

5.semester - IoT

Af Jannik Thygesen, studienummer 202107062

1.0 Introduktion

Sideløbende med undervisningen i "EH5IOT-01 Internet of Things" arbejdes der på et IoT-projekt. Arbejdet med dette projekt dokumenteres løbende i denne rapport og ved afslutningen af semeseret, så afleveres rapporten via Digital Eksamen.

Projeket skal have disse grundlæggende krav opfyldt(fuld liste ses på Brightspace, Exam Project):

- Krav 1: Skal drives af en Particle Argon eller lignende
- Krav 2: være forbundet til internettet og bruge data fra internettet
- Krav 3: Skal være forbundet til lokal enhed, fx sensor eller ligende og bruge data fra lokal enhed
- Krav 4: Systemet skal **samlet** have en bedre funktionalitet end tilsvarende system uden adgang til internettet

2.0 Ide- projektbeskrivelse for IoT-projektet

Forløbig ide:

Rumstationen ISS flyver konstant rundt om jorden og viser sig ved forskellige geografiske steder på jorden til forskellige tidspunkter. Ønsker man at se ISS med egne øjne, så skal primært to krav være opfyldt: 1. den skal være synlig fra den by du befinder dig i og 2. det skal gerne være mørkt.

Derfor er ideen, at få data fra ISS Tracking API ned til Particle Argon. Derudover skal der tilkobles en magnetometer til Partile Argon som kan måle retningen du kigger i. På baggrund af de to oplysninger skal det fremgå af konsolen hvorvidt du peger Particle Argon i den rigtige retning for at øge dine chancer for at se rumstationen på himmelen.

3.0 Krav analyse

- Krav 1: Projektet drives af en Particle Argon. Denne udleveret af underviser. Der er desuden indkøbt en Particle Photon 2 som backup og til udvikling i fremtiden.
- Krav 2: Fra Particle Console oprettes et Webhook der peger på en webpage med positionoplysninger for ISS som API data. Da API-linket returnerer flere oplysninger end der skal bruges i dette projekt, så er der lavet en sortering af disse under "Advanced Setting".
- Krav 3: Der er valgt et HMC5883 magnetometer til dette projektet. Denne lokale sensor måler magnetisk styrke i et X-Y-Z-plan, og resultatet af dette kan bruges til beregningen af retningen sensoren vender.
- Krav 4: Ved at sammenkoble ovenstående tre krav, så opfylder systemet kravene til projektet ved at indeholde internetdata samt lokal data og derudover, at virke bedre end ikke at være forbundet til internettet. I dette projekt ses det ved at give oplysninger om placeringen af rumstationen ISS. Uden disse oplysninger ville det være umuligt for brugeren at vide hvilken retning der skulle kigges i.

4.0 System design

Grundlæggende er systemet opbygget af både hardware og software. Her forsøges at opstille hvordan de enkelte dele arbejder sammen når systemet er i drift.

- Hardware

En HMC5883 magnetisk sensor er forbundet til Particle Argon. Der er brugt to pull-up modstande og ledninger til denne forbindelse.

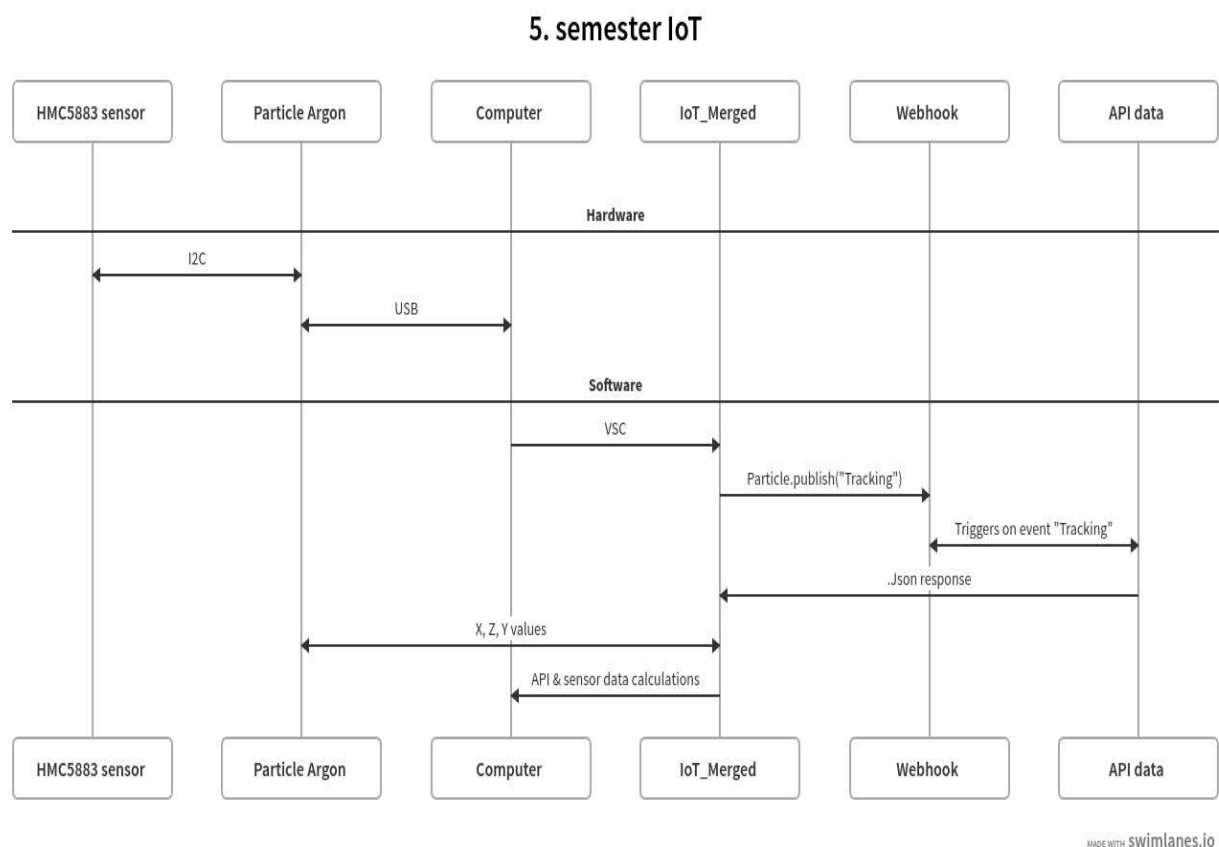
Particle Argon er forbundet til en computer med USB kabel.

- Software

Under afvikling af koden(IoT_Merged.cpp), så aktiveres et Webhook med et event kaldet "tracking". Når dette event bliver kaldt, så hentes API data fra det angivne link og myHandler funktionen bliver aktiveret, hvilket betyder det efterspurgte data printes i terminalen.

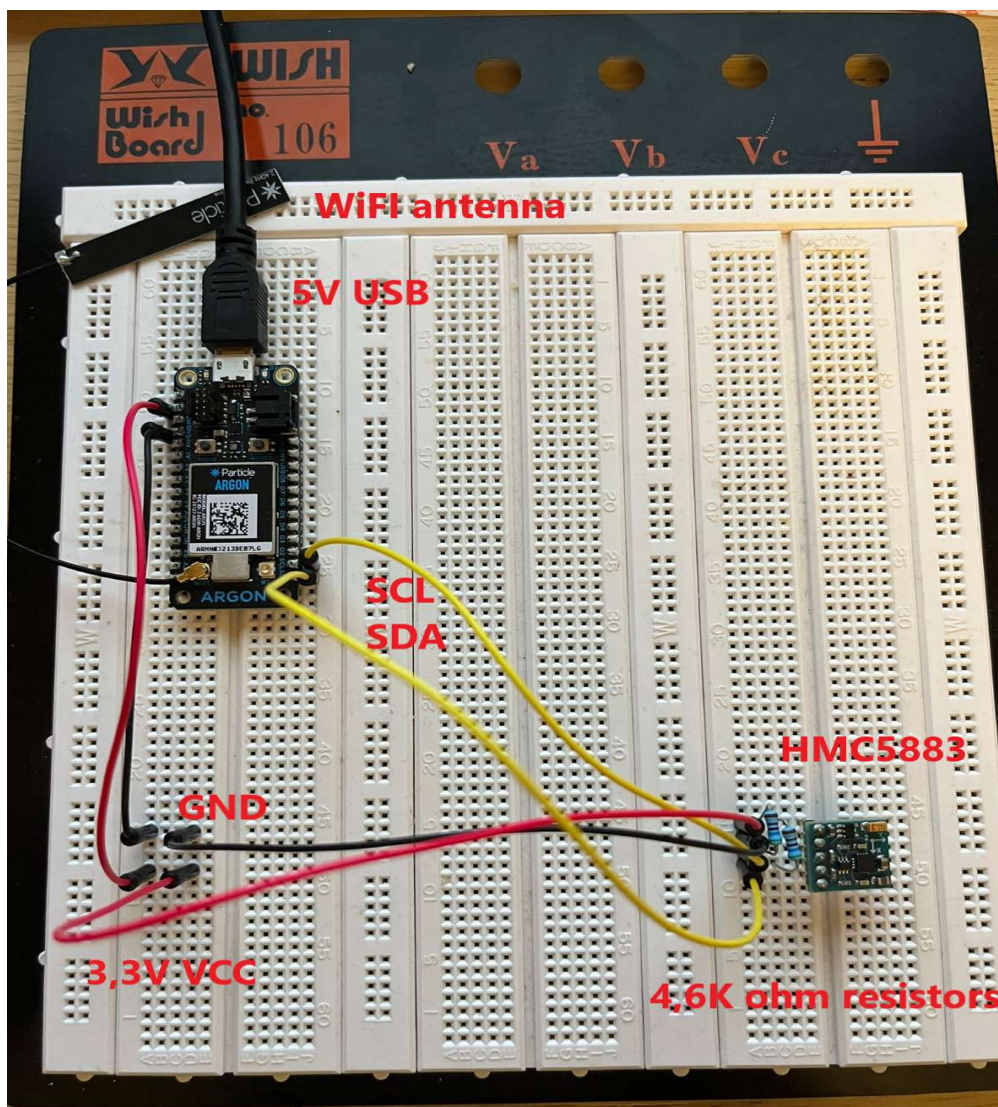
Udover at hente API data, så sker der også en lokal indhentning af data fra HMC5883 sensoren. Under afvikling af koden(IoT_Merged.cpp), så sættes der en I2C forbindelse op mellem Particle Argon og sensoren. Dette inkluderer bl.a. at angive variabler(data) til overførslen af data mellem sensor og Particle Argon samt foretage løbende omregninger af denne. Når data er modtaget bliver det placeret som 16-bit signed integers. Disse tal bruges herefter til at beregne "Heading"(0 til 360 grader) og igen til beregne "Direction"(North, West, South, East). Resultatet af disse omregninger bliver både vist på Particle Console med Particle.Publish ligesom "Direction" printes i terminalen sammen med API dataen.

En grundlæggende oversigt over systemet kan ses herunder og i pdf-filen "swimlanes-systemoversigt":



5.0 Implementering

For at få indarbejdet de opstillede krav til opgaven, så arbejdes der primært med disse én ad gangen. Dette sikre større overblik og kontrol over de enkelte funktioner, men gør det også nemmere for projektet, da der løbende kan bygges ovenpå eksisterende løsninger efterhånden som de kan verificeres. Hele systemet forbindes som vist på tegningen herunder og i png-filen "systemSamlet":



- Vedr. krav 1:
Particle Argon forbindes med USB-kabel til computeren og herefter oprettes forbindelse til lokalt Wi-Fi netværk. Der oprettes en konto på "Particle Console" som giver mulighed for at overvåge aktiviteten for ens device, oprette og redigere Webhooks, følge med i events samt meget andet.
Som IDE installeres Microsofts Visual Studio Code(herefter VSC) og der installeres udvidelsespakken "Particle Workbench" som giver mulighed for at interagere med Particle Argon direkte fra VSC, herunder at opdatere firmware, installere 3. parts biblioteker, compile kode og flashe denne direkte.
- Vedr. krav 2:
Der oprettes et Webhook kaldet "ISS Tracking" med et event kaldet "Tracking". Da der skal importeres data ind til Particle Argon, så angives request type som "Get" og der angives følgende link som kilde til positionsoplysninger for ISS:

<https://api.wheretheiss.at/v1/satellites/25544>

Under Advanced Settings angives følgende parametre til Response Template:

Latitude: {{latitude}},
Longitude: {{longitude}},
Visibility: {{visibility}},

Dette gøres, da API-linket returnerer en del informationer som ikke er relevante for dette projekt, f.eks. hastigheden, timestamps og position ifht. solen, så bruges ovenstående til at kun lade de relevante data passere til koden når eventet "tracking" bliver kaldt.

- **Vedr. krav 3:**

Der indkøbes en [HMC5883 sensor fra Arduinotech](#) til brug i dette projekt. Denne er valgt, da den opfylder de basale krav om at kunne måle magnetisk kraft i X, Y og Z retning, den kan drives med 3.3V direkte fra Particle Argon samtidigt med, der var umiddelbart understøttelse af denne sensor gennem [Particle Liberaies HMC5883](#).

Det lykkedes dog ikke at få denne firmware til køre på Particle Argon - formentligt grundet for store versionsforskelle mellem denne oprindelige firmware også de nyere der er tilgængelige. Der forsøges at bruge ældre firmware til Particle Argon, men uden held. Derfor importeres og bruges [følgende kode fra ControlEverything](#). Koden gør brug af I2C kommunikationsprotokol til at forbinde mellem sensoren og Particle Argon. Efter aflæsninger omdannes målinger til enheden "Gauss" som repræsentation for mængden af magnetisk kraft.

- **Vedr. krav 4:**

Efter der er installeret og opsat et programmeringsmiljø(krav 1) til brug med Particle Argon, og efter der er kan hentes API-data ind og vist på terminalen(krav 2), og efter der er etableret forbindelse til en lokal sensor(krav 3), så laves en samlet kode hvor alle funktionerne bliver samlet samme sted.

6.0 Test & Verifikation

Som angivet i ovenstående afsnit, så testes hver enkelt funktion med tilhørende kode individuelt før der fortsættes til næste funktion(angivet som krav).

- **Krav 1** Dette krav testet ved at være et fungerende udviklingsmiljø inden arbejdet med denne opgave startes. Dog opnås aldrig adgang til funktionen "local compile" som sikre, at der kan compiles på lokalt på device. Der fremkommer fejl relateret til C++ compileren, og selvom denne fjernes og installeres igen, så udbedres problemet ikke. Løsningen bliver i stedet, at bruge "cloud compile" som gør, at beskeden om at compile et program sendes via internettet i stedet for lokalt via USB-kabel. Dette tager en smule længere tid, men accepteres som en fornuftig løsning.





- **Krav 2** Der tages udgangspunkt i den første kode der arbejdes med i dette kursus, hvilket er [koden tilhørende det første Webhook](#). Denne modificeres til brug i dette projekt og den færdige kode kan ses i "API_data.ino" filen. Et eksempel på hvordan output ser ud kan findes i "API_data_OUTPUT.png" filen og herunder:

```
The ISS space station is currently at:
Latitude: -3.6921956648715,
Longitude: 108.26296427311,
Visibility: eclipsed
The ISS space station is currently at:
Latitude: -4.2011569320519,
Longitude: 108.62365276233,
Visibility: eclipsed
The ISS space station is currently at:
Latitude: -4.7098921161936,
Longitude: 108.98485272418,
Visibility: eclipsed
```

- **Krav 3** For dette krav importeres den tidligere omtalte kode fra anden Github side. Herefter forbindes sensoren til Particle Argon via SCL og SCA som vist på tegningen i afsnit 5.

Efter sensoren er forbundet så indlæses indholdet af "HMC5883.cpp" til device og et eksempel på hvordan resultatet ser ud ses i "HMC5883_OUTPUT.png" filen og herunder:

Events

<div><div></div><div>Search for events</div></div>		
NAME	DATA	DEVICE
Unpause to reveal 25 queued events.		
Direction:	West	ArgonT800
Heading:	290.409883	ArgonT800
Magnetic Field in Z-Axis :	-464	ArgonT800
Magnetic Field in Y-Axis :	-129	ArgonT800
Magnetic Field in X-Axis :	48	ArgonT800

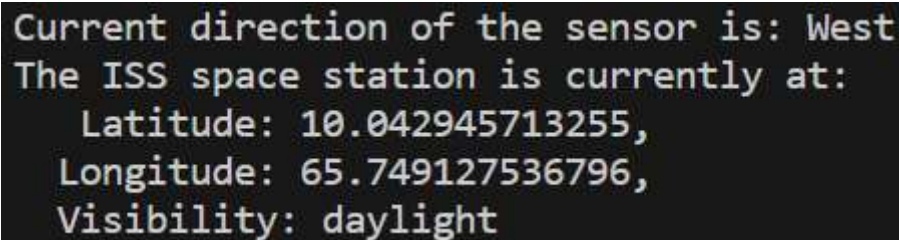
På billedet kan det ses hvordan aflæsningerne for X, Z og Y planen sammen med omregningen til "heading" og "direction" vises på Particle Console(som Event). Data bliver flyttet fra sensoren til Particle Argon via I2C protokollen, og herefter til Particle Console med en Particle.publish kommando.

For at efterprøve aflæsningen og den efterfølgende omregning, så sammenlignes resultatet med et kompas fra en indbygget app på en iPhone 15. Desværre viser det sig, at angivelsen af retningen ikke er stabil. Der vises oftest forkert retning, ligesom retningen ændres selvom sensoren ikke flyttes. Sensoren kan udemærket registrere ændringer, men disse ikke altid korrekt. Ved at kigge på de data der fremkommer af koden, så formodes problemet at ligge i de data der ligger til grund for udregningen, da selv ændringer giver store spring for værdierne X, Y og Z. Der ses også eksempler på disse værdier kan skifte selvom sensoren er fuldstændigt i ro. Der forsøges flere tiltag for at sikre stabil retningsangivelse.

- Gennemgået kode for fejl
- Gennemgået de fysiske forbindelser fra Particle Argon til Sensor og udskiftet ledninger
- Gennemgået miljøet for ydre, forstyrrende magnetisk påvirkning og flyttet sensoren så langt væk fra Particle Argon som muligt
- Gennemgået lodninger på sensoren for oplagte fejl ved lodning af pins
- Arbejdet med kalibrering

Desværre har det ikke været muligt at finde en oplagt fejl som har kunne udbedres. Det er vurderingen, at den mest sandsynlige fejl er mangelfuld kalibrering og/eller dårlig forbindelser ved lodning.

- **Krav 4** Efter arbejdet med ovenstående krav - især omkring krav 3 - samles de to koder til en enkelt. Dette gøres for at kunne drive projektet fremad trods ikke alle funktionerne virker helt efter hensigten. Den samlede kode kan ses i "IoT_Merged.cpp" filen og et eksempel på hvordan resultatet ser ud ses i "IoT_Merged_OUTPUT.png" filen og herunder:



```
Current direction of the sensor is: West
The ISS space station is currently at:
  Latitude: 10.042945713255,
  Longitude: 65.749127536796,
  Visibility: daylight
```

På billedet ses resultatet af fra importen af API-data samt resultatet af indlæsning og omregningen sensor data i terminalen på VSC.

7.0 Konklusion

Samlet set lykkedes det at designe et IoT-system som arbejder med de krav der er stillet til opgaven. De enkelte krav til projektet er brudt ned til mindre størrelser, og herefter implementeret og testet en ad gangen. I sidste ende er det lykkedes at arbejde med data fra internettet såvel som data som lokal data fra en sensor. Det er desværre ikke helt lykkedes at lave en pålidelig afmåling af den lokalt indhentede data fra HMC5883 sensor. Selvom der er gjort flere forsøg på at udbedre dette, så lykkedes det desværre ikke i sidste ende.

8.0 Perspektivering og fremtidigt arbejde

Der er flere oplagte tiltag der kan arbejdes med såfremt der ønskes flere og/eller bedre funktioner af dette IoT-system. Først og fremmest er det oplagt, at arbejde med det indkomne API-data til systemet. Selve angivelsen længdebred og breddegrad giver ikke den store information til brugeren om den nuværende lokation af ISS. Derfor var der arbejdet med en udvidelse af koden som skulle tage indholdet fra API feedbacket og placere det i variabler der herefter kunne bruges til omregning og videre behandling. På den måde skulle der ske en løbende omregning af længdegred og breddegrad til mere konkret stedsangivelse på jorden. Yderligere kunne systemet designes således, at brugeren fik besked på hvornår ISS var tæt på ens nuværende placering og brugeren var derfor ikke afhængig af at kontrollere dette konstant.

Der er også oplagte forbedringer til den del af koden der arbejder med den lokale sensor data. Udover at arbejde med en mere pålidelig sensortilslutning, så kunne de tænkte variabler med indholdet af API data bruges til sammenligning med den data fra den lokale sensor til angivelse af position og retning.

8.1 Aspekter af Cyber Security for dette IoT-system

Sideløbende med arbejdet af dette projekt har der været undervisning i forskellige emner der relaterer sig til IoT-enheder og deres funktioner. Et af disse emner har været kryptografi og emner indenfor cyber security, som det ville være relevante at gøre sig overvejelser om hvordan det kan implementeres i dette system. Selvom placeringen af ISS altid er kendt for offentligheden, så kunne samme IoT-system som omtalt i denne rapport bruges til at tracke en satellit, køretøjer som biler, både, fly eller lignende samt mange andre ting. Såfremt deres placering skulle holdes hemmelig for alle andre end brugeren af dette system, så kunne lokationsdata krypteres.

F.eks. kunne der installeres et bibliotek som [OpenSSL](#) hos både afsender og modtager. Med dette bibliotek kunne lokationsdata først krypteres fra plaintext til ciphertext. Dette gøres fra den afsendende enhed med en nøgle. Herefter kan den kodede ciphertext sendes af sted og modtageren kan bruge sin key til at oversætte ciphertexten tilbage til plaintext og få dens nuværende placering. På den måde kan man afsende lokationen på en enhed, men kun den rigtige modtager af informationerne vil være i stand til at afkryptere beskeden med sin nøgle. Denne fremgangsmåde kaldes for symmetrisk kryptering og kræver at både afsender og modtager deler den samme nøgle.

8.2 Aspekter af Machine Learning for dette IoT-system

En anden måde dette IoT-system kan forbedres på, er gennem brugen af Machine Learning (ML). Brug af ML kræver, at man indsamler data som man bruger til at bygge en model som herefter kan bruges af ens IoT-enhed. Konkret for dette projekt, så kunne man forestille sig at bygge et datasæt bestående af positionsoplysninger med ISS til bestemte tidspunkter på døgnet. Disse data kunne så klassificeres med de korrekte labels og gemmes som et datasæt. Herefter kan man designe en ML algoritme på ens Particle Argon, dette kunne f.eks. være en neural model der er sammesat af forskellige lag bedst egnet til at håndtere disse data. Det er også muligt at vælge en klassisk model som f.eks. RandomForest som fungerer som en regressionsmodel.

Når der er valgt en passende algoritme, så vil denne kunne modtage datasættet og trænes til at genkende sammenhængen mellem tidspunktet på dagen, ens geografiske placering også positionen for ISS rumstationen. Dette kunne øge brugbarheden af IoT-systemet ved at være mindre afhængig af API-data for at kunne fastslå placeringen af ISS på himmelen. Såfremt der er manglende internetforbindelse, vil det alene være resultatet af ML modellen som bruges til at angive retningen for ISS.