holde øje med hvor og hvornår der er forhøjet risiko for brand i naturen.

(bilag 14 Brugerundersøgelse af Beredskabsstyrelsen)

5.4 Personaer og User Stories

Peter, Lyst Vandrer:

Peter er 34 år gammel og er alene-boende. Peter har en hund, en Golden Retriever, Thor som han lufter i Amager-fællede.

Da han bor på Kongelundsvejen er han en enkelt spyt-kast fra Amager-fællad, med en kort gåtur på 4 minutter er han på naturreservatet. Bil-mekaniker der arbejder I Kirstinehøj

Sofie, Miljøaktivisten:

Sofie bor på Amagerbro, spaniensgade

Hun er 27 år og er webdesigner for en dansk startup, der hedder Glamour. Firmaet har udvist interesse i at have et neutralt Co2 footprint.

Sofie bor sammen med sin kæreste, Robin, og er begge 2 meget investeret i miljøet og dansk natur-kultur.

De er begge medlemmer af en forening, der samlet er med til at rydde op i deres boligområde.

Bo, Skovmanden:

49 år gammel og bor i Dragør. Skovmand på Amager Fælled Naturcenter.

Han arbejder med skoleelever og underviser dem i naturen i deres baghave. Han oplever, at det er ufattelig nemt at pirre ungdommens interesse i naturmiljøet, hvis de blot får en smule af se. Han er gift til sin kone Helene.

Anders, Landmand:

Anders er en landmand, der driver både kvæg og landbrug. Hans landjord ligger lige knapt på 100 hektar, og ligger lige op ad et naturreservat i Jylland. Han er gift og far til 4 unger.

Han er ikke mere eller mindre klimaorienteret, og er præget politisk mere på baggrund af hans erhverv.

Jørgen Jørgensen, Miljøminister:

Som miljøminister er målet at fuldføre ens parti miljøpolitik, som stort set bliver afspejlet i befolkningens mening. I takt med at befolkningen er blevet mere miljøbevidst, så er partierne blevet mere og mere enige om mærkesager. Miljøet er vigtigt, fordi vælgerne synes det er vigtigt.

Heinrich Mûeller, Businessmand repræsentant:

Heinrich er 40 år gammel og har arbejdet i et større tech-firma i 16 år. Hans interesser ligger ikke i klima, men mere i aktier og bedre vækst i firmaet.

Han har en kat der hedder Torben, som holder mest til sig selv.

User stories

Miljøministeren Jørgen Jørgensen.	Story nr : 1
<p>Jørgen's behov er at blive valgt igen, som egentlig er enhver politikers primære mål. Grundet det store skift i dansk og international politik, til at fokusere mere på grønnere miljø politisk mærkesager.</p> <p>Dansk natur der brænder er dårligt for miljøministeriet og regeringen, men løsninger og bud skal helst være så billigt og effektivt som muligt, ellers kan der ikke opnås politisk enighed på tværs af rød og blå blok.</p>	<p>Prioritet :Høj</p> <p>Politiker bør vægtes højt på grund af deres høje indflydelse. Der behøves ikke engang sigtes efter ministerier, da man også kan gå efter borgmestre på det kommunale niveau.</p> <p>Hvis man har støtte fra kommuner eller et ministerium, så kan man sikre sig økonomisk finansiering fra statens side.</p>
Hvis Jørgen kan blive valgt igen på baggrund af denne tekniske løsning, og bekæmpe konsekvenserne af global opvarmning, så kan han udføre mere ambitiøst klima-politik.	<p>Størrelse / Effort : Høj størrelse, Stor effort</p> <p>At overbevise en borgmester og sikre sig kommunal økonomisk størrelse er ikke nemt, men det ville hjælpe ekstremt med at få løftede produktet fra jorden.</p>

Peter, Lyst vandrer og bilmekaniker	Story nr : 2
<p>Peter's behov er egentlig meget simpel. Han passer sit erhverv og han passer sin hund. Hans social liv er på plads, og egentlig er han tilfreds med hvor han er. Behovet ligger i at bibeholde det han har. Hans hund er hans prioritet. Tørke i Amager-fælde ville være forfærdeligt.</p>	<p>Prioritet : Lav</p> <p>Almindelige borgerer er ikke vigtige for selve løsningen, da deres indflydelse er lav.</p> <p>Dette betyder ikke at det er ligegyldigt, da løsningen sagtens kunne finansieres privat.</p>
Beskyt den danske natur, så Thor kan få sine gåtur i et mere spændende miljø.	<p>Størrelse / Effort :Lav</p> <p>Der er egentlig ikke meget at hente fra private personer, men deres behov er til gengæld meget nemt at se til og opnå.</p>

Bo, Skovmanden	Story nr : 3
Ude på Amager-fællede er der et naturcenter der bl.a holder små uddannelses togter fra de nærliggende skoler. Tørke og hedebløger presser det lokale miljø og den biodiversitet der fandtes på reservatet.	Prioritet : Mellem Amager fællede har mulighed for både at leje bålpladser ud til private, hvor man kan købe brænde fra det lokale naturcenter. Der er 4 shelters hvor der er mulighed for overnatning, og det er her løsningen reelt kan testes, og gør gavn.
Hvis fælleden kan brandsikres, så kan folk på naturcenteret sove trygt. Det er hele hans erhverv og arbejde der er på spil.	Størrelse / Effort: Mellem Her kan løsningen blive testede for effektivitet. Det er også i dette miljø, hvor løsningen bliver brugt.

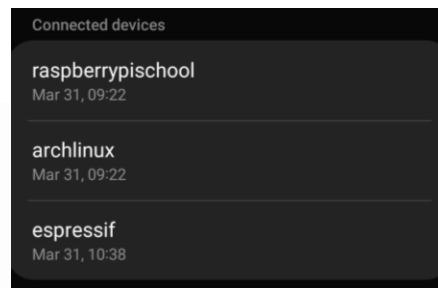
Sofie, Miljøaktivisten	Story nr : 4
Sofies behov for at vedvare naturen som miljøaktivist og at hun er meget investeret i dansk natur-kultur gør selvfølgelig at det er en vigtig ting for hende. Derimod har det ikke den store effekt på hendes arbejde i Glamour, som vil være Co2 neutralt.	Prioritet: Lav Det er mere af personlige årsager end noget der kan have effekt på hendes levevis. Naturbrande har ikke en effekt på hendes arbejde da hun er webdesigner. Selvom Glamour vil være Co2 neutralt betyder det ikke at naturbrande er en prioritet for dem.
Beskyt den danske natur som Sofie går meget op i.	Størrelse / Effort :Lav Hun er en privatperson med en kærlighed til naturen, men ikke mere end det. Der er ting hun kan gøre, men det er ikke meget og det er nemt at opnå.

6. Kravspecifikation og accepttest metode

ID:1	Krav: ESP'en og RPI kan connect til Internettet	Prioritet : (1)
Kategori: Software	Accepttest: Der printes i shell at de connector til internettet og det kan ses på internet udbydere at de er connected. Yderligere vil vi kræve at løsning kan connecte til internettet hvis den miste forbindelsen.	Passed

ESP32:

```
>>> %Run -c $EDITOR_CONTENT
Temperatur: 23.0 C
Humidit: 38.0 P
Carbondioxide: 300.200009 Corrected PPM
23, 38, False, 55.69194647082459, 12.554169821375735
Temperatur: 23.0 C
Humidit: 38.0 P
Carbondioxide: 300.200009 Corrected PPM
23, 38, False, 55.69194647082459, 12.554169821375735
Connected to 192.168.239.54 MQTT broker, subribed to ESP32Data topic
```



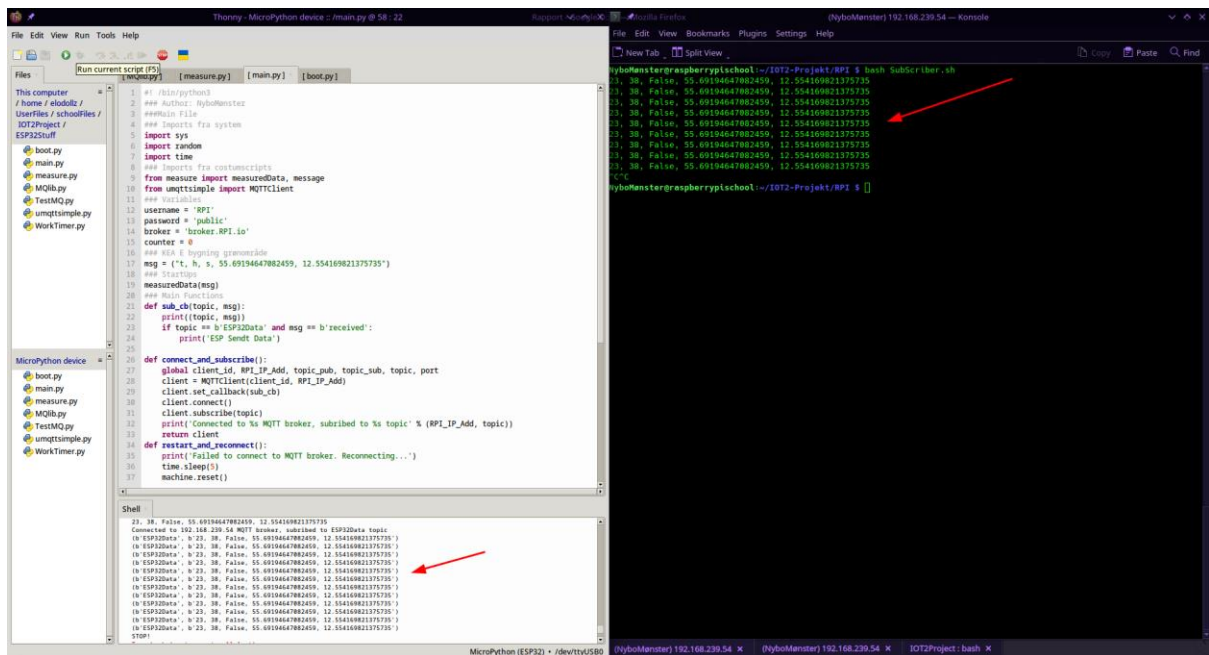
RPI:

```
mosquitto_sub -h localhost -t "ESP32Data"
```

ID:2	Krav: ESP'en Data fra DHT11 til og har en placering, med X og Y koordinater.	Prioritet : (1)
Kategori: Software	Accepttest: At den kan printet målt data fra ESP'en	Passed

```
23, 38, False, 55.69194647082459, 12.554169821375735
Connected to 192.168.239.54 MQTT broker, subribed to ESP32Data topic
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
(b'ESP32Data', b'23, 38, False, 55.69194647082459, 12.554169821375735')
STOP!
```

ID:3	<p>Krav:</p> <p>Den kan sende data, fra ESP32 til RPI</p>	<p>Prioritet</p> <p>: (1)</p>
<p>Kategori:</p> <p>Software</p>	<p>Accepttest:</p> <p>ESP'en kan sende en string af data til RPI (RaspberryPI) og RPI kan modtage data'en.</p>	<p>Passed</p>



ID:4	<p>Krav:</p> <p>MQ135 kan måle PPM og koden printer, om der er røg eller ej.</p>	<p>Prioritet : (1)</p>
<p>Kate gori: Software</p>	<p>Accepttest:</p> <p>At den kan printet målt data fra ESP'en og sende den til RPI, Data'en er real.</p> <p>3 stadigere:</p> <ul style="list-style-type: none"> - ingen røg (normalt klima) - lidt røg - meget røg 	<p>failed</p>

ID:5	<p>Krav:</p> <p>Data'en kan sættes ind på en hjemmeside, og vises i graf og en table.</p>	<p>Prioritet : (2)</p>
<p>Kate gori: Software</p>	<p>Accepttest:</p> <p>Data'en indsættes i en SQLite3 Database og kan hentes, indsættes i en HTML side, i en table så man kan se målingerne.</p>	<p>failed</p>

ID:6	<p>Krav:</p> <p>Holdbarhed</p>	<p>Prioritet</p> <p>: (3)</p>
<p>Kate</p> <p>gori:</p> <p>Hardware</p>	<p>Accepttest:</p> <p>Lade den falde fra 2 meters højde,</p>	<p>ikke</p> <p>testet</p>

ID:7	<p>Krav:</p> <p>Levetid, på batteri niveau</p>	<p>Prioritet</p> <p>: (3)</p>
<p>Kate</p> <p>gori:</p> <p>Hardware</p>	<p>Accepttest:</p> <p>Den kan holde sig kørende 6 timer på 2 batteriere af 1800 mAh. med 7.4 V, som nedreguleres til 5 V. til ESP'en</p>	<p>ikke</p> <p>testet</p>

6.1 Rationale for prioritering af krav

Vi begrundet vores valg af prioritering ud fra at hovedproblemet er at informere om en evt skovbrand, så derfor har vi prioritet krav ID 1 - 4 med prioritet 1. Vi gjorde særligt det fordi disse krav er til for, om vores løsning kan connect til internettet og sende data'en, og om den faktisk kan måle data'en, med DHT11 og en MQ135.

For at gøre det nemt for brugere at anvende vores løsning har vi vores prioritet ID 5 som kan indsætte dette på vores hjemmeside. Vores løsning var dog ikke at gennemføre, eftersom der var nogle problemer med at få hentet data'en fra SQLite3.

De mindste prioritet ID 6 - 7 er om vores løsning kan holde ude i et real test miljø, særligt om den kan holde batteri, og om den kan holde til miljøet, fra naturen, dyr og at de ikke ødelægger vores løsning, eller evt vejret. Vi har valgt at køre med 2 batteri, for at vores løsning kan holde sig kørende i en acceptabel mængde af tid, siden det vil være meget upraktisk at trække ledninger ud til hele skovarealerne i Danmark.

7. Analyse

Til vores løsning har vi en lille ordning af sensorer og en GPS tracker, men uden en ESP32 DEVKIT, kan disse sensorer ikke gøre deres formål. Gennem ESP32'eren's evne til at kommunikere wireless (UART protokollen)

Den tekniske opsættelse af ESP32 vil dog ikke supplere strømmen til modulerne, da støjniveauet er for højt, og da ESP'en også selv skal bruge strøm, så kan den ikke effektivt levere det videre.

MQ135

MQ135 gassensor som fungerer ved at når luften kommer ind ved MQ135 sensor element, hvor de forskellige gasser i luften bliver ioniseret, når luften kommer tæt på sensorelementet. Den måde som sensoren interface virker er, når der ikke er noget gas i høj nok koncentration, ville den digitale output være 1 og den analoge output er 1023 som er max value. Når der er gas til stede, ville den digitale være 0 og analoge output er mindre 1023. Der bliver brugt et potentiometer på chippen kan man controller turning off punktet af den digitale pin til en værdi fra analog pin. Sensoren har en belastningsmodstand mellem 2 kOhm til 47 kOhm, hvor den lavere værdi gør sensoren mindre sensitiv og den højere gør den mindre præcis. (NILET, 2022)

Dette projekt bruges MQ135 til at opfange CO₂ fra luften, så man kan forudse, om der er ved at komme naturbrande.

DHT11

DHT11 er en fugtigheds- og temperatursensor, som bruger eksklusive digital signal opsamlings teknik, hvor dens sensor-elementer er connected med en 8 bit enkelt-chip computer, som hedder MCU (microcontroller unit). Når MCU sender start signalet til DHT11, som gør at DHT11 skifter fra standby til kørende, efter at MCU bliver færdig med at sende start signalet, ville DHT11 begynde at sende en 40 bit data som reflekteret den relative fugtighed og temperatur. I dette projekt bruges DHT11 til at måle både fugtigheden og temperaturen i naturen for at kunne bestemme hvornår der er mest fare for at der opstår naturbrande.

MCU

MCU (microcontroller unit) er en lille computer, som bruger et VLSI integreret kredsløb, MCU'en har en eller flere CPU, samt med memory og programmerbart in-output parameter. (Wikipedia) (Lutkevich, 2019) MCU'en bliver brugt til DHT11, hvor det er en del af sensoren.

7.1 Arkitektur af pathfinding

Pathfinding

version 1

Her er den ideelle opsættelse af vores løsning. Et sæt serie forbundet batterier der føres igennem en LDO, nedsat til en spænding på 5 volt. Strøm forsyner en ESP32, der forsyner resten af komponenterne gennem et LAB-kort. En kondensator er placeret uden for hvert modul for at reducere strømmens "støj". Dette er en u-optimeret løsning, da ESP32 er dårligt bygget til at forsyne strøm.

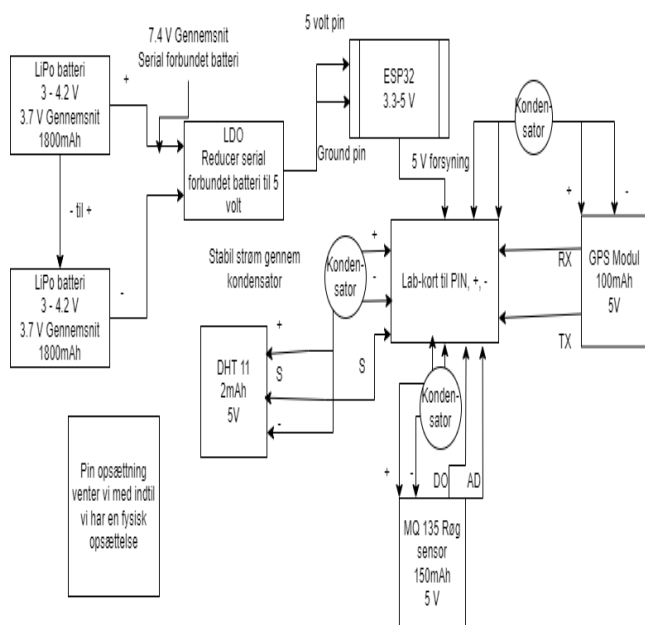


Figure 1

Pathfinding

version 2

Her er den optimale opsætning af vores løsning. Det er stadig et set serieforbundet batterier som går ind i en LDO som regulerer vores spænding fra 7,4V til 5V og forsyner vores ESP32, MQ135 og DHT11. Vi har også en GPS som bliver forsynet med 3,3V igennem vores ESP32s 3,3V output. Vi har så vores rx og tx (Transmit and Receive) data fra vores gps som sidder i vores ESP32 og leverer data derigennem. Vores DHT11 bruger en one wire connection og vores MQ135 bruger sit analog output til at connecte til vores ESP32

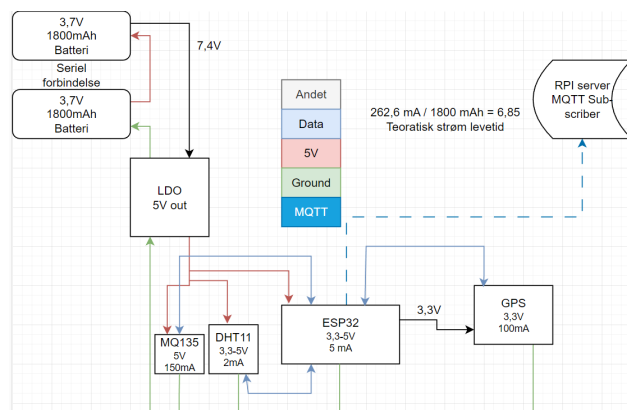


Figure 2

Programmering
pathfinding version 1
Version 1 er et meget stort træ²

2

Programmering
pathfinding version

² Større billede kan findes i Bilag

Processen startes når RPI'en³ bliver tændt. RPI anmoder om data fra vores SQLite database, som ligger lokalt på vores RPI (RPI'en fungerer som en server i dette tilfælde). Næste trin er at visualisere den data på en HTML hjemmeside, der viser den data som er blevet indsendt fra en ESP32. Den kører et loop som tjekker fejl hvert andet sekund.

Hvis der ingen fejl er, så bliver databasen opdateret, indtil man slukker for RPI'en, eller hvis ESP 32 løber tør for strøm

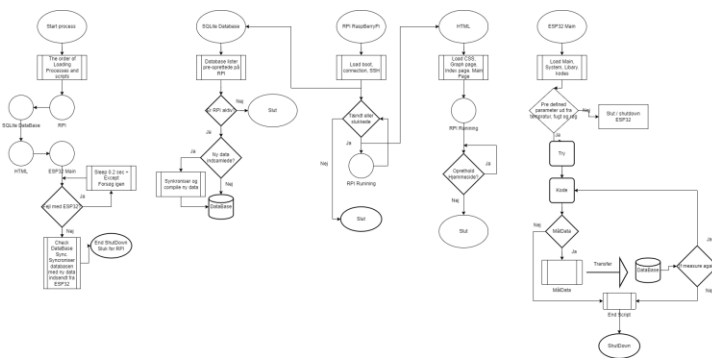


Figure 3

³ En raspberry-Pi, forkortes til RPI

7.2 Den endelige arkitektur

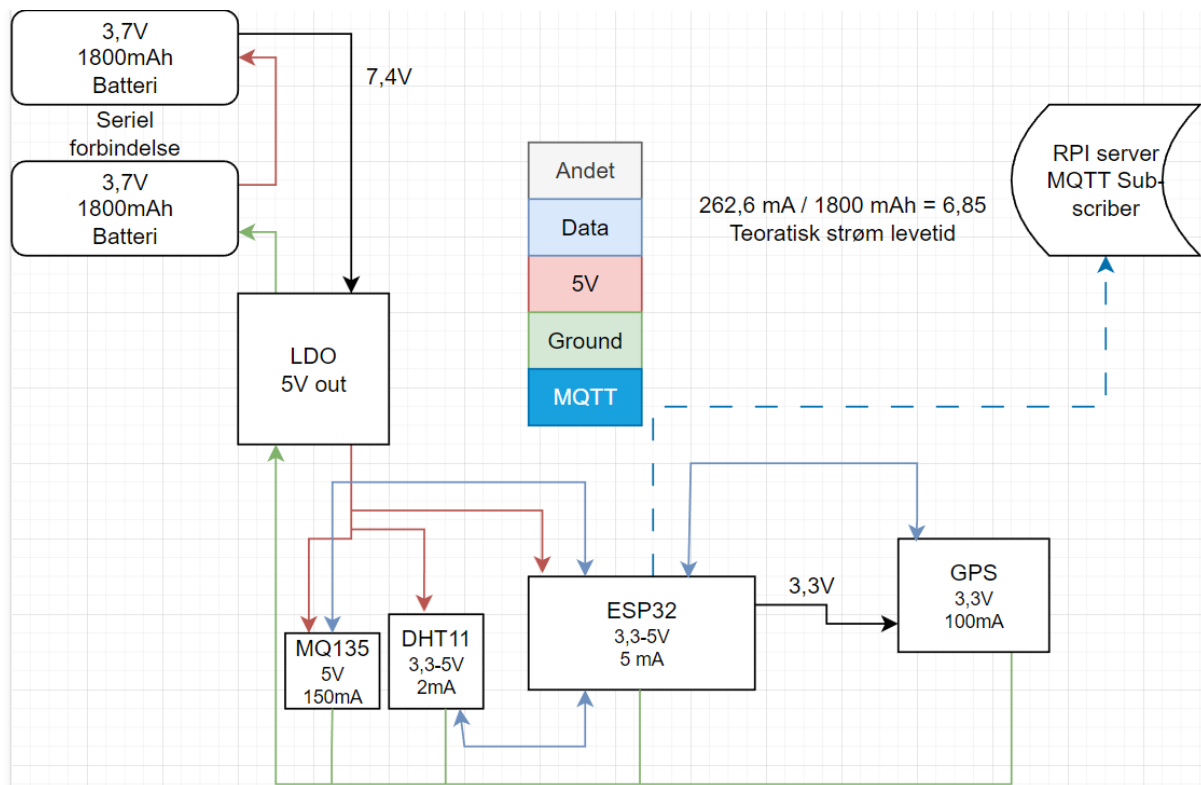


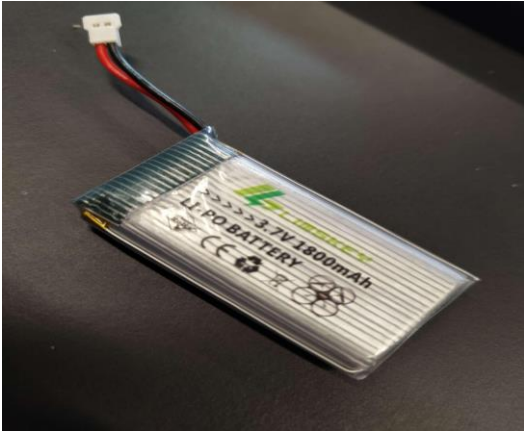
Figure 4

Den endelige arkitektur på den for indlejrede systemer som ses ovenfor. Raspberry Pi er ikke en del af denne figur, da dens funktion er at simulere en desktop / server. Da disse systemer kan variere drastisk, bliver der ikke taget meget holdning til dens skematik.

Til sammenligning af det ældre pathfinding til det nyere, kan man se at der er blevet en lang bedre struktur. Der er blevet farve kodet til vigtige linjer, som mindsker forvirring af krydset linjer, og der er dannet bedre overblik over de forskellige moduler og deres placering.

En ESP 32 Dev Kit har et maksimum strøm levering, ifølge dens datablad, på 1200 mA. Da det kun er GPS'en der får suppleret dens strøm fra ESP 32, så er det ikke et problem. En Esp 32 har kun 2 pins til strøm levering (et 3 volts og et 5 volts) og med 3 moduler der skal forsynes strøm, bliver dette problem undgået ved at supplere strømmen eksternt til både DHT11 og MQ135. Den forventede levetid på 6,85 er langt over minimumskravet til projektet.

7.3 Teori af relevante elektroniske/programmering/netværk/server arkitekturblokke

Blok	navn
Blok 1	<p>navn: LiPo Batteri</p> <p>Dette batteri har en kapacitet på 1800 mAh, og har en spænding profil der går fra 3.0 V til 4.2 V. Dens gennemsnitlige spændings niveau ligger på 3,7 V</p> <p>Batteriet kan max. aflade med 2 C, hvilket svarer til 3600 mA, som er langt under løsningens forbrugsniveau på 230 mA.</p> <p>Batteriet er 52 mm lang, 30 mm bred, 9 mm tyk, og vejer 13 gram.</p> <p>I dette projekt bruges 2 batterier til at forsyne en ESP32, gennem en LDO, ved hjælp af en serie forbindelse</p>  <p><i>Figure 5</i></p>

blok 2

navn: LDO

Output spænding 1.25V til 37V DC

Output strøm 1.5A

max input-output spændingsforskel 40V DC

Operativ temperatur 0 C- 125 C

dimensioner 53mm x 22 mm x 15 mm

LDO'ens formål er at reducere strømmen fra den serielle forbundet batteri kilde fra dens gennemsnitlige spænding fra 7,4 til 5 volt, da flere af vores komponenter har 5 volt som deres operative spænding

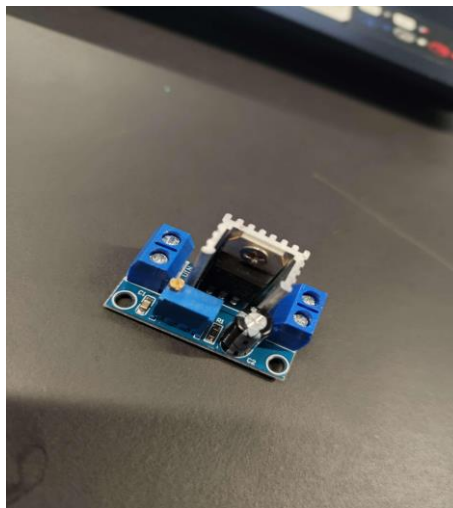


Figure 6

Blok 3

navn: ESP32

Forsyningsspænding 5 V

IO Spænding 3.3V

Flash hukommelse 4 MB

SRAM 520 KB

Clockfrekvens 240 MHz

ADC input pins 16

DAC output 2

PCB Størrelse 51.52 x 25.04 x8.54 mm

ESP'ens formål er at håndtere sensor data (GPS, DHT11, MQ135) og videregive den information til en ekstern bruger (Raspberry-pi serveren) database

Det er den i stand til da den har både wifi, bluetooth og ethernet evner.

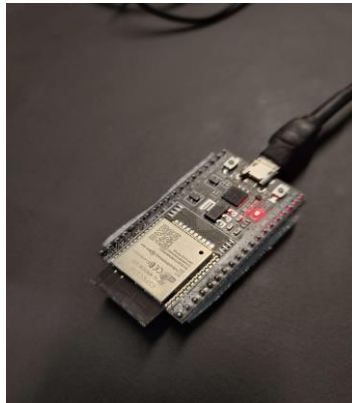


Figure 7

<p>Blok 4</p>	<p> navn: DHT 11 operativ spænding 5V Temperatur 0-50 c fejl med ± 2 C Fugtighed 20-90% RH $\pm 5\%$ RH fejl sampling periode 1 sekund interface digital DHT11 sensoren har evnen til at måle luftfugtigheden i et område, sammen med temperaturen, og er derfor rigtig vigtig i forhold til løsningen af vores problemformulering. </p>  <p><i>Figure 8</i></p>
<p>Blok 5</p>	<p> Navn: MQ135 Operativ spænding 5V Strømforbrug 150 mA Varmemodstand $31\Omega \pm 5\%$ Varmeforbrug $\leq 800\text{mW}$ Sansning modstand $30\text{ K}\Omega - 200\text{ K}\Omega$ Detektionsgrænse 10 - 1000 ppm Interface Analog, Digital MQ135'ens primære funktion er at måle kulstof niveauet i luften. Samtidig med DHT11's evne til at måle temperatur og fugt, kan denne kombination af moduler i teorien måle forudsætningerne for naturbrande, og er derfor et vigtigt led til løsningen af vores problemformulering. </p>



Figure 9

Blok 6

GPS Modul

Operativ spænding 3-5V

Interface UART

Default Baud Rate 9600 bps

PCB dimensioner 25mm x 35mm

Antenne dimensioner 12mm x 12mm

kabel længde 20 mm

GPS modulets formål i løsningen er at kunne fremvise geografisk data. Hvis et netværk af vores løsning var i brug i en skov, så ville evnen til at tracke dem via satellitdata være uvurderlig. Denne information kan også give præcise lokationsdata til risiko zoner for brande



Figure 10

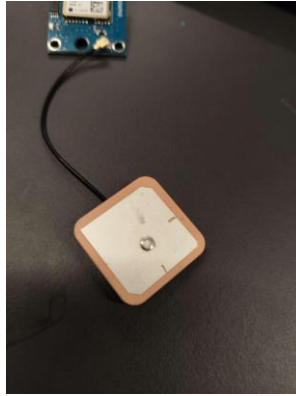


Figure 11

Blok 7

Navn: Raspberry PI 3+

Broadcom BCM2837B0, Cortex-A53, 64-bit SoC @ 1.4 GHz

Hukommelse: 512MB LPDDR2 SDRAM

2.4 GHz og 5 GHz IEEE 802.11.b/g/n/ac wireless LAN,

Bluetooth 4.2/BLE

gigabit Base Ethernet

40-pin extended GPIO

1 USB 2 ports

1 × full size HDMI, MIPI DSI display port, MIPI CSI camera port, 4 pole stereo output og composite video port

touchscreen display

Micro SD port for loading your operating system and storing data

5 V/2.5 A DC via micro USB connector / 5 V DC via GPIO header

Raspberry Pi 3+, træder i kraft som en server til vores løsning. Den simulerer en bruger der huser data'en som vores ESP32 sender.

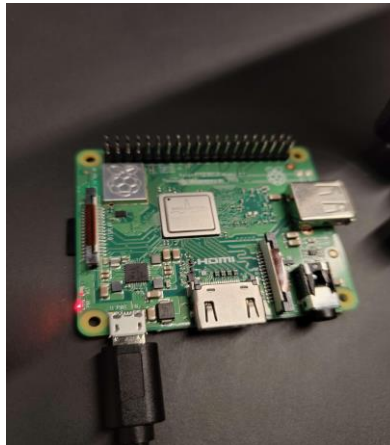


Figure 12

8. Løsningsdesign

8.1 Elektriske skematikker, målinger og udregninger (Indlejrede systemer)

Hvis I har kunne lave en elektrisk skematik af jeres løsning (på DUT niveau), så kan I med fordel indsætte den her. Har I beregnet komponentværdier, strømforbrug og målinger så indsættes det også her.

DHT11 Luftfugtighed og temperatur sensor	Strømforbruget i mA ligger mellem 0,5 - 2,5 mA. Luftfugtighed sensoren kan operere på både 3V og 5V
MQ 135	Strømforbrug i mA er 150 mA. Den operative spænding er på 5 V
GPS	Strømforbruget i mA er på 100 mA. Vores GPS modul har en fleksible operativ spænding på enten 3,3 V eller 5 V
ESP32	En ESP32 har et strømforbrug på 5mA, og en operativ spænding på enten 3,3 V eller 5 V

Med ampere målene sat på plads, kan der nu udregnes den teoretiske levetid på vores serielle forbundet batterier

Vi ved at vores LiPo batterier har en maks strømmåling på 1800 mA og en spænding på 3,7 V. Tallene fra vores komponentliste lægges så sammen (mA) og divideres med strømmen på vores batteri. Da det er et serieforbundet batteri, så fordobler vi desværre kun vores spænding, da mange af vores komponenter har en operativ spænding på 5 V. Med vores batteri og dens sølle 3,7 V, kan det ikke opnås.

Det samlede strømforbrug i mA på vores komponenter ligger på 262,5 mA (et ca beløb) og den forventede levetid på løsningen er 6,85 / 6½ timer.

8.2 Kode beskrivelse (Programmering)

Boot.py



```
boot.py x main.py x measure.py x
2  ###Author: NyboMønster
3  ### Imports fra ESP'en selv
4  import esp
5  esp.osdebug(None)
6  #import webrepl
7  #webrepl.start()
8  import network
9  import time
10 import ubinascii
11 import machine
12 import micropython
13 import gc
14 import random
15 gc.collect()
16 ### Imports fra scripts
17 from umqttsimple import MQTTClient
18 #from CreditsInformation import creditsInformationList
19 ##Variables:
20 cooldownTimer = 5
21 last_message = 0
22 counter = 0
23 message_interval = 5
24 ### Credits:
25 ssid = 'HotSpot'
26 password = 'Daniel2901Nybo!'
27 RPI_IP_Add = '192.168.239.54'
28 ESP32_IP_Add = '192.168.239.88'
29 port = 1883
30 client_id = f'python-mqtt-{2901}'
31 #client_id = ubinascii.hexlify(machine.unique_id())
32 topic_sub = b'Received'
33 topic_pub = b'measuredData'
34 topic = "ESP32Data"
35
36 ### MainCode
37 #creditsInformationList()
38 station = network.WLAN(network.STA_IF)
39
40 station.active(True)
41 station.connect(ssid, password)
42
43 while station.isconnected() == False:
44     pass
45 print('Connection succesful')
46 print(station.ifconfig())
```

Figure 13, Boot.py

Første step i ESP32'en er, når den starter så loader den boot. I boot startes der med at importere følgende moduler og scripts som indlæses i linje 4 - 16. Modulerne i boot holdes stadig aktive og kan tilgås i alle scripts på ESP'en. De vigtigste er ESP, Machine, Network, MQTTClient, Ubinascii. Disse gør at vores ESP kan fungere og tage input fra pins og

connect til netværket og sende til en specifik IP-adresse og portnummer. Det vigtigste variable er i Credits listen, hvor netværket og specifikation til forbindelsen af RPI og ESP32 og netværket oprettes, alle angives i strings, undtagen portnummer, fordi modtagerne RPI kan ikke modtage data hvis det er en string, så derfor skal portnummeret var i integer format.

Topic og client-ID er idét på den forbindelse der skabes mellem RPI og ESP'en, her sættes et ens client ID, på den MQTT forbindelse som laves, og et topic. Dette gøres fordi MQTT opretter en forbindelse som kan sende og modtage data mellem 2 enheder, Topic bruges til at identificere hvilken specifik ESP det er hvis man sætte flere til. Client_iD, bruges til at sætte port nummeret på den forbindelse som oprettes fra ESP'en.

I main Code, oprettes en variabel som bruger funktionen fra netværk, her sættes den til aktiv, og efterfølgende sættes netværksforbindelser navne og passwordet til den variable. Hvis den går ind og tjekker at forbindelsen er falsk, stopper den med at prøve at oprette forbindelse, til sidst printer den at connection var succesfuld, og konfiguration på forbindelsen, særligt IP adressen og beskeden der er sendt. (Mkyong, 2018)

Main.py

```
[ boot.py ] [ main.py ] [ measure.py ] [ MQlib.py ]

1  #! /bin/python3
2  ### Author: NyboMønster
3  ###Main File
4  ### Imports fra system
5  import sys
6  import random
7  import time
8  ### Imports fra costumscripts
9  from measure import measuredData, message
10 from umqttsimple import MQTTClient
11 ### Variables
12 username = 'RPI'
13 password = 'public'
14 broker = 'broker.RPI.io'
15 counter = 0
16 ### KEA E bygning grønomsråde
17 msg = ("t, h, s, 55.69194647082459, 12.554169821375735")
18 Originmsg = ("t, h, s, 55.69194647082459, 12.554169821375735")
19 ### Main Functions
20 def sub_cb(topic, msg):
21     print((topic, msg))
22     if topic == b'ESP32Data' and msg == b'received':
23         print('ESP Sendt Data')
24
25 def connect_and_subscribe():
26     global client_id, RPI_IP_Add, topic_pub, topic_sub, topic, port
27     client = MQTTClient(client_id, RPI_IP_Add)
28     client.set_callback(sub_cb)
29     client.connect()
30     client.subscribe(topic)
31     print('Connected to %s MQTT broker, subribed to %s topic' % (RPI_IP_Add, topic))
32     return client
33
34 def restart_and_reconnect():
35     print('Failed to connect to MQTT broker. Reconnecting...')
36     time.sleep(5)
37     machine.reset()
38
39 try:
40     ### StartUps
41     client = connect_and_subscribe()
42 except OSError as e:
43     restart_and_reconnect()
```

Figure 14, Main.py, Først del

```
39  ### Main Loop
40  while True:
41      try:
42          client.check_msg()
43          if(time.time() - last_message) > message_interval:
44              msg = message
45              client.publish(topic, msg)
46              last_message = time.time()
47              counter += 1
48      except OSError as e:
49          restart_and_reconnect()
50      except TypeError:
51          print("Type error somewhere in code, functional")
52      except:
53          print("STOP!")
54          exit
55          time.sleep(2)
```

Figure 15, Main.py, Anden del

Formålet med vores main file er at tjekke om forbindelsen stadig er aktiv og sende en besked i gennem forbindelsen. For at opnå dette importerer vi igen modulerne, som kan ses på første del af main filen på de ovenstående figure(linje 5-9). For at få hentet noget data at sende, køre vi funktion fra measure, som hedder measuredData. Efterfølgende har vi variable sat op med credits på RPI'en som ESP'en skal sende til, sammen med en counter så vi kan se hvor mange beskeder der er sendt.

Efter dette skabes 3 funktioner, en funktion som bruges til troubleshooting, hvis ESP'en modtager et feedback fra subscriberen om at den har modtaget beskeden, printer den at beskeden er modtaget. Nummer 2 funktion er at den tjekker om forbindelsen er stadig aktiv og connector til Subscriberen. Der startes med at lave nogle variable global, så indholdet kan hentes. Efter sættes MQTTClient med de information som blev hentet i linje 24, sat til en variable kaldt client. Client bruges efterfølgende igen til at connect og sende den subscriber eller fremsende en forbindelse som er topic, som var sat i boot til at være "ESP32Data".

Sidste funktion i main er lavet til at oprette forbindelse hvis forbindelsen er død og skal starte. Først forsøger den at sende til client variablen, ellers køre den funktionen "restart_and_reconnect"

Til sidst har vi vores main loop, som holder alt kørende. Først forsøger den at køre client.check_msg hvilket gør at den tjekker om der er nogle beskeder som ikke er hentet endnu af ellers køre den en wait_msg funktion.

Hvis tiden imellem har været højere end 5, så køre den følgende if statement, den sætte message fra measure modulet, til msg variablen, efterfølgende køre den client.publish funktionen fra MQTTClient klassen, fra umqttsimple modulet, her giver den topic

“ESP32Data” og msg “t, h, s, x, y” for Tempature, humdity, smoke, Xlocation, Ylocation.
Her sætter den last_message variabelen til tiden, og increment variabelen counter, med 1.

Tilafslutning af main loop, fordi loop’et skal kunne stoppes, der er sat 2 excepts ind, hvis der midste forbindelse køre den igen, restart funktionen. i tilfælde af typeerrors printer den at der er typeError,

Hvis den skal stoppe loopet, printer den en STOP!, og exit, og sleep for 2 sekunder for at man har tid til at anvende den standard CTRL + C. (Mkyong, 2018)

Measure.py

```
[ boot.py ] [ main.py ] [ measure.py ] [ MQlib.py ]
1  #!/bin/python3
2  ### Author: NyboMønster
3  ###Imports fra system
4  from machine import Pin
5  import dht
6  import random
7  #import MQlib
8  ###Variables
9  dhtsensor = dht.DHT11(Pin(25))
10 #mq135 = MQlib.MQ135(Pin(26))
11 # Location: KEA Parkerings grønareal
12 message = ("t, h, s, 55.69194647082459, 12.554169821375735")
13 message.split(", ", 5)
14 ###MainCode
15 def measuredData(message):
16     dhtsensor.measure()
17     t = dhtsensor.temperature()
18     #print('Temperatur: %3.1f C' %t)
19     h = dhtsensor.humidity()
20     #print('Humidit: %3.1f P' %h)
21     cb = mq135.get_corrected_ppm(t, h)
22     #t = random.randint(1, 100)
23     #h = random.randint(1, 100)
24     #cb = random.randint(1, 2000)
25     #print('Carbondioxide: %3.1f CB' %cb)
26     #print(t,h)
27     if (t is not None) and (h is not None) and (cb is not None):
28         print('Temperatur: %3.1f C' %t)
29         print('Humidit: %3.1f P' %h)
30         print('Carbondioxide: %3f Corrected PPM' %cb)
31     else:
32         print("Invalid sensor readings")
33     Swaped = str(t)
34     newmsg = message.replace("t", Swaped)
35     Swaped = str(h)
36     newmsg = newmsg.replace("h", Swaped)
37     if cb > 2500:
38         newmsg = newmsg.replace("s", "True")
39     else:
40         newmsg = newmsg.replace("s", "False")
41     print(newmsg)
42     message = newmsg
43     return message
```

Figure 16, Measure.py

Formålet med dette script er at indhente dataen fra vores moduler vi har sat til vores ESP32, her under DHT11 og MQ135, som importeres i linjerne 4- 6. Hvorefter vi sættes variablerne for scriptet, fra linje 7-13. Særligt her, har vi hardcoded lokationen af ESP'en fordi den skal alligevel side på et træ, så vi valgte at fravælge GPS modulet, og spare på komponent budgettet. Vores main kodes funktion er at kalde funktionerne fra modulerne vi har importeret, specielt DHT og MQ135. Disse funktioner giver os, målende på vores hoved information vi skal bruge, temperature, humidity, carbon i ppm. (linje 14 -43)

Der er indsat et "if statement" som tjekker at DHT11 og MQ135 faktisk har målet noget, så de tjekker om der er ikke noget, i variablen, og hvis der er noget så køre den funktionen ellers printer den "invalid sensor readings"

Linje 34 oprettes en local variable som bruges til at indsætte målingerne i den text variable kaldt "message" som skal sendes. newmsg bruges som en local variable til at holde de ny ombyttede variable med. hvor Swaped erstattes for hver gang der skal indsættes noget nyt i variablen.

I linje 37 er der indsat en der fortæller om det brænder eller ej, måleren er sat til 2500 ppm, fordi standarden i luften burde ligge mellem 250-400, efter testning kan dette ændres afhængigt af test klimaet.

Linje 43, tager data'en ud af vores funktion og sætter den til en variable, som vi kan sende over til main, for at blive sendt til vores RPI.

RPI-Kodning

I følgende delafsnit vil dokumentation af vores RPI fortsætte. Her vil vi dokumentere, hvilke scripts kører på vores RaspberryPi (RPI). Der er inkluderet en installations gennemgang af de steps, der skal til for at få Mosquitto client til at virke på en RPI. Formålet med dette er dokumentation for hvordan vores opsætning er lavet, også selvom installation ikke nødvendigvis er en kode, er det stadig en kommando med prefixed installations instrukser som udføres.

Subscriber.bash

```
$ FinalSubscriber.bash X
$ FinalSubscriber.bash
1  #!/bin/bash
2  ##Variables
3  messages=""
4  pastmessages=""
5  testmessages="123.4, 123.4, False, 123.4, 123.4"
6  SentMessages="Messages sent"
7  DataBase="/home/NyboMønster/IOT2-Projekt/RPI/measuredData.db"
8  IFS=', '
9  ##Functions
10 while :
11 do
12 mosquitto_sub -h localhost -t "ESP32Data" -C 1 -R $messages | while read TEMP HUMD SMOKE XLOCATION YLOCATION; do
13     echo $TEMP $HUMD $SMOKE $XLOCATION $YLOCATION
14     sqlite3 $DataBase "INSERT INTO measuredData(TEMP,HUMD,SMOKE,XLOCATION,YLOCATION) VALUES ($TEMP,$HUMD,$SMOKE,$XLOCATION,$YLOCATION);"
15     echo $SentMessages
16 done
17 done
18
19 |
```

Figure 17, FinalSubscriber.bash

Scriptet her er kodet i bash, dette kan ses på først linje fordi, alle scripts på et linux system bruger, #! til at se, hvor den kompilere de skal køre følgende scripts med ligger hen, det er derfor meget vigtigt at path'en er 100% korrekt.

På linjerne 3-7 oprettes variablerne. # bruges til at kommentere ting med, i undtagelse fra først linje hvor det bruges til at initialisere kompileringen. Fra linje 10-17 har vi main loopet, som kaldes ved "while : " "do" og "done". Det er vigtigt, at der er et mellemrum imellem while og : fordi ellers kaldes den som en forkert variable, eller errorvariable.

Fordelen ved bash er, at man kan indsætte kommandoer, som man normalt ville bruge i terminalen, hvilket gør det nemmere at arbejde med. kommandoen henter en funktion som tager argumenterne -h for hosting, og localhost er en global variable, for RPI'en IPadresse, i vores tilfælde (192.168.239.54). hvor den så lytte for et topic med argumentet -t som er kaldt "ESP32Data". I vores tilfælde skulle vi kun bruge kommandoen 1 gang hvilket vi kan gøre ved at bruge argumentet -c for "count" og -R er brugte for om indholdet skal læses og hvilken variable det skal indsættes i.

Bash fungerer ved, at den kan læse, hvad den selv har printet, hvilket er det vi gør i linje 12, efter absolute tegnet, her læse den outputtet som kommer fra linje 12. Den sætter det så til variablen i linje 3 med "\$messages" alle variable efter printer den så den nye data, med echo messages, hvor den så afslutter loopet med done på linje 16.

Linje 14 har vi så den endelige kommando som indsætter i vores sqlite3 DataBase, den henter navnet igennem den absolutte path fra variablen. og begynde indsætte data'en. Den tager så vores variabler som er angivet med \$ foran fordi det er variabler som indeholder vores nye ændringer igennem scriptet og ikke bare de standard hardcoded værdiere. til sidste printes i shell at data'en er indsat, og loopet stoppets på linje 17. (GNU, 2022) (SQLite, 2023) (sqlitetutorials, 2022)

mainPage.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>IOT</title>
8   <link rel="stylesheet" href="Style.css">
9 </head>
10 <body>
11   <h1>FOREST SAVER</h1>
12   <nav>
13     <div class="link">
14       <a href="mainPage.html">MAIN</a>
15     </div>
16     <div class="link">
17       <a href="graph.html">GRAPH</a>
18     </div>
19     <div class="link">
20       <a href="table.html">TABLE</a>
21     </div>
22   </nav>
23   <div class="info">
24     <h2>INFO</h2>
25     <p>Naturbrände har en meget stor negativ effekt på vores klima i dag. Ikke nok med at der udledes CO2 så mister vi også skove og områder med planter som man fotosyntese
26     <strong>Forest Saver is the way to go</strong>
27   </div>
28 </body>
29 </html>
```

Figure 18, mainPage.html

I toppen af vores mainPage.html har <html> som er fil formatet så alt inde for det tag bliver loaded med siden. Vores <title> tag definerer hvad siden kommer til at hedde, og vores <link rel="stylesheet" href="Style.css"> linker vores hjemmeside med vores Stylesheet (CSS) og gør at det bliver loaded med siden. Under der har vi vores <body> tag som er der alle vores fysiske bokse som bliver det man kan se når siden loader.

<h1> som står for overskriften på siden. Vi har lige under en navigation bar, som har tagget <nav>. Det er et tag, der bliver brugt, for at indikere, at det er en navigation bar. Inde i den er der 3 tags. Tags er en måde at linke på fx. til en anden hjemmeside. Den følger, hvad der står i dens href mellem de 2 anførselstegn. I vores tilfælde fører den til table.html og graph.html.

Under har vi et <h2>, som fungerer lidt ligesom vores <h1>, men i stedet for at være en overskrift er det en underoverskrift. Under der har vi et <p> tag, som står for paragraph. Det er et tag, som bliver brugt, som din normale tekst til fx. paragraffer.

table.html

```
26 <?php
27 include('SQLite3Importer.php');
28 ?>
29 <div class="table">
30 <div class="table_item">ID</div>
31 <div class="table_item">Temperature</div>
32 <div class="table_item">Humidity</div>
33 <div class="table_item">Co2</div>
34 <div class="table_item">Location X</div>
35 <div class="table_item">Location Y</div>
36 <div class="table_item"><p id="id1"></p></div>
37 <div class="table_item"><p id="t1"></p></div>
38 <div class="table_item"><p id="h1"></p></div>
39 <div class="table_item"><p id="s1"></p></div>
40 <div class="table_item"><p id="x1"></p></div>
41 <div class="table_item"><p id="y1"></p></div>
42
43 </div>
```

Figure 19, table.html

Vores table.html ligner meget vores mainpage, men det er uden <p> tagget, hvor at der i stedet er en <div> med class table, som bliver givet display: grid; i vores style.css og en masse <div> med class table_item og de individuelle har fået id id1, t1, h1, x1, y1.

graph.html

Vores graph.html ligner meget de 2 andre men med en <div> med class graph indsat i stedet for at tage imod vores graph.

```
23 <div class="graph">
24
25 </div>
```

Figure 20

style.css

I vores style.css, har vi startet med en reset på de fleste ting.

```
1 /* Resets */
2 html, body{
3     max-width: 100%;
4     font-style: none;
5     box-sizing: border-box;
6     background-color: rgb(208, 252, 228);
7     font-family: 'Times New Roman', Times, serif;
8 }
9 a{
10     text-decoration: none;
11     color: black;
12     font-size: 2rem;
13 }
```

Figure 21, Style.css

Max-width gør, at body ikke kan blive mere end 100% af sidens bredde. Font-style: none gør, at alt tekst ikke kommer til at være kursiv. Box-sizing: border-box; gør, at padding ikke kan vokse ud fra dens box. Font-family: bestemmer hvilken font vi bruger på siden, med mindre der er undtaget i andre classes. Fonterne er sorteret fra venstre til højre efter prioritet, så hvis din pc ikke supporter fx. "Times New Roman" ville den så tage "Times" som er dens sekundære font.

Text-decoration: none gør at alle links får fjernet deres underlinje. Color black gør så at den bliver sort, så den ligner normal text. Font-size er størrelsen på teksten i tagget og rem er default størrelsen på ens browser (flest tilfælde 16 pixels) 2 rem er så 32 pixels. Ved at lave den her "reset" gør det, det lidt nemmere, at sætte siden op, da det gør nogle ting er sat op til at tilpasse siden, så den ser lidt bedre ud og gør det nemmere for den, som sætter siden op, da det modarbejder nogle lidt irriterende problemer, som kunne dukke op, hvis man ikke havde sat dem op på forhånd.

```
14 /* Global */
15 nav{
16     display: flex;
17     margin: auto;
18     font-weight: bold;
19     font-family: sans-serif;
20 }
21 h1{
22     text-align: center;
23     margin: auto;
24     font-size: 3rem;
25     font-family: sans-serif;
26 }
27 h2{
28     font-size: 2rem;
29     text-align: center;
30     font-family: sans-serif;
31 }
32 .link{
33     display: block;
34     margin: 30px auto;
35 }
36 a:hover{
37     color: green;
38     transition: 1s;
39 }
```

Figure 22

Her er vores "global", som kommer til at være noget, som er universelt for alle siderne. Display: flex i vores nav gør at alt som er i vores <nav> bliver lagt på samme linje. Margin auto gør at den giver en automatisk margin på alle sider så den bliver centreret. Font-weight: bold gør at den tekst som er i tagget bliver **bold**. Text-align: center gør at teksten bliver sat i midten af den boks som den er i. Font-family i deres egne tags gør at de får en anden font er den som var i vores "reset". Display: block gør at den får properties til at det er en block ligesom et image og gør at man kan give den margin: auto for at centrere den. Første tal efter margin er top og bottom og den anden er venstre højre. Margin er en property som skubber det tag som den er i. A:hover gør at det der står i tagget sker når man hover over den med ens mus. Transition: 1s er hvor lang tid det tager for den at transition og 1s står for 1 sekundt.

```
40  /* Main */
41  .info{
42      margin: 0 auto;
43      width: 800px;
44      line-height: 1.5;
45      font-size: 1.3rem;
46  }
47  p{
48      text-align: justify;
49  }
```

Figure 23

I vores “main” har vi klassen info som linker til en div i vores main side og giver den margin 0 auto. Width: 800px gør at tekst boksen er 800px bred. Line-height: 1.5 gør at der bliver en linjeafstand på 0.5x den normale linjeafstand.

Text-align: justify gør at linjerne i vores tekst bliver mere aligned for enden og kan skabe nogle lidt større mellemrum for at aligne dem for enden. Det er kun for designmæssige årsager.

```
52  /* Graph */
53  .graph{
54      width: 60%;
55      height: 500px;
56      margin-left: auto;
57      margin-right: auto;
58      border: 2px solid black;
59  }
```

Figure 24

I vores graph har vi en class som linker til en div i vores html og giver den en width på 60% så den er på bredde 60% af siden og en height på 500px. Den har margin left og right auto så den er centeret uden at margin i toppen eller bunden. Til sidst er der en border på 2px som er solid black. Det skaber en firkant rundt om vores graph div.

```
60 /* Table */
61 .table{
62     display: grid;
63     grid-template-columns: 10% auto auto auto auto auto;
64     width: 60%;
65     margin-left: auto;
66     margin-right: auto;
67 }
68 .table_item{
69     border: 1px solid black;
70 }
```

Figure 25

I vores table har vi 2 classes som linker til 2 forskellige <div> i vores html. Vores table bliver den ydre div som sætter det hele op i system og table_item bliver det som kommer til at fylde den ud. Display grit skaber et array af bokse som bliver sat op i grid-template-columns. I vores grid-template columns giver vi dem deres størrelse og hvor mange bokse der skal være til at fylde den ud. Den første længde er 10% da den skal være mindre i de andre. De andre er sat til auto, så de bliver lige bredde og automatisk tilpasser sig bredden af siden. Dens width gør at den er 60% af sidens bredde og margin left og right sat til auto centrere den. Vores table_item er dem som kommer til at fylde ud i vores grid array. De er blevet givet en border på 1 pixel som er solid black som giver det et grid look når det er sat sammen.

9. Test af løsning

9.1 Pathfinding testresultater

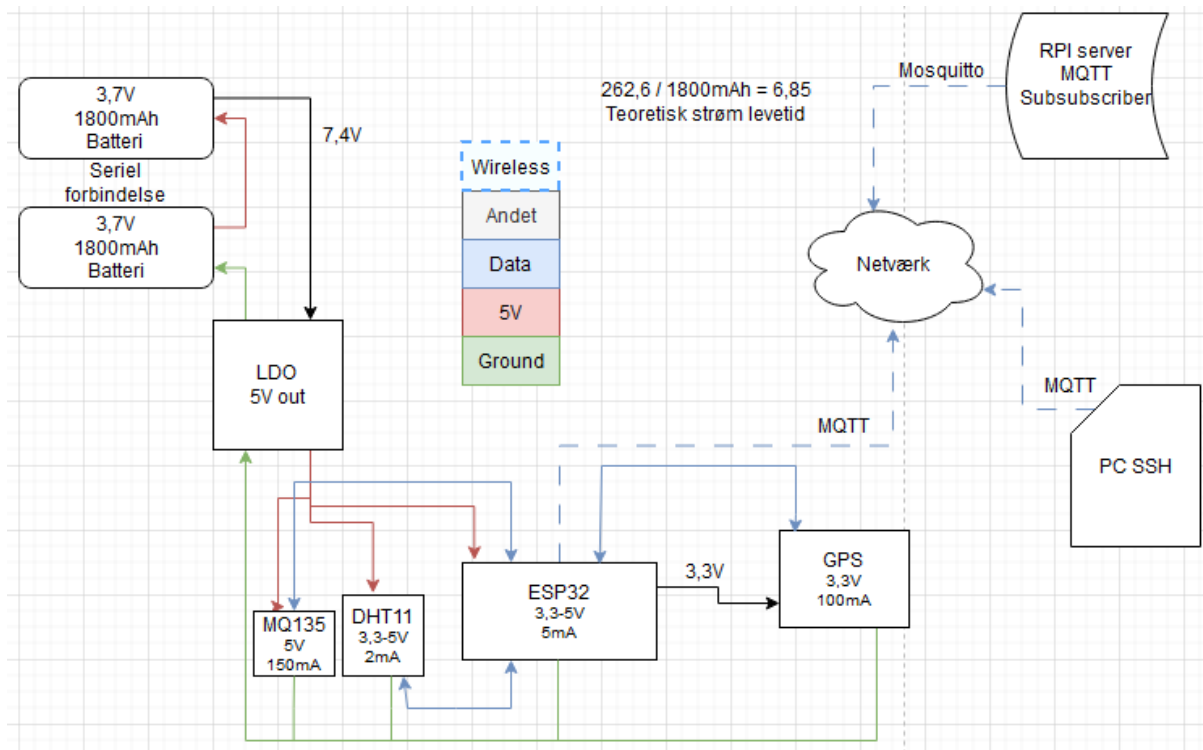


Figure 26

Under vores test opsætning vil vi anvende et lokalt netværk, i vores tilfælde et mobilt hotspot som skal virke som vores internet udbydere. Det vigtigste for vores løsning er at de alle bare er på samme netværk. Vi vil i denne test sætte særligt fokus på vores ESP32 der har forbindelse med UART, og Onewire til vores hhv. GPS modul og vores DHT11 sensor. Vores MQ135 sensor er sat til for at simulere et større strømforbrug.

9.2 Dokumentation af DUT

DUT: Device under testing.

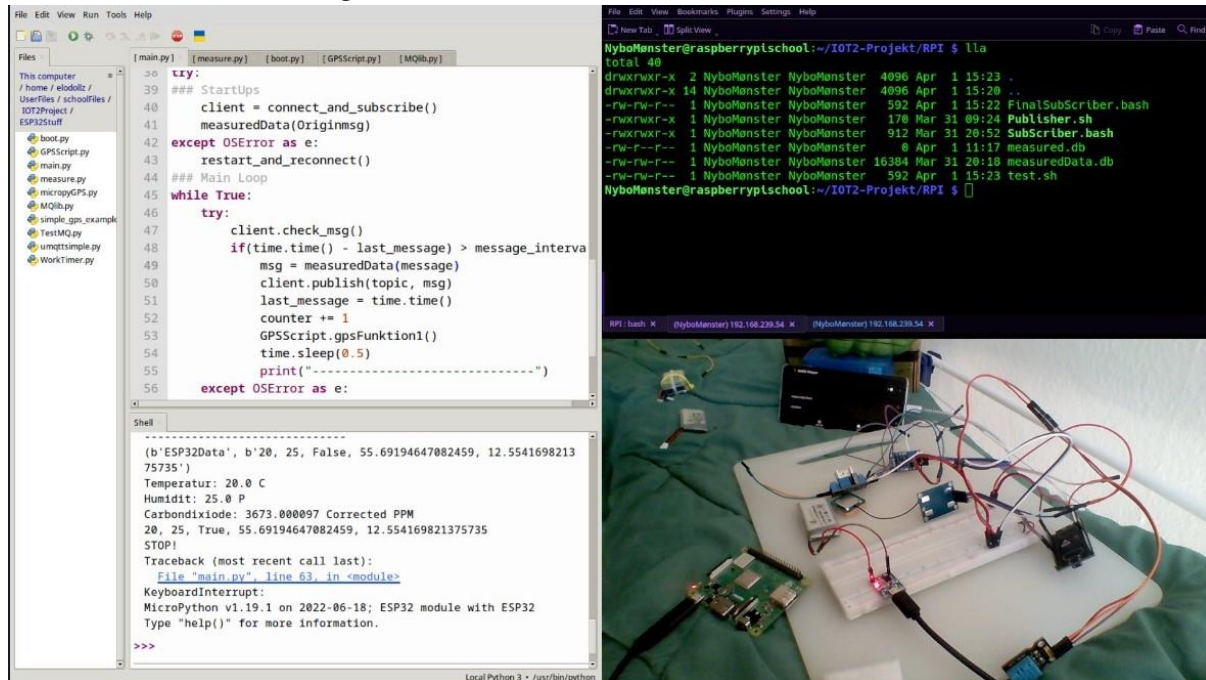


Figure 27, DUT

Opsætning skal startes med at sikker, double tjek at kablerne til ESP'en er sat rigtigt i, i vores tilfælde skal signal kablet i Pin 25, fra DHT11 målerne (den orange). Alle kabler er farvekodet røde kabler er positive spændinger, og brune er negative/ground spændinger, med den ene undtagelse af vores MQ135 som er blå til hvid som er positiv spænding, og lilla og sort er negative spænding. MQ135 målene sættes i Pin 26, Analog Pin, til ESP32'erne. Både DHT11 og vores MQ135, modtager 5 V spænding, hvilket er grund til der er farvekodet Rød og grund, på vores opsætning. (figur 27)

Vores GPS module er sat på med Rx til pin 16, og vores Tx Pin til 17. TX pin 17 er det lilla kable, og vores RX kable til Pin 16, er det grå kable. Her til GPS modulet er der taget andre farvekode fordi det er 3,3 V det forsynes med, i dette tilfælde har vi Blå til Ground og Hvid til 3.3 V spændingen.

Under vores opsætning vil vi have koblet vores ESP32 til vores computer, så vi kan læse outputtet i Shell terminalen. Vi forsøger så vidt muligt at simulere at vores løsning skal være operativ i de krævet 6 timer, så under denne test opstilling vil vi kunne teste i 4 timer med hvor det så kun er MQ135 der tager strøm fra vores MQ135, og hvis vores løsning løber tør for strøm vil vi anse testen som fejlet. (dette er gjort af hensyn til ESP32'en transisterne ikke bliver fryet, hvis man sætte både 5V fra bat, og 5V fra computerne kortsletter man sin ESP, så for testning muligheder vi har lavet denne midlere tidige ændring, for at kunne udføre testen og måle data'en i en periode af 2 timers testning. (Se bilags mappen, Bilag 10, Video af test session 4 timer)

Table:	Nuværende:	Ønsket Stadie:
Højde:	Nuværende: cirka 30 cm	Ønsket Stadie: cirka 15 cm(pc med 60 cm)
Vægt:	Nuværende: cirka 400g+pc 2.8kg	Ønsket Stadie: cirka 250g (+pc 2.8 kg total)
Indgangsspænding:	Nuværende: V3.3 og 5V	Ønsket Stadie: V3.3 og 5 V
Netværksprotokoller:	Nuværende: MQTT	Ønsket Stadie: MQTT
sikkerhedsfunktioner:	Nuværende:stop ved fatal	Ønsket Stadie: Stop hvis fatal error detected

9.3 Udførsel af brugertest

Dato: 01/04/2023 – kl:13:00

Test beskrivelse og resultater: se billag 12, billede af opsætning og billag 10, video 4 timers test af opsætning

Beskrivelse:

Løsning blev testet i et indendørs miljø, 1 meter fra vinduet, på et kollegieværelse for at simulere vind effekt på miljøet. Vores løsning er sat op på et plastik bord (et bredt køkken, skæbræt af plastik) med et tæppe under for at gøre afskaffelse muligheden nem og lukke for evt brandfare. Vores løsning testes ikke i vores kasse som skal være den endelige produkt for at gøre inspektion nemt, og tilgængeligt. Et stopur startes med 2 timer, og på samme tid startes ESP'en og RPI modtager koden startes. Test udføres med 2 lipo batterier, med samlet 7.4 V reduceret til 5V leveret til MQ135 og DHT11 sensoren. Batterierne skal være fuldt opladt til teststart.

Formålet:

Teste om løsning kan holde strøm til MQ135 i 4 timer, og om vores løsning kan måle data, og sende den til en RPI. under denne testning kommer vi til at connecte til vores RPI med ssh på dens ip address (nuværende 192.168.239.54). Testen kommer til at være fokuseret på at teste kravene i ID 1 og 2, hvor data'en skal hentes fra DHT11 målerne og gemme dem i en variable, for ID 2 skal den kunne printe data'en i en liste i shell med X og Y koordinater. For ID 1 skal den kunne connect til internettet. Dette testes ved om den printes og, i sessionen ses det om den kan holde stabil forbindelse i mindst 10 minutter af gangen, uden at have behov for at reconnect. Vi sætter i denne test session mindst 10 minutter af gangen, fordi testen foregår i et storbyområde, hvorimod når den skal ud i en skov, hvor der ville være minimal forstyrrelse af andre signaler.

Resultater:

I indledning af test session ser vi at vores løsning connector til vores internet i dette tilfælde et mobilt hotspot, hvor efter scriptet på vores RPI sættes i gang for at teste om den faktiske connector eller om den kunne printe i shell at den sender noget. Allerede inden for de første 2 minutter er der oprettet forbindelse og de første beskeder sendt igennem netværket og modtaget af vores RPI. Allerede grundet dette ville vi vurdere vores test successful, siden vores mål og krav var kun at den bare skulle connect til vores internet og måle DHT Data. Vi fortsætter dog vores test til de fulde 2 timer i første test session her, for at kunne sikre også at den kan gen forbindes til internettet hvis den skulle blive uforudset afbrudt. vi gør særligt dette fordi det vil være meget upraktisk hvis man manuelt skulle ud i en skov for at restart enheden, hvis den mister forbindelsen.

Dato: 1/04/2023 – kl: 18:00

Test beskrivelse og resultat: Se Bilag billede 12, og bilag video 15, video af testing session 2

Beskrivelse:

Vores opsætning følger meget det samme fra test session 1, med primær forskel her hvor vi vil forsøge at måle ppm, og se om der bliver indsat data i vores SQLite3 database. Her er MQ135 måleren sat til Pin 26, med analog pin. Under denne test session kan det blive nødvendigt at frakoble MQ135 målerne, efter som den i tidligere test session har givet problemer med resten af vores test set up. MQ135 måler ud fra de Quick-And-Dirty test vi har lavet, erfaret til at være en meget følsom sensor. Se bilags video 7 for dokumentation af at MQ135 virkede 1 enkelt gang af alle vores Quick-And-Dirty tests. Udelukkende på grund af denne for viden vil vi godtage at MQ135 frakobles og testen så kun gennemføres med at kunne indsætte data'en i vores SQLite3 Database. Hvis dette er tilfældet, må vi antage, at vores MQ135 krav til at være fejlt. Det gøres udelukkende for at vores anvender af vores løsning ikke skal bekymre sig om en enkelt måler virker, alternativt må vi finde en anden carbon måler til at måle om der skulle være røg på vores test miljø. For at test om der er røg i test miljøet, vil vi tænde en lighter ved test opsætning, lighteren holdes cirka 1 meter fra, målerne, efter 1 minut hvis ingen ændring, rykkes lighteren tættere på, gentage igen vent 1 minut, og se om der sker ændringer i målingerne. Dette gentages igen, med cirka 25 cm fra målerne, cirka 10 cm fra måleren, og cirka 2-3 cm fra målerne. Anden session vil vi anvende en røg maskine eller en vape til at teste om vores måler kan opfange den stigende ppm som sker i miljøet, med samme forudsætning som før med hhv. , cirka 1 meter fra, 50 cm, 25cm, 10 cm, 2.5 cm, vores test måler. Vape eller røg maskine er det tætteste vi kommer på en måde at teste vores løsning på, en sikker måde der ikke kræver brand tilladelse.

Formålet:

Vores formål her er at test om vores test opsætning kan modtage og printe data'en der bliver sendt over internettet. Særligt her vil der forsøges at printe indholdet af SQLite3 databasen en gang hver cirka 15-20 minut, dette bliver gjort manuelt, fordi man skal ud af databasen og ind igen før den opdateres med de nye data målinger. Her vil der holdes øje med at vores løsning kan måle en PPM værdi, og sende data over internettet.

Resultater:

Vores løsning kan indsætte data'en i vores SQLite3 database, desværre kan den dog ikke connecte med Satellitten for vores GPS, og vores hjemmeside kan heller ikke vise. Ud fra dette kan vi desværre kun sige testen er fejlet

9.4 Udførsel af Accepttest på DUT

Test af krav ID: 1, ID: 2 & ID: 3

ID:1	<p>Krav:</p> <p>ESP'en og RPI kan connect til Internettet</p>	<p>Prioritet</p> <p>: (1)</p>
<p>Kategori:</p> <p>Software</p>	<p>Accepttest:</p> <p>Der printes i shell at de connector til internettet og det kan ses på internet udbydere at de er connected. Yderligere vil vi kræve at løsning kan connecte til internettet hvis den miste forbindelsen.</p>	<p>Passed</p>

Diskussion:

Testen er gennemført, uden problemer. Vores ESP32 connector til vores internet, uden problemer. Vi kan også se at vores løsning kan genforbinde til vores internet hvis den skulle gårt varit miste forbindelsen. Her ville det være ideelt hvis vores løsning så også kunne gemme data'en og samle det målte data til en længere data samling, og sende den afsted. Ulemper dog ved dette er hvis man den længere samlet data bliver for lang, vil modtageren (RPI) ikke kunne modtage den nyere data, i mens den behandler den forrige lange data, der blev sendt. Ud fra dette vælger vi så dog at prioritere den nyere data, og må sige at den målte data imens ESP'en genforbinder går tabt.

ID:2	Krav: ESP'en Data fra DHT11 til og har en placering, med X og Y koordinater.	Prioritet : (1)
Kategori: Software	Accepttest: At den kan printet målt data fra ESP'en	Passed

Diskussion:

Under vores test session vil vi kunne at vores løsning kan fint måle vores ønskede data og printe den i terminalen. Under denne test session er koordinaterne dog statiske, grundet at vores GPS har problemer med at forbinde satellitterne. Den kan dog godt forbinde nogle af dem og modtage et klokkeslæt. Klokken er dog 2 timer bagud, fra den reale tid, dette skal tages med i overvejelserne til den endelige prototype.

ID:3	Krav: Den kan sende data, fra ESP32 til RPI	Prioritet : (1)
Kategori: Software	Accepttest: ESP'en kan sende en string af data til RPI (RaspberryPI) og RPI kan modtage data'en.	Passed

--	--	--

Diskussion:

Vores data bliver modtaget, men den går tabt og bliver overskrevet af ny data der kommer ind, dette vil vi forsøge at løse i næste test session. Men vi kan bekræfte at vores løsning er stabile og kan sende data igennem et internet, og modtages af en server host, der kører linux, i vores tilfælde kører den Ubuntu Server, 22.04 LTS, versionen.

ID:4	<p>Krav:</p> <p>MQ135 kan måle PPM og koden printer, om der er røg eller ej.</p>	Prioritet : (1)
Kategori: Kate Software	<p>Accepttest:</p> <p>At den kan printet målt data fra ESP'en og sende den til RPI, Data'en er real.</p> <p>3 stadigere:</p> <ul style="list-style-type: none"> - ingen røg (normalt klima) - lidt røg - meget røg 	failed

Diskussion:

Vi kunne ikke få MQ135 til at virke undtagen en gang, så vi kunne ikke måle om den kunne opfange noget røgen. Vi kunne ikke finde fejlen i koden, og da vi prøvede at skifte de pin som MQ135, for at se om det løste problemet, men det lykkes ikke, og vi antager at ESP 32 ikke kunne få forbindes ADC fra MQ135, da den gave og inputtet 0, så vi forestiller os at der var noget galt med pin eller forbindelsen i MQ135, da gav os det resultat selvom vi ikke brugte ADC, men digital.

ID:5	<p>Krav:</p> <p>Data'en kan sættes ind på en hjemmeside, og vises i graf og en table.</p>	<p>Prioritet : (2)</p>
<p>Kategori: Kate Software</p>	<p>Accepttest:</p> <p>Data'en indsættes i en SQLite3 Database og kan hentes, indsættes i en HTML side, i en table så man kan se målingerne.</p>	<p>Webpage-failed SQLite-Success</p>

Diskussion: Efter vores løsning her kan vi dokumentere at vores løsning kan modtage data'en vi sender, og måler dog kun med DHT11, så ergo er MQ135 test delen fejlet, efter som vores test opsætning ikke kunne fungere med MQ135 måleren sat til. Vores test opsætning kan dog godt sende data'en ind i vores Sqlite3 database.

10. Implementering af løsning i drift (eventuel)

Løsningen vil eventuel være i stand til at kunne måle CO₂ niveauet samt fugtighed og temperaturen ude i naturområder, hvor den ville sende de målinger til kundes database, hvor de er i stand til se, om der er høj sandsynlighed for at der sker en naturbrand. Hvis løsningen opdager og der kommer røg, ville den opfange det og kunne give et signal, siger at der ryger.

11. Praktisk projektplanlægning og ledelse

11.1 WBS

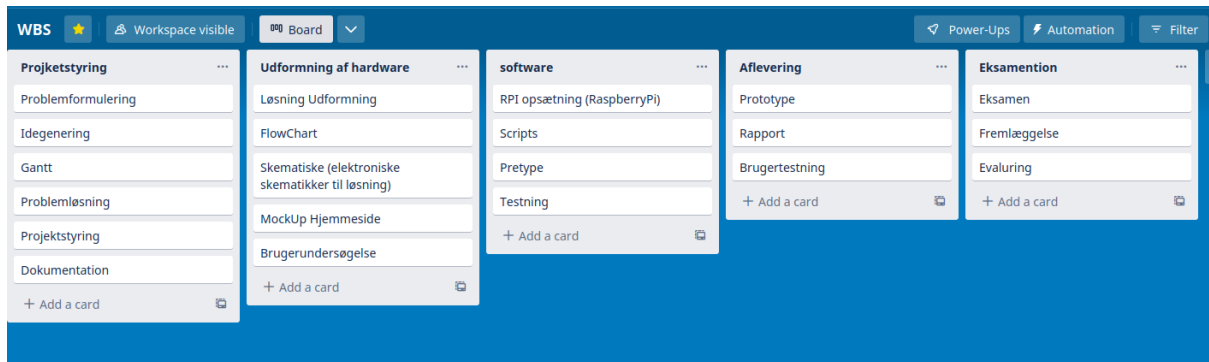


Figure 28

Til vores WBS opdelte vi det i 5 emner, fordi det gav mest mening for vores projekt, her til IOT2 projekt. Særligt vores hardware del og udformning har vi sat mest tid af til fordi det er nok det emne vi forventer vil tage længst tid.

11.2 Gantt

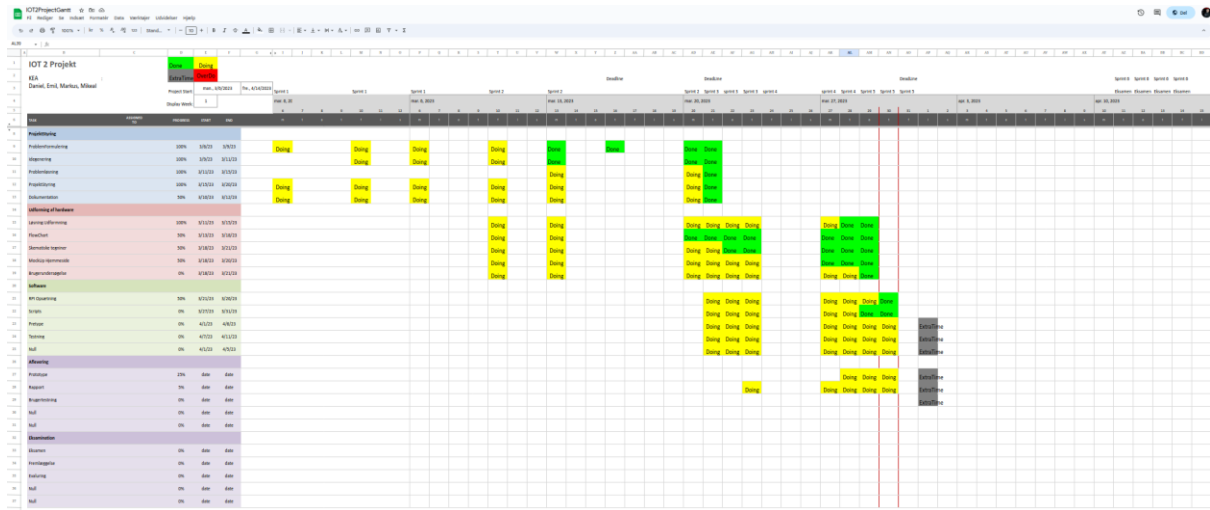


Figure 29

Vores Gantt Diagram har vi angivet 4 farve-koder, med tekst som har følgende betydning. Grøn er færdig, gul er igang, grå var dem som vi manglede extra på, og rød ting som er over en deadline. Vores grå og røde er dog på dette billede i figur X blevet overskrevet af grønne fordi vores grå er ikke inde på gantt'en ude over de sidste 6 som kunne være et eksempel på hvordan det skulle se ud.

11.3 Scrum

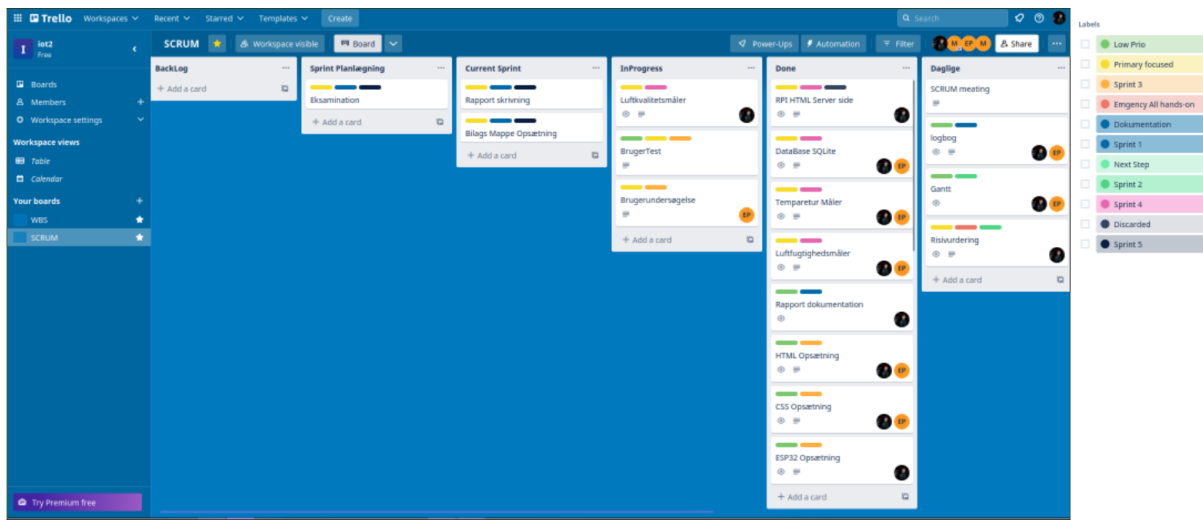


Figure 30, (fuldt billede størrelse i bilag)

Til dette forløb anvendte vi scrum til vores projektstyring. Ude til højre er farvekoderne angivet. Opgaverne havde vi inddelt i 2 forskellige kategorier. sprint numrene fra 1-6, og fra lav til høj prioritet. Vi valgte at tilføje disse labels i fordi hvis der var noget i projektet som skulle laves før noget andet kunne komme videre, var det vigtigt at vi fik det løst hurtigst muligt. Hvorimod vi også har lav prioritet, hvis noget godt kan vente til den sidste ende af det nuværende sprint vi er i gang med.

11.4 Projektanalyse

SWOT

Styrker Mangel på fysisk løsning. Simpel produkt. Billigt. Lille og let produkt.	Svagheder Nemt produkt at kopiere. Nemt produkt at stjæle.
Muligheder Stort marked i verden. Voksende marked på grund af klimaforandringer.	Trusler Store firma. korrupte lande.

SWOT tabellen (tabel x) ovenover er blevet lavet ud fra, hvordan løsningen ville være som et produkt. Derfor er der blevet tænkt på, hvordan produktionen kunne blive påvirket af forskellige faktorer. SWOT analysen er blevet udarbejdet efter vores forventninger, derfor ville den kun give et indblik i, hvordan vi tænker vores løsning ville kunne klare sig på markedet.

Interessentanalyse

<p>Gidsler</p> <p>Beboere tæt på skove</p> <p>Landmænd</p> <p>byggeri</p> <p>Naturforskere</p> <p>Skovbruge</p>	<p>Ressourcepersoner</p> <p>Naturstyrelsen</p> <p>kea lærer vejledere</p>
<p>extern interessent</p> <p>Miljøaktivister</p> <p>Tech-firma</p> <p>Brandfare.dk</p> <p>Familie</p>	<p>Grå eminence</p> <p>Staten</p> <p>Beredskabsstyrelsen</p>

I interessentanalysen tabellen (tabel x) ovenover bliver der vist, hvilken interessenter der er i projektet. De vigtigste interessenter er Naturstyrelsen og beredskabsstyrelsen, hvor Naturstyrelsen er den, som både skal købe vores løsning, mens beredskabsstyrelsen er dem, som skal bruge vores løsning. Under Ressourcepersoner er der Naturstyrelsen og KEA lærerne, da Naturstyrelsen er den som skal købe løsningen til Beredskabsstyrelsen og KEA lærerne er dem, som kan hjælpe os med faglig viden.

11.5 Risikovurdering

Hvad kan der ske?	Konsekven	Sandsynlighed	Risiko Point	Forebyggende	Ansvarlige
gruppen ikke laver noget	5	3	15	Folk mødes til tiden, så man kan komme i gang.	Os alle i gruppen
Projektet fejler	5	2	10	grundig undersøgelse om problemet faktisk er reelt	
Naturkatastrofer / skolen lukker	2	1	2	Internet, arbejde hjemmefra	Os alle i gruppen
Uforventet fejl; kode, projektarbejde	4	4	16	Undersøgelse i fritiden, man prøver at forstå og lære det gældende i timerne	os alle i gruppen
Ressourcemangel	5	4	20	Planlæg materialeliste i god tid, specielt under udformning af løsning	Alle i gruppen, Daniel/Emil ansvarlige for at få skaffet delene
Afveje i projektet / dårlige projekt styrelse	3	3	9	Aktiv deltagelse og alle gennemtjekker materialet vi laver, om der er noget som kan være forkert	Os alle
Defekt udstyr sensor	5	2	10	Være sikker på at havde ekstra sensor eller havde filmet at det har virket	Os alle
Sygdom/ andre forhindringer	5	3	15	Skriver hvis man er forhindret i at lave det, som er blevet aftalt	Os alle
Kundernes ved ikke hvad de vil havde / har ikke en samlet produktkrav	5	1	5	Havde en god kommunikation med kunderne og havde en prioriteringsliste for hvilke kunder, som vi mener er mest vigtige at arbejde med.	Os alle

I risikoanalyse tabellen viser det hvilken problemer, som vi tror der kunne opstå gennem projekt og giver dem en score, for hyppighed og konsekvens og derefter gangens de to tal sammen, hvor så får man den samlede score, derefter laves der et løsningsforslag og hver er andsvarliget.

12. Konklusion

I projektet var der lavet en løsning som var i stand til at opfange temperatur og luftfugtighed, og den kunne også måle CO₂ i en gang (se bilag 7, dokumentation af MQ135 virker en gang, i bilagsmappen). Løsning kan indsamle noget data og ud fra givne parametre informere om 5 ting, temperaturen i området, luftfugtigheden i området, om der er røg eller ej, placering i X&Y koordinater. Løsning ville være i stand til at kunne opfange det miljø, hvor der ville være størst mulighed for, at der ville opstå naturbrande. Vores løsning kan på afsluttende tidspunkt godt sende data'en til vores RPI(RaspberryPI) som hoster vores Database og vores hjemmeside. Vores løsning har også et GPS modul på som kan fortælle tiden, men GPS modulet virker dog til at have store problemer med at connect til satellit og give en placering. På grund af dette har vi været nødt til at hardcoded, hvor vores løsning skal placeres, da GPS placering ikke nødvendigt for løsningen funktion, da den hovedsageligt kun skal opsamle data og sende dem videre til databasen. Vores løsning kan på nuværende tidspunkt ikke holde table på vores hjemmeside, ikke holdes opdateret på grund af problemer med PHP. Dette ville være en af de næste steps hvis man fortsætter projektet. Her ville det også være oplagt at hoste siden på en real server med et domain så man kan tilgå den rundt omkring i hele Danmark, og ikke kun indenfor en vis rækkevidde.

Derfra kan der konkluderes, at man kunne godt lave en IOT produkt som er i stand til at måle ude i naturområder og advare om høj chance for naturbrande. yderligere kan det konkluderes at vores løsning kan sende data'en til vores RPI, fra vores ESP. Data'en bliver lavet om til en tekst streng af data.

13. Projektforløbet

I dette projekt har der været godt samarbejde i gruppen, der var problemer med mødetiden, da alle ikke mødte op til den aftalte tid og kunne være bedre koordineret med kommunikation. Der var også noget problem med koncentration i løbet af projektet, men det løste sig, da et gruppemedlem tog initiativ og hjalp med at holde fokus. Der var nogle i gruppe, som havde lagt en stor indsats ind i projektet, hvor de har lagt store dele af deres fritid ind i det, for at få det væsentlige for projektet til at virke, mens andre havde lagt væsentlige mindre ind i det. Arbejdsfordelingen i gruppen kunne godt have været bedre, da mange af de store opgaver blev fordelt til de samme personer, det gjorde at de andre ikke vidste, hvad de skulle lave. I fremtiden kunne man have bedre kommunikation, så arbejdsfordelingen kunne blive mere ligeligt, så alle havde samme arbejdsbyrde, så gruppearbejde ville blive mere effektivt. For at forbedre gruppearbejde kunne der også have lavet en bedre projektplan, så det var mere overskueligt, hvad der skulle laves, og hvor der kunne kommes med forslag til, hvad de enkelte punkter skulle omhandle, dette ville gøre at der var en klar ide om, hvad der var forventet i de enkelte afsnit. Der kunne også have været en bedre kommunikation under afleveringerne, da der var nogle problemer med at gennemtjekke af dem, og havde medført at der havde sket en væsentlig fejl, som kunne nemt have undgået, hvis man havde gennemtjekket afleveringen.

14. Perspektivering

For videreudvikling af vores løsning kunne vi gøre at vores hjemmeside ville give en besked om når der er fare for naturbrande, man kunne tilkoble nogle sprinkler, som ville automatisk køre hvis det blev for tørt. Løsningen kunne også eksporteres til andre lande, da løsningen ikke behøver speciel informationer. Dette gør at løsningen nemt kunne implementeres andre steder, hvilket ville gøre markedet meget større, da der mange lande, som har kæmpe naturbrande. Vores løsning ville måske også kunne bruges til andre ting, da man kunne også anvende den i drivhuse, da det ville hjælpe vækst af planter, fordi planter optager CO₂ og de vokser bedre når CO₂ indholdet er omkring en hvis mængde, der samme gælder også for varme. Det var også planlagt i starten at løsningen også skulle kunne måle jord-fugtigheden, dette ville gøre at løsningen ville også kunne bruges til at hjælpe med landbrug. Løsning kunne også bruges til måle CO₂, som man kunne også bruge den som en røgalarm og samt måle af indeklimaet. Vores løsning kunne blive videreudviklet til også at kunne slukke små brande, hvis løsningen kunne kalde droner, som ville være i stand til at slukke ilden, så den ikke spreder sig.

15. References

- (n.d.). Retrieved 03 23, 2023 from Wikipedia: <https://en.wikipedia.org/wiki/Microcontroller>
- BRS, Beredskabsstyrelsen. (2018). *BRS Naturbrande 2018*. BRS Beredskabsstyrelsen. Retrieved 03 17, 2023
- CLI.github.com. (2023). *Manuel CLI Github*. Retrieved 03 21, 2023 from CLI Github: <https://cli.github.com/manual/index>
- Crontab. (2023). *Crontab*. Retrieved 03 26, 2023 from Crontab, Guru: https://crontab.guru/#*_0-23_*_*_*
- DanskBeredskaber. (n.d.). *Naturbrand*. Retrieved 03 17, 2023 from Danskberedskaber: <https://danskeberedskaber.dk/naturbrand/>
- Dee, S. G. (2022, februar 25). *Weforum*. Retrieved 03 17, 2023 from Here's how this man connected humans to global warming in 1938: <https://www.weforum.org/agenda/2022/02/climate-change-global-warming-carbon-dioxide-fossil-fuels/>
- Ed. (2019, september 20). *RandomnerdTutorials*. Retrieved 03 25, 2023 from ESP32 raspberryPI lamp server: <https://randomnerdtutorials.com/esp32-esp8266-raspberry-pi-lamp-server/>
- Flask dokumentation. (2010). *Quickstart*. Retrieved 03 21, 2023 from Flask Dokumentation: <https://flask.palletsprojects.com/en/2.2.x/>
- GNU, 5.2 . (2022, September). Retrieved 03 28, 2023 from Bash manual: <https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html>
- Lutkevich, B. (2019, november). *Microcontroller MCU*. Retrieved 03 23, 2023 from techtarget: <https://www.techtarget.com/iotagenda/definition/microcontroller>
- Mkyong. (2018, 12 14). *How to convert float to string*. Retrieved 03 26, 2023 from Mkyong, python, how to convert float to string: <https://mkyong.com/python/python-how-to-convert-float-to-string/>
- NILET, G. (2022, 04 13). *MQ135 interfacing with arduino*. Retrieved 03 20, 2023 from IOT: https://www.nielit.gov.in/gorakhpur/sites/default/files/Gorakhpur/alevel_iot_13april20_SM.pdf
- Pankaj. (2022, august 3). *Nohup linux command*. Retrieved 05 28, 2023 from Digital Ocean: <https://www.digitalocean.com/community/tutorials/nohup-command-in-linux>

SQLite. (2023, 03 22). *SQLite Dokumentation*. Retrieved 03 23, 2023 from SQLite:
<https://www.sqlite.org/quirks.html>

sqlitetutorials. (2022). *SQLitetutorials*. Retrieved 03 26, 2023 from sqlite-python, insert.

Tutorialpoint. (n.d.). *SQLite PHP*. (Tutorialpoint) Retrieved 03 28, 2023 from Tutorialpoint:
https://www.tutorialspoint.com/sqlite/sqlite_php.htm

VisionTIR. (2022, april 18). *Forest fire detection*. Retrieved 03 17, 2023 from VisionTIR:
<https://visiontir.com/forest-fire-detection/>