

Royer Akram Khalil, Patrich sodon Lyager Poulsen, Daniel Nybo,

Jakob Wilquin, Kasper Lyng Tindahl Nielsen

Studiegruppe 3

Klasse b, 2022-2024,

IOTProjekt 1, Vokalo vest løsning,

Nikolaj, Malene, Kevin, og Dan

Antal anslag



[Vokalo Avanceret Træningsvest]

1. Resumé

Der blevet arbejdet på at finde en løsning til at kunne effektivisere en fodboldspiller og hans træning og derfor finde svare på hvordan man kan lave en IoT løsning der fremmer træning. Ud fra de kriterier er der blevet arbejdet med forskellige programmeringsmetoder som brug af micropython til at kunne skrive koderne og til at få udstyret til at kommunikere med hinanden, og indlejrede system metoder for at kunne lave de enkelte detaljer til udstyret, såsom arbejde med fusion360 og lodning af komponenter. diagrammer og flowcharts blev brugt for at kunne simplificere arbejdsmetoderne, både for projektgruppen og eventuelle andre skulle bygge det samme. Både WBS, Gantt og Kanban er blevet brugt for at kunne planlægge hele projektets fremgang og arbejdstider samt er deadlines lagt indenfor gruppen. Alle krav der blev lagt for gruppen, blev ikke udført, da gruppen ikke havde den fulde ekspertise i diverse kravspecifikationer, hvilket gjorde at der blev sat ekstra fokus på de første prioriteter af kravene, så det kunne blive fuldført helt. Så om IoT løsningen kan gøre en fodboldtræning mere effektivt er uvist af da projektet ikke blev gennemført med alle prioriteterne til det.

2. Indholdsfortegnelse

1. Resumé	3
2. Indholdsfortegnelse	4
3. Indledning	6
3.1 Beskrivelse af virksomhed	6
4. Problemformulering	7
5. Indledende undersøgelse	8
5.1 Ideudvikling	8
5.2 Research	8
5.3 Brugerundersøgelse / Design Thinking	8
5.4 Persona og User stories	8
6. Kravspecifikation og accepttest metode	9
6.1 Rationale for prioritering af krav	10
7. Analyse	11
7.1 Arkitektur af pathfinding	11
7.2 Den endelige arkitektur	13
7.3 Teori af relevante elektroniske/programmering/netværk/server arkitekturblokke	13
8. Løsningsdesign	15
8.1 Elektriske skematikker, målinger og udregninger (Indlejrede systemer)	15
8.2 Router/Switch opsætning og Interface forbindelser (Netværk)	15
8.3 Kode beskrivelse (Programmering)	15
8.4 Konfiguration af valgte servere (Server)	16

9. Test af løsning	17
9.1 Pathfinding testresultater	17
9.2 Dokumentation af DUT	17
9.3 Udførsel af brugertest	18
9.4 Udførsel af Accepttest på DUT	19
10. Implementering af løsning i drift (eventuel)	20
11. Praktisk projektplanlægning og ledelse	21
11.1 WBS	21
11.2 Gantt	21
11.3 Kanban / Scrum	21
11.4 Projektanalyse	21
11.5 Virksomhedsanalyse	21
12. Konklusion	22
13. Projektforløbet	23
14. Perspektivering	24
15. Litteraturliste	25
16. Bilag	26

3. Indledning

Skader i sport i dag er mere udbredt end nogensinde, og en af de mest almindelige skader vi ser idag er knæskader. Ca. 14.9 ud af 100 fodboldspillere oplever knæskader et tidspunkt i deres karrierer. (*Rahnama et al., 2009*) Derfor går man mere efter forebyggelse af skaderne for at sørge for at chancerne for at det sker er mindre. I dette problem bliver der kigget på, hvordan knæskader bliver reduceret ved hjælp af IoT teknologi. Ved hjælp af en vest der er udstyret med (sensorer) vil man kunne måle de forskellige opbremsninger og landinger fra en fodboldspiller og med den data, kan man få dataene til at kunne optimere en spillers chance for at undgå en skade fra for "store" bevægelser.

Udover at hjælpe med at kunne forhindre knæskader, skal løsningen have flere komponenter af IoT-løsninger til rådighed, til at modernisere en fodboldtræning og dermed gøre den mere effektiv da det er ønskede egenskaber fra Vokalo.

Dataene i vesten bliver testet før brug, til at kunne beregne og vurdere, hvad der er en for "stor" bevægelse, så man som udgangspunkt kan undgå det til en træning eller træningskamp.

Løsningerne bliver bearbejdet ud fra den viden og metoder i indlejrede systemer, programmeringer og virksomhed. Ud fra den erfaring bliver der arbejdet på løsningen til problemet.

3.1 Beskrivelse af virksomhed

Vi samarbejder med en virksomhed ved navn Vokalo. Vokalo blev grundlagt i år 2019, og har hovedkvarter i Århus. De har et team som består af 8 personer.

Vokalo et firma som henvender sig til fodboldspillere. Vokalo har udviklet en vest, som kan kommunikere mellem spiller og træner. Alt dette gør de for at optimere spillerne.

4. Problemformulering

Problemstilling

Efter gennemgang af opgaveformuleringen, og kravene som blev stillet af kunden, er vi kommet frem til nedenstående problemstilling. Vores arbejdsproces af forskellige problemformuleringer, kan ses i bilag 1.

‘Det er et problem at, fodboldspiller får unødvendige knæskader ved pludselige standsninger, landinger og skarpe vendinger.’

Problemformulering

Vi vælger at gå med en bred problemformulering, eftersom det kan hjælpe med en god problemløsning senere hen. Området af IoT der mindsker knæskader og ben skader er meget bredt, derfor afgrænser vi området til at fokusere på stød og g-påvirkninger, som knæet udsættes for under en træningskamp.

Hvordan mindskes knæskader ved hjælp af IoT(Internet of Things)løsning-

Vi sætter fokus på knæskader, meget pludselige standsninger samt landinger efter et hop. Vi kommer til at undersøge med en trykmåler som kan måle hvor meget tryk ens knæ og ben udsættes for. Særligt når ens kropsvægt skiftes til at kun støttes af det ene ben.

Hvad kan løsning, gøre for at forbedre fodboldtræning?

‘Hvordan mindsker man knæskader, ved pludselige standsninger, og skarpe drejning under løb, under en fodbold træning.’

Ud fra dette vil vi lave en løsning som kan hjælpe med at reducere belastning på knæet og benene. Vi forventer at se en oversigt og totalt tryk hen over en træningssession, og en tæller som kan se hvor mange gange, spilleren, har haft for meget tryk på det ene ben.

5. Indledende undersøgelse

5.1 Ideudvikling

Udviklingen af ideerne blev udarbejdet ved meget ydrestyret idéundfangelse, fordi der blev holdt møder jævnligt i idegenereringsfasen. På den måde blev der lagt vægt på ideudviklingen på prioriter til projektet. Ved hjælp af værktøjer som Cirkel Teknik, udarbejdede ideerne sig mere ideelt til vores projekts problemformulerings krav.

Cirkelteknik til idéudvikling	
	Beskrivelse
Grundidé A	Hvordan mindskes knæskader i Fodbold
Tilføjelse B	Hvordan knæskader kan opstå under en fodboldkamp eller træning.
Tilføjelse C	Hvordan mindsker man knæskader, ved pludselige standsninger, hurtige retningsskift og landinger, under en fodboldtræning
Tilføjelse D	Hvad kan træningen hjælpe med at formindske knæskader under en fodboldkamp/træning?

Sammenfatning	<p>Ud fra tilføjelserne der er blevet lavet, så blev alle ideerne lagt sammen så den kunne gøres mere specifik til IoT formål og derfor blev den endelige ide ligeledes:</p> <p>Hvordan mindskes knæskader ved hjælp af IoT(Internet of Things)løsning</p>
---------------	---

5.2 Research

Projektets team fandt frem til den endelige løsningsforslag ved hjælp af research på forskellige videnskabelige artikler fundet på nettet som vi har fundet ud af at knæskader såsom korsbåndsskader er en af de mest normale skader der opstår under fodbold (*Chong, 2004*). Ud fra den artikel som hoved information, kunne gruppen finde på relevante ideer til en løsning, som så skulle kunne skabes ud fra problemformuleringen der blev skrevet.

Ved hjælp af de metoder der blev brugt til ideudviklingen, kunne projektgruppen derfor mere præcist finde ud af hvad der skulle researches om mere specifikt og derfor finde videnskabelige artikler der passer til problemformuleringen.

Det endelige løsningsforslag blev valgt ud fra den information der blev samlet ind fra de artikler projektgruppen har fundet frem til.

6. Kravspecifikation og accepttest metode

De tekniske krav for projektet er:

ID: 1 Kategori: UI	Krav: Løsningen skal som minimum kunne fremvise lokation og batteridata via et online dashboard, med en opdateringsrate på minimum 2 opdateringer i minuttet.	Prioritet: 1
Accepttest: Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres. Step 2) Indenfor 4 minutter bliver position og batteriniveau data præsenteret i et online dashboard. Step 3) Et stopur sættes i gang, og der måles hvor mange opdateringer dashboardet får over en periode på 5 minutter. Vil der over perioden være 10 eller flere opdateringer, så vurderes kravet som bestået.		
ID: 2 Kategori: Power	Krav: Den kropsbårne løsning skal være batteridrevet, og skal have over 115 minutters batterilevetid.	Prioritet: 1

<p>Accepttest:</p> <p>Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>Step 2) Et stopur sættes i gang, og testbrugeren påbegynder en trænings session på 2 x 45 minutter, med en 15 min pause i mellem.</p> <p>Hvis løsningen stadig har over 10% batteri ved afslutningen af forløbet på 115min, så vurderes kravet at være opfyldt.</p>		
<p>ID: 3</p> <p>Kategori:</p> <p>Connectivity</p>	<p>Krav: Løsningen skal være opkoblet til internettet, og være mobil.</p>	<p>Prioritet: 1</p>
<p>Accepttest:</p> <p>Step 1) Løsningens hardware del placeres på en testbruger, aktiveres, og dashboardet observeres</p> <p>Step 2) Testbrugeren går 1km væk fra startpunktet og vender tilbage til startpunktet.</p> <p>Hvis løsningen opretholder internetforbindelsen og ingen pakkeab har gennem hele gåturen, vurderes kravet at være opfyldt.</p>		

<p>ID: 4</p> <p>Kategori:</p> <p>Mechanical</p>	<p>Krav: Løsningens hardware del skal være kropsbåret, robust, og ikke være til fare for spiller under en fodboldkamp.</p>	<p>Prioritet:</p> <p>1</p>
<p>Accepttest:</p> <p>Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>Step 2) Testbrugeren ligger sig ned på græsplæne og ruller rundt 10 gange.</p> <p>Step 3) Testbrugeren bliver tacklet 10 gange</p> <p>Hvis testbrugeren ikke oplever nogen gener, stammende fra løsningens hardware del, fra denne test, og at elektronikken ikke udviser tegn på skader, så vurderes kravet at være opfyldt.</p>		
<p>ID: 5</p> <p>Kategori:</p> <p>Sensing</p>	<p>Krav: Løsningen skal kunne måle og visualisere brugerens position.</p>	<p>Prioritet:</p> <p>1</p>

Accepttest:

Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.

Step 2) Testbrugeren går i en cirkel med en diameter på 10 meter.

Step 3) Testbrugers position visualiseres via dashboardet

Hvis dashboardet fremviser en cirkel, med en diameter på 10 meter (+-3 meter), vurderes kravet at være bestået.

ID: 6

Kategori:

Sensing

Krav: Løsningen bør indikere hvis spilleren er tacklet, og fremvise antal af tacklinger direkte på løsningens hardwaredel.

Prioritet: 2

Accepttest:

Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.

Step 2) Der observeres displayet på løsningens hardwaredel at antallet af målte tacklinger er 0

Step 3) Testbrugeren bliver tacklet 10 gange

Hvis der observeres at displayet fremviser tallet 10, efter at step 3 er gennemført, så vurderes kravet at være opfyldt.

ID: 7**Kategori:**

UI

Krav: Løsningen bør kunne fremvise batteriniveau på løsningens hardwaredel.**Prioritet: 2****Accepttest:**

Step 1) Løsningens batteri lades op til 100%, og batterispændingen måles med et multimeter.

Step 2) Løsningens hardwaredel placeres på et bord, og batteriniveauet bliver observeret både på dashboardet(krav 1), men også via en passende indikator på løsningens hardwaredel.

Step 3) Der foretages batterimålinger med multimeter hver halve time, over en periode på 3 timer.

Hvis batteriniveauet der måles med multimeter, og batteriniveauet fremvist på løsningens hardwaredel, samt dashboard er indenfor 15% nøjagtighed, så vurderes det at kravet er opfyldt.

ID: 8 Kategori: sensing	krav: Løsningen kan registrere det højeste belastningstal hver gang fodboldspilleren lander under træning, og så få dem vist på et dashboard. "Belastningstal": G-Kræft Sensor der giver en nummerværdi, ud fra hvor højt og hårdt man lander.	Prioritet: 3
Accepttest: Step 1) Testbrugereren ifører sig løsningens hardware del, og hardwareløsningen aktiveres. step2) Testbrugereren hopper 10 gange Testen skal kunne fremvise den højeste tryk på fodboldspillere efter hver landing og har dem sendt på et dashboard en af gang. Den har som min. registret 10 landinger betragtes det som gennemført.		

ID: 9 Kategori: sensing	Krav: Løsningen kan registrere fodstilling og knæ bevægelse samtidig.	Prioritet: 3
---	--	---------------------

Accepttest:

step1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.

Step2) Test Brugeren bevæger sit knæ mens foden sidder fast i jorden, så skal den kunne fremhæve en tone.

Hvis der observeres en lyd, når test brugeren har bevæget sit knæ mens foden sidder fast i jorden, betragtes det som fuldført.

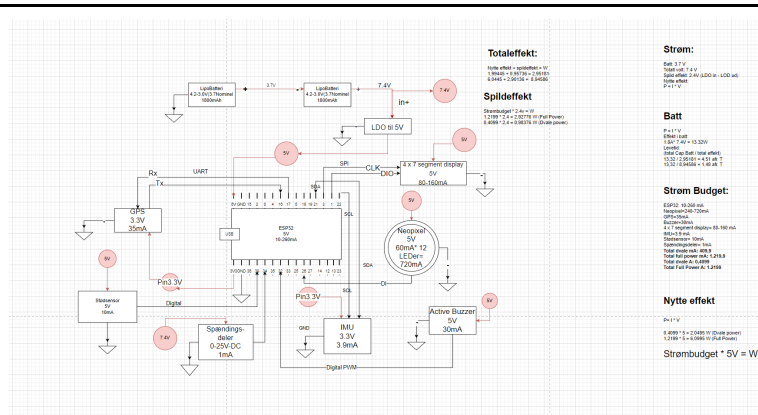
6.1 Rationale for prioritering af krav

Prioriteterne er opdelt således, de første fem har en første prioritet, derfor har vi fokuseret mest på dem, og fordi det er de krav der er blevet stillet til projektgruppen, som vi skal udføre. De første prioriteter er også sat sådan, da det er en IoT-løsning projekt, skal der være en basis form for IoT udstyret på vesten, og for at måle gruppens færdighed ud fra det, der er blevet lært i undervisningen. Anden prioriteterne er nogle lidt mere avancerede krav, som i dette tilfælde projektgruppen kun nåede at kunne gennemføre den ene af. Tredje Prioritet kravene er dem der blev lavet ud efter den helt nye problemformulering, og derfor har de heller ikke kunne nås og blive gennemført.

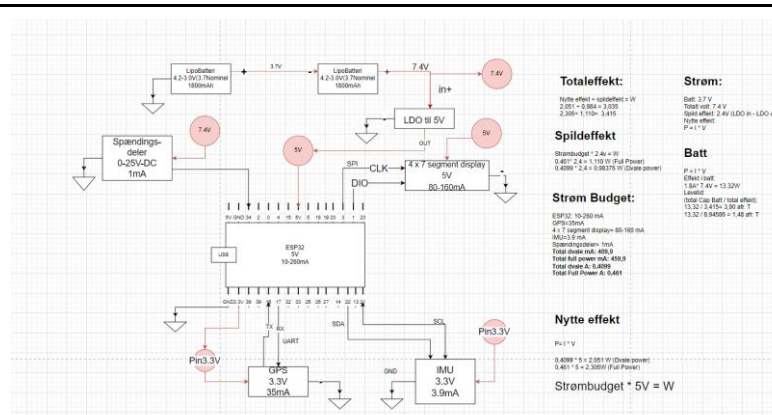
7. Analyse

7.1 Arkitektur af pathfinding

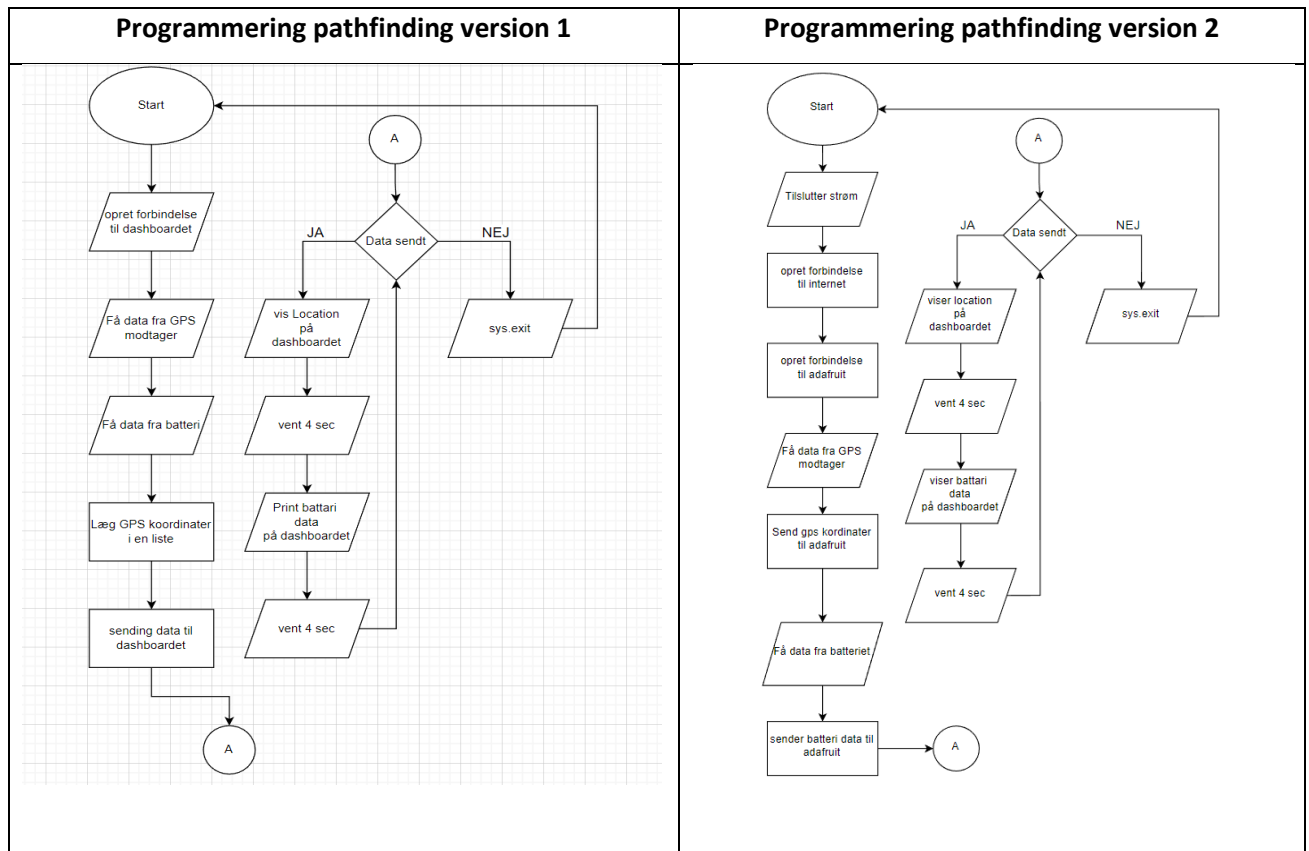
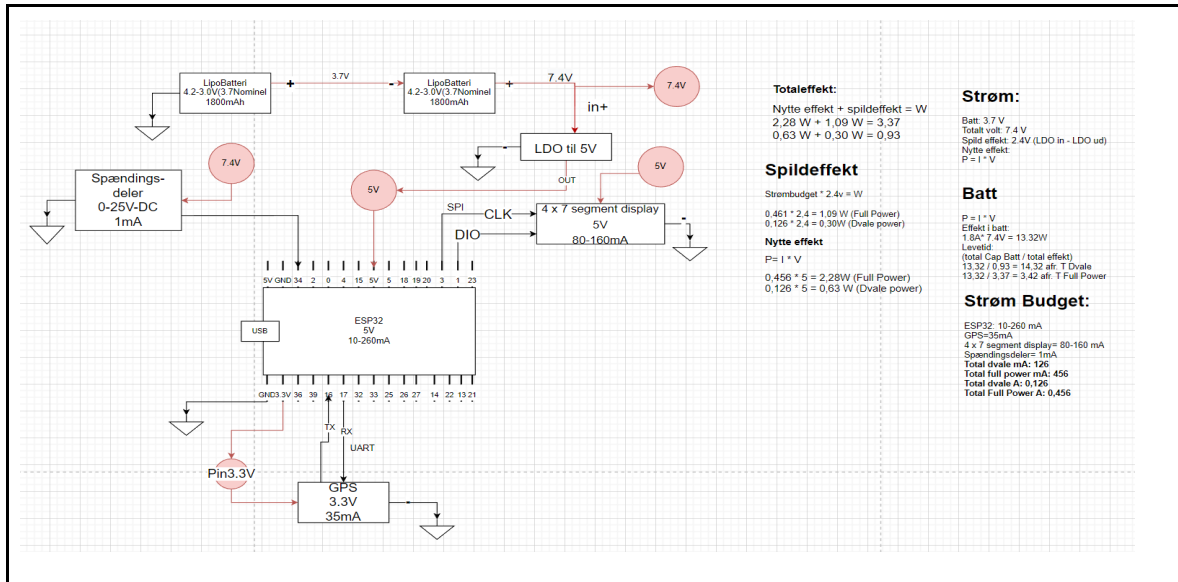
Version 1



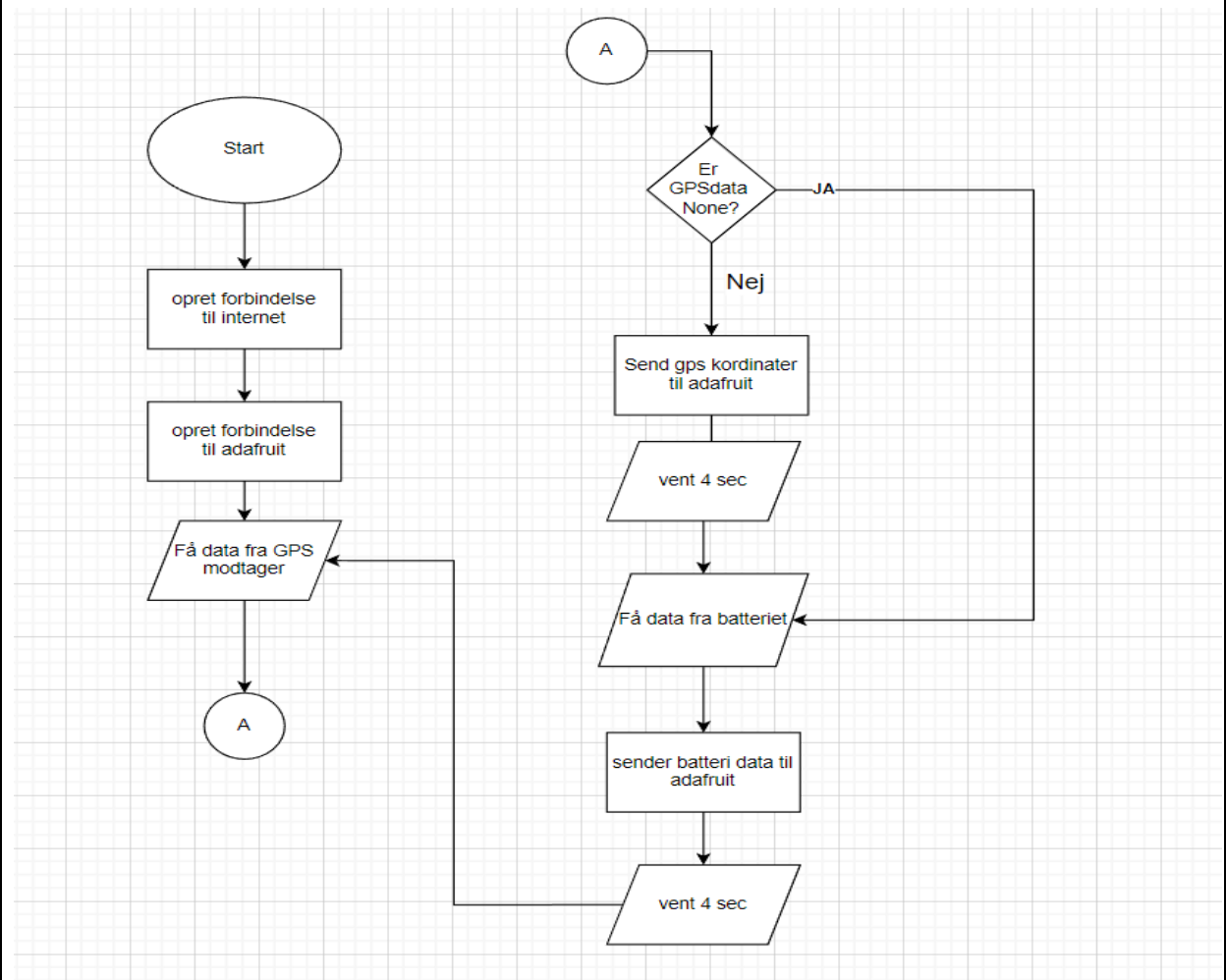
Version 2



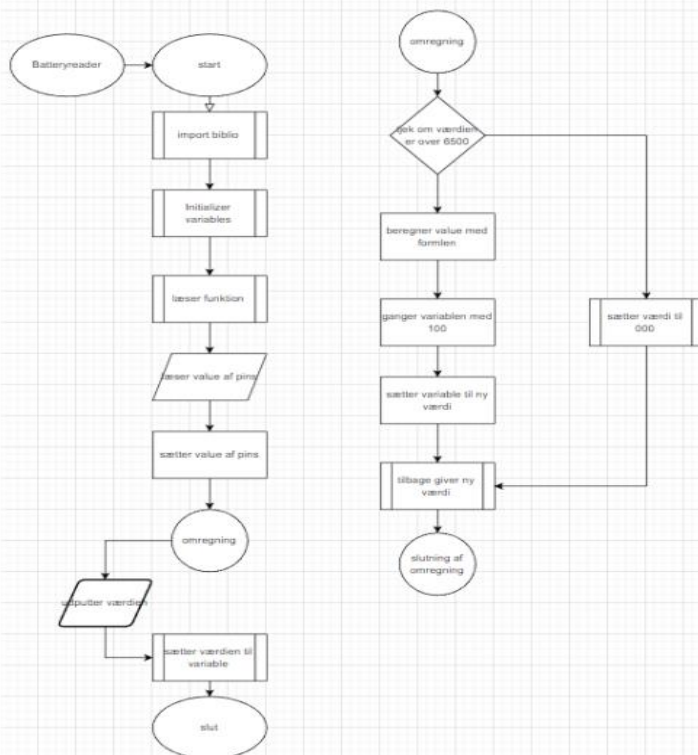
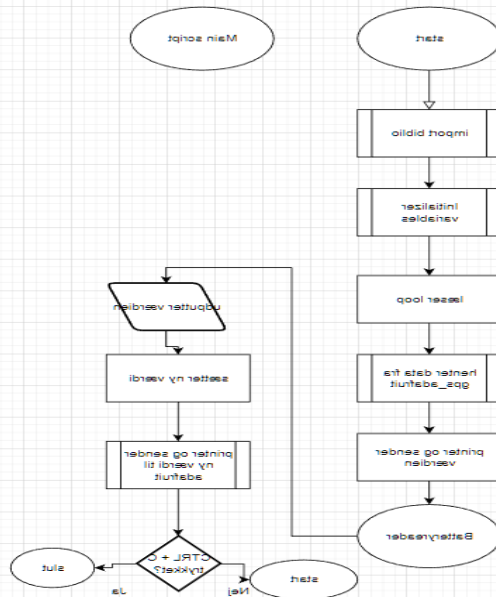
Version 3



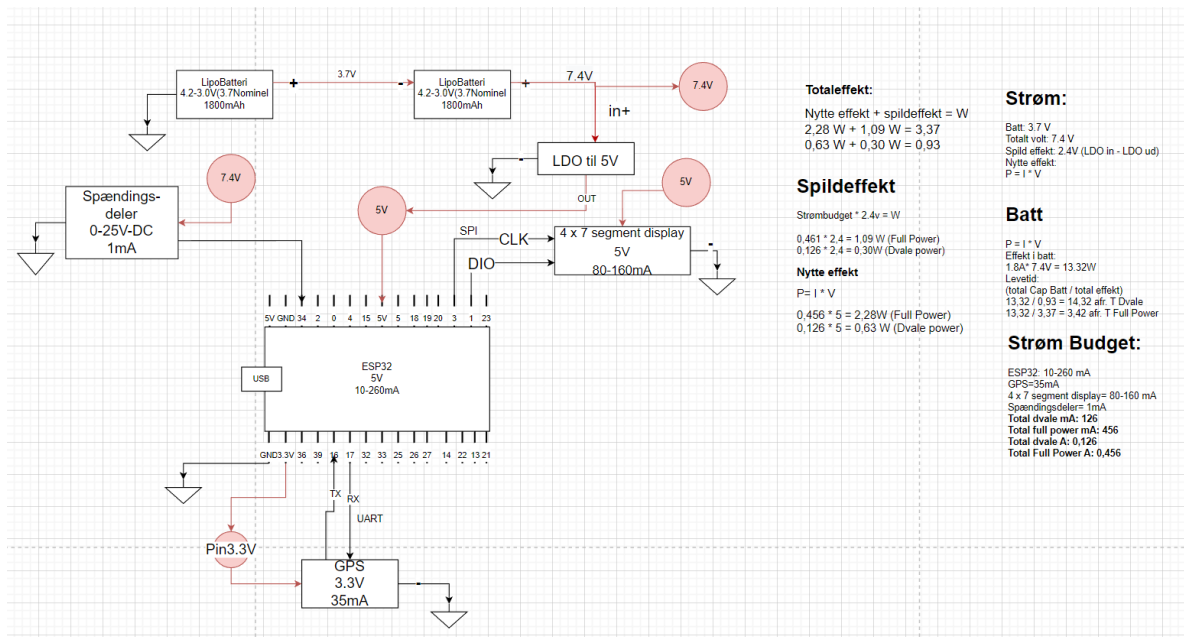
Programmering pathfinding version 3




Programming pathfinding version 4



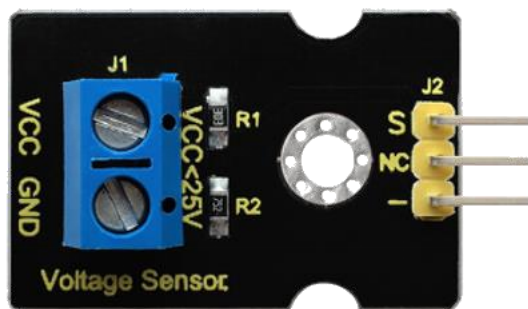
7.2 Den endelige arkitektur



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

Blok navn: ESP32	
	<p>ESP32 er en mikrocontroller, som er placeret på et DevkitC v4 board. Det er udstyret med en 2.4 GHz Wi-fi og Bluetooth chip.</p>
	<p>ESP32'eren er også udstyret med en spændingsregulator, 2 knapper, LED og en USB til UART-konverter, der gør at der kan komme kommunikation til ESP32'eren via et USB.</p>
	<p>ESP32'eren har en IO spænding på 3.3V og en forsyningsspænding på 5V.</p>
	<p>ESP32'eren er 52.52 mm lang, 24.04 mm bred, 8.54 mm tyk, og vejer 8 gram.</p>
	<p>ESP32 er designet til at kunne bruges med mobil elektronik og IoT applikationer.</p> <p>I dette projekt bruges ESP32'eren til at kunne kommunikere og kontrollere diverse IoT komponenter.</p> <p>ESP32'eren får strøm gennem et Batteri (beskrevet i blokken "Batteri")</p>

Blok navn: Spændingsdeler

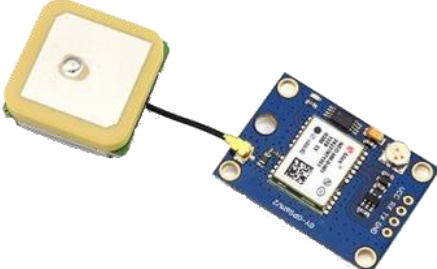


Spændingsdeleren benyttes hvis man driver en microcontroller (ESP32) via batteri vil der som reelt være en spænding som er højere end 3.3V som ADC-benene max kan læse.

Komponenten indeholder 2 modstande som har en 1 til 5 størrelsesforhold.

Komponentens operative spænding er 0-25V DC, og strømforbruget er 1mA

Komponentens størrelse er 35 mm lang, 20 mm bred og 14 mm tyk.

Blok navn: GPS	
	<p><i>Modulet gør at man kan modtage GPS information via en UART-forbindelse.</i></p> <p><i>Modulets operative spænding er 3-5V og har en baud rate på 9600bps.</i></p> <p><i>PCB: 25mm x 35mm.</i></p> <p><i>Antenne: 12mm x 12mm.</i></p> <p><i>Kabel længde: 20mm.</i></p> <p><i>I dette projekt bruges GPS'en så det kan muliggøres at IoT-løsningen kan sende GPS information til Adafruit så man kan se IoT-løsningens lokation.</i></p>

Blok navn: LIPO Batteri



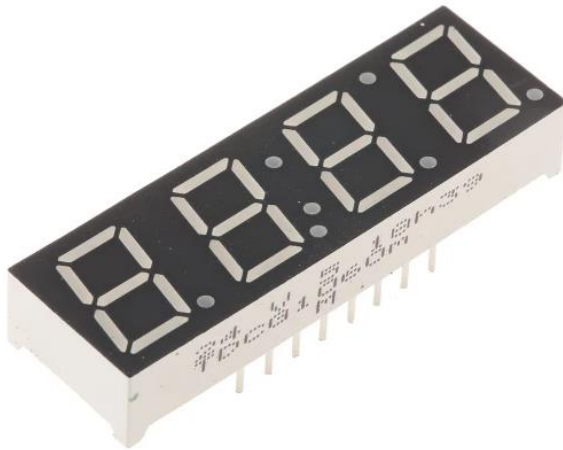
Et Lipo batteri er et perfekt batteri at bruge til mindre IoT-løsninger, som har 1800 mAh til rådighed

Den operative spænding er 4.2V - 3V og med en gennemsnitsspænding på 3.7V

Dimensioner: 55 mm lang, 30 mm bred, 10 mm tyk og 13 g vægt.

I dette projekt bruges LIPO-batterierne til at kunne give strøm til hele IoT-løsningen.

Blok navn: 4 x 7 Segment display

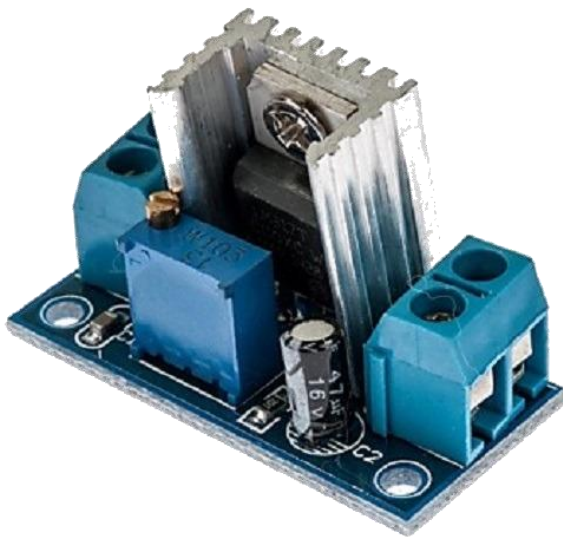


Dette display kan bruges til at vise tal eller bogstaver. Displayet har en IC driver som hedder "TM1637" og den gør at alle LED'er i displayet bliver kontrolleret ordentligt.

Operativ spænding er 3.3-5V

Dimensioner 45 mm lang, 21 mm bred, 10mm tyk og 7.2 vægt.

Blok navn: LDO



En LDO er spændings regulator der tager en høj inputspænding og leverer en lavere spænding i output siden.

Output spænding kan være fra 1.25 til 37V DC.

Max input-output spændingsforskel 40V DC

53 mm lang, 22 mm bred x 15 mm tyk.

i dette projekt bruges LDO'en til at kunne regulere en spænding fra batterierne fra en input på 8,4-6,6V til en regulering på 5V.

Blok navn: Batteryreader

i filen "Batteryreader.py" linje 19 - linje 25

I starten af linje 19 initializer man en if statement som tjekker om en condition er opfyldt, i vores kode tjekker den om volts måling er højere end 6500, fordi 6500 er det minimum for at vores system kan være operativ. Linje 21 sættes en lokal variabel til værdien af værdien af volts som er målt, tidligere i linje 16. 9335 er max værdien af vores batteri kapacitet. hvor 6500 er minimum kapacitet som vores løsning kan køre på.

her beregnes den til en procentdel, (formlen er i kode beskrivelsen sammen med billede figur af koden) linje 22 ganges værdien med 100 så vi får værdien i procent dele. og ikke 1 procentdele, Man kunne godt have ganget med 100 i linje 21 men vi valgte at gange med 100 for så kan man også nemmere debugge med en print af volts værdien hvis der skulle blive problemer.

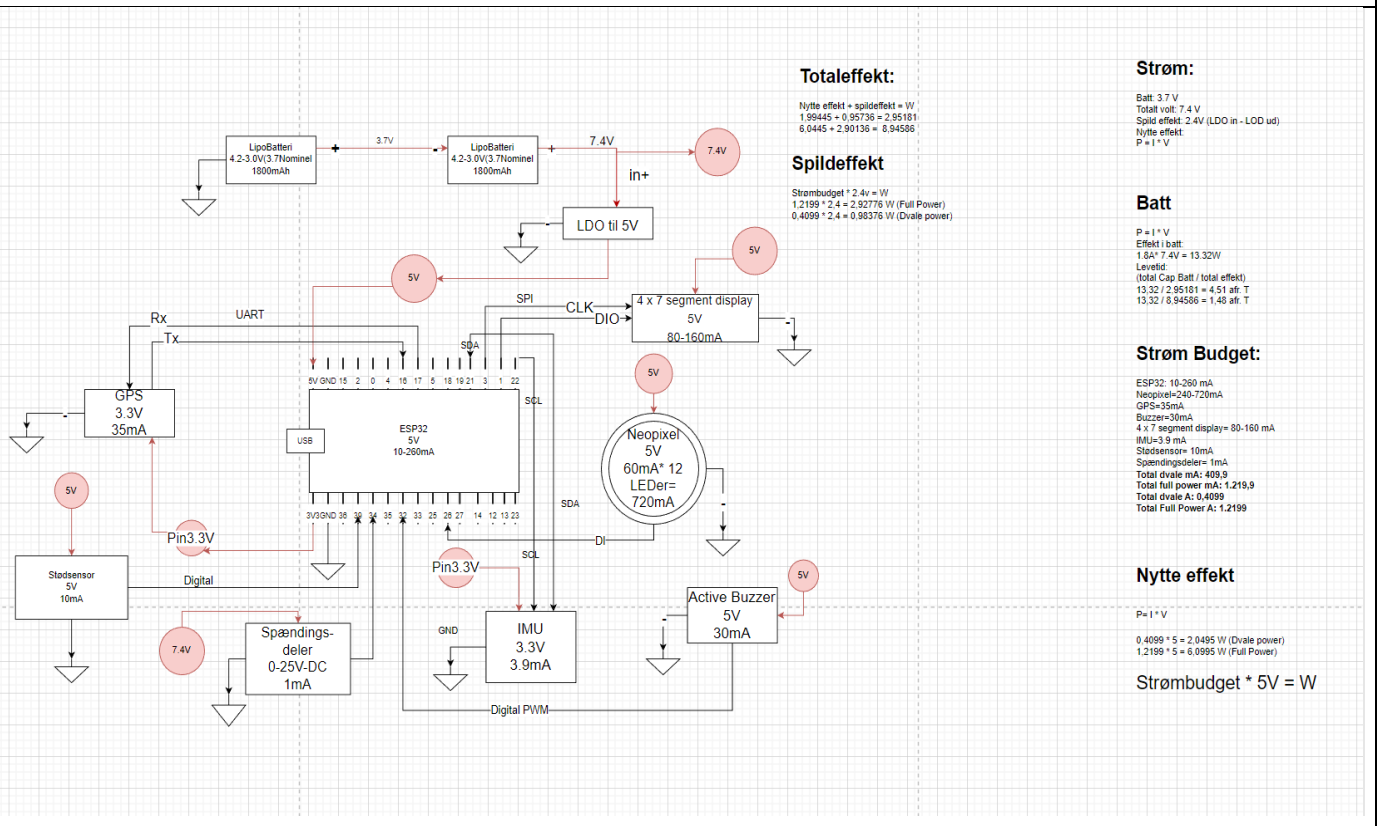
I linje 23 sættes variablen af voltages til battery percentage. I linje 24 skabes en else: condition for if statement så, hvis condition i if statement ikke er opfyldt så udfører den else:

her sættes battery percentage til 000 (kun 3 fordi de er 3 decimaler på batterimålerne)
de lyserøde er integreres tal værdier som bruges til matematiske beregninger.

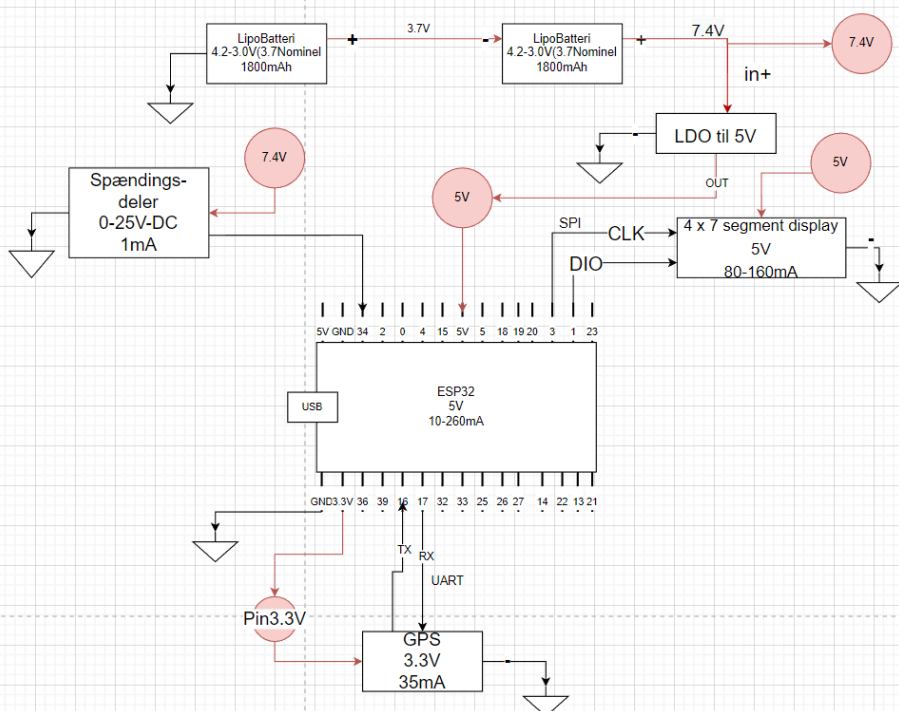
8. Løsningsdesign

8.1 Elektriske skematikker, målinger og udregninger (Indlejrede systemer)

Skematik og strøm budget version 1



skematik og strøm budget version 2



Totaleffekt:
 Nytte effekt + spildevækt = W
 $2,28 \text{ W} + 1,09 \text{ W} = 3,37$
 $0,63 \text{ W} + 0,30 \text{ W} = 0,93$

Spildevækt

Strømbudget * 2.4v = W
 $0,461 * 2,4 = 1,09 \text{ W}$ (Full Power)
 $0,126 * 2,4 = 0,30 \text{ W}$ (Dvale power)
Nytte effekt
 $P = I * V$
 $0,456 * 5 = 2,28 \text{ W}$ (Full Power)
 $0,126 * 5 = 0,63 \text{ W}$ (Dvale power)

Strøm:

Batt: 3.7 V
 Totalt volt: 7.4 V
 Spild effekt: 2.4V (LDO in - LDO ud)
 Nytte effekt:
 $P = I * V$

Batt

$P = I * V$
 Effekt i batt:
 $1,8 \text{ A} * 7,4 \text{ V} = 13,32 \text{ W}$
 Leveltid:
 (total Cap Batt / total effekt)
 $13,32 / 0,93 = 14,32 \text{ afr. T Dvale}$
 $13,32 / 3,37 = 3,92 \text{ afr. T Full Power}$

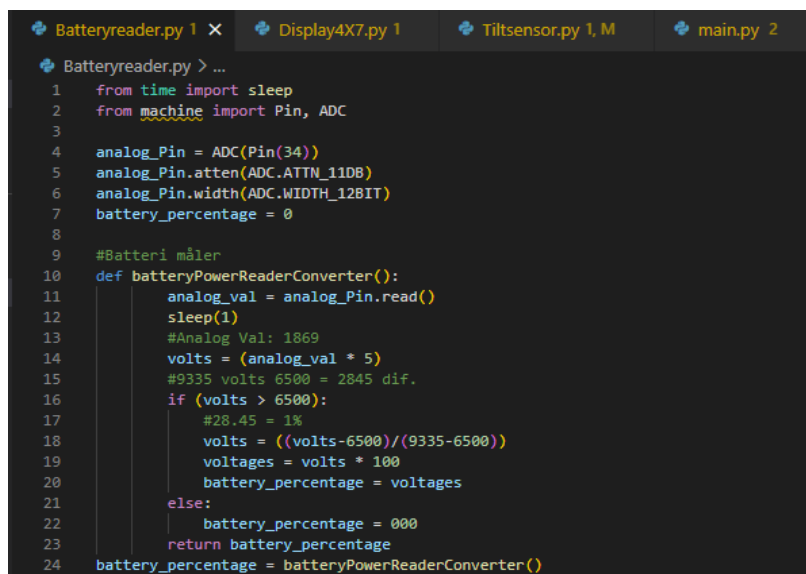
Strøm Budget:

ESP32: 10-260 mA
 GPS=35mA
 4 x 7 segment display= 80-160 mA
 Spændingsdeler= 1mA
 Total dvale mA: 126
 Total full power mA: 456
 Total dvale A: 0,126
 Total Full Power A: 0,456

Siden vi ikke har kunne finde et elektronisk skematikprogram og lave et skematik i, så valgte vi at kombinere det med strøm budget. Dette er også grunden til at ESP32 Devkit V4 har alle pins med. Vi har udregnet med den gennemsnitlige spænding for batterierne på 7.4 volt.

8.3 Kode beskrivelse (Programmering)

Igennem de kodeeksempler, som vi valgte og tage med i denne rapport vil vi anvende forskellige libraries som kan findes online eller på ESP32. Libraries er klasser til allerede skabt funktioner som man kan anvende til at omregne data. Særligt Pin klassen bruges gentagne gange fordi vi skal måle en værdi som vores ESP32 har målt, og den er tilgængelige på en pin.



```

Batteryreader.py 1 X  Display4X7.py 1  Tiltensor.py 1, M  main.py 2

Batteryreader.py > ...
1  from time import sleep
2  from machine import Pin, ADC
3
4  analog_Pin = ADC(Pin(34))
5  analog_Pin.atten(ADC.ATTN_11DB)
6  analog_Pin.width(ADC.WIDTH_12BIT)
7  battery_percentage = 0
8
9  #Batteri måler
10 def batteryPowerReaderConverter():
11     analog_val = analog_Pin.read()
12     sleep(1)
13     #Analog Val: 1869
14     volts = (analog_val * 5)
15     #9335 volts 6500 = 2845 dif.
16     if (volts > 6500):
17         #28.45 = 1%
18         volts = ((volts-6500)/(9335-6500))
19         voltages = volts * 100
20         battery_percentage = voltages
21     else:
22         battery_percentage = 000
23     return battery_percentage
24 battery_percentage = batteryPowerReaderConverter()

```

Se bilag nr. 8.3:

Kode eksempel 1: Batterimåler

Det primære der er at lægge mærke til er omregning af batteri niveauet i linje 19 funktionen. Vi valgte at sætte den i en funktion, fordi den værdi der måles kan alligevel ikke forsyne vores system, hvis det ikke kan levere mere end 6.5 volts, så derfor lavede vi det sådan at den kun regner, hvis

værdien er højere ellers siger den bare at der ikke er noget batteri tilbage. I else, er der sat 00 ind fordi i displayet viser den allerede 0 på første plads og P for procent på den fjerde plads. I linje 13 sættes værdien den læser til variablen, og i linje 27 returnere værdien til en variabel efter funktionen er færdig med at køre. Formlen der er brugte er: $((\text{Current value} - \text{minimum value}) / (\text{Maximum value} - \text{minimum value}) * 100)$

```

1  import tm1637
2  from machine import Pin
3  from time import sleep
4  import Batteryreader
5
6  tm = tm1637.TM1637(clk=Pin(4), dio=Pin(2))
7
8  tm.brightness(5)
9  variableY = "LIVE"
10 sleep(1)
11 tm.show("PPPP")
12 def display():
13     variableX = Batteryreader.battery_percentage
14     variableX = int(variableX)
15     if(variableX >= 100):
16         tm.show("{}{}{}{}".format("0", variableX, "P"))
17     else:
18         tm.show("{}{}{}{}".format(variableX, "P"))

```

Kode eksempel 2: Display4X7 battery viser på vesten

linje 1-4 importer forskellige libraries og fra andre scripts, "Batteryreader" er et script vi har lavet. I linje 6 sættes tm, ved brug af classen fra tm1637, og sættes til pin 4 og 2 til clk og dio hhv. i linje 8 sættes der variablen. Det vigtigste er linje 13 og 14, eftersom de sætter værdien af batteri procenten, hvor variablen gøres til en int værdi. I linje 12 skabes funktionen til display. I linje 16 initialisere vi et if statement som tjekker om batteriet er 100% eller over. her går vi ned og sætter "P" ind på alle 4 pladser for procent. efter sættes variablen int værdien ind på pladserne, og hvis værdien er under 100, bruges if statementen til at sætte et 0 foran, så der vil displayet vise følgende: "089P".

Kode eksempel 3: Main filen

Formålet med main filen her, er at samle alle andre kode scripts. Vi vælger at tage udgangspunkt i loopet, som køre alle andre scripts. Linje 48-55 er kode fra materiale filerne, som tillader at stoppe loopet og adafruit connection med ctrl + C.

```

main.py > ...
1  import umqtt_robust2 as mqtt
2  from machine import Pin, ADC
3  from time import sleep
4  #from GPSInformation import gps_main, GPSPrint, GPSSatellitesUsed, GPSTiden, gps_to_adafruit
5  import gps_funktion
6  import neopixel
7  from colorpicker import set_color
8  from Display4X7 import display, variableY
9  import Tiltensor
10 import Batteryreader
11
12 # Her kan i placere globale variabler, og instanser af klasser
13 r = 255
14 g = 255
15 b = 255
16 GPSInformation = ['', '']
17 set_color(0,0,0)
18
19 while True:
20     try:
21         # Denne variabel vil have GPS data når den har fået kontakt til satellitterne ellers vil den være None
22         gps_data = gps_funktion.gps_to_adafruit
23         print(f"\ngps_data er: {gps_data}")
24         sleep(4)
25         mqtt.web_print(gps_data, 'dani636e/feeds/iot.iotmaps/csv')
26         set_color(r, 0, 0)
27         sleep(4) # vent mere end 3 sekunder mellem hver besked der sendes til adafruit
28         Batteryreader.battery_percentage = Batteryreader.batteryPowerReaderConverter()
29         print("The battery percentage is:", Batteryreader.battery_percentage, "%")
30         mqtt.web_print(Batteryreader.battery_percentage, 'dani636e/feeds/iot.iotbatteri')
31         sleep(4)
32         set_color(0,g,0)
33         display()
34         Tiltensor.tiltSensor(Tiltensor.direction, Tiltensor.countTackels)
35         print(Tiltensor.countTackels)
36
37         gps_funktion.gpsSecondFunktion()
38         GPSInformation = gps_funktion.GPSTiden
39         mqtt.web_print(GPSInformation, 'dani636e/feeds/iot.iotfeed')
40         GPSInformation = gps_funktion.GPSSatellitesUsed
41         sleep(4)
42         mqtt.web_print(GPSInformation, 'dani636e/feeds/iot.iotfeed')
43         set_color(r, g, b)
44         # Jeres kode skal slutte her
45         sleep(0.5)
46         if len(mqtt.besked) != 0: # Her nulstilles indkommende besked
47             mqtt.besked = ""
48         mqtt.sync_with_adafruitIO() # igangsæt at sende og modtage data med Adafruit IO
49         print(".", end = '') # printer et punktum til shell, uden et enter
50     # Stopper programmet når der trykkes Ctrl + c
51     except KeyboardInterrupt:
52         print('Ctrl-C pressed...exiting')
53         mqtt.c.disconnect()
54         mqtt.sys.exit()

```

Vores loop starter med at hente listen i linje 23, og sætter det til variabelen `gps_data`, hvorefter det printes i næste linje til shell. For at mqtt beskederne ikke spammer adafruit er vi nødt til at have et sleep per besked sendt til adafruit, så her er en af de første sleeps på 4 sekunder, siden det er et infinity loop som kører så længe condition er true, men der er ikke angivet nogle parametre for en false scenario hvilket gør det til et infinity loop, med mindre man afbryder koden med Ctrl + C.

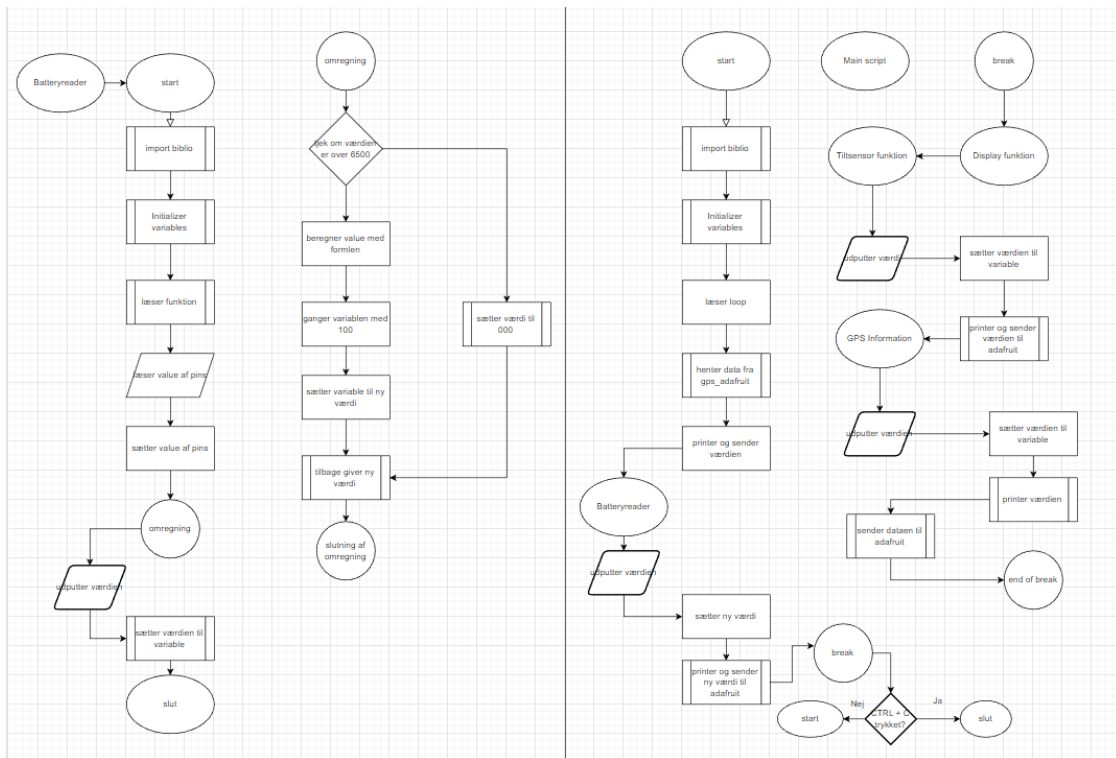
linje 26, gør brug af MQTT fra libraries som er hentet fra github (tilgængeligt fra materialerne udgivet af vejlederne). linje 27 kalder et script der sætter en neopixel ring til rød, for at vise koden er startet og der er sendt gps data til adafruit. (Vi bruger neopixel ringen i denne kode til at se hvor langt koden er nået, men den er ikke anvendt i det endelige løsning og produkt)

linje 28 er sleep til at undgå blokering fra adafruit.

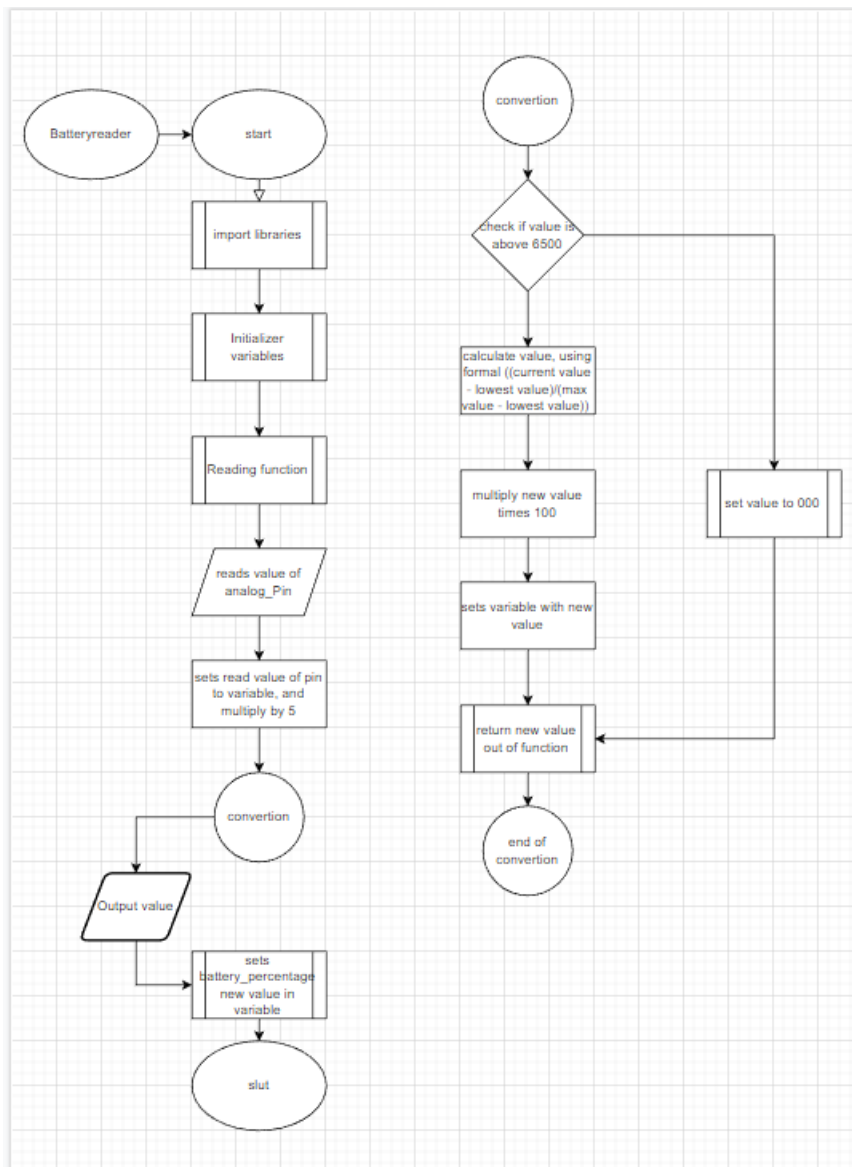
linje 29 henter vores batterimåler som henter værdien af variabelen og sætter den til en variabel angivet i main, og opdatere den i loopet. Linje 30 printer værdien i shell. hvorefter linje 32 sender dataen til adafruit med MQTT, efterfulgt af en sleep på 4 sekunder for at undgå blokering. I linje 34 sættes neopixel ringen til grøn for at vise koden er nået forbi batterimålerne. (for at gøre troubleshooting nemmere). I linje 35 kaldes display funktion som opdaterer batteri niveauet på displayet, med den nye variable fra batteri procenten.

linje 40-47 er mere til administrationen af produktet, som viser hvor mange satellitter som GPS'en har adgang til og hvad tiden er. Disse printes i shell og sendes til adafruit. (vi vil ikke gøre mere i dybden med dem siden det ikke er nødvendigt for det endelige løsning, bare nice-to-have, for anvenderne af løsning kan se om dataen, kan være påvirket af mangel på GPS forbindelse).

Flowcharts:



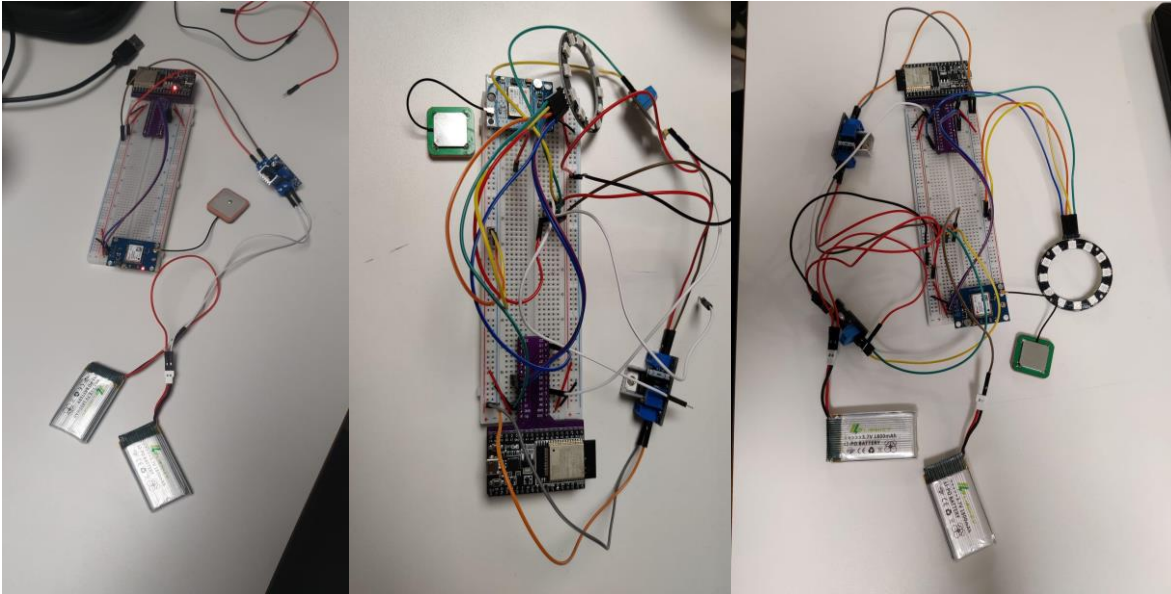
MainScript flowchart. Vi har opdelt dette flowchart ved hjælp af et break, hvor det første break er den nødvendige kode til at løse kravene for prioritet 1, og efter break er så til prioritetene 2. Vi anvender forskellige processer, hvor nogle er forudbestemt, som at printe en værdi fra en variabel. Flowcharts for Display og Tiltensor kan findes i bilag, GPS har vi valgt ikke at lave flowcharts til siden det er mere for administrator af løsning, script skulle have vist og sendt en liste af information til Adafruit dashboardet, her i blandt altitude, satellites og tiden, mm.



9. Test af løsning

9.1 Pathfinding testresultater

I starten satte vi GPS'en op til at kører på bread-boardet, for det var den første prioritet vi skulle have styr på, Hvor vi skulle sætte to batterier til for at få GPS'en tændt og sende dens data/koordinater til adafruit-io. Ved hjælp af "LDO" som er vores spændingsregulator og en spændingsdel, som måler batteriniveauet. Det gik fint med at sende GPS data, og vise batteri procent på dashboard, men den batteriprocent vi fik via dashboardet var ikke særlig præcis, da vi testede den. Bagefter lavede vi om på regnestykket for at få batteriniveauet mere præcist. Det endte med at det hele slukket, da batteriet var på 27%. Så skulle vi lave regnestykket om igen og igen, efter det begyndte den at vise et helt forkert tal, som endte på ca. -4000, ellers så var den over 100% og det var ikke det vi skulle have som resultat. Vi endte med at bruge mere end 4 dage. Efter vi fik lidt hjælp fra de andre grupper og nogle af lærerne fik kigget på det, fandt vi løsningen. Det næste problem som vi fik var at resultatet ikke blev opdateret, den blev ved med at sende den samme batteristatus selvom vi havde haft vores komponenter til at være tændt i lang tid. Der fandt vi så ud af, at det skulle sættes ind i en "while løkke" så den opdaterede batteriprocenten. Det gjorde vi så og så var problemet løst. Vi endte med at kunne nå at lægge 4x7 segment displayet på som kunne vise, vores batteri procent på selve vesten.



9.2 Dokumentation af DUT

Vi har lavet en lille video som viser, hvordan vores vest fungerer samt, hvordan man starter det hele op.

Se billeder i bilag nr. 9.2:

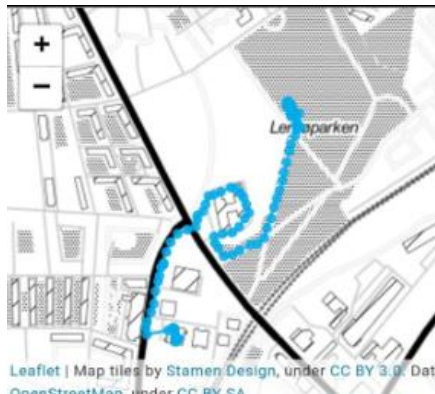
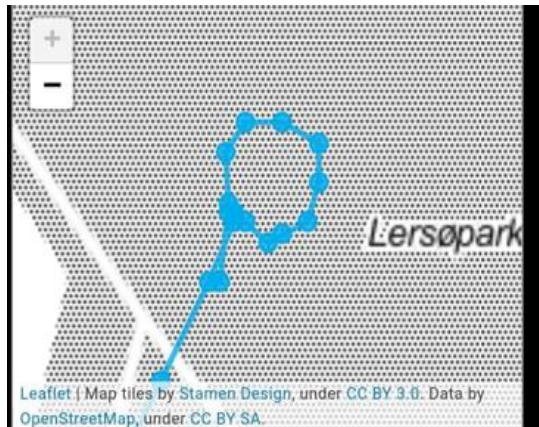
<https://youtu.be/mUeazu0-cvQ>

9.3 Specifikationer på vesten

	Nuværende DUT stadie	Ønsket endelig stadie
Højde, bredde, længde	<p>Kassen på ryggen</p> <p>H 2,5 cm. B 3,5cm L 10,5 cm</p> <p>Display på højre skulder</p> <p>H 1,6cm B 2,8cm L 4,6cm</p> <p>Kassen på højre bryst</p> <p>H 2cm B 3,3 cm L 10cm</p> <p>Mobil hotspot på venstre bryst.</p> <p>H 1,8 cm B 5,7 cm L 9,6 cm</p>	<p>Ønsket har været at tingene skulle passe til vestens lomme, samt at vores kasser skulle være robuste</p>
Vægt	412g	

Se bilag nr. 9.3:

9.4 Udførsel af brugertest

Dato	Test beskrivelse og resultater
26.10.22	<p>Vi fik sat gps og batteri på et breadboard og skulle så ud og teste, hvor godt gps'en virkede. Vi gik i samlet flok ned mod en fodboldbane som lå tæt på. Gps'en sendte data til adafruit som fulgte via telefonen. se bilag. Da vi kom tilbage til klassen, kunne vi se at batteriniveauet ikke havde ændret sig, og vi blev derfor nødt til at få ændret det</p> 
1.11.22	<p>Vi var ude og teste vores gps og batteri igen. Denne gang skulle vi teste om vores gps kunne give en cirkel på adafruit med en diameter på 10 m. Vi fik også tjekket vores batteri og denne gang virkede den</p> 

9.5 Udførsel af Accepttest på DUT

ID: 1	Krav: Løsningen skal som minimum kunne fremvise lokation og batteridata via et online dashboard, med en opdateringsrate på minimum 2 opdateringer i minuttet.	Prioritet: 1
Kategori: UI	<p>Accepttest: Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>Step 2) Indenfor 4 minutter bliver position og batteriniveau data præsenteret i et online dashboard.</p> <p>Step 3) Et stopur sættes i gang, og der måles hvor mange opdateringer dashboardet får over en periode på 5 minutter.</p> <p>Vil der over perioden være 10 eller flere opdateringer, så vurderes kravet som bestået.</p>	Det vurderes at kravet er fuldført.

ID: 2	Krav: Den kropsbårne løsning skal være batteridrevet, og skal have over 115 minutters batterilevetid.	Prioritet: 1
Kategori: Power	<p>Accepttest: Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>Step 2) Et stopur sættes i gang, og testbrugeren påbegynder en trænings session på 2 x 45 minutter, med en 15 min pause i mellem.</p> <p>Hvis løsningen stadig har over 10% batteri ved afslutningen af forløbet på 115min, så vurderes kravet at være opfyldt.</p> <p>Vil der over perioden være 10 eller flere opdateringer, så vurderes kravet som bestået.</p>	<p>Det vurderes at kravet er fuldført.</p> <p><i>9.6 Bilag:</i></p>

ID: 3	Krav: Løsningen skal være opkoblet til internettet, og være mobil.	Prioritet: 1
Kategori: Connectivity	<p>Accepttest: Step 1) Løsningens hardware del placeres på en testbruger, aktiveres, og dashboardet observeres</p> <p>Step 2) Testbrugeren går 1km væk fra startpunktet og vender tilbage til startpunktet.</p> <p>Hvis løsningen opretholder internetforbindelsen og ingen pakkeab har gennem hele gåturen, vurderes kravet at være opfyldt.</p>	Det vurderes at kravet er fuldført.

ID: 4	Krav: Løsningens hardware del skal være kropsbåret, robust, og ikke være til fare for spiller under en fodboldkamp.	Prioritet: 1
Kategori: Mechanical	<p>Accepttest: Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>Step 2) Testbrugeren ligger sig ned på græsplæne og ruller rundt 10 gange.</p> <p>Step 3) Testbrugeren bliver tacklet 10 gange</p> <p>Hvis testbrugeren ikke oplever nogen gener, stammende fra løsningens hardware del, fra denne test, og at elektronikken ikke udviser tegn på skader, så vurderes kravet at være opfyldt.</p>	<p>Det vurderes at kravet er fuldført.</p> <p>https://youtube.com/shorts/LILmrMog2-E?feature=share</p>
<p>Diskussion og evt. ønskede ændring af krav:</p> <p>Vi valgte at gøre det indenfor på gulvet, da græsset var meget vådt, men vi mener at, hvis vesten kan klare at rulle på gulvet, kan den også klare det på græsset</p>		

<p>ID: 5</p> <p>Kategori: Sensing</p>	<p>Krav: Løsningen skal kunne måle og visualisere brugerens position.</p> <p>Accepttest: Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>Step 2) Testbrugeren går i en cirkel med en diameter på 10 meter.</p> <p>Step 3) Testbrugers position visualiseres via dashboardet</p> <p>Hvis dashboardet fremviser en cirkel, med en diameter på 10 meter (+-3 meter), vurderes kravet at være bestået.</p>	<p>Prioritet: 1</p> <p>Det vurderes at kravet er fuldført.</p> <p>Se bilag nr. 9.5:</p>
--	--	---

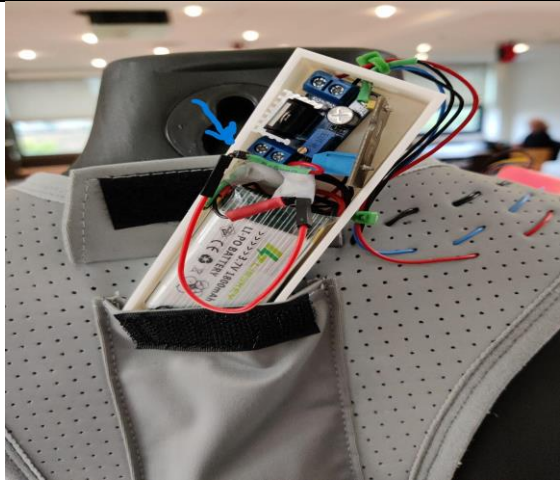
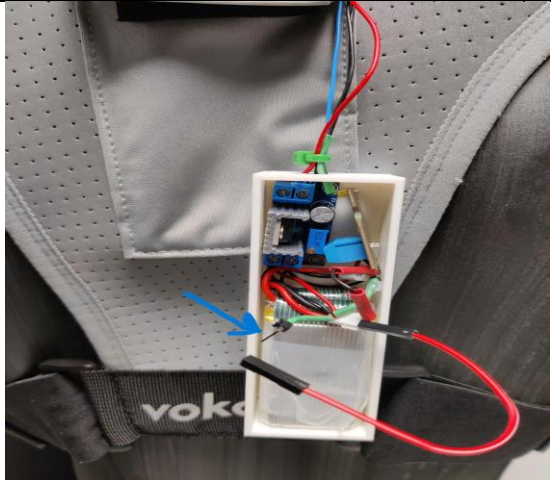
<p>ID: 6</p>	<p>Krav: Løsningen bør indikere hvis spilleren er tacklet, og fremvise antal af tacklinger direkte på løsningens hardware del.</p>	<p>Prioritet: 2</p>
<p>Kategori: UI</p>	<p>Accepttest: Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>Step 2) Der observeres displayet på løsningens hardware del at antallet af målte tacklinger er 0</p> <p>Step 3) Testbrugeren bliver tacklet 10 gange</p>	<p>Ikke lavet</p>

	Hvis der observeres at displayet fremviser tallet 10, efter at step 3 er gennemført, så vurderes kravet at være opfyldt.	
ID: 7	Krav: Løsningen bør kunne fremvise batteriniveau på løsningens hardwaredel.	Prioritet: 2
Kategori: UI	<p>Accepttest:</p> <p>Step 1) Løsningens batteri lades op til 100%, og batterispændingen måles med et multimeter.</p> <p>Step 2) Løsningens hardwaredel placeres på et bord, og batteriniveauet bliver observeret både på dashboardet (krav 1), men også via en passende indikator på løsningens hardwaredel.</p> <p>Step 3) Der foretages batterimålinger med multimeter hver halve time, over en periode på 3 timer.</p> <p>Hvis batteriniveauet der måles med multimeter, og batteriniveauet fremvist på løsningens hardwaredel, samt dashboard er indenfor 15% nøjagtighed, så vurderes det at kravet er opfyldt.</p> <p>Hvis der observeres at displayet fremviser tallet 10, efter at step 3 er gennemført, så vurderes kravet at være opfyldt.</p>	<p>Det vurderes at kravet er fuldført.</p> <p><i>9.7 Bilag:</i></p>

ID: 8	<p>krav: Løsningen kan registrere det højeste belastningstal hver gang fodboldspilleren lander under træning, og så få dem vist på et dashboard.</p> <p>“Belastningstal”: G-Kræft Sensor der giver en nummerværdi, ud fra hvor højt og hårdt man lander.</p>	Prioritet: 3
<p>Kategori:</p> <p>Sensing</p>	<p>Accepttest:</p> <p>Step 1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres.</p> <p>step2) Testbrugeren hopper 10 gange</p> <p>Testen skal kunne fremvise den højeste tryk på fodboldspillere efter hver landing og har dem sendt på et dashboard en af gang. Den har som min. registret 10 landinger betragtes det som gennemført</p>	Ikke lavet

ID: 9	Krav: Løsningen kan registrere fodstilling og knæ bevægelse samtidig.	Prioritet: 3
Kategori: Sensing	Accepttest: step1) Testbrugeren ifører sig løsningens hardware del, og hardwareløsningen aktiveres. Step2) Test Brugeren bevæger sit knæ mens foden sidder fast i jorden, så skal den kunne fremhæve en tone. Hvis der observeres en lyd, når test brugeren har bevæget sit knæ mens foden sidder fast i jorden, betragtes det som fuldført.	Ikke lavet

Testopstilling

Bruger tilgår batteriet	Bruger udtager batteriet
	

Her vises hvor batteriet sat til Pinen.(Strøm til)
Tag det rød lang kabel og sæt det til den ledige
Pin.

Her vises hvor Batteriet er sat fra Pinen.(Strøm
fra)

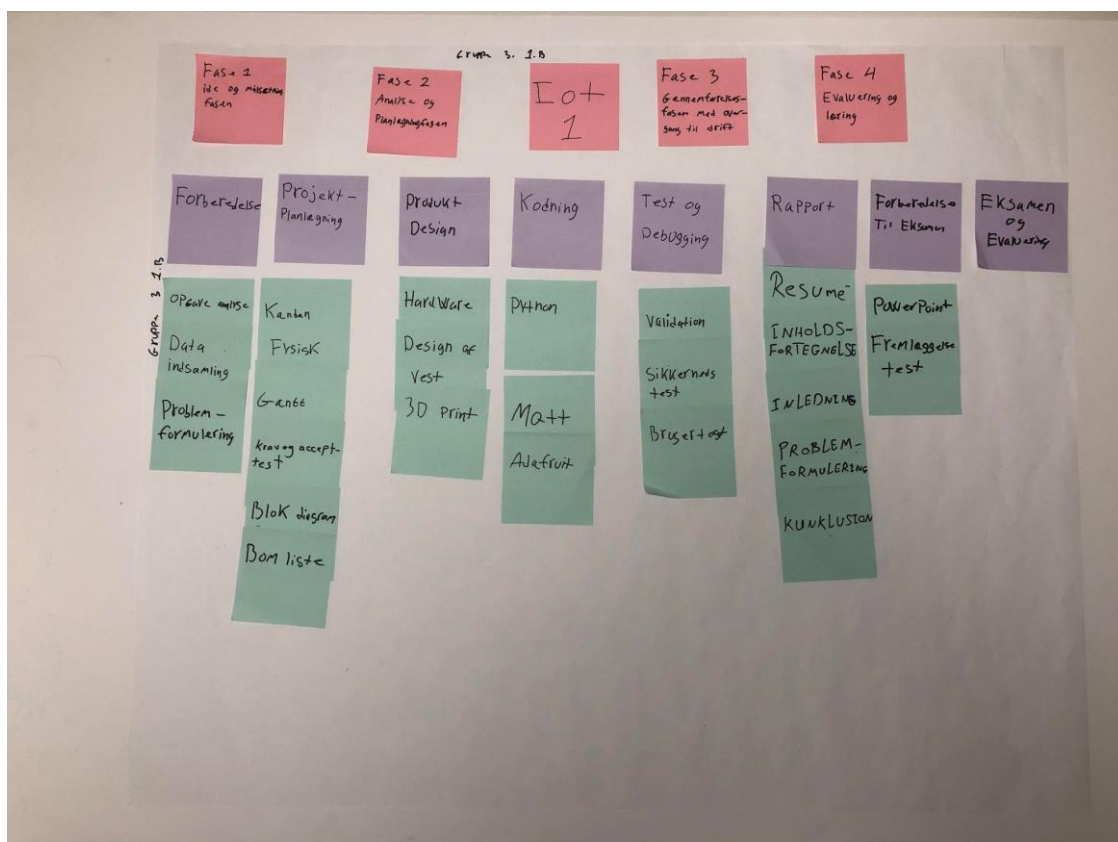
10. Praktisk projektplanlægning og ledelse

10.1 WBS

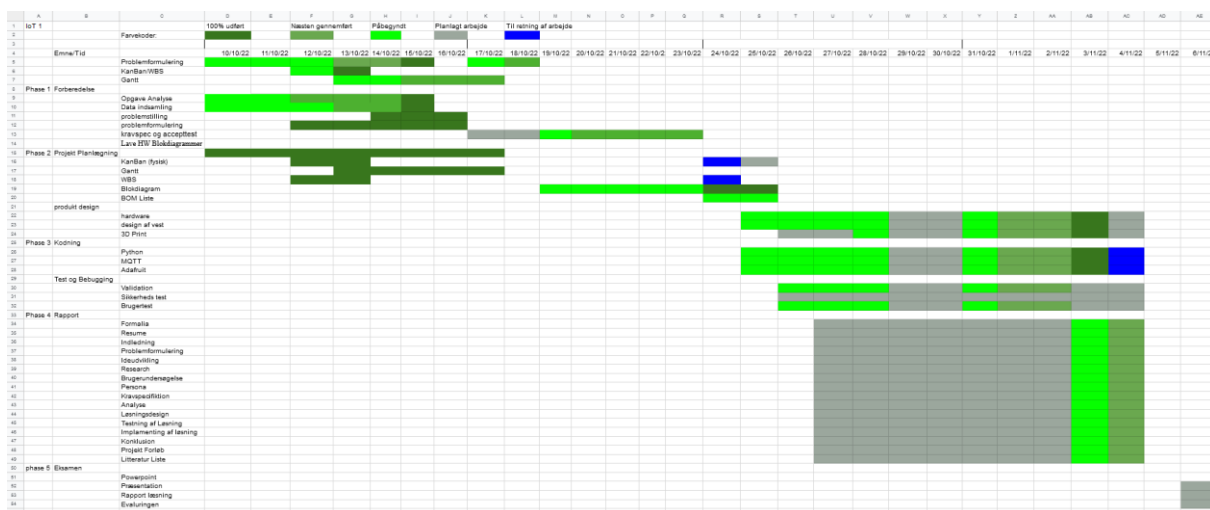
For at kunne gøre projektet nemmere at håndtere er det bedst at nedbryde projektets overordnede faser til flere rækker af mere overskuelige opgaver, og det hjælper med at opfylde projektets formål.

Da en WBS ikke har nogle restriktioner til sig, da det er en overordnet planlægger, var det derfor et ideelt værktøj at bruge til at kunne sætte sig i overblik over hele projektets gang.

Projektgruppen har brugt WBS til at sætte en struktur i workflowet, ved at opdele hele projektet i 4 forskellige faser, med de 4 faser har vi sat nogle hovedmål i de forskellige faser. Ud fra hovedmålene kan man derfor så sætte de forskellige opgaver der skal fuldføres. WBS'en har sat workflowet for projektgruppen og den rækkefølge som tingene er blev udført i.



10.2 Gantt



Figur 1, figur af gantt

(ser fuldt gantt i bilag)

Gennem vores projekt har vi anvendt gantt diagram, til at hjælpe med tidsplanlægning.

Opgavebeskrivelsen krævede at vi brugte gantt, og ikke måtte anvende andre værktøjer som

“HackNPlan” som er en automatiserer version af gantt, i disse programmer modtager man også mail

om deadlines og mm. Gantt har dog stadig tilbudt en del redskaber som specielt tidsplanlægning,

som kan ses på “Figur af gantt” hvordan vi har inddelt vores arbejde med forskellige farvekoder. Den

grønne farve er hvor langt i arbejdes processen vi er. (hhv. lige begyndt, cirka halvejs/næsten værdi,

og helt færdig), hvor grå er hvad vi havde planlagt, og som vi kan se blev en del af rapporttiden brugt

på at få løsningen færdig, fordi det er svært at skrive en rapport om en løsning der ikke er

fungerende. Blå er ekstra hvor vi fandt ud af at vi manglede at ændre noget, i vores hidtil færdige

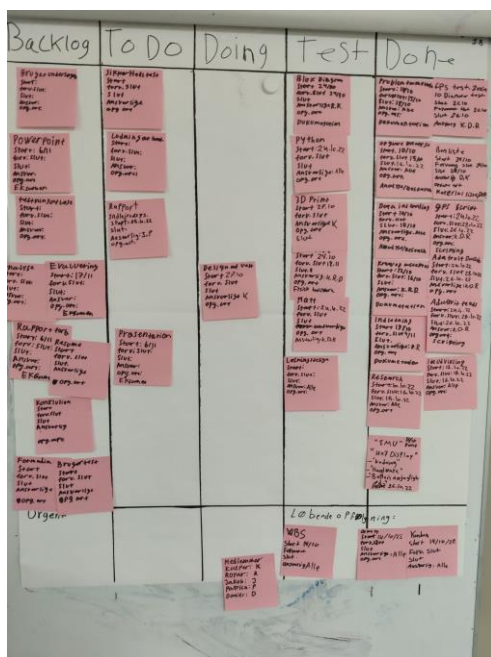
punkter. (som f.eks. koden, der var noget rengøring af koden som er indikeret med blå, under den

4/11).¹

¹ (vedhæftet (Gantt.HTML), fordi det var den eneste måde at exportere gantt uden det ødelagde layout)

Konceptet i gantt er virkelig god til at tidsstyre og som vi kan se burde vi nok havde brugte mere tid i uge 2 på vores løsning, for at komme i mål til tiden, som vi enlig havde planlagt.

10.3 Kanban



Vi brugte kanban boardet fysisk, dette har en del restriktioner som online kanban board ikke har. Kanban er stadig værktøjet som vores gruppe brugte til at skabe overblik over de opgaver der skal laves. En feature som online kanban board værktøjer som f.eks. "Trello" er man kan gå tilbage i sin tidslinje og se hvor hvornår ting blev færdiggjort og hvornår der var sat deadline til, heriblandt vil man også modtage notifikation om at der er deadlines man skal nå. Det tilbyder den fysisk dog ikke. Kanban board hjælp os stadig med at se, hvilke opgaver vi burde fasttrack og hvilke vi godt kunne vente med. Kanban for vores gruppe var primært overblik og arbejdsopgaverne, og fordeling af dem.

Se bilag nr. 10.3:

10.4 Logbog

Vi har ført logbog under hele projektet. Den ligger i bilagsmappe.

11. Konklusion

Vi har under vores arbejdsproces, arbejdet med at skabe en videreudvikling af Vokalo vesten som kan måle og informere træneren om eventuelt, spiller som overanstrenger sig under en træningssession. Under vores arbejdsproces kom vi frem til at skabe en løsning som har et elektrisk kredsløb, her under processen tog anvendelsen af batteri og batterimåler meget længere tid end forventet og derfor var vi nødt til at skubbe vores originale tidsplan eftersom batteri er en vigtig komponent til vores løsning. grundet denne tidsplan nået vi ikke i mål med alle kravene til vores produkt. Det kan også stærkt ses på vores uge 2, i vores gantt diagram.

Den endelige Løsning til "Hvordan mindskes knæskader ved hjælp af IoT (Internet of Things)" løsning- " kunne derfor ikke gennemføres, fordi vi valgte at fokusere på prioritet 1 og 2, fordi ekspertisen for prioritet 3 ikke var god nok til at kunne fuldføre de prioriteter. Og for at kunne at fuldføre så meget, som muligt indenfor den givne deadline blev der fokuseret mere på de to første prioriteter. Derfor mener vi det har en større relevans for det Vokalo ønskede fra løsningen.

12. Projektforløbet

Under starten på vores projektforløb var hele gruppen til stede og alle var klar til at komme i gang, dog skete der hurtigt ændringer i gruppen. Vi var alle seriøse i starten, men havde svært ved at finde en god problemformulering og kravspecifikation, som gjorde at vi følte vi ikke kom nogen vegne. Ugen efter fik en fra gruppen corona og var derfor væk fra gruppen i en uge. Vi havde også et andet medlem af gruppen, som havde det svært med nogle af lærerne, og derfor valgte at droppe ud fra skolen. Her blev til kun 3 mennesker som der arbejdede i gruppen. I den sidste uge blev vi 4 i gruppen igen efter den syge var rask igen. Og så ville halvdelen af gruppen helst arbejde hjemmefra og halvdelen vil møde op fysisk i skolen. Fejlen var at vi ikke fik skrevet alle de her ting ind i vores gruppe kontrakt. Der var heller ikke en leder til at styre gruppen. For her var tankegangen, at alle er voksne og skal være ansvarlige for det man skal under projektet.

Til næste gang, der kan forbedres mange ting. for eksempel, snakke mere sammen, om hinandens interesser (under de faglige emner) og måske lidt personligt også. Dvs. forbedrer kommunikationen mellem gruppemedlemmer. Tage kontrakten mere seriøs og sætte nogle regler som skal overholdes. For eksempel, regler om forberedelse før vi møder op, eller når vi er i de sidste faser, alle skal møde op uanset hvad. (sygdom undtaget). En ting, som kan forbedres til næste gang, kan være, at man får opgaverne delt op mellem gruppemedlemmer og tager tingene en ad gangen. i forhold til kravene i projektet

13. Perspektivering

først punkt for fremtidige proces af vores løsning vil være og få 3. prioriteres krav opfyldt, eftersom det er primært dem som sætter fokus på knæskaderne som kan ske hos fodboldspilleren.

Andet punkt kunne være at inddrag G-kraften og beregner hvor meget ens ben enlig belastes under træning. En af de svære mål her er dog at man jo ikke kan sætte en måler ind i benet og måle på den måde, men man kan måle cirkuleret tryk på spillernes ben.

14. Litteraturliste

(Dr.dk/nyehder, 2022)

<https://videnskab.dk/krop-sundhed/otte-typiske-skader-i-en-fodboldkamp>

Fronter (Programmering #1.pdf, Programmering #2.pdf)

Fronter Indlejrede systemer (INDL 03.pdf)

Kea IT labs

Chong, R. (2004). *annals.edu.sg*. Hentet fra <https://annals.edu.sg/pdf200405/V33N3p298.pdf>

Nader Rahnama, E. B. (9-14. 05 2014). Hentet fra link.springer:

<https://link.springer.com/article/10.1007/s11332-009-0070-1>

15. Bilag

8.3 Bilag:

Koder blev brugt til at vise den rigtig batteri % indtil vi fik den rigtigt der ligger øverst på rapporten

8.3:

```
import umqtt_robust2 as mqtt
from machine import Pin, ADC
from time import sleep
#import simple_gps_example

analog_Pin = ADC(Pin(34))
analog_Pin.atten(ADC.ATTN_11DB)
analog_Pin.width(ADC.WIDTH_12BIT)
# Her kan i placere globale variabler, og instanser af klasser

while True:
    try:
        # Denne variabel vil have GPS data når den har fået kontakt til
        #
        analog_val = analog_Pin.read()
        #print("Raw analog value: ", analog_val)
        #sleep(1)
        volts = (analog_val * 0.00095235)*5

        #print("The voltage is:", volts, "v")
        sleep(4)
        battery_percentage = volts*50 - 710
        print ("The battery percentage is:", battery_percentage, "%")
```

```
import umqtt_robust2 as mqtt
from machine import Pin, ADC
from time import sleep
#import simple_gps_example

analog_Pin = ADC(Pin(34))
analog_Pin.atten(ADC.ATTN_11DB)
analog_Pin.width(ADC.WIDTH_12BIT)
# Her kan i placere globale variabler, og instanser af klasser

while True:
    try:
        # Denne variabel vil have GPS data når den har fået kontakt til
        #
        analog_val = analog_Pin.read()
        #print("Raw analog value: ", analog_val)
        #sleep(1)
        volts = (analog_val * 0.00095235)*5*0.94

        #print("The voltage is:", volts, "v")
        sleep(4)
        battery_percentage = volts*50 - 320
        print ("The battery percentage is:", battery_percentage, "%")
```

```
analog_Pin = ADC(Pin(34))
analog_Pin.atten(ADC.ATTN_11DB)
analog_Pin.width(ADC.WIDTH_12BIT)
# Her kan i placere globale variabler, og instanser af klasser

while True:
    try:
        # Denne variabel vil have GPS data når den har fået kontakt til
        #
        analog_val = analog_Pin.read()
        #print("Raw analog value: ", analog_val)
        #sleep(1)
        volts = (analog_val * 0.00095235)*5

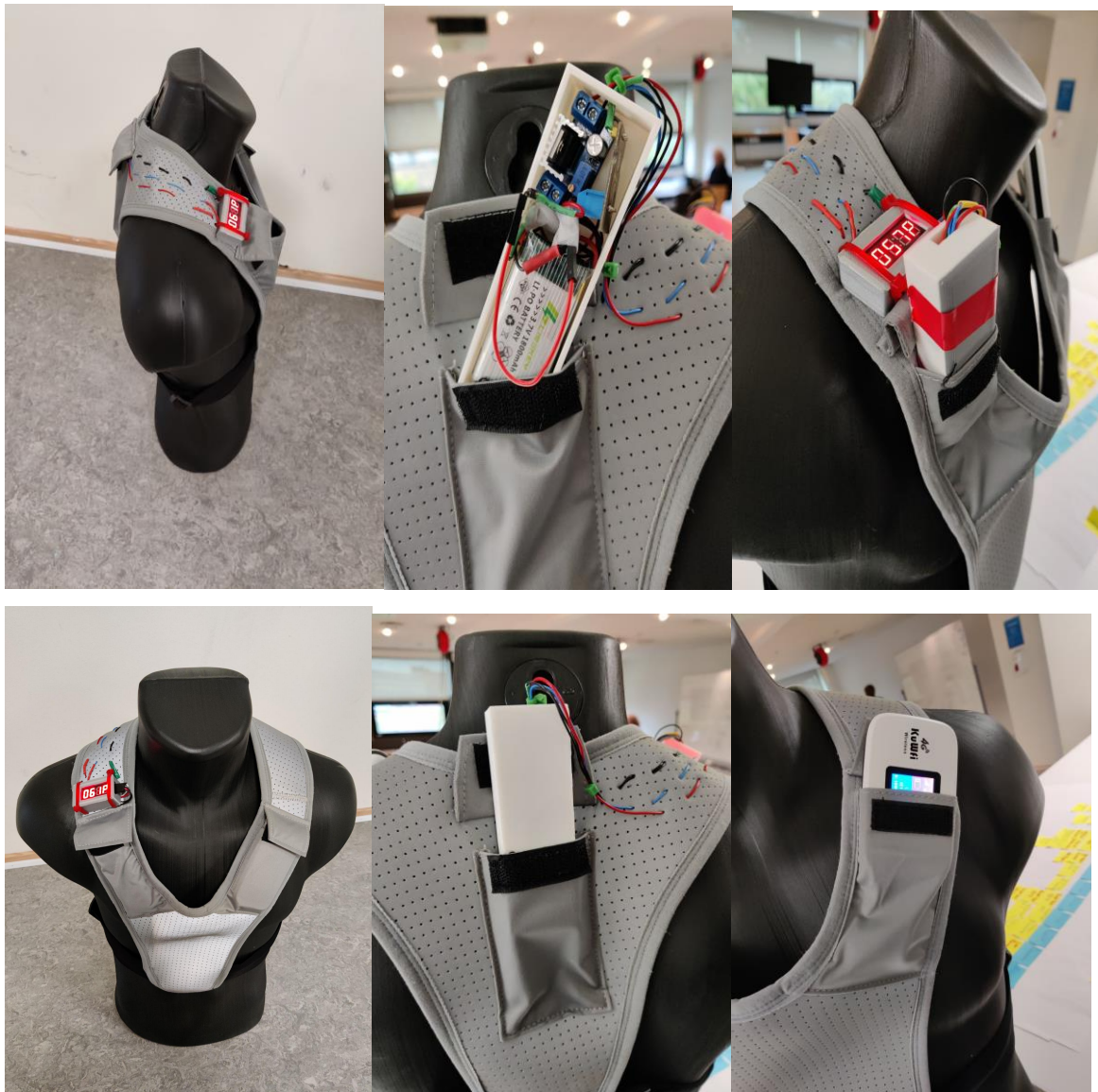
        #print("The voltage is:", volts, "v")
        sleep(4)
        battery_percentage = volts*100 - 320 /2
        print ("The battery percentage is:", battery_percentage/2, "%")

        mqtt.web_print(battery_percentage/2)
```

```
1 import umqtt_robust2 as mqtt
2 from machine import Pin,ADC
3 from time import sleep
4
5 analog_Pin = ADC(Pin(34))
6 analog_Pin.atten(ADC.ATTN_11DB)
7 analog_Pin.width(ADC.WIDTH_12BIT)
8
9 while True:
10     try:
11
12         analog_val = analog_Pin.read()
13         #print("Raw analog value: ", analog_val)
14         #sleep(1)
15         volts = (analog_val * 0.00095238)*5
16
17         # print("The voltage is:", volts, "v")
18         # sleep(1)
19         battery_percentage = volts*100 - 320
20         print ("The battery percentage is:", battery_percentage, "%")
21
22         mqtt.web_print(battery_percentage)
23
24         sleep(0.5)
```

9.2 Bilag

Billederne viser hvordan vesten ser ude og hvad blev lagt ind i hver kasse:



9.3 Bilag

Vægten på vesten

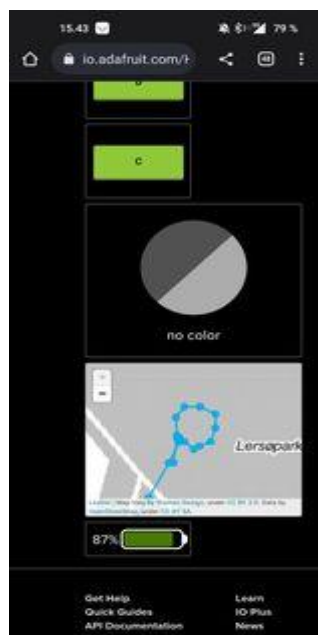
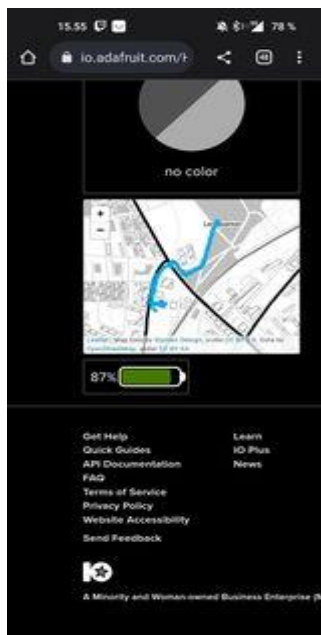


9.5 Bilag:

ID 1

2 km tur m. Batteri %

Circle på 10m Diameter



9.6 Bilag:

Den første 45 min



15 min pause



efter den 2. 45 min



9.7 Bilag:

ID 7

Batteri % på hardware delen



10.3 Bilag

