

Design og udvikling af præcis autonom magnetfeltsmåler

Bachelorprojekt

Ananda Narasipur Holck Nielsen
Vejleder José M G Merayo
Juni 2017

s144054 ANANDA N. HOLCK NIELSEN

VEJLEDER JOSÉ M G MERAYO

JUNI 2017

DANMARKS TEKNISKE UNIVERSITET

Resumé

This report will describe the process of designing the first prototype of a hand held wireless instrument for measuring magnetic fields with respect to the Earth's gravitation. An analysis of the problem will lead into the detailed outline of the used components and their importance to the overall project as well as the work done to configure these components, including the software written for different useful components.

The later part of the report will include the process of calibrating the sensors and also examples of measurements done with the prototype with an analysis of the measured fields as well of an analysis of the precision of the sensors. The product reached in this project is a working 'flat-sensor' that is few steps away from a first prototype. The final part of the report will discuss the current limitations to the product and the steps needed to reach the first prototype. Lastly design ideas and improvement points for the final product will be addressed briefly.

Indholdsfortegnelse

I.	Introduktion	7
	A. Krav	7
II.	Problemanalyse	9
	A. Måleenheden	9
	B. Modtagerenheden	11
III.	Teori	12
	A. Magnetfelter og magnetometre	12
	B. Magnetometre	14
	C. Accelerometre	15
	D. Orientation	17
IV.	Komponenter	20
	A. Magnetometer: LIS3MDL	20
	B. Accelerometer: MMA8451	22
	C. Trådløs transceiver: SPIRIT1	23
	D. Mikrocontroller: Teensy-LC	27
	E. Solpanel	28
	F. Batteri	29
	G. Strømforsyning/-styring	30
	H. Skærm: SSD1306	30
V.	Komponentsammensætning	32
VI.	Kalibrering	34
	A. Testopstillingen	34
	B. Accelerometer kalibrering	36
	C. Magnetometer kalibrering	37

D. Yderligere kalibreringer	38
VII. Målinger	39
A. Feltvisualiseringer	39
B. Magnetometer sensitivitet	44
C. Orienteringspræcision	46
VIII. Endeligt produkt og fremtidigt arbejde	48
A. Produktdesign	48
B. Systemopgraderinger	50
IX. Konklusion	52
X. Appendix	53
A. Kalibreringspositioner	53
B. Magnetiske objekter til målingerne	54
C. Opstilling til magnetfeltsmåling	55
D. Bx og By konturer fra målinger	57
E. Tabeller for undersøgte komponenter	62
XI. Kode	71
A. C++ kode	71
B. Matlab kode	107

Figurer

1	Design blokdiagram for sender-/måleenheden. Sorte og røde linjer repræsenterer strømforsyningslinjer og blå linjer repræsenterer data linjer. De stiplede blå linjer skal vise alternative data linjer hvis instrumentet skulle bruges mens forbundet til en computer	10
2	Design diagram for modtagerenheden. Sorte og røde linjer repræsenterer forsyningslinjer og blå linjer repræsenterer data linjer	11
3	Visualisering af stangmagnetens B-felt. Billede fra [22] side 889	12
4	B-feltet omkring en ledning med strøm i	13
5	Magnetfelt(sort) som følge af en ledningsring med strøm i(rød)	13
6	Visualisering af Hall effekt. En elektron påvirkes af kraften F_B i feltet B mens den bevæger sig med hastighed v . Ladningsophobningen skaber feltet E som modvirker kraften F_B med F_B . Dette giver 'Hall-spændingen'	14
7	Gennemskæring af kapacitivt accelerometer	17
8	Roll og pitch definitioner og deres tilhørende vinkler	18
9	Accelerometer og magnetometer	21
10	SPSGRF 868MHz Spirit1 trådløs tranceiver og breakout board	23
11	Tilstandsdiagrammet for SPIRIT1 chippen. Figur fra SPIRIT1 datablad[14]	24
12	PCB design til montering af SPSGRF på et fumlebræt	25
13	Råprintet efter ætsning(til venstre) og udskåret print klar til lodning	25
14	Teensy LC μ -controller	27
15	550 mW solpanel	28
16	150 mAh LiPo batteri	29
17	Seed LiPo Pro solcelle og batteri strømstyring	30
18	Adafruit breakout af SSD1306 126 x 96 pixel skærm	31

19	Måle/sende modulet på fumlebræt med strømstyring, batteri og solpanel	32
20	Modtager komponenterne monteret på fumlebræt	33
21	Testopstillingen til kalibrering af sensorer	35
22	Nærbillede af toppladen som fumlebrættet placeres på samt vinklerne for plastik- klodserne	35
23	Magnetfeltet målt for en lille skruetrækker. Pilene viser magnetfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akser- ne viser placeringen af magnetometeret på målebordet	40
24	Magnetfeltet målt for en firkantet køleskabsmagnet på 5x10 cm. Pilene viser mag- netfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magnetometeret på målebordet	41
25	Magnetfeltet målt for en jernvinkel. Placeringen af jernvinklen kan ses af det ly- serøde outline. Pilene viser magnetfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magne- tometeret på målebordet	42
26	H-formet metalstang på undersiden af jernvinklen	43
27	Magnetfeltet målt for en lille rund knapmagnet med en diameter på 1 cm placeret fladt under bordpladen. Pilene viser magnetfeltets vektorfelt i XY-planen. Kontur- kortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magnetometeret på målebordet	43
28	Magnetfeltet målt for en lille rund knapmagnet med en diameter på 1 cm placeret på højkant under bordpladen. Pilene viser magnetfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser pla- ceringen af magnetometeret på målebordet	44
29	2000 målinger med et stationært instrument. Taget indenfor. Spredning over disse 100 sekunder er i den lave ende af det erfarede	45
30	9500 målinger med stationært instrument. Taget udenfor. Der var over 5 meter til nærmeste bygninger hele vejen rundt	46
31	10000 målinger med stationært instrument. Taget indendørs	47
32	Design til senderen. Magnetometer og accelerometer er placeret i det lille rum for enden af armen og resten af elektronikken er placeret i det store rum. Armen bruges både til at give afstand og til at løbe ledninger igennem	49

33	Design til modtagerenheden. En boks der indeholder elektronikken med stærm og knap på toppen. Et USB stik kan sættes i siden, ind til Teensy'en	50
34	De tre positioner der skulle bruges til kalibrering af sensorerne	53
35	Objekterne hvis magnetfelter blev målt	54
36	Jernvinklen tapet til bundet af bordet	55
37	Måleopstilling med måler og sender forbundet til computeren	56
38	Konturplot for Bx fra den lille skruetrækker	57
39	Konturplot for By fra den lille skruetrækker	57
40	Konturplot for Bx for jernvinklen	58
41	Konturplot for By for jernvinklen	58
42	Konturplot for Bx for køleskabsmagneten	59
43	Konturplot for Bx for køleskabsmagneten	59
44	Konturplot for Bx for den lille magnet placeret fladt under bordet	60
45	Konturplot for By for den lille magnet placeret fladt under bordet	60
46	Konturplot for Bx for den lille magnet placeret på højkant under bordet	61
47	Konturplot for By for den lille magnet placeret på højkant under bordet	61

Tabeller

1	Datapakken der blev brugt og størrelsen af hvert felt. I alt 32 bytes og 5 bit	27
2	Målinger fra accelerometer til kalibrering. Gennemsnit af 1000 målinger over 10 sek	36
3	Målinger fra magnetometer til kalibrering. Gennemsnit af ca 1000 målinger over 10 sek. Da der kun kigges på Z-komposanten af Jordens magnetfelt er der kun opgivet 1 forventet værdi	37
4	Tabel over undersøgte μ -controller. Den grønmarkerede er den valgte	62
5	Tabel over undersøgte accelerometre. Det valgt er markeret med grøn	63
6	Tabel over de undersøgte magnetometre til instrumentet. Den grønne linje viser det valgte magnetometer	64

I. Introduktion

Når der skal foretages målinger i felten og uden at bruge for meget tid til opsætning af målestationer er det nødvendigt med håndholdte og pålidelige måleapparater. Et håndholdt apparat skal være trådløst og gerne selvforsynende så det ikke er instrumentet der er den begrænsende faktor. Sådanne apparater ville også give mulighed for automatiske målinger uden brug for menneskelig kontakt over lange perioder. Til dette er det oplagt at bruge Solen til at forsyne instrumentet og dertil have et batteri til nætter og regnvejrskdage.

Ved måling af magnetfelter skal der også tages højde for, at magnetfelter er vektorfelter. Det er derfor ikke uvæsentligt hvordan et instrument er orienteret gennem ens målingssæt. Man skal derfor også kunne kompensere for hvordan instrumentet er orienteret. Der bliver derfor nød til både at måles magnetfelterne og orientering, f.eks. i forhold til tyngdekraften.

Hvis instrumentet skal kunne være autonomt og kunne måle felter, hvor der f.eks. ikke er mennesker skal de indsamlede data kunne sendes til noget dataindsamling så det kan analyseres og bruges. En form for trådløs kommunikation skal derfor etableres. Det er derfor også nødvendigt at have en modtager der kan videregive informationen til en computer. En lille skærm så man kan se felternes størrelse i realtid på modtageren ville også være oplagt.

Projektet skal munde ud i en testet og fungerende prototype af systemet - måler, sender og modtager dog uden størrelses optimering og PCB design.

A. Krav

- Bestemme orienteringen af sensorerne med en præcision på $< 1^\circ$ i forhold til tyngdekraften
- Bestemme magnetfeltet med en præcision på $< 0.1 \mu\text{T}$
- Bestemme magnetfeltets 3 dimensionselle vektor
- Bestemme magnetfelter op til 1 mT i styrke
- Have et lavt nok energiforbrug til flere timers brug uden behov for opladning
- Trådløs kommunikation inden for 10m
- Mulighed for opladning fra Solen

- Udvikling af 'flat-sensor'
- Short-range trådløs modtager til at se data i realtid
- Mulighed for logging af data på en computer

II. Problemanalyse

A. Måleenheden

For at løse dette problem skal der laves to enheder - senderen og modtageren. Hvis der først kigges på senderen, skal det undersøges hvilke komponenter der er nødvendige. Da vi er interesserede i at måle magnetfelter skal der selvfølgelig et magnetometer til. Dette skal være meget lille og have et lavt strømforbrug, men samtidig have nok præcision til at overholde de $0.1 \mu\text{T}$. Da magnetfeltets 3 komponenter skal bestemmes skal det være 3-akset, eller der skal bruges 3 1-aksede magnetometre der er placeret ortogonalt i forhold til hinanden.

Da det er nødvendigt at bestemme orienteringen af instrumentet skal der bruges et 3-akset accelerometer til at måle hvilke accelerationer, i hver retning, instrumentet er udsat for. Dette kan bruges til at udregne instrumentets hældninger da det ikke skal bevæge sig og dermed kun er udsat for tyngdeaccelerationen. Der er som sagt altså brug for et 3-akset accelerometer der kan måle accelerationer på $\pm 1g$. Dette skal selvfølgelig også være så småt som muligt, samtidig med at have et så lavt strømforbrug som muligt. Igen kan 3 accelerometre placeret ortogonalt i forhold til hinanden også bruges, i stedet for ét der måler i 3 akser.

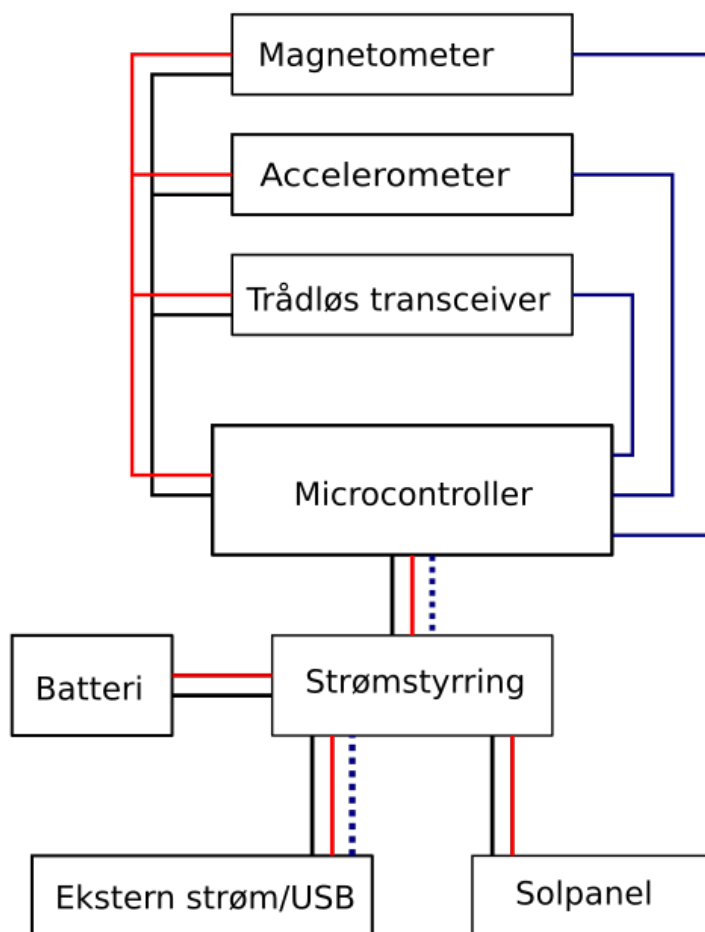
På grund af behovet for trådløs kommunikation skal der også bruges et trådløst modul. Som minimum skal det bare være i stand til at sende data, dog ville det være praktisk fra starten at bruge et modul der også kan modtage data, altså en trådløs transceiver, da det ville være oplagt at kunne ændre indstillinger for modulet uden at skulle have fysisk fat i det. Da der bare er behov for kommunikation mellem senderen og modtageren er der ikke behov for kommunikationsprotokoller som WiFi og Bluetooth, og en simpel RF-transceiver burde være helt tilstrækkelig. Da trådløs kommunikation kan tage meget energi er det vigtigt at finde et energieffektivt modul.

Instrumentet har behov for at kunne holde styr på flere sensorer og moduler, og har derfor brug for en μ -controller til dette. Denne skal gerne være lille, men stadig have udregningskraft nok til at kunne regne med floats og understøtte flere typer seriel-kommunikation til at kommunikere med sensorer.

Alle disse ting vil selvfølgelig ikke kunne virke uden strøm og til dette skal bruges et batteri. Da det ønskes at gøre instrumentet så lille som muligt er et lille lithiumbatteri optimalt da en celle giver en spænding på ca 4 V og derfor vil kunne forsyne μ -controlleren som så kan forsyne resten af systemet. Strømmen til batteriet skal skaffes fra et solpanel, samt mulighed for at kunne oplade fra f.eks. en computer, så man ikke er helt afhængig af Solen. Det er derfor

nødvendigt med et strømforsyningskredsløb der kan stå for at regulere spændinger og oplade batteriet korrekt.

Det sidste som instrumentet mangler er sit solpanel. Solpanelet skal helst bare have samme størrelse som arealet af elektronik og hylster, for ikke at gøre instrumentet større. Derudover ønskes solpanelet selvfølgelig at være så effektivt som muligt. Designdiagrammet for de nødvendige dele og hvordan disse kan spille sammen ses på figur 1 nedenfor.

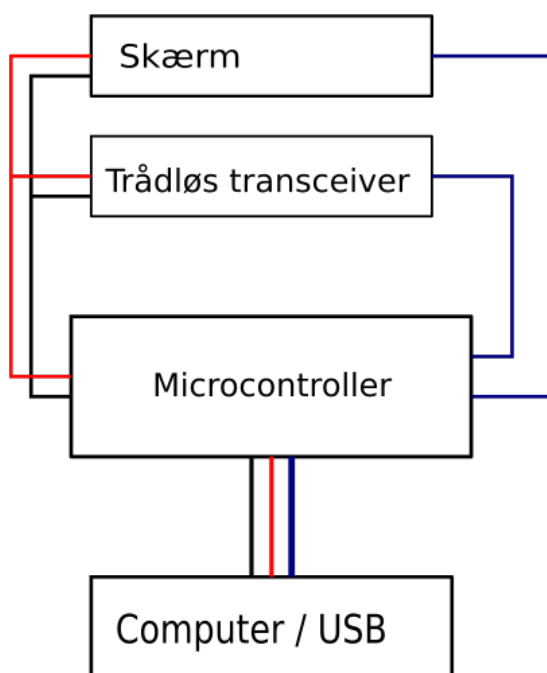


Figur 1: Design blokdiagram for sender-/måleenheden. Sorte og røde linjer repræsenterer strømforsyningslinjer og blå linjer repræsenterer data linjer. De stiplede blå linjer skal vise alternative data linjer hvis instrumentet skulle bruges mens forbundet til en computer

B. Modtagerenheden

For at gøre modtageren så simpel som mulig vil den så vidt som muligt blive designet med de samme komponenter som selve måleinstrumentet. Den har brug for en μ -controller til at styre det hele. Det er denne som skal snakke med den trådløse transceiver som modtager data fra måleenheden. Dertil kommer en skærm som den skal udskrive de modtagne data i realtid. Denne skærm er ikke nogen stor skærm, men blot en lille en der kan vise magnetfeltets 3 komponenter, den totale styrke og måleenhedens orientering.

Til sidst skal μ -controlleren også sende alle de modtagne data videre til computeren så man kan gemme dem og behandle dem som det ønskes. Design digrammet for modtageren kan ses på figur 2 nedenfor.

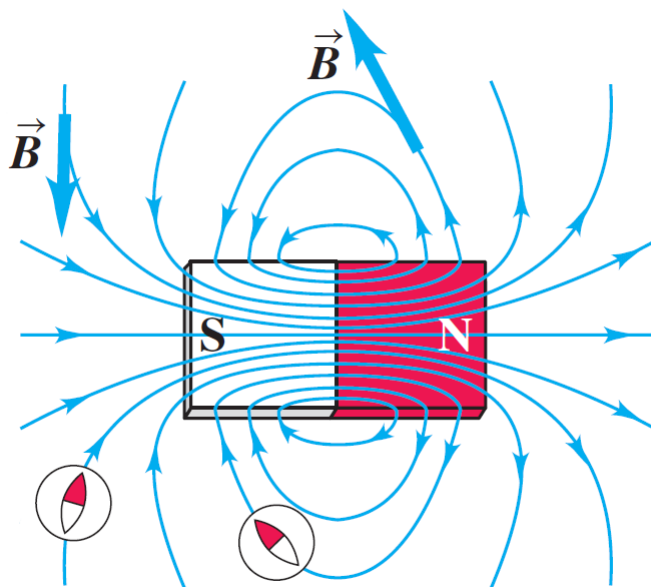


Figur 2: Design diagram for modtagerenheden. Sorte og røde linjer repræsenterer forsyningslinjer og blå linjer repræsenterer data linjer

III. Teori

A. Magnetfelter og magnetometre

Magnetfelter er felter skabt af magnetiske materialer eller strømmen i f.eks. en ledning. Et magnetfelt er et vektorfelt og kan beskrives ved dets $\vec{B}$ -felt som siger noget om styrken af magnetfelt samt dets retning. Styrken siger noget om den kraft, som feltet vil påvirke en bevægende ladet partikle med. Dette $\vec{B}$ -felt, som angives i tesla, kan visualiseres som linjer der går fra den ene 'ende' af det magnetiske materiale, nordpolen, til den anden 'ende', også kaldet sydpolen. En enlig pol kan ikke eksistere og man vil altid se magnetiske ting som dipoler. En sådan dipol kunne være en almindelige stangmagnet. Her vil feltlinjerne gå fra nordpolen til sydpolen uden for magneten og fra syd til nord inde i magneten som det kan ses på figur 3. Som det også kan ses på figuren er $\vec{B}$ -feltet et vektorfelt og vil derfor have bestemte retninger afhængig af hvor man ser på det. Det er derfor vigtigt at også kende disse retninger for at kunne sige noget om et magnetfelt.

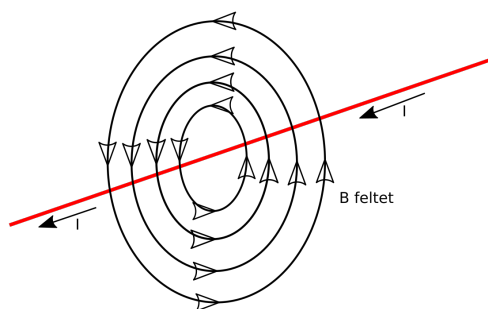


Figur 3: Visualisering af stangmagnetens $\vec{B}$ -felt. Billede fra [22] side 889

Da der kun kan eksistere magnetiske dipoler vil man hvis man knækkede magneten fra figur 3 over blot ende med to magnet med en nord og en sydpol.

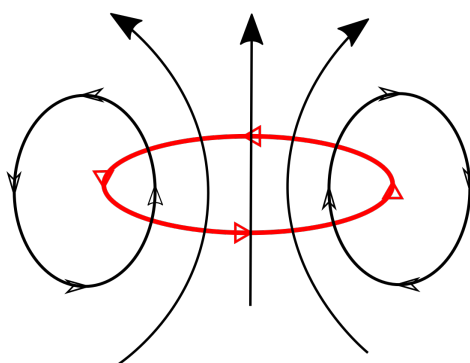
Som nævnt skabes der også et magnetisk felt omkring en ledning med strøm i. I dette tilfælde

vil B-feltet være rundt om ledningen, afhængig af strømmens retning som det ses på figur 4.



Figur 4: B-feltet omkring en ledning med strøm i

Hvis vi forestiller os en lukket kreds med ladningsbevægelse, altså som en sluttet ledning hvor der løber en konstant strøm vil, som følge af feltet omkring en ledning, fås et felt som på figur 5.



Figur 5: Magnetfelt(sort) som følge af en ledningsring med strøm i (rød)

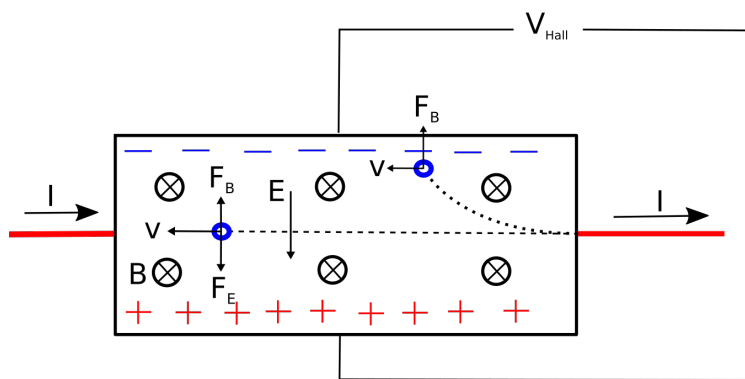
Denne cirkulation af ladning rundt i ring skaber altså et magnetfelt, hvor man kan se toppen af ringens plan som en nordpol og bunden som en sydpol. Denne ring kan altså ses som en magnetisk dipol. Den vektor som peger lige ud af planen kaldes det magnetiske dipolmoment. En elektron der kredser om sin kerne, eller bare elektronens spin kan, da elektroner er ladede partikler, ses som magnetiske dipoler med et magnetisk dipolmoment. I de fleste materialer vil alle disse dipolmomenter dog pege i tilfældige retninger og materialet vil derfor have et samlet dipolmoment (summen af alle de små dipolmomenter) på 0 og vil derfor ikke være magnetisk udadtil og skabe et magnetisk felt. Nogle materialer kan dog få peget mange af disse dipolmomenter i samme retning og blive magnetiseret når det påvirkes af et eksternt felt og kan

dermed skabe sit eget magnetfelt. I hvor stor en grad dette er muligt er forskelligt fra materiale til materiale. Nogle kan holde fast på denne magnetisering, som f.eks. jern, mens andre bare vil blive magnetiseret som følge af et magnetisk felt og selv begynde at udsende et felt så længe det eksterne felt er til stede. Når dette fjernes vil alle dipolmomenterne igen orientere sig tilfældigt og materialet har ikke længere et magnetfelt.

B. Magnetometre

Magnetiserede materialer eller elektriske strømme vil som beskrevet ovenfor udsende magnetiske felter. Hvis man vil måle dette $\mathbf{B}$ -felt kan man bruge et magnetometer. Magnetometre er sensorer der kan omsætte magnetfelter til værdier vi kan regne på og forstå.

Der findes mange slags magnetometre, men de mest normale der laves på IC niveau er de såkaldte Hall Effect Magnetometre, MEMS (MicroElectroMechanical Systems) baserede magnetometre og Magnetoresistive magnetometre og. De sidste fungerer ved at have et magnetoresistivt materiale som ændrer modstand som følge af det magnetiske felt. Modstanden virker ved at have en plade af det magnetoresistive materiale. Når et magnetfelt er til stede vil feltet påvirke elektronernes vej på pladen. Denne ændring i ruten vil svare til en ændring i modstanden.



Figur 6: Visualisering af Hall effekt. En elektron påvirkes af kraften F_B i feltet B mens den bevæger sig med hastighed v . Ladningsophobningen skaber feltet E som modvirker kraften F_B med F_E . Dette giver 'Hall-spændingen'

Hall effekt magnetometre bruger den såkaldte 'Hall effekt' der er illustreret på figur 6. Hall effekten skyldes at magnetfelter vil påvirke elektronerne i en metalplade der løber strøm igennem. Når magnetfeltet påvirker elektronerne vil de samles på den ene side af pladen således at der er en stor negativ ladning på den ene side og en stor positiv ladning på den anden. Denne ladnings-

forskel, når den bliver stor nok, vil skabe et elektrisk felt der vil modvirke kraftpåvirkningen af magnetfeltet således at den resulterende kraft på elektronerne er nul. Feltet vil resultere i en potentiel forskel der giver 'Hall-spændingen' over pladen som kan måles. Størrelsen af spændingen vil være relateret til magnetfeltet og kan på denne måde bruges til at bestemme magnetfelter. Da denne spænding kan være både positiv og negativ er det også muligt at bruge 'Hall effekt' til at bestemme retning.

De MEMS-baserede sensorer bruges også den kraftpåvirkning som sker på grund af et magnetfelt. Der findes forskellige måde at gøre dette, men princippet bag er at magnetfeltet påvirker en mekanisk struktur. Denne ændring kan så enten omsættes til en spænding ved hjælp af f.eks. strain gauges eller ved hjælp en piezoresistens. Den spænding kan herefter omsættes til magnetfeltværdier. MEMS sensorer kan også fungere ved at man måle ændringen af en strukturs mekaniske resonansfrekvens som følge af magnetfeltspåvirkningen. Dette kan så omregnes til et magnetfelt.

Alle disse tre slags magnetometre kan laves meget små, har lave strømforbrug og kan laves billigt så de er oplagte til dette projekt.

C. Accelerometre

Et accelerometer er et instrument der kan måle den acceleration som det udsættes for. Acceleration er et udtryk for ændringen af hastighed pr. tidsenhed. Det er altså den afledte af hastighed.

$$\vec{a} = \frac{d\vec{v}}{dt} \quad (1)$$

Som det kan ses af ligningen er acceleration en vektor og man skal derfor være opmærksom på en accelerations retning, og et accelerometer skal altså ikke bare kunne finde størrelsen af accelerationen, men også i hvilken retning accelerationerne er.

Alle objekter på Jorden påvirkes af en tyngdeacceleration, g , som gør at alt vil accelerere ind mod Jorden. Dette skyldes gravitationskræften mellem to objekter som det er fra Newtons gravitationslov.

$$F = \frac{G \cdot M_{\oplus}}{r^2} \cdot m \quad (2)$$

Hvor $M_{\oplus}$ er Jordens masse, r er radius af Jorden og G er Newtons gravitationskonstant. Kigges der på Newtons anden lov, ligning 3, der siger at den resulterende kraft er lig massen af et objekt

gange den samlede accelerationen af objektet kan tyngdeaccelerationen defineres som i ligning 4

$$\vec{F} = m \cdot \vec{a} \quad (3)$$

$$g = |\vec{a}| = \frac{G \cdot M_{\oplus}}{r^2} \quad (4)$$

Det er netop denne g 's påvirkning som vi skal bruge accelerometeret til at bestemme.

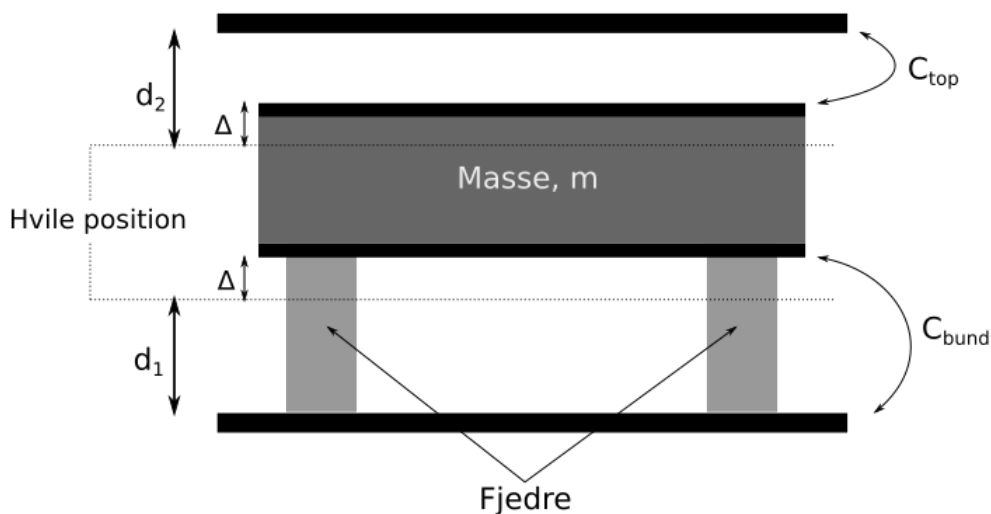
En måde at bestemme en acceleration på er ved at se hvordan en masse rykker sig relativt til en anden masse. Den ene masse vil rykkes på grund af en kraftpåvirkning takket være accelerationen. Ud fra denne afstand kan man således bestemme kraften og dermed accelerationen.

Et apparat til at gøre dette er et kapacitivt accelerometer. Accelerometeret er konstrueret ved at have en plade monteret således at det er stationært i forhold til hele accelerometerets 'hus'. En anden plade er sat på den frie masse med en afstand d_1 mellem dem. Disse to plader vil skabe en kapacitans, C_{bund} . Pladerne vil således kunne bevæge sig længere eller tættere på hinanden som den frie masse påvirkes af en kraft. Denne ændring af afstanden mellem pladerne vil også resultere i en ændring af kapacitansen. Typisk vil man også have en plade på den anden side af den frie masse således at man får en ekstra kapacitor, C_{top} . Disse to kapacitansers ændringer skulle derfor gerne ændres lige meget i hver sin retning. Altså differencen er ens, men med en 180° faseforskydning. Princippet bag et sådan accelerometer kan ses på gennemskæringen på figur 7. Et rigtigt accelerometer vil have 4 fjedre placeret således at bevægelse sker jævnt i den ønskede op og ned retning. Ved at have et kredsløb til at omsætte disse kapacitanser til en spænding kan dette måles og ændringen kan omregnes til en afstandsændring, Δ . Ved at have anbragt den frie masse på nogle siliciumsfjedre kan vi ud fra afstandsændringen og kendskab til fjedrene, altså fjederkonstanten k , få kraften, F_m , fra ligning 5. Denne kan bruges med Newtons anden lov og den frie masses masse, m , til at finde accelerationen.

$$F_m = k \cdot \Delta \quad (5)$$

Afstandene som den frie masse bevæger sig vil typisk ikke overgå $20\mu\text{m}$ og capacitive accelerometer er derfor meget små og konstrueres som IC'er i silicium hvilket også gør dem billige og med lavt strømforbrug.

Udover kapacitiv effekt kan man også bruge piezoresistiv effekt til at bestemme acceleration. Ved at måle hvilken hvordan modstanden ændres som følge af kraftpåvirkningen fra accelerationen



Figur 7: Gennemskæring af kapacitivt accelerometer

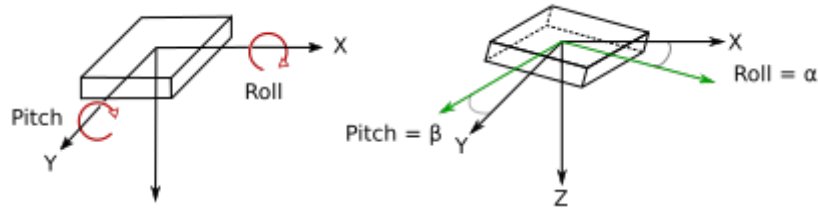
kan dette omsættes til en acceleration. Også piezoelektrisk effekt kan bruges til at måle acceleration, da det piezoelektrisk materiale vil omsætte kraftpåvirkningen direkte til elektricitet som kan måles om omregnes til acceleration.

En sidste undersøgt metode til måling af acceleration er ved hjælp af termiske gas accelerometre. Disse fungerer ved at have en gas som sin frie masse. Gassen opvarmes fra et varmelegeme. Nogle termometre vil måle temperaturen forskellige steder i gassen. Som denne gas påvirkes af acceleration vil temperaturen stige omkring nogle termometre og falder omkring andre. Denne forskel i temperature vil være svarende til en acceleration. Selvom disse accelerometre også kan laves meget små og billige bruger de mere strøm end andre da der skal holdes gang i et varmelegeme.

Alle disse accelerometre er alle typer af MEMS sensorer der bruger en mekanisk bevægelse til at bestemme accelerationen.

D. Orientation

Når accelerationen kan måles på 3 ortogonale akser kan dette bruges til at udregne hvordan instrumentets hældning er. På figur 8 kan ses hvilke vinkler i forhold til tyngdekraften der ønskes fundet. Roll er rotationen omkring x-aksen og pitch er rotationen omkring y-aksen.



Figur 8: Roll og pitch definitioner og deres tilhørende vinkler

For at udregne disse vinkler bruges formlerne fra databladet[4] som kan ses på ligning 6 og 7.

$$\text{Roll} = \alpha = \arctan \left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}} \right) \quad (6)$$

$$\text{Pitch} = \beta = \arctan \left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}} \right) \quad (7)$$

Disse værdier spænder pga. $\arctan()$ mellem -90° og 90° . For at få en fuld rækkevidde på 360° kan bruges fortegnet på a_z til at bestemme om apparatet peger op eller ned.

Orienteringsvinklerne bruges til at få alle magnetfeltsmålingerne til at være i forhold til tyngdekraften, således at det i princippet er lige meget hvordan instrumentet holdes. Det kan altså godt holdes lidt skævt og stadig give de samme målingerne uden at der skal bruges lang tid på at sætte instrumentet ordentligt hver gang det rykkes. Dette gør også at man kan bruge det håndholdt uden problemer.

For at få magnetfeltet udtrykt i forhold til tyngdekraften skal vektorfeltet roteres. Til at rotere feltet, som er opgivet som en vektor skal der bruges rotation. En vektor kan roteres omkring en akse ved hjælp af en rotationsmatrix og de vinkler den er drejet - pitch, β , og roll, α . Ved at rotere i modsat retning af disse to vinkler kan vores vektor blive roteret så den har z-aksen sammenfaldende med tyngdekraften. Det roterede magnetfelt kan altså fås ved to rotationer. En omkring x-aksen og en omkring y-aksen som i ligning 8.

$$B_{rot} = R_y(\beta)^{-1} \cdot R_x(\alpha)^{-1} \cdot B_{original} \quad (8)$$

Hvor R_x og R_y er rotationsmatricerne omkring hhv. x- og y-aksen. I 3 dimensioner ser de to nødvendige matricer ud som i ligning 9 og 10.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad (9)$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \quad (10)$$

Det samlede roterede felt bliver derfor:

$$\begin{bmatrix} B_{x,rot} \\ B_{y,rot} \\ B_{z,rot} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}^{-1} \cdot \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}^{-1} \cdot \begin{bmatrix} B_{x,original} \\ B_{y,original} \\ B_{z,original} \end{bmatrix} \quad (11)$$

Ganges disse matriser sammen får vi den endelige roterede vektor.

$$\begin{bmatrix} B_{x,rot} \\ B_{y,rot} \\ B_{z,rot} \end{bmatrix} = \begin{bmatrix} \cos(\beta)B_{x,original} - \sin(\beta)\sin(\alpha)B_{y,original} - \sin(\beta)\cos(\alpha)B_{z,original} \\ \cos(\alpha)B_{y,original} - \sin(\alpha)B_{z,original} \\ \sin(\beta)B_{x,original} + \cos(\beta)\sin(\alpha)B_{y,original} + \cos(\beta)\cos(\alpha)B_{z,original} \end{bmatrix} \quad (12)$$

IV. Komponenter

A. Magnetometer: LIS3MDL

Det brugte magnetometer til instrumentet er et LIS3MDL fra STMicroelectronics. Det er blevet valgt efter at have undersøgt hvad der var muligt at skaffe inden for små, billige og energieffektive magnetometre. Fokus har ligget på magnetoresistive, Hall effekt og MEMS magnetfelt sensorer. De undersøgte magnetometre fremgår af tabel 6 i appendix E., hvor deres opløsning, hvilke teknologi der er brugt, deres rækkevidde og energioplysninger fremgår. LIS3MDL er valgt da det er et 3 akset magnetometer der har større rækkevidde og bedre opløsning end de andre fundne magnetometre, uden at hverken pris eller energiforbrug er meget højere. På figur 9a kan ses breakoutboardet til LIS3MDL som det er brugt. Det er et 3-akset digitalt elektromekanisk, MEMS, magnetometer. Det kan kommunikere med μ -controlleren ved hjælp af enten I²C eller SPI, som er det der er blevet brugt.

Magnetometeret kan konfigureres til 4 forskellige intervaller af magnetfelter, $\pm 400\mu\text{T}$, $\pm 800\mu\text{T}$, $\pm 1200\mu\text{T}$ eller $\pm 1600\mu\text{T}$. Værdierne gemmes i 16 bit registre hvilket giver sensitivitet på hhv. 15nT, 30nT, 44nT og 60nT per LSB. Den har et meget lavt strømforbrug og med en målerate på 20Hz bruger den mellem $40\mu\text{A}$ og $270\mu\text{A}$. [11]

Det lave strømforbrug sammen med den høje opløsning, samt et stort interval af magnetfeltstyrker førte til valget af dette magnetometer.

Konfiguration af magnetometer

Til at kommunikere med magnetometeret blev det valgt at bruge SPI da dette er hurtigere end I²C. For nemt at kunne bruge magnetometeret skulle der skrives noget kode til μ -controlleren og der er derfor blevet lavet et lille C++-bibliotek til at bruge magnetometeret. I koden kan der laves et LIS3MDL objekt som indeholder alle de nødvendige funktioner der skal bruges for at kunne konfigurere og læse data fra magnetometeret. Biblioteket bruger Arduinos SPI bibliotek¹ til at styre SPI kommunikationen. Magnetometerbiblioteket har funktioner til at initialisere magnetometeret til nogle forudbestemte indstillinger, læse registre, tjekke om der er nye målinger klar, hente magnetfeltsdata og omregne til fysiske størrelser, ændre datahastigheden, ændre måle størrelserne og præcisions niveauet. Objektet vil også holde styr på indstillingerne så omregningerne til de fysiske størrelser passer.

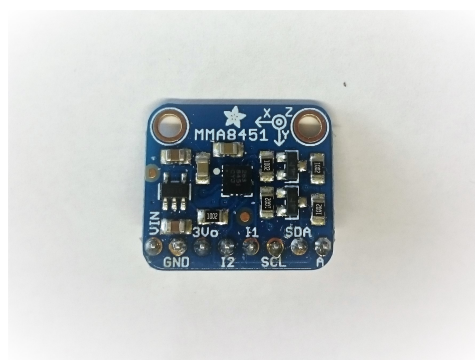
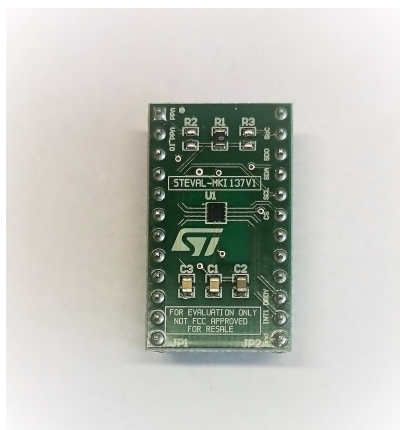
¹<https://www.arduino.cc/en/Reference/SPI>

Når magnetometeret initialiseres indstilles det til følgende indstillinger:

- Magnetfelt størrelse: $\pm 400\mu\text{T}$
- XYZ-akser sat til at operere 'Ultra-High-Performance-Mode'
- Måle hvert 50 ms (20Hz)
- Den indbyggede temperatursensor er slået fra
- Dataregistre opdateres ikke før de er blevet læst
- Flere andre ikke anvendte funktioner er slået fra

Magnetometerets operationstilstand (punkt 2) er det der afgøre støjen på målingerne. I 'Ultra-High-Performance', der er den bedste, er der en RMS støj på $0.31\mu\text{T}$ for X og Y akser og en støj på $0.42\mu\text{T}$ for Z-aksen. Disse værdier for standard afvigelsen ved præcisionsmålingerne blev også opnået. Hvad værdierne er for de andre indstillinger er desværre ikke opgivet i databladet og er ikke blevet testet da der ønskes så høj præcision som overhoved muligt.

Al kalibrering af magnetometeret foregår på μ -controller niveau, da magnetometeret kun tilbyder offset justering og ikke med høj nok præcision. Magnetometeret klare sig selv med de givne indstillinger, men sender et interrupt når data er klar til at blive læst. Ellers venter det bare på instruktioner fra μ -controlleren.



(a) ST LIS3MDL magnetometer breakout (b) Adafruit MMA8451 accelerometer breakout

Figur 9: Accelerometer og magnetometer

B. Accelerometer: MMA8451

For at kunne holde styr på instrumentets orientering bruges et MMA8451 accelerometer der kan ses på figur 9b. Accelerometeret er valgt efter at have undersøgt hvad der var tilgængeligt af kapacitive, MEMS, termiske og piezoresistive accelerometre. De andre undersøgte accelerometre kan ses i appendix E. på tabel 5. Det er blevet valgt da det både har en god opløsning og et lavere strømforbrug end de andre. Dertil kommer at MMA8451 findes som et breakoutboard fra Adafruit som tilbyder et godt software bibliotek til brug af accelerometeret.

Det er et kapacitivt accelerometer der kan måle alle 3-akser med en opløsning på 14-bit. Accelerometeret bruger I²C serial kommunikation til at kommunikere med μ -controlleren. Det kan konfigureres til at måle i intervallerne $\pm 2g$, $\pm 4g$ og $\pm 8g$, hvilket giver en bitopløsning på hhv. 0.244mg, 0.488mg og 0.977mg pr LSB. Da accelerometeret skal bruges til at bestemme orientering og dermed kun skal kunne måle tyngdekraftens påvirkning af de 3 komponenter er der kun brug for at kunne måle kræfter på $\pm 1g$.

MMA8451 har et maksimalt strømforbrug på $165\mu A$ med en datarate på 800Hz. Dog kan magnetometeret maksimalt måle med 80Hz, hvilket for accelerometeret giver et strømforbrug på $44\mu A$. [12]

Konfiguration af accelerometeret

AdaFruit tilbyder et godt bibliotek til Arduino som kan bruges til Teensy-LC μ -controlleren således at accelerometeret er klar til brug. [18] Som for magnetometeret er det egentlig bare 2 C++-biblioteker der bruger Arduinos egne biblioteker til I²C kommunikation².

Med biblioteket laves der et MMA8451 objekt som har alle funktionerne til at initialisere, læse fra accelerometeret, skrive til det og få informationer om den nuværende konfiguration. Når accelerometeret initialiseres sættes det til følgende indstillinger:

- Det samler målinger med 12.5Hz
- Der måles accelerationer på $\pm 2g$
- 'Low Noise' funktionen er aktiveret, som mindsker støj med ca. 20%
- 'High resolution mode' er aktiveret for at få 14-bit data og dermed højeste opløsning

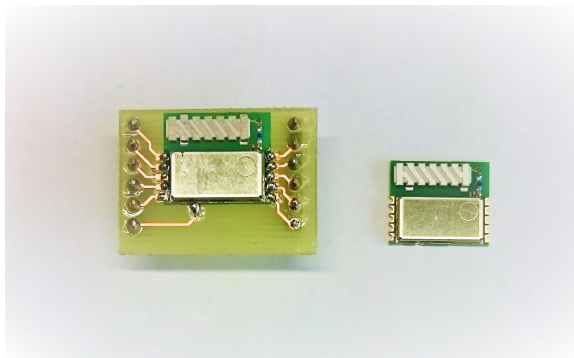
²<https://www.arduino.cc/en/Reference/Wire>

- Accelerometeret sender et interrupt på sin I1-pin når ny data er klar
- Interruptet er sat til 'Active-High'
- Et orienteringsregister der gemmer accelerometerets orientering aktiveres
- Alle andre funktioner er slået fra

Da accelerometeret er sat til 'High Resolution' og 'Low Noise' bruger det selvfølgelig ekstra strøm og når faktisk op på sit maksimale $165\mu\text{A}$ ved alle frekvenser da den bruger oversampling til for at opnå reduceret støj og højere opløsning. Da der ønskes så høj præcision som muligt er dette dog nødvendigt. Accelerometerets data bruges som sagt til at udregne instrumentets orientering hver gang der er nye målinger klar.

C. Trådløs transceiver: SPIRIT1

Da man ikke vil have en ledning mellem sin målestation og selve måleinstrumentet er det vigtigt at kunne kommunikere trådløst. Til dette bruges et trådløst modul fra STMicroelectronics. Det brugte modul er en trådløs transceiver, dvs. at den både kan bruges som sender og modtager. Transceiveren er et SPSGRF-modul med en antenne konfigureret til 868MHz, der bruger SPIRIT1 chippen. På figur 10 kan ses modulet sammen med det nødvendige print til montering af modulet på et fumlebræt.[14][16]

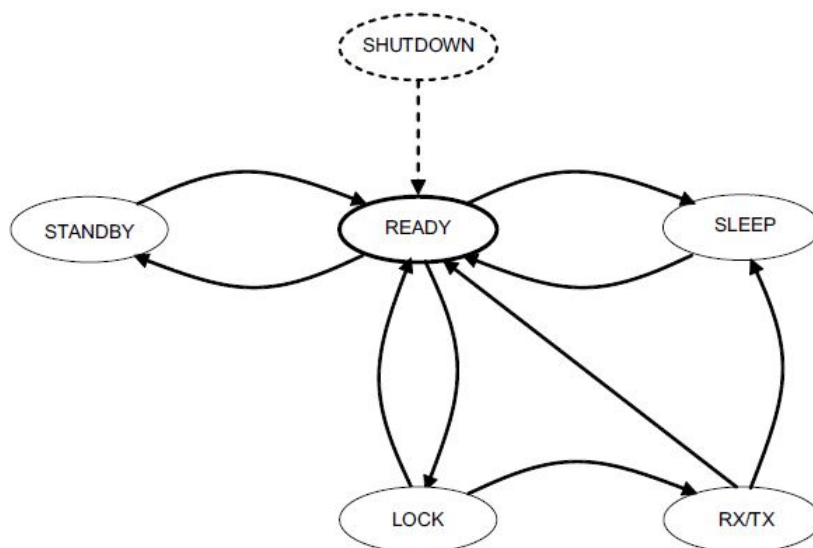


Figur 10: SPSGRF 868MHz Spirit1 trådløs transceiver og breakout board

SPSGRF har mange konfigurationsregistre der kan bruges til at vælge hvordan data skal sendes, hvilken modulation der skal bruges, hvilken effekt der skal sendes med og mange andre ting. Der kommunikeres med μ -controlleren ved hjælp af SPI.

Modulet kan sende data med op til 500kbps med en maksimal effekt på 12dBm. Ved maksimal effekt bruger modulet 23mA, dog tager en sending ikke meget tid og i sin 'READY'-tilstand bruger den kun 400 μ A således at strømforbruget ikke bliver alt for højt.

SPIRIT1 chippen opererer i tilstande og kan være i forskellige tilstande afhængig af hvad den skal foretage sig. Tilstandsdiagrammet og de mulige tilstandsovergange fremgår af figur 11. Skift mellem tilstandene sker ved såkaldte kommandoer (Command strobes) som sendes over SPI. F.eks. kan data skrives til chippen i sin 'READY'-tilstand og ved at sende en 'TX'-kommando vil chippen automatisk pakke denne data som man har konfigureret det og sende den af sted på den ønskede måde. Herefter vil den returnere til sin 'READY'-tilstand og være klar igen.

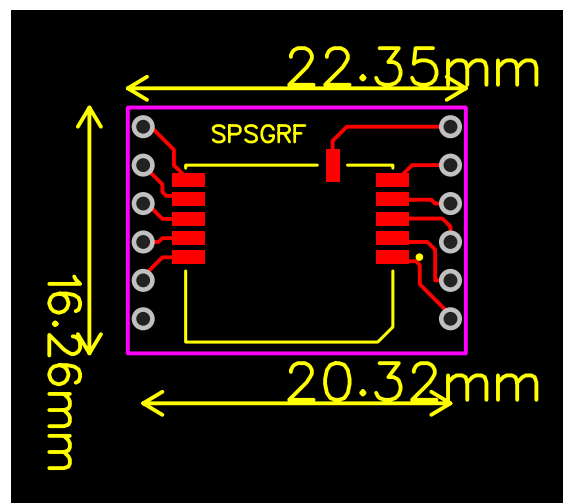


Figur 11: Tilstandsdiagrammet for SPIRIT1 chippen. Figur fra SPIRIT1 datablad[14]

At holde styr på hvilke tilstand den er i er vigtigt og i begyndelsen af arbejdet med chippen voldte dette store problemer da chippen hele tiden endte i en "ukendt" tilstand når data skulle sendes. Den var ukendt da den ikke fremgik af databladets tilstandsoversigt. Det viste sig at det i stedet var en fejlmeddelelse som blev gemt i tilstandsregistreret. Fejlen opstod på grund af fejlkonfigureret. Da det endelig blev fundet i databladet hvad denne "tilstand" betød kunne arbejdet med chippen fortsætte og den kunne blive konfigureret korrekt.

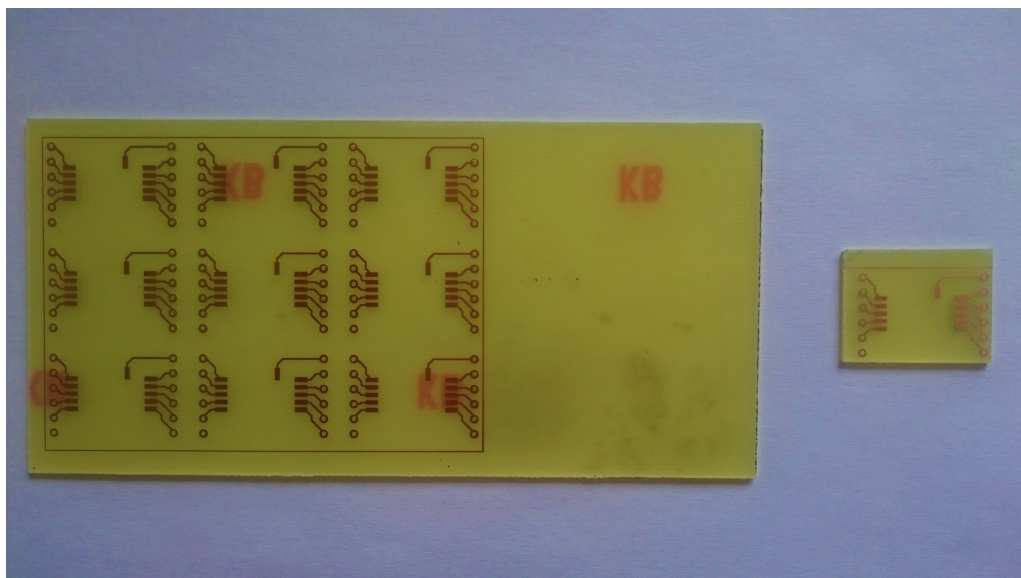
Breakout board til SPSGRF

Som sagt var det nødvendigt at lave et 'breakout board' til den trådløse transceiver, da SPSGRF er beregnet til overflade montering. Først blev der lavet et PCB design som kan ses på figur 12.



Figur 12: PCB design til montering af SPSGRF på et fumlebræt

Herefter printes PCB'et på et transparentt papir så alle kobber-linjerne er sorte. Det transparente papir lægges oven på råprintet, en plade af glasfiber beklædt med et kobberlag på den ene side, og det anbringes i en maskine der belyser printet med UV-lys. Med kredsløbet overført til råprintet kan det overskydende kobber ætzes væk i et syrebade og hullerne til ben kan bores. Det endelige produkt kan ses på figur 13. Pladen kan nu skæres i passende stykker, hvor SPSGRF'en kan loddess på. Herefter kan der også loddess ben på printet så det kan monteres på fumlebrættet.



Figur 13: Råprintet efter ætsning (til venstre) og udskåret print klar til lodning

Konfiguration af SPIRIT1

Som for magnetometeret er der skrevet et C++-bibliotek til håndtering af transceiveren. Biblioteket gør det muligt at sende data til transceiveren, læse data, ændre dens tilstand og skrive til konfigurations registrene. Der er også funktioner til at hente al modtaget data.

Som sagt var der problemer med konfigurering af transceiveren da den indeholder flere hundrede konfigureringsmuligheder og uden dybere kendskab til trådløs kommunikation svært at vide hvad der var vigtigt. Heldigvis tilbyder STMicroelectronics et program, SPIRIT1 DK[15], der kan hjælpe en med at finde de nødvendige konfigurationer. De endelige konfigurationer var:

- Bruger 2GFSK modulation
- Basisfrekvensen er 867.999967 MHz
- Sender med ca 250 kbps eller 31kBps
- Bruger den simple 'Basic Package' til at pakke data
- Sender med maksimal kraft: 12dBm
- Bruger ikke noget acknowledge mellem sender og modtager
- Tjekker ikke om sender og modtager adresser passer
- Pakkerne skal indeholde 24 bytes data

Når data var klar blev det sendt til det trådløse modul som så pakkede det og sendte det med de valgte indstillinger. Pakken så ud som i tabel 1 der også angiver længden af hvert felt. Feltet 'Preamble' er en lang serie af 1010 1010... som kan bruges til at identificere at en pakke er på vej. 'Sync'-feltet bruges til at synkronisere elektronikken til at læse pakken korrekt. I de brugte pakker består 'Sync' af 4 bytes af '1000 1000'. Herefter kommer størrelsen af 'Data' delen af pakken. Efter dette kommer der 24 bytes data som er udgjort af accelerometer værdier i XYZ-retning og magnetometer data i XYZ-retning. Til sidst kommer en byte med en CRC værdi der bruges til at tjekke om de modtagne værdier stemmer med det afsendte. Hvis det ikke gør smides pakken væk ellers ligges den i chippens RX-FIFO der gemmer modtaget data hvorefter det kan læses af μ -controlleren.

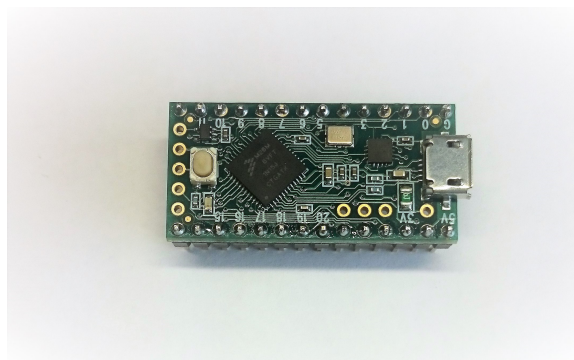
Tabel 1: Datapakken der blev brugt og størrelsen af hvert felt. I alt 32 bytes og 5 bit

3 bytes	4 byte	5 bit	24 bytes	1 byte
Preamble	Sync	Length	Data	CRC

D. Mikrocontroller: Teensy-LC

μ -controlleren der skal holde styr på alle sensorer og læse og sende data er en Teensy LC der kan ses på figur 14. Inden TeensyLC blev valgt er andre μ -controllere blevet undersøgt. Disse kan ses i tabel 4 i appendix E.. TeensyLC'en er blevet valgt da den har en god balance mellem regnekraft, hukommelse, porte og energiforbrug. Dens overmand Teensy3.2 blev hurtigt fravalgt da denne havde meget mere kraft end hvad var nødvendigt. LC'en havde dog det mere end de andre samt, at den byggede på Arduino hvilket jeg også har erfaring med så programmering og at komme i gang ikke ville blive en flaskehals.

Den bruger en ARM Cortex M0+ processor med en clockhastighed på 48MHz. Den bygger på 32-bit arkitektur. Dens indbyggede hukommelse består af en 8kb RAM og 62kb FLASH. Derudover understøtter den de vigtige serielle forbindelser som SPI, I²C og USB, samt har en masse analoge og digitale indgange og udgange.[17]



Figur 14: Teensy LC μ -controller

Da M0+ processoren er designet til lavt energiforbrug i små applikationer er den ideal til dette instrument. Da programmet som skal køres ikke er et stort program er der heller ikke brug for meget hukommelse der bare bruger unødvendig strøm. Dette sammen med dens mange porte og lille størrelse er grunden til den er valgt til projektet.

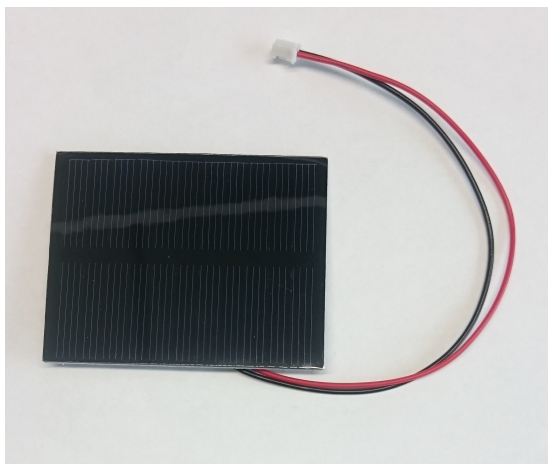
Teensy kan programmeres ved hjælp af Arduinos compiler og bruge alle Arduinos biblioteker hvilket gør den hurtig at komme i gang med og nem at finde support til.

Teensy'en er sat op til at bruge serial(UART) kommunikation til debugging og udskrive data til computeren. Derudover er SPI og I²C hardwaren initialiseret til kommunikation med sensorer. Også tre digitale porte er konfigureret til reagere på interrupts fra sensorer. Den er programmeret til at vente på målingerne fra accelerometer og magnetometer. Når begge målinger er klar sendes de trådløst og den går tilbage til at vente.

Modtager enheden står og venter på at den har modtaget data, hvorefter den vil lave de nødvendige udregninger og udskrive realtid data til skærmen og sende data til computeren over UART.

E. Solpanel

Solpanelet brugt på magnetmåleren er et lille monokrystallisk siliciums solpanel på 55 mm x 70 mm eller 38.5 cm². Solcellen kan afgive en effekt på ca. 550 mW under optimale forhold og med en effektivitet på 17%.^[1]



Figur 15: 550 mW solpanel

For at teste om det var muligt at opnå disse tal blev der foretaget en målinger i fuld sollys omkring kl. 11.00 i slutningen af maj således at solen var højt på himmelen. I Danmark giver solen ca. 1 kW/m²³. Det betyder at det maksimale udbytte der kan opnås af en solcelle af denne størrelse er

$$P_{max} = 1000 \text{ W/m}^2 \cdot 38.5 \text{ cm}^2 = 3.85 \text{ W} \quad (13)$$

Ved målingerne blev der opnået en åbenkredsspænding, V_{OC} , på 6.6 V og en kortslutningsstrøm,

³<http://www.solcelleinfo.dk/t/soleffekt/>

I_{SC} , på 81.5 mA. Dette giver en effekt og dermed effektivitet på

$$P_{solcell} = V_{OC} \cdot I_{SC} = 6.6 \text{ V} \cdot 81.5 \text{ mA} = 537.9 \text{ mW} \quad (14)$$

$$\mu = \frac{P_{solcell}}{P_{max}} = \frac{537.9 \text{ mW}}{3.85 \text{ W}} \cdot 100\% = 13.97\% \quad (15)$$

En simpel måling viser altså at det er muligt at komme tæt på de opgivne tal og var målingen foretaget i højsommer og når solen var højest på himlen kunne man komme tættere på de 17%. Solpanelet er valgt fordi størrelsen er passende og den giver nok energi til holde gang i instrumentet.

F. Batteri

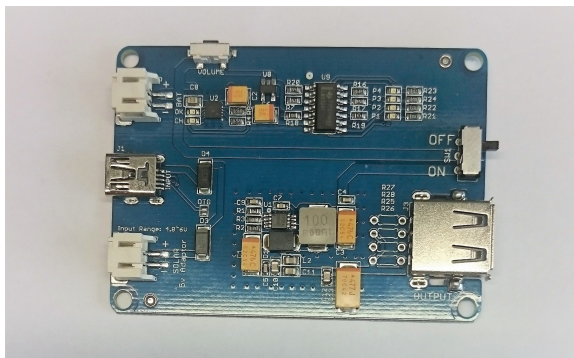
Det brugte batteri er et en celled litium-ion polymer batteri, LiPo-batteri, på 150 mAh og en spænding på 3.7 V. Batteriet tåler en opladning på 3C, dvs. at det kan oplades med $150 \text{ mA} \cdot 3 = 450 \text{ mA}$. Batteriet er valgt da LiPo-batterier har en meget høj energitæthed og derfor kan være meget små. Det valgte batteriet er et hobbybatteri til model RC fartøjer og vejer kun 15 g, og fylder lige godt 3 cm^2 , og kan ses på figur 16. Trods sin lille størrelse kan det afgive en meget stor mængde strøm - hele 45C, eller 6.75 A. [13]



Figur 16: 150 mAh LiPo batteri

Da instrumentet ikke bruger meget strøm er der ikke brug for et stort batteri. Derfor er dette lille batteri valgt, da det selv når der ikke er sol vil kunne holde det kørende i et par timer. Da batteriet tillader hurtig opladning i forhold til sin kapacitet kan det også bruges effektivt hvis man skal tage mange målinger inden for kort tid.

Ved en simpel test kunne instrumentet sende i lige godt 3 timer på et fuldt opladt batteri indendørs uden sollys.



Figur 17: Seed LiPo Pro solcelle og batteri strømstyring

G. Strømforsyning/-styring

Til at styre strømforsyning og opladning af batteriet bruges LiPo Pro boardet fra Seedstudios. Det er på størrelse med solpanelet og kan ses på figur 17. Boardet har 4 porte - en til solcellen, en til batteriet, en til USB fra en computer og et USB til at forbinde til μ -controlleren. Boardet klare selv at regulere spændingerne således at batteriet kun bliver opladet med 4.2V og μ -controlleren forsynes med 5V fra USB-stikket. Boardet er også forsynet med en lille knap, der kan bruges til at tænde for op til 4 LED'er for at vise hvor meget strøm der er tilbage[10]. Denne funktion er begrænset i dette instrument, men man kunne 'hacke' boardet således at man kan måle batterispændingen og dermed bruge denne information i selve instrumentet til at styre strømforbruget.

En anden meget brugbar funktion af boardet er den direkte forbindelse mellem de to USB-stikket således at μ -controlleren kan programmeres eller kommunikeres med uden at man skal skille instrumentet ad. Dette er selvfølgelig mest brugbart i starten når der skal debugges og uploades ny kode mens prototypen udvikles.

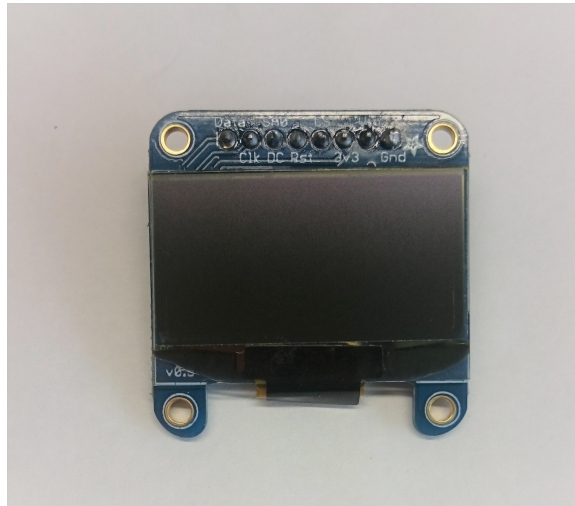
Til opladning af batteriet fra solcellen ville det være meget effektivt hvis boardet brugte MPPT, Maximum Power Point Tracking, så energien fra solcellen overføres med mindst muligt tab, men et sådan board vil skulle udvikles fra bunden, hvilket der ikke var tid til.

H. Skærm: SSD1306

Til at kunne se den modtagne data i realtid bruges en lille skærm på 1.3 tommer. Displayet er et SSD1306 monteret på et board fra AdaFruit så der kan kommunikeres med det ved hjælp af I²C. Skærmen er 128 x 68 pixel og giver mulighed for tekst i forskellige størrelser, figurer og

anden grafik.

Da skærmen er fra Adafruit Industries tilbyder de et stort bibliotek så det er nemt at bruge skærmen og udskrive data i realtid. Der bruges de to biblioteker `Adafruit_GFX`[2], der giver adgang til forskellige grafik på skærmen og `Adafruit_SSD1306`[3] der er driveren til selve skærmen. Disse drivere gør at den er klar til at bruge med det samme.

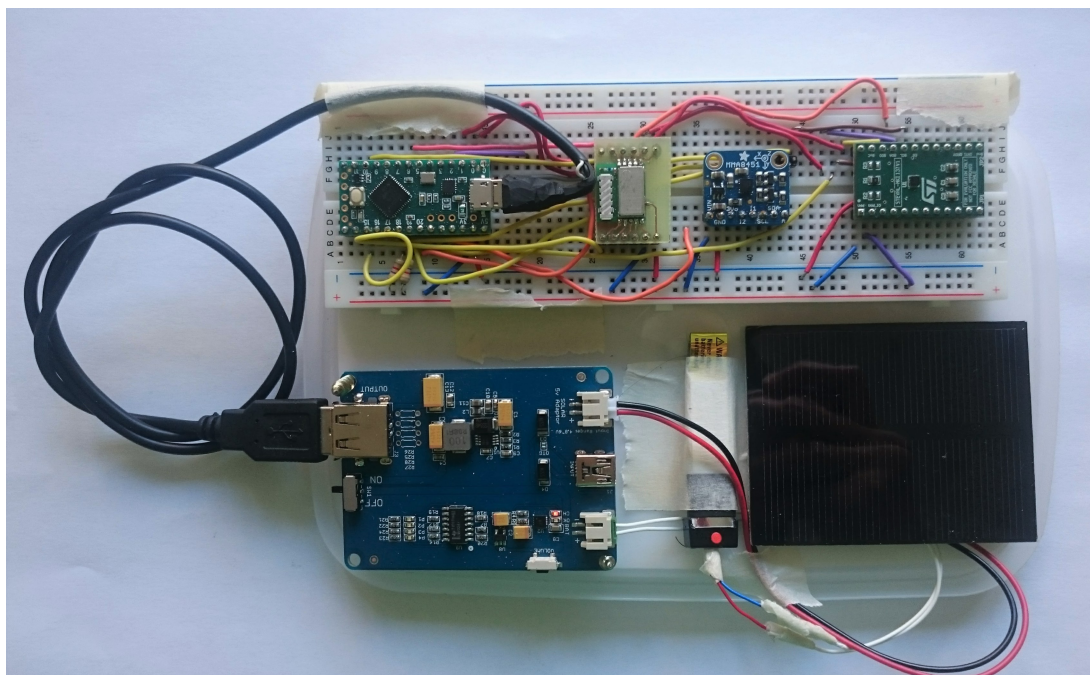


Figur 18: Adafruit breakout af SSD1306 128 x 96 pixel skærm

V. Komponent sammensætning

Med styr på alle komponenterne skal det hele bare sættes sammen. Da dette bare er en udviklingsmodel er det hele sat sammen på et fumlebræt. For nemt at kunne lave ændringer er der ikke loddet noget sammen og det hele er sat sammen med stik og ledninger der kan sættes i fumlebrættet.

Selve måleenheden er sat sammen som det fremgår af blokdiagrammet på figur 1. På figur 19 kan ses hvordan den endelige sammensætning ser ud.



Figur 19: Måle/sende modulet på fumlebræt med strømstyring, batteri og solpanel

På fumlebrættet, set fra venstre, er Teensy-LC μ -controller $\rightarrow$ SPSGRF Trådløs transceiver $\rightarrow$ MMA8451 Accelerometer $\rightarrow$ LIS3MDL Magnetometer. Magnetometeret er sat så langt væk fra alt andet som muligt for ikke at blive påvirket af magnetfelter rundt om ledninger.

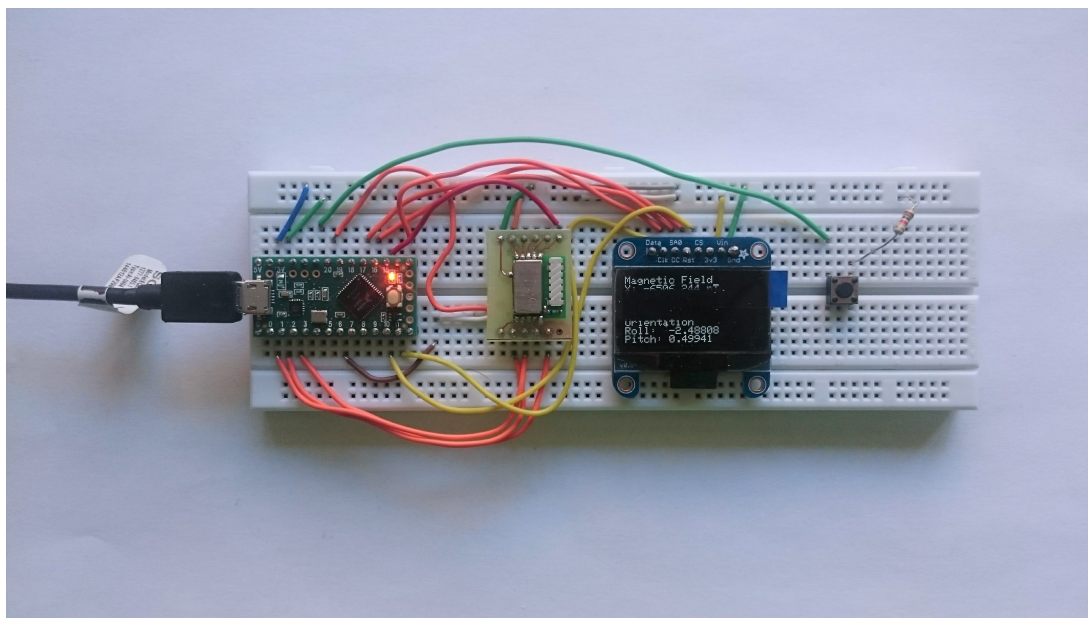
Under fumlebrættet kan ses, fra venstre: LiPo Rider strømstyring $\rightarrow$ LiPo batteri $\rightarrow$ solpanel. Senderen fungerer ved at en 5V spænding forsynes til TeensyLC μ -controlleren af LiPo Rider boardet gennem USB stikket fra batteriet. Batteriet kan oplades gennem LiPo Rider boardet ved hjælp af solcellen eller fra et USB stik fra en computer.

Når TeensyLC'en er blevet tændt initialiserer den alle sensorerne til de indstillinger beskrevet oven for. Med programmet i gang venter Teensy'en på interrupt fra magnetometeret og acce-

lerometeret. Når nye målinger er modtaget fra begge sensorer bliver de pakket og sendt af sted via den trådløse transceiver. Herefter starter den forfra så længe der er strøm på.

Modtageren er også sat sammen som det fremgår af blokdiagram figur 2 dog med en lille tilføjelse. Der er blevet sat en lille knap på der skal bruges til at aktivere sending af målinger videre til computeren. Når knappen trykkes tages gennemsnittet af 50 målinger og dette sendes til computeren.

På figur 20 kan ses fumlebrættet med de monterede komponenter som bruges af modtageren. Fra venstre: TeensyLC μ -controller $\rightarrow$ SPSGRF trådløs transceiver $\rightarrow$ SSD1306 Skærm $\rightarrow$ knap. Modtageren fungerer ved, at den når den tændes initialisere den trådløse transceiver og skærmen. Herefter venter den på at en pakke er modtaget med data. Med data modtaget foretages en kalibrering af data. Herefter udregnes orientering og der laves en rotation af magnetfeltet så dets Z-aksen flugter med tyngdekraften. Herefter udskrives orienteringen og det roterede magnetfelt på displayet og den venter på den næste måling. Hvis der er blevet trykket på knappen udskrives data stadig løbende på skærmen, men der foretages også en gennemsnitsudregning af 50 målinger som herefter sendes til computeren gennem USB stikket.



Figur 20: Modtager komponenterne monteret på fumlebræt

VI. Kalibrering

For at kunne måle magnetfelter og orientering præcist er det nødvendigt at få kalibreret sensorerne således at deres offset minimeres. For at kalibrere sensorerne skal deres måleværdier hentes i en kendt opstilling hvorefter disse værdier kan sammenholdes med de forventede værdier og på den måde kan kalibreringsværdierne bestemmes. Til en kalibrering skal der foretages 6 målinger. Disse målinger er for at bestemme dels hvor 0 værdien ligger og dels for at bestemme om der skal ganges en faktor på for at værdierne passer med det forventede. For at bestemme offset eller 0 punktet laves der to målinger som er modsatrettede. Man skulle derfor gerne få samme værdi bare med et fortegnsskift. For at se om værdien nu er rigtig skal det der måles være en kendt størrelse så man kan få afvigelsen fra denne.

Når målingerne er lavet kan skaleringsfaktoren, α , og offsettet, β , findes.

$$\alpha = \frac{teoretisk_{positive} - teoretisk_{negativ}}{målt_{positive} - målt_{negativ}} \quad (16)$$

$$\beta = teoretisk_{positiv} - \alpha \cdot målt_{positiv} \quad (17)$$

Endelig kan de kalibrerede værdier nu findes ved hjælp af α og β , hvilket giver en lineær kalibrering som det fremgår af ligning 18.

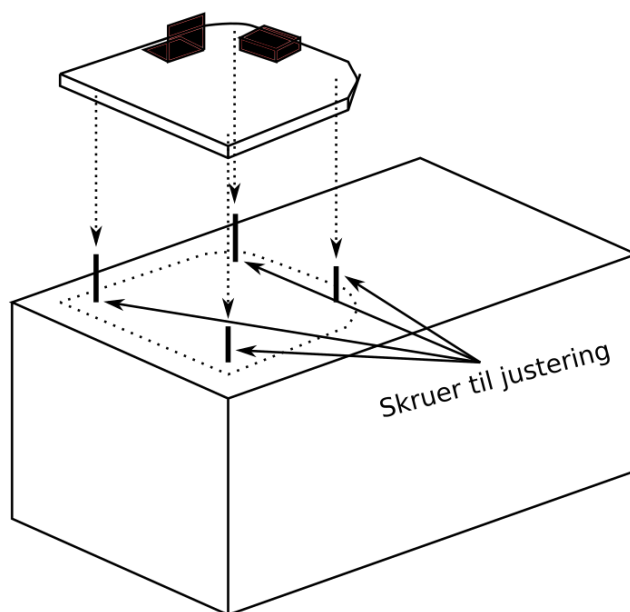
$$val_{kalibreret} = \alpha \cdot val_{målt} + \beta \quad (18)$$

$$\begin{bmatrix} val_{x,kalib} \\ val_{y,kalib} \\ val_{z,kalib} \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & \alpha_z \end{bmatrix} \cdot \begin{bmatrix} val_{x,målt} \\ val_{y,målt} \\ val_{z,målt} \end{bmatrix} + \begin{bmatrix} \beta_x \\ \beta_y \\ \beta_z \end{bmatrix} \quad (19)$$

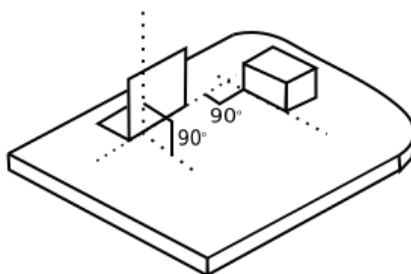
A. Testopstillingen

Testopstillingen blev lavet hovedsageligt af træ og plastik, da den helst ikke skulle kunne påvirke magnetfeltet med metal. Opstillingen bestod af en trækasse der kunne sættes på jorden for at få et stabilt underlag. I toppen af kassen var sat 4 skruer der kunne justeres for at lave en plan overflade til selve monteringspladen. Med de 4 skruer justeret så de var i vatter, kunne en plade placeres ovenpå og der kunne så tjekkes om pladen var plan som det kan ses på figur 21. Ellers kunne skruerne nemt justeres yderligere indtil pladen var plan. Med pladen plan

kan fumlebrættet monteres på pladen i den ønskede position. Ved hjælp af en fastmonteret plastikkloids og en plastik vinkel kunne fumlebrættet nemt sættes på samme måde hver gang for at mindske måleusikkerhederne. Begge sensorer er monteret på fumlebrættet således at de er i plan med fumlebrættets flader, da det er nemmere at få sat større objekter præcist. Placeringen af plastikklodserne kan ses på figur 22. Placeringen af fumlebrættet på testopstillingen kan ses på billederne i appendix A..



Figur 21: Testopstillingen til kalibrering af sensorer



Figur 22: Nærbillede af toppladen som fumlebrættet placeres på samt vinklerne for plastikklodserne

Tabel 2: Målinger fra accelerometer til kalibrering. Gennemsnit af 1000 målinger over 10 sek

	Forventet			Målt		
	a_x, g	a_y, g	a_z, g	a_x, g	a_y, g	a_z, g
Pos 1, x positiv	1	0	0	0.9733	0.0096	0.0092
Pos 2, x negativ	-1	0	0	-1.0027	0.0158	-0.0004
Pos 3, y positive	0	1	0	-0.0113	1.0053	0.0198
Pos 4, y negative	0	-1	0	-0.0118	-0.9780	0.0081
Pos 5, z positiv	0	0	1	-0.0138	0.0030	1.0039
Pos 6, z negative	0	0	-1	-0.0116	0.0228	-0.9837

B. Accelerometer kalibrering

For accelerometeret skal der kalibreres i forhold til tyngdekraften. Dette gøres ved at placere accelerometeret således at 2 af akserne vil måle 0g og den sidste måler 1g. Ved at gøre dette 2 gange for hver akse, således at man har en måling hvor accelerationen bør være 1g og en hvor den bør være omkring -1g, kan man se hvor meget hver akse afviger fra den ønskede værdi. De målte værdier fremgår af tabel 2 sammen med de forventede værdier.

Med måleværdierne på plads kan vi udregne α og β som beskrevet ovenfor.

$$\alpha_x = \frac{1g - (-1g)}{0.9733g - (-1.0027g)} = 1.0111$$

$$\beta_x = 1 - 1.0111 \cdot 0.9733g = 0.0149 \text{ g}$$

$$\alpha_y = \frac{1g - (-1g)}{1.0053g - (-0.9780g)} = 1.0084$$

$$\beta_y = 1 - 1.0084 \cdot 1.0053g = -0.0138 \text{ g}$$

$$\alpha_z = \frac{1g - (-1g)}{1.0039g - (-0.9837g)} = 1.0063$$

$$\beta_z = 1 - 1.0063 \cdot 1.0039g = -0.0101 \text{ g}$$

Med disse værdier fås kalibreringsligningen der skal bruges når den målte data skal kalibreres:

$$\begin{bmatrix} a_{x,kalib} \\ a_{y,kalib} \\ a_{z,kalib} \end{bmatrix} = \begin{bmatrix} 1.0111 & 0 & 0 \\ 0 & 1.0084 & 0 \\ 0 & 0 & 1.0063 \end{bmatrix} \cdot \begin{bmatrix} a_{x,målt} \\ a_{y,målt} \\ a_{z,målt} \end{bmatrix} + \begin{bmatrix} 0.0149 \text{ g} \\ -0.0138 \text{ g} \\ -0.0101 \text{ g} \end{bmatrix} \quad (20)$$

C. Magnetometer kalibrering

På samme måde kan der laves 6 målinger med magnetometeret i de samme positioner og bruge dette til at kalibrere ud fra. Til denne måling blev der kalibreret i forhold til Jordens magnetfelt og helt præcist så z-komponenten af Jordens magnetfelt. Kalibreringen blev foretaget mandag den 5. juni i en park i København. Der blev valgt en park for at mindske effekten af andre felter der kunne skyldes strømforsyningsledninger, huse osv. ved hjælp af NOAA's hjemmeside⁴ blev størrelsen af Jordens magnetfelt samt z-komponenten fundet i København den pågældende dag. I følge NOAA var magnetfeltets størrelse som følger:

$$|B_{Jorden}| = 50306.5 \text{ nT}$$

$$B_z = 47313.3 \text{ nT}$$

Tabel 3: Målinger fra magnetometer til kalibrering. Gennemsnit af ca 1000 målinger over 10 sek. Da der kun kigges på Z-komponenten af Jordens magnetfelt er der kun opgivet 1 forventet værdi

	Forventet			Målt			
	B_x , nT	B_y , nT	B_z , nT	B_x , nT	B_y , nT	B_z , nT	Offset, nT
Pos 1, x positiv	47313.3	-	-	49850.7	-24592.4	29803.4	2537.4
Pos 2, x negativ	-47313.3	-	-	-47398.1	8387.9	29940.9	-84.8
Pos 3, y positive	-	47313.3	-	14961.7	37542.2	31470.7	-9771.1
Pos 4, y negative	-	-47313.3	-	-12690.7	-55490.6	30870.0	-8177.3
Pos 5, z positiv	-	-	47313.3	-4743.9	7915.7	84583.4	37270.1
Pos 6, z negative	-	-	-47313.3	4508.1	9089.3	-12099.8	35213.5

Som det kan ses af tabel 3 så har magnetometeret meget store afvigelser og en kalibrering er derfor meget vigtig for at kunne bruge dens data til noget. Som for accelerometeret kan α og β for de 3 akser udregnes.

⁴<https://www.ngdc.noaa.gov/geomag-web/#igrfgrid>

$$\alpha_x = \frac{47313.3\text{nT} - (-47313.3\text{nT})}{49850.7\text{nT} - (-47398.1\text{nT})} = 0.97304$$

$$\beta_x = 47313.3 - 0.97304 \cdot 49850.7\text{nT} = -1193.3 \text{ nT}$$

$$\alpha_y = \frac{47313.3\text{nT} - (-47313.3\text{nT})}{37542.2\text{nT} - (-55490.6\text{nT})} = 1.01713$$

$$\beta_y = 47313.3 - 1.01713 \cdot 37542.2\text{nT} = 9128.0 \text{ nT}$$

$$\alpha_z = \frac{47313.3\text{nT} - (-47313.3\text{nT})}{84583.4\text{nT} - (-12099.8\text{nT})} = 0.97873$$

$$\beta_z = 47313.3 - 0.97873 \cdot 84583.4\text{nT} = -35470.9 \text{ nT}$$

Med α og β værdierne på plads kan kalibreringsligningen sættes op:

$$\begin{bmatrix} B_{x,kalib} \\ B_{y,kalib} \\ B_{z,kalib} \end{bmatrix} = \begin{bmatrix} 0.97304 & 0 & 0 \\ 0 & 1.01713 & 0 \\ 0 & 0 & 0.97873 \end{bmatrix} \cdot \begin{bmatrix} B_{x,m\ddot{a}lt} \\ B_{y,m\ddot{a}lt} \\ B_{z,m\ddot{a}lt} \end{bmatrix} + \begin{bmatrix} -1193.3 \text{ nT} \\ 9128.0 \text{ nT} \\ -35470.9 \text{ nT} \end{bmatrix} \quad (21)$$

D. Yderligere kalibreringer

For begge kalibreringer er det antaget at der ikke er nogen usikkerhed på måleaksernes placering i forhold til hinanden. Der er derfor ikke nogen korrelation mellem de enkelte akser. Det er altså antaget at produktionen af sensorerne er så god at vi kan se bort for små usikkerheder her. For at kunne bekræfte denne antagelse er det nødvendigt at lave nogle målinger hvorefter man kan udregne den egentlige vinkel mellem de enkelte akser og på den måde se om antagelsen holder.

VII. Målinger

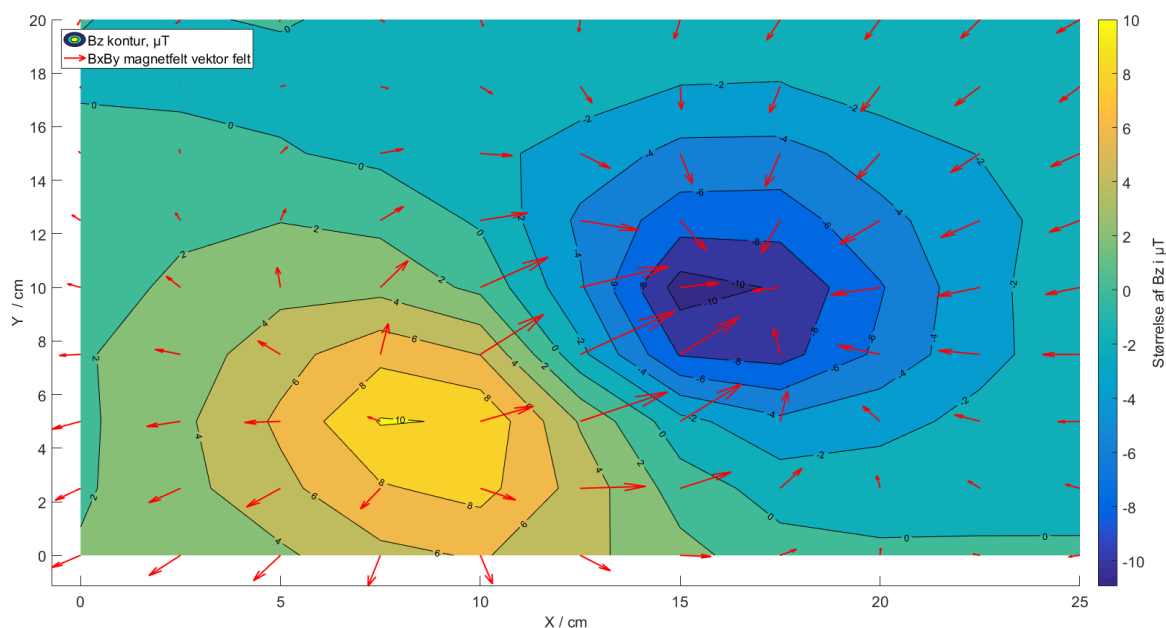
A. Feltvisualiseringer

For at teste om systemet også kunne bruges til noget, blev der foretaget flere målinger af magnetfelterne omkring flere forskellige magnetiske og metalliske objekter. Målingerne blev foretaget på et træbord hvorpå der var tegnet et koordinatsystem og med magnetiske objekter tapet fast på undersiden af bordpladen. Bordpladen var 3.4cm tyk og selve magnetometeret er placeret 2.3cm fra bunden af instrumentet. Alle måleobjekterne er derfor placeret 5.7cm under magnetometeret. Opsætningen kan ses i appendix C.

Måleinstrumentet blev placeret oven på bordet og der blev foretaget en mængde systematiske målinger ud fra det påtegnede koordinatsystem. Disse målinger gav nu en matrix fyldt med vektorerne for magnetfeltet i alle målepunkterne. Som beskrevet overfor er hver måling et gennemsnit af 50 målinger taget inden for et par sekunder på hver position. Med en fyldt matrix med målinger var det muligt at begynde at undersøge om det var muligt at identificere de magnetiske felter og om det var muligt at sige noget om objektet magnetfeltet stammede fra. Alle objekterne der er målt på kan ses i appendix B.

Efter de nødvendige målinger var blevet lavet og anbragt i en matrix blev hjørne målingerne brugt til at bestemme størrelsen af baggrundsmagnetfeltet. Med en gennemsnitlig størrelse for baggrundsfeltet bestemt blev dette trukket fra alle målingerne således at kun det ønskede felt var tilbage. Det blev herefter plottet i et XY-plot med målepositioner på akserne. For hvert målepunkt er så indtegnet magnetfeltets vektor i XY-planen. Oven på dette blev så lagt et konturplot for at vise størrelsen af Z-komponenten for det målte felt. Ud fra dette plot er det muligt at beskrive magnetfeltet. Et sådan plot kan f.eks. ses på figur 23. Til alle afsnit er vist et plot, men for at kontur plot for B_x og B_y kan ses i appendix D. for at få en idé om styrken af disse komponenter.

Den første måling blev lavet på en lille svagt magnetiseret skruetrækker. Skruetrækkeren er 12.5 cm lang, men selve metaldelen er lige godt 9 cm. Der blev foretaget 99 målinger for skruetrækkeren på et 20 x 25 cm område med 2.5 cm mellem hver måling. Den grafiske repræsentation af disse målinger kan ses på figur 23.

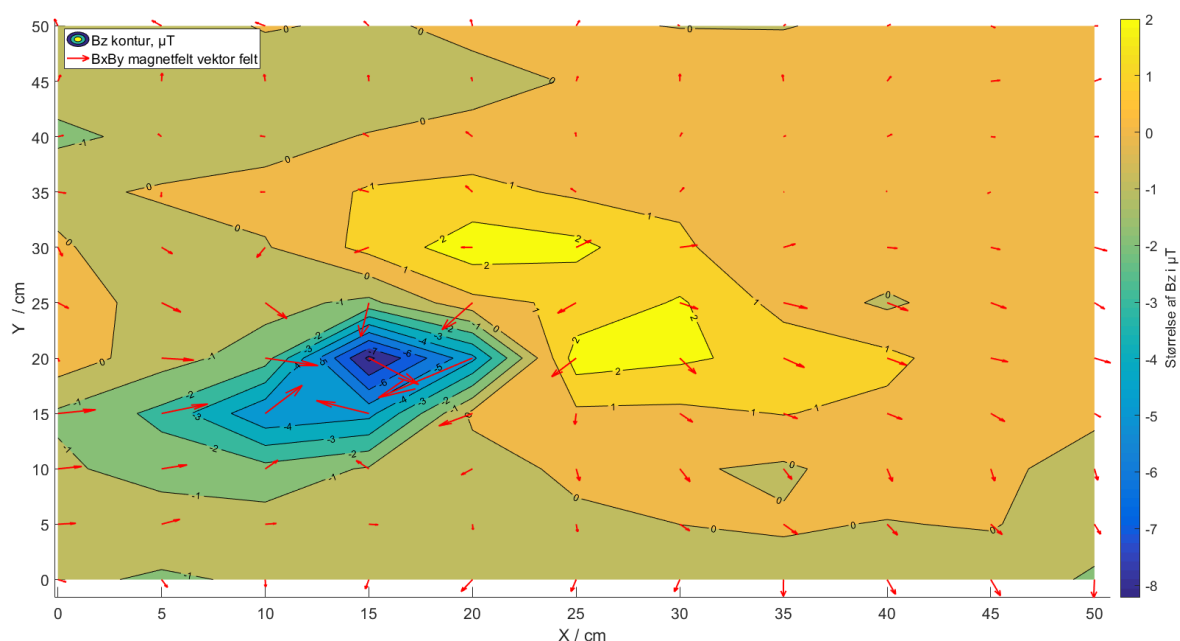


Figur 23: Magnetfeltet målt for en lille skruetrækker. Pilene viser magnetfeltets vektorfelt i XY-planen.

Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magnetometeret på målebordet

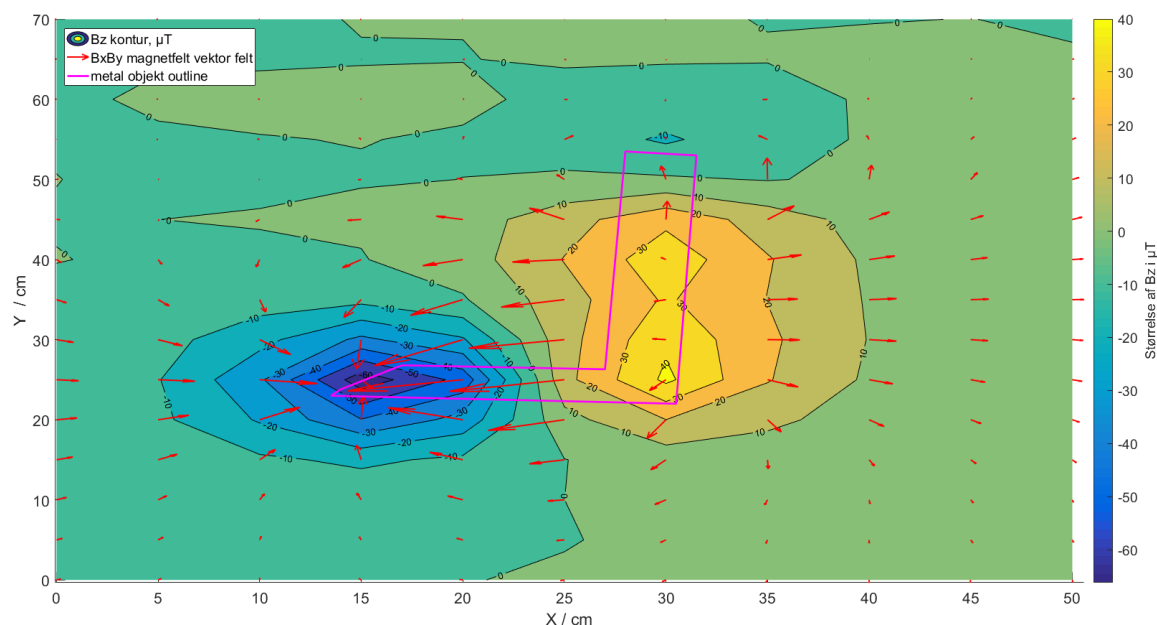
På plottet er ses det tydeligt at skruetrækkeren er en magnetisk dipol. Det ses at skruetrækkeren har en magnetisk nordpol i det gule område og en sydpol i det blå. Feltlinjerne som tegnes følger også det man ville forvente af en stangmagnet, hvilket skruetrækkeren i bund og grund er, og går fra nord til syd i fine buer. Der er ingen tvivl om placeringen og orienteringen af skruetrækkeren. Den ene ende af metallet må befinde sig i ved nordpolen og den anden ved sydpolen. Vi kan også se at felterne er lige stærke ved nord og sydpolen så den er placeret i samme plan som bordet og er ikke vinklet væk fra bordet.

Den næste måling var af en flad firkantet køleskabsmagnet på 5 x 10 cm. Resultatet af målingerne er fremgivet i figur 24. Den præcise udformning af dette felt er ikke helt så entydigt som for skruetrækkeren. Det er tydeligt at der sker noget omkring $X = 15\text{cm}$ og $Y = 20\text{cm}$ og der befinder sig noget der ligner en sydpol. De gule pletter antyder også noget der ligner en nordpol og vi ser da også en antydning af nogle feltlinjer fra nord til syd. Det er svært at sige noget om hvad der skaber dette felt da feltet ikke har en tydelig form.



Figur 24: Magnetfeltet målt for en firkantet køleskabsmagnet på 5x10 cm. Pilene viser magnetfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magnetometeret på målebordet

Næste måling var på en metal vinkel lavet af stål. På den lange led var vinklen 32 cm og 4 cm bred. På den korte led er vinklen 17 cm og 4.1 cm bred. Metallet er 1mm tyk, med undtagen af bunden af den korte del hvor der var sat en lille H-formet metal stang som det kan ses på figur 26. Der er derfor mest metal i bunden hvilket burde give de største udsving på magnetfeltet.



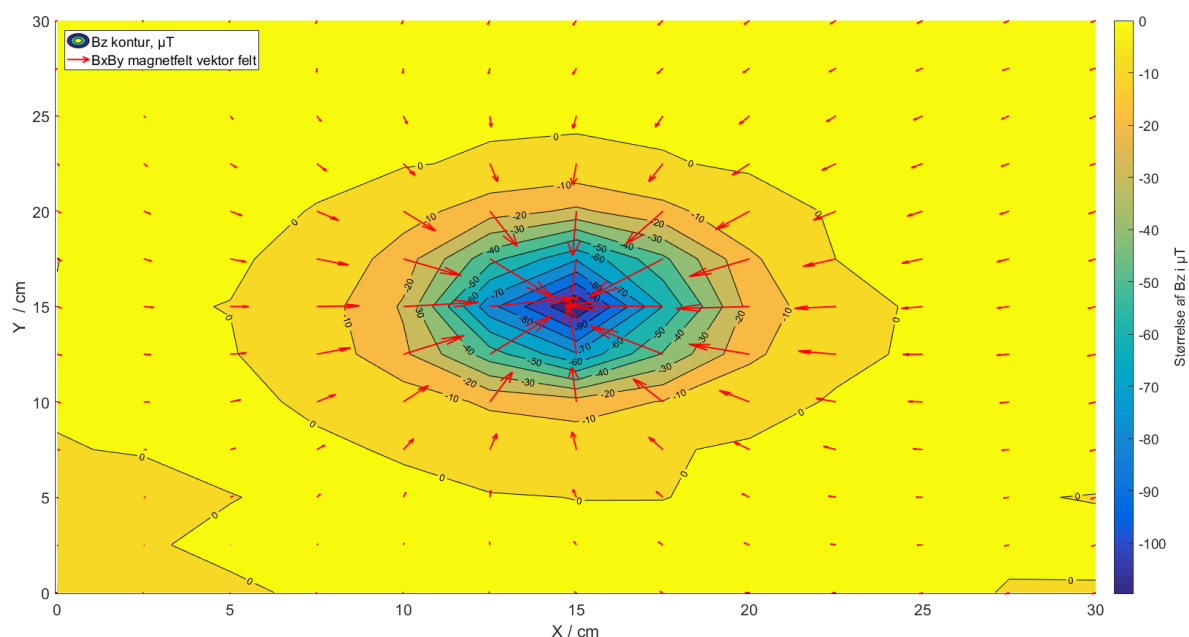
Figur 25: Magnetfeltet målt for en jernvinkel. Placeringen af jernvinklen kan ses af det lyserøde outline. Pilene viser magnetfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magnetometeret på målebordet

På figur 25 kan ses magnetfelt plottet for jernvinklen. På plottet ses det tydeligt at vi har noget magnetisk. Vi kan se at der er en sydpol koncentreret omkring spidsen af vinklen og at der er en nordpol på den lange del af vinklen. Det ses tydeligt at feltet udspringer fra den lange del af jernpladen. Feltlinjerne er meget tydelige og feltet er i samme størrelse som Jordens magnetfelt. Hvis man skal prøve at sige noget om hvad der laver feltet kan man se at det ikke er noget symmetrisk. Det er tydeligt at det går ud i en form for spids ved det blå område og at der er en stor flade i det gule område man ville nok godt kunne se at det er noget L-formet der udsender felter.



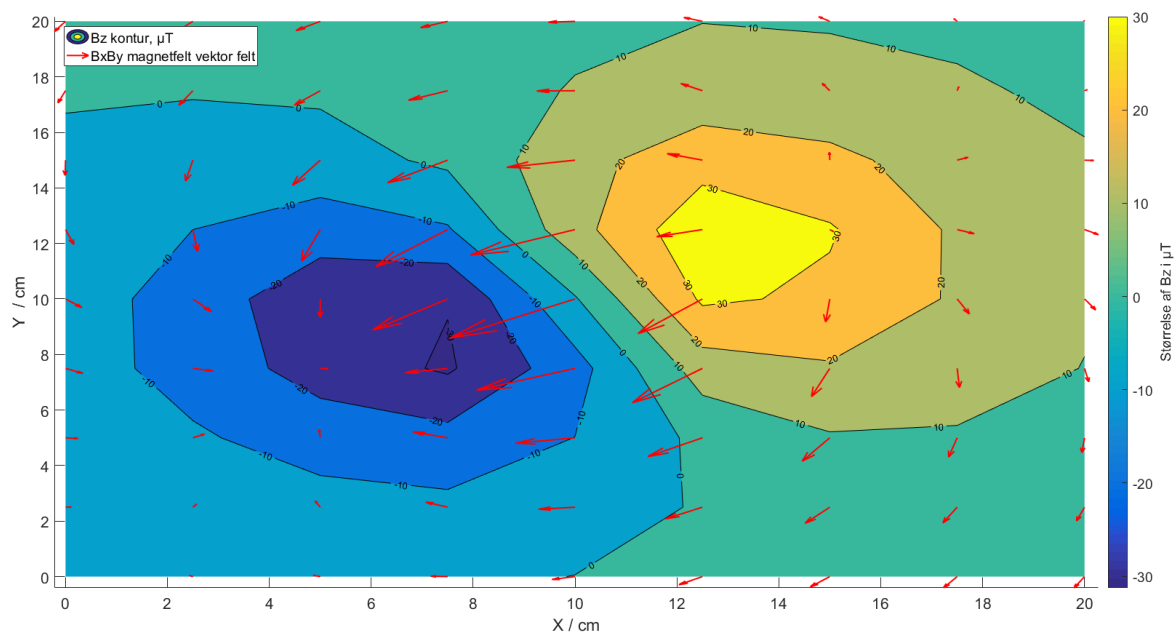
Figur 26: H-formet metalstang på undersiden af jernvinklen

Det sidste sæt målinger blev lavet på en lille, men stærk, knapmagnet med en diameter på 1 cm og en tykkelse på 1 mm. Der er blevet lavet 169 målinger med 2.5cm afstand mellem hver. På de første målinger er magneten placeret fladt under bordet hvilket kan ses på figur 27. Her kan det tydeligt ses at den side der vender mod instrumentet er magnetens sydpol. Det er tydeligt at vi har en stærk magnet som er orienteret så vi kigger ned gennem midten af den da der ikke er nogen antydning af nordpoler på plottet.



Figur 27: Magnetfeltet målt for en lille rund knapmagnet med en diameter på 1 cm placeret fladt under bordpladen. Pilene viser magnetfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magnetometeret på målebordet

For næste sæt målinger, hvor der blev lavet 121 målinger med 2.5 cm mellem hver, blev magneten placeret på højkant så den faktisk bare fungerede som en meget lille stangmagnet. Det er også helt tydeligt at det er det vi ser. Plottet ligner meget plottet for skruetrækkeren som man også ville forvente. Dog med forskellen at polerne er meget tættere på hinanden.



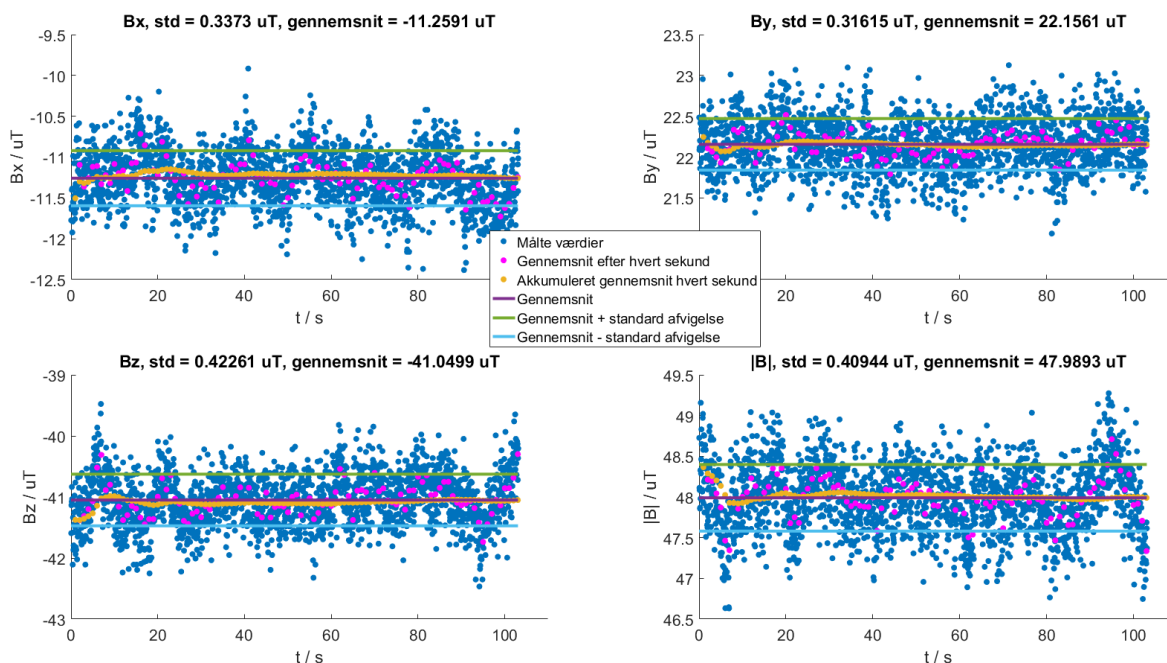
Figur 28: Magnetfeltet målt for en lille rund knapmagnet med en diameter på 1 cm placeret på højkant under bordpladen. Pilene viser magnetfeltets vektorfelt i XY-planen. Konturkortet viser størrelsen af Z-komponenten af magnetfeltet. Akserne viser placeringen af magnetometeret på målebordet

Alle disse målinger viser at det er muligt at bestemme magnetfelter ved hjælp af dette instrument også helt ned i μT størrelserne som det var målet. Med mere tid kan der også regnes på disse data så man kan sætte nogle tal på magnetfelterne og i sidste ende også deres placering dybdemæssigt.

B. Magnetometer sensitivitet

For at bestemme hvor sensitivt magnetometeret var og hvor præcist man kunne håbe at måle magnetfelterne blev der lavet nogle målinger hvor magnetometeret var stationært og placeret så isoleret som det kunne. Her blev der lavet målinger over 100 sekunder og over 500 sekunder for at se hvor meget variation der var i disse målinger, som det ønskes skal være så ens som mulige. Første måling over 100 sekunder gav ca 2000 målinger. Et plot af disse målinger kan ses på figur 29. Det fremgår tydeligt her at der er større spredning end hvad er ønsket. Standardafvigelsen

ligge mellem 0.3 og 0.5 μT for alle akserne og dette sæt målinger er et mere jævnt fordelt sæt over 100 sekunder en noget interval af 100 sekunder fra de længere målinger på figur 30 og 31.

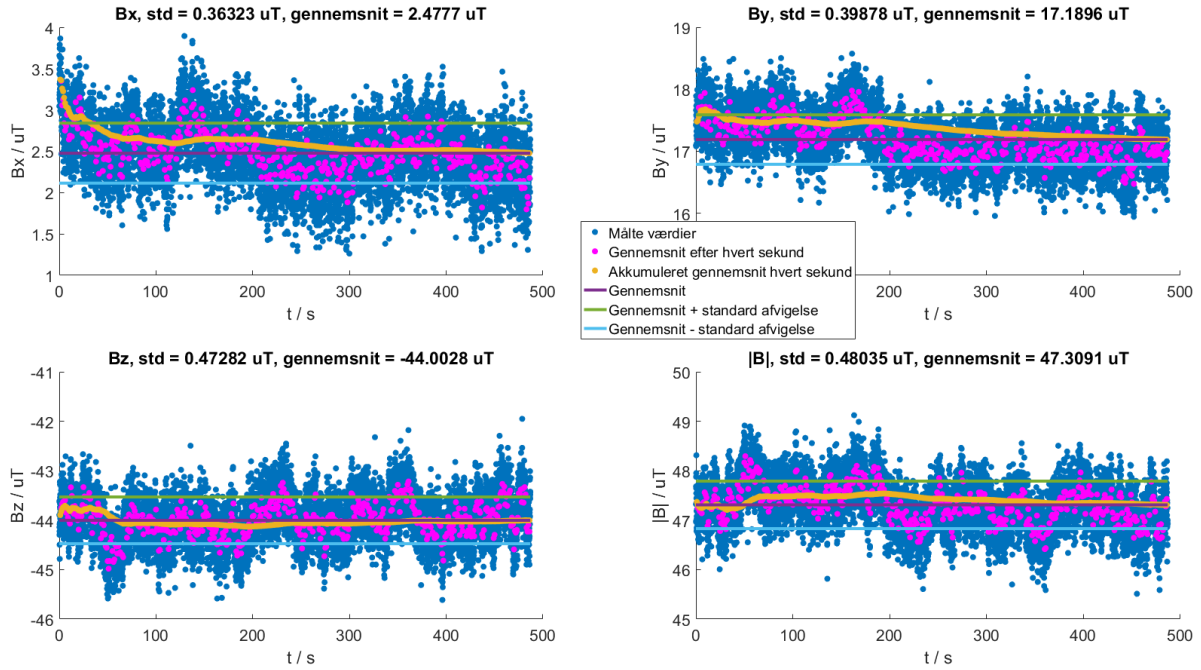


Figur 29: 2000 målinger med et stationært instrument. Taget indenfor. Spredning over disse 100 sekunder er i den lave ende af det erfarede

En standardafvigelse på mellem 0.3 og 0.5 μT er alt for meget hvis vi gerne vil have en præcision på 0.1 μT . For en enkelt måling er altså kun præcis ned til 1 μT . Men normalt man ikke kun kigge på en måling for at bestemme værdien så vi må også kigge på hvordan det ser ud gennemsnitligt. Kigges på de lyserøde prikker på figurene, som er gennemsnittet af 1 sekunds målinger ses det allerede at stort set alle prikkerne er inden for standardafvigelsen, hvilket for X og Y komponenten gør os i stand til at bestemme felterne med en præcision på 0.6 μT og 0.8 μT for Z komponenten. Går vi endnu videre og ser på det akkumulerede gennemsnit kommer vi faktisk inden for 0.1 μT standardafvigelsen efter 10 sekunder. Så hvis vi har et stabilt felt og kan nøjes med en måling hver 10 sekund, er det muligt med en præcision på 0.2 μT . Dette er selvfølgelig ikke helt så godt som ønsket, men muligvis med et filter på dataen kan de 0.1 μT opnås.

Kigges der på figur 30 og 31 ses målingerne over længere tid. Her ser kurverne ikke så fine ud som på figur 29. Om dette skyldes magnetometeret eller om det har kunne være små fluktuationer i

magnetfelter hvor der er blevet målt, eller begge dele, er ikke til at sige. For at definitivt kunne sige hvor præcist magnetometeret er skal der måles hvor man er helt sikker på magnetfeltets udseende. Ud fra plottene kan det dog siges at spredningen ser ud til at være meget konstant omkring de lyserøde middelværdier, så antagelserne fra figur 29 stadig holder.



Figur 30: 9500 målinger med stationært instrument. Taget udenfor. Der var over 5 meter til nærmeste bygninger hele vejen rundt

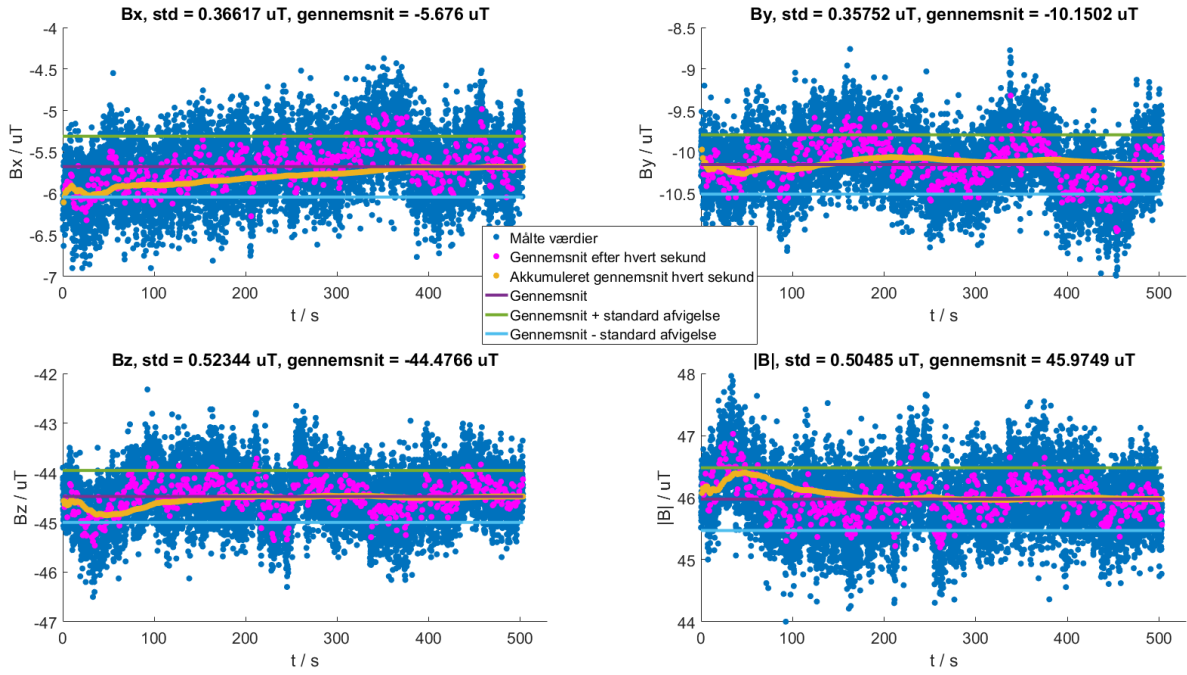
C. Orienteringspræcision

Et af kravene til instrumentet var bestemmelse af præcision med 1° s nøjagtighed. Til dette blev standard afvigelsen af accelerometerets x,y og z værdier for den 8 minutter lange stationære måling beregnet:

$$\sigma_{ax} = 0.001167g \quad (22)$$

$$\sigma_{ay} = 0.001326g \quad (23)$$

$$\sigma_{az} = 0.000899g \quad (24)$$



Figur 31: 10000 målinger med stationært instrument. Taget indendørs

Herefter kan udregnes roll og pitch vinklerne hvor standardafvigelsen bruges til at forværre vinklen i forhold til idealtilfældet. Så når vi f.eks. har g flugtende med Z-aksen vil vi få vinklerne

$$\alpha = \arctan \left(\frac{\sigma_{ax}}{\sqrt{\sigma_{ay}^2 + (1g + \sigma_{az})^2}} \right) = \arctan \left(\frac{0.001167g}{\sqrt{(0.001326g)^2 + (1g + 0.000899g)^2}} \right) = 0.067^\circ \quad (25)$$

$$\beta = \arctan \left(\frac{\sigma_{ay}}{\sqrt{\sigma_{ax}^2 + (1g + \sigma_{az})^2}} \right) = \arctan \left(\frac{0.001326g}{\sqrt{(0.001167g)^2 + (1g + 0.000899g)^2}} \right) = 0.076^\circ \quad (26)$$

Som det kan ses er disse afvigelser ikke nok til at afvige fra den korrekte vinkel på 0° med en hel grad og instrumentet holder sig derfor inden for den 1° som er nødvendigt.

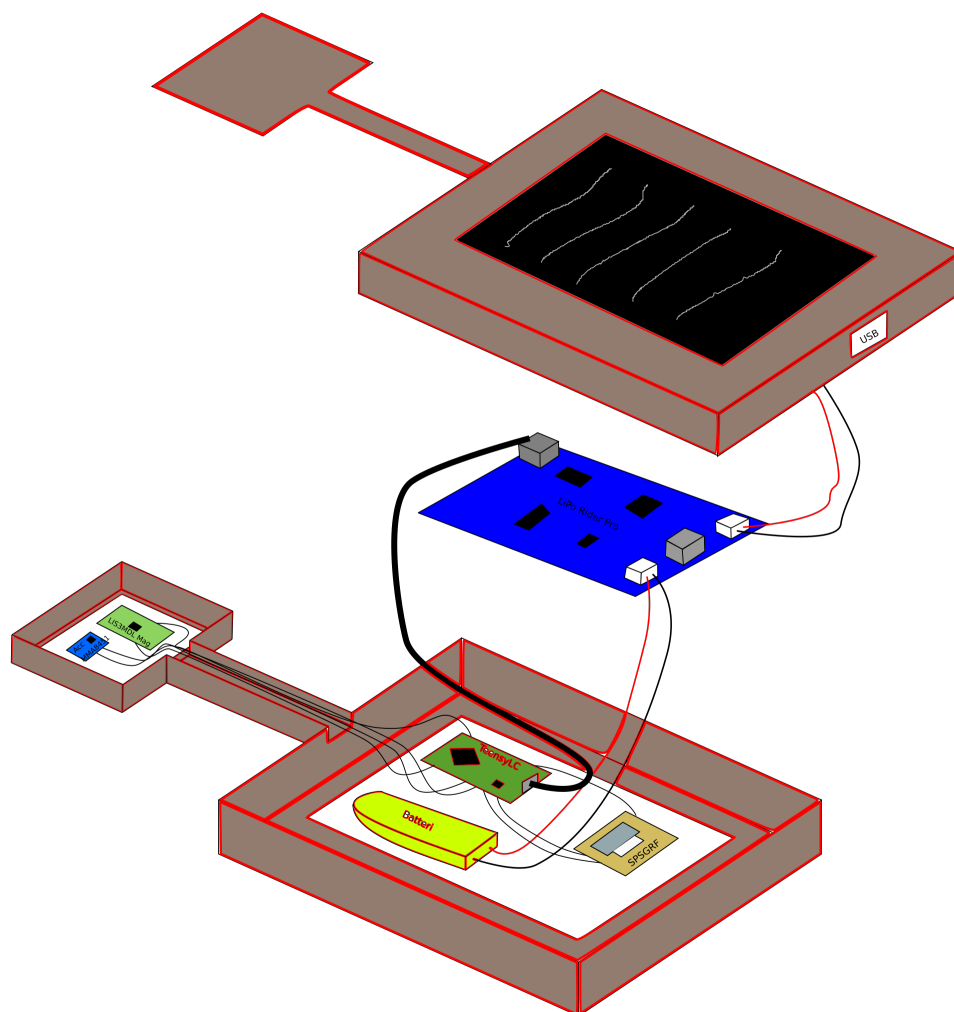
VIII. Endeligt produkt og fremtidigt arbejde

A. Produktdesign

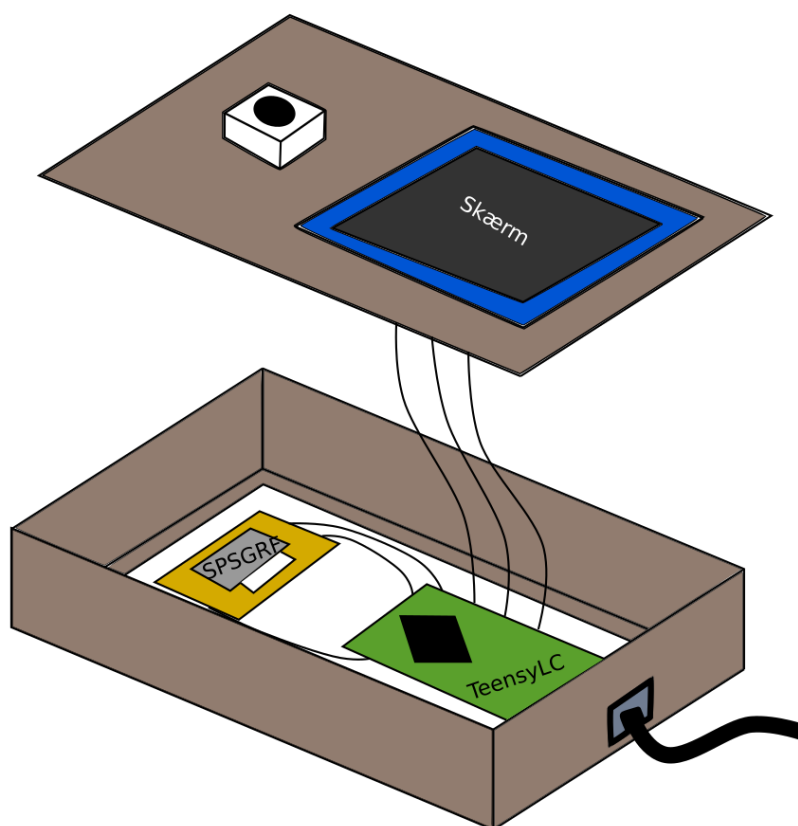
I løbet af projektet er der blevet lavet en såkaldt 'flat-model' af instrumentet. D.v.s. at der er lavet et instrument uden at der er tænkt på størrelse og design. Alt er åbent så det er muligt at lave ændringer og målinger på alle sensorer. Det næste skridt er derfor at få lavet en grov prototype. En grov prototype ville bestå af at få loddet alle komponenterne sammen så forbindelserne er helt sikre og det skal gøres så småt som muligt. Herefter skal der designes og laves en form for hylster så alle komponenterne ikke er eksponeret.

Der skal designes en 3D model til både sender og modtager. Herefter kan disse modeller 3D-printes så prototypen kan samles. Modtageren vil være den simpleste da denne bare skal have Teensy, skærm, en knap og trådløs transceiver. Idéen er en kasse som indeholder al elektronikken, med skærmen stikkende ud på toppen med en knap og et USB til Teensy'en. Ved at printe den i 2 dele kan komponenterne monteres på hver sin halvdel og derefter skrues sammen. Udkastet til designet kan ses på figur 33.

Senderen er ikke helt så simpel som modtageren. Både fordi den har flere komponenter, men også fordi man gerne vil have magnetometeret isoleret så den ikke bliver påvirket af de andre dele. En måde at gøre dette er ved at montere magnetometeret væk fra resten, ude på en slags arm. Senderen bliver derfor delt op i to bokse. Den ene boks indeholder de to sensorer og den anden indeholder Teensy, batteri, LiPo Rider og den trådløse transceiver. Begge sensorer skal monteres sammen da det er vigtigt at de er orienteret helt ens, for at man kan bruge målingerne til noget. Med en arm med en lille boks med de to sensorer kan resten pakkes i en større boks som det blev gjort for modtagerenheden. Et hult rør mellem de to bokse vil sørge for at holde fast på det hele og ledningerne til sensorene vil kunne løbes igennem røret. Udkastet til designet kan ses på figur 32.



Figur 32: Design til senderen. Magnetometer og accelerometer er placeret i det lille rum for enden af armen og resten af elektronikken er placeret i det store rum. Armen bruges både til at give afstand og til at løbe ledninger igennem



Figur 33: Design til modtagerenheden. En boks der indeholder elektronikken med stærm og knap på toppen. Et USB stik kan sættes i siden, ind til Teensy'en

B. Systemopgraderinger

Da dette projekt er gået ud på at lave et proof-of-concept system, har der ikke været tid til at gå i dybden med hver enkelt ting. Det trådløse system virker, men kan sagtens optimeres meget og pakkehåndteringen kan også gøres bedre så der ikke mistes pakker og data bliver forvrænget. Det var målet at den skulle kunne sende over så lange afstande som muligt, og som en start gerne over 10 meter. Som det virker nu begynder signalet dog at forsvinde og være ustabil efter 2-3 meter. Noget arbejde på evt. at optimere antennen og mere viden omkring hvilke konfigurationer af RF-linket på SPSGRF modulet vil kunne gøre noget for at få forbedret denne afstand. Også på software siden vil der kunne laves en del opgraderinger som vil give endnu flere funktioner såvel som øget brugervenlighed. Her kan f.eks. nævnes at magnetometeret som nævnt tidligere kan måle felter helt op til 1.6 mT. Som det er nu er det dog kun konfigureret til at

måle op til 0.4 mT. En inkorporering af en automatisk funktion der indstiller den nødvendige rækkevide ud fra de målte data vil derfor være oplagt.

Med opgradering af den trådløse kommunikation vil der også kunne kigges på en måde at sende kommander fra computeren eller bare fra modtagerenheden tilbage til måleinstrumentet. Dette ville kunne bruges til at ændre f.eks. datarater, følsomhed eller andre ting.

En sidste ting der kunne være godt at kigge på er en form for 'Power-management' system, således at Teensy'en holder øje med batteriet og reagere på dette. F.eks. ved at sætte tempoet ned og evt. sende data med større mellemrum så batteriet hurtigt kan blive opladt igen.

Ud over det produktmæssige fremtidige arbejde er der også stadig ting der kan gøre i forhold til produkttest. Det ville være oplagt at få lavet nogle målinger der kan bruges til at bestemme den præcise følsomhed af instrumentet så man ikke er i tvivl om hvilke opgaver instrumentet kan bruges til. Der er blevet lavet nogle målinger og givet et skøn for sensitiviteten, men for et meget præcist resultat skal der laves målinger i mere kontrollerede forhold.

Det vil også være godt hvis der kan laves et studie omkring strømforbruget af de enkelte dele. Derudover vil det være godt at finde ud af hvor lang tid instrumentet kan køre og hvor lang tid det vil tage at oplade under forskellige forhold.

IX. Konklusion

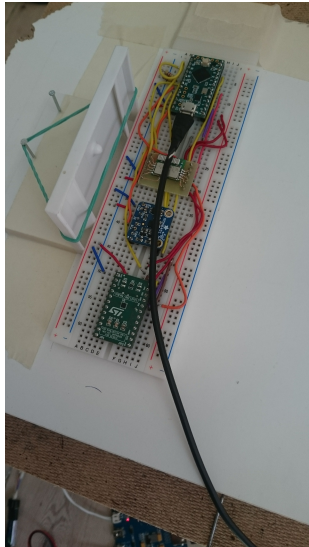
I løbet af dette projekt er der blevet lavet en 'flat' version af et trådløst måleinstrument til måling af magnetfelter og få dem defineret i forhold til tyngdekraften. Det endelige produkt er ikke en prototype endnu og stadig på fumlebrætstadiet, men det er blevet vist at konceptet virker og den ønskede funktionalitet af instrumentet er opnået. Alle de forudtænkte komponenter er blevet konfigureret og har på nuværende tidspunkt fungeret som håbet og antaget, med undtagen af den trådløse tranceiver som ikke helt har den ønskede rækkevidde. Dette kan som diskuteret skyldes antenner eller konfigurationer som der skal arbejdes videre på. Derudover kan magnetometeret på nuværende tidspunkt ikke komme op på den ønskede sensitivitet, men med et godt designet filter kunne dette problem løses. Endvidere er der blevet udviklet softwarebiblioteker til magnetometeret og den trådløse transceiver som vil kunne bruges til andre instrumenter.

I løbet af projektet er der succesfuldt lavet kalibreringer af de to essentielle sensorer og det har været muligt at måle og beskrive flere forskellige objekters magnetfelter selvom disse var svagere end Jordens. Det var ud fra målingerne muligt at få gode informationer om magnetfelter og målingerne kunne foretages helt uden besvær.

Selvom det endelige produkt ikke er nået så langt som man kunne have håbet, da meget af det indledende arbejde med komponenterne har taget længere tid end først antaget, burde der ikke være meget arbejde før en brugbar prototype kan være lavet.

X. Appendix

A. Kalibreringspositioner



(a) Z aksen opad



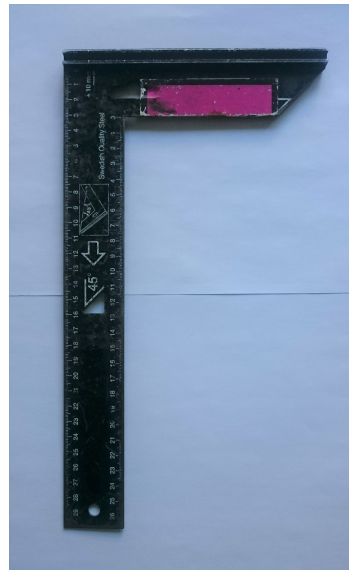
(b) Y aksen opad



(c) X aksen opad

Figur 34: De tre positioner der skulle bruges til kalibrering af sensorerne

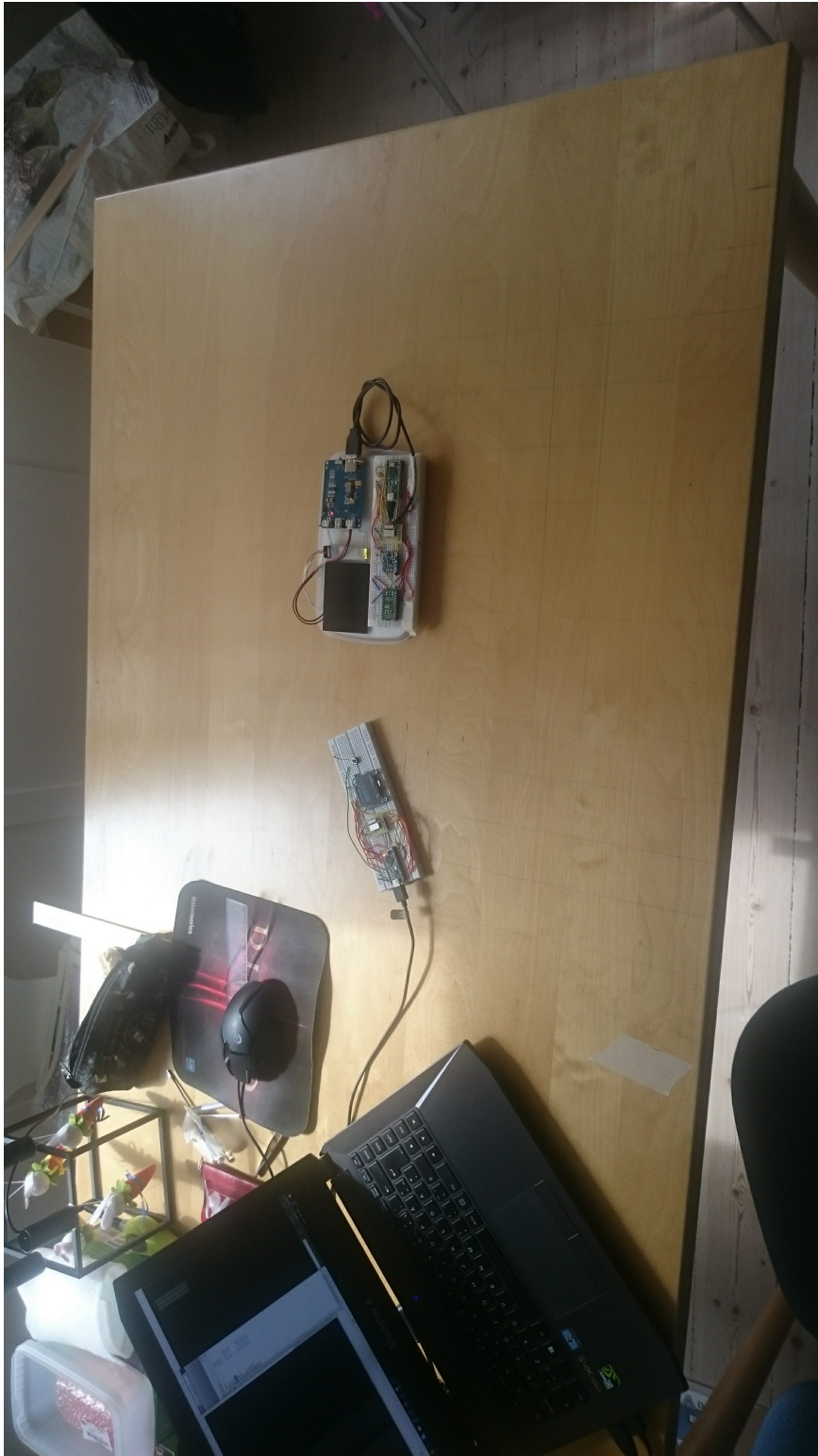
B. Magnetiske objekter til målingerne

*(a) Knapmagnet**(b) Jernvinkel**(c) Lille skruetrækker**(d) Flad køleskabsmagnet**Figur 35: Objekterne hvis magnetfelter blev målt*

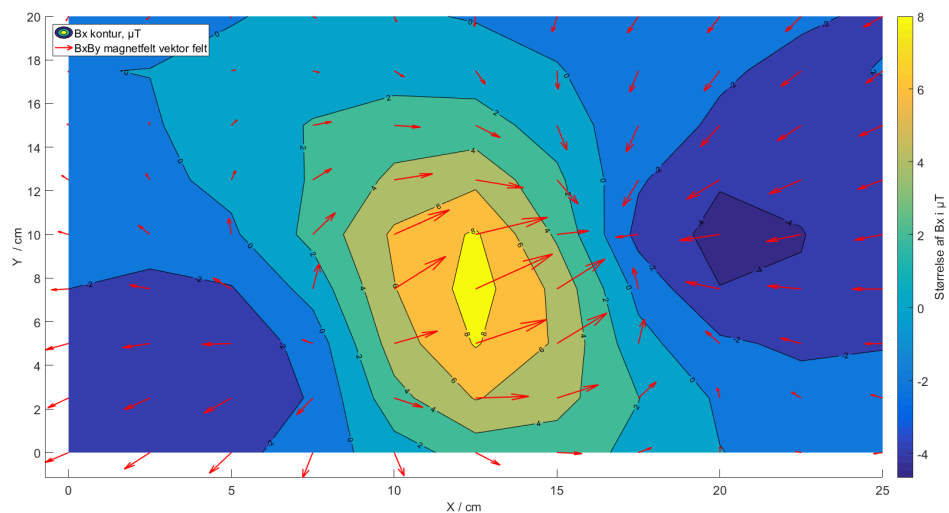
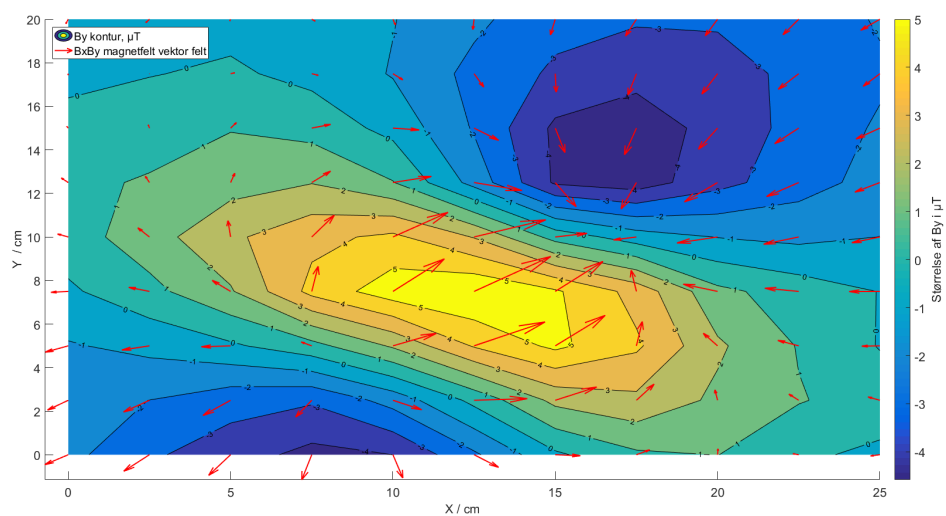
C. Opstilling til magnetfeltsmåling

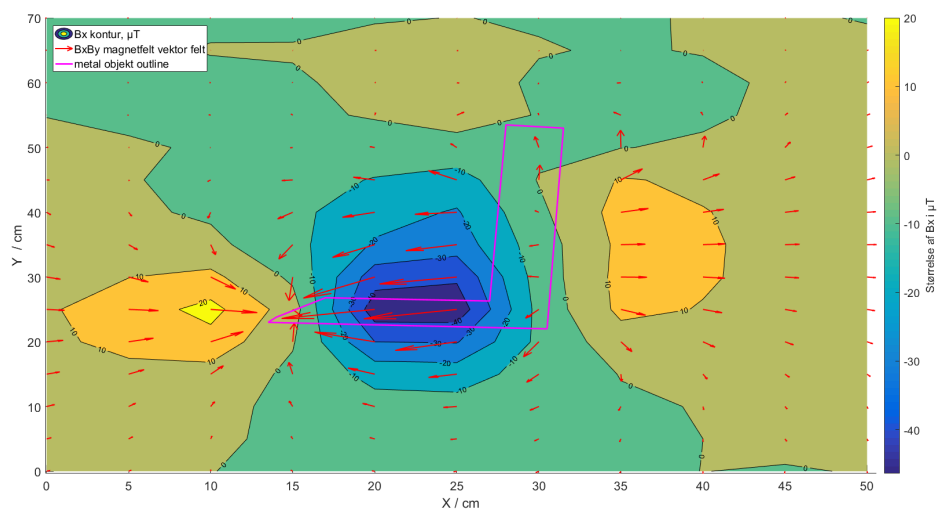


Figur 36: Jernvinklen tapet til bundet af bordet

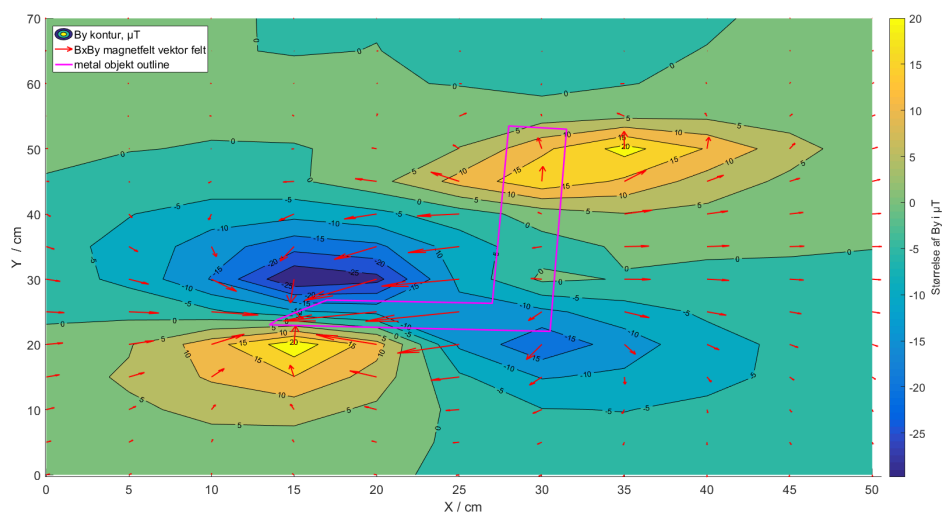


Figur 37: Måleopstilling med måler og sender forbundet til computeren

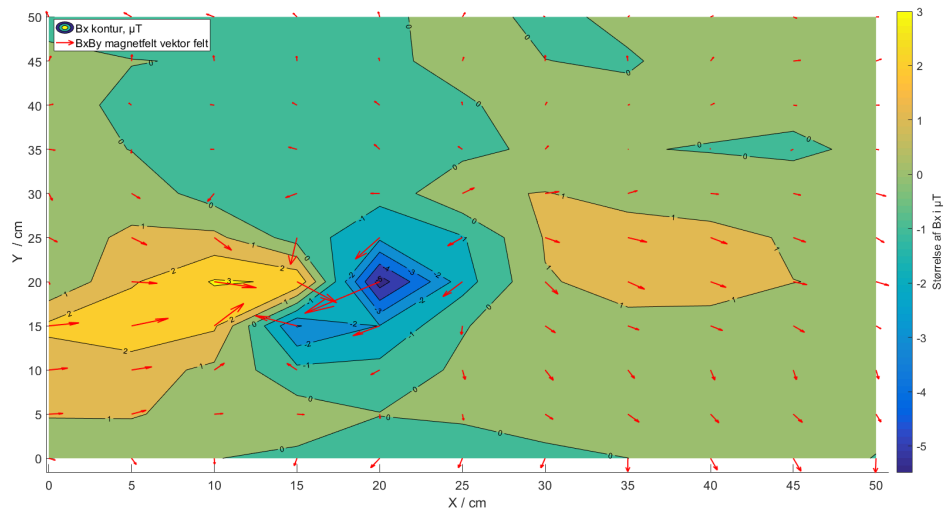
D. B_x og B_y konturer fra målingerFigur 38: Konturplot for B_x fra den lille skruetrækkerFigur 39: Konturplot for B_y fra den lille skruetrækker



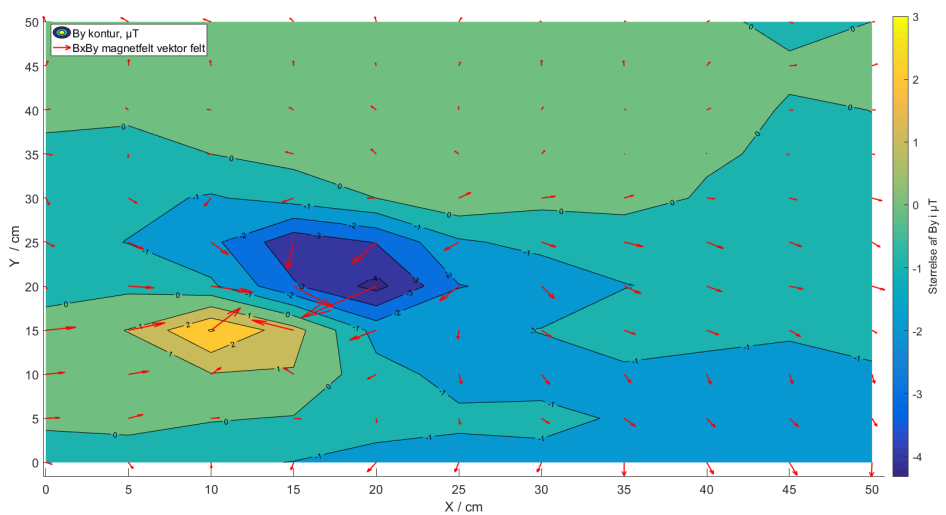
Figur 40: Konturplot for B_x for jernvinklen



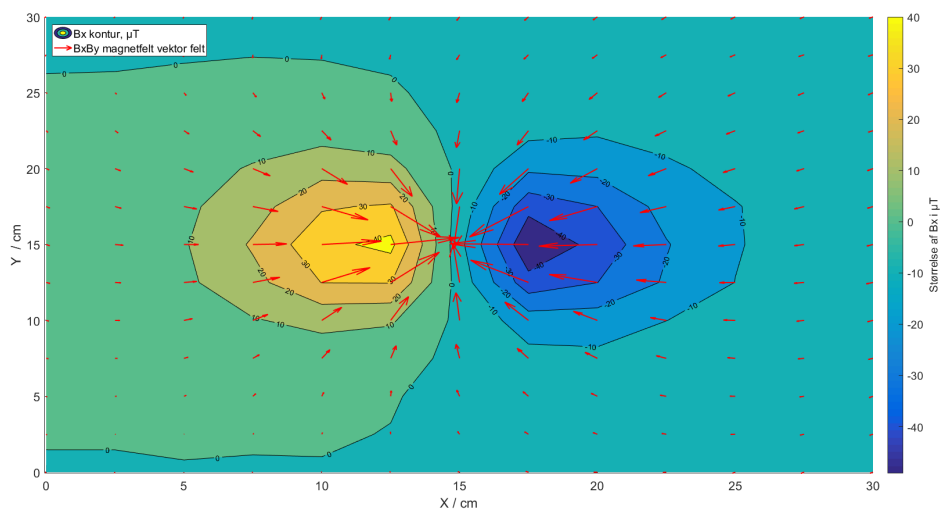
Figur 41: Konturplot for B_y for jernvinklen



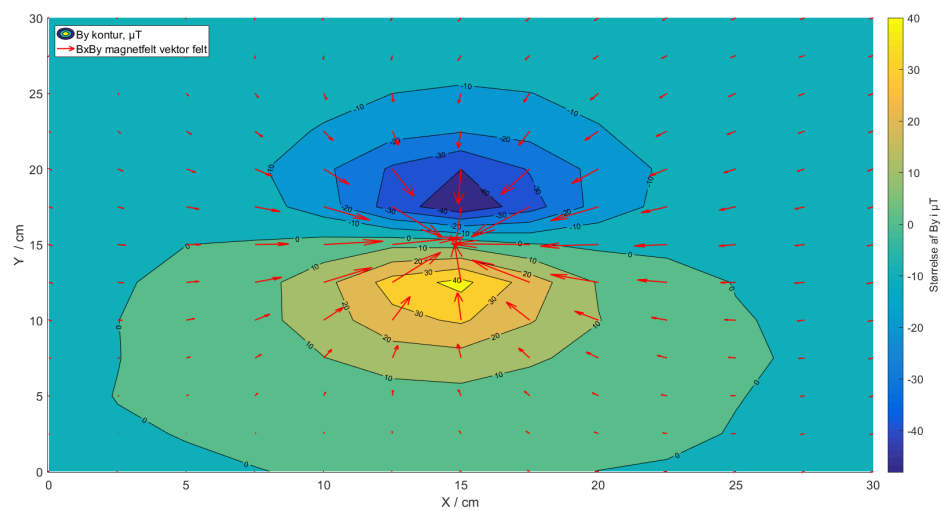
Figur 42: Konturplot for B_x for køleskabsmagneten



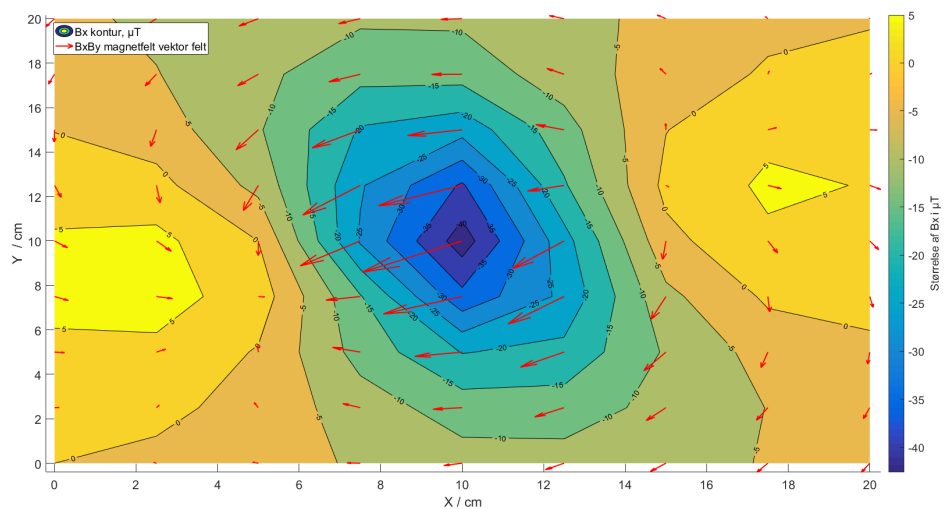
Figur 43: Konturplot for B_y for køleskabsmagneten



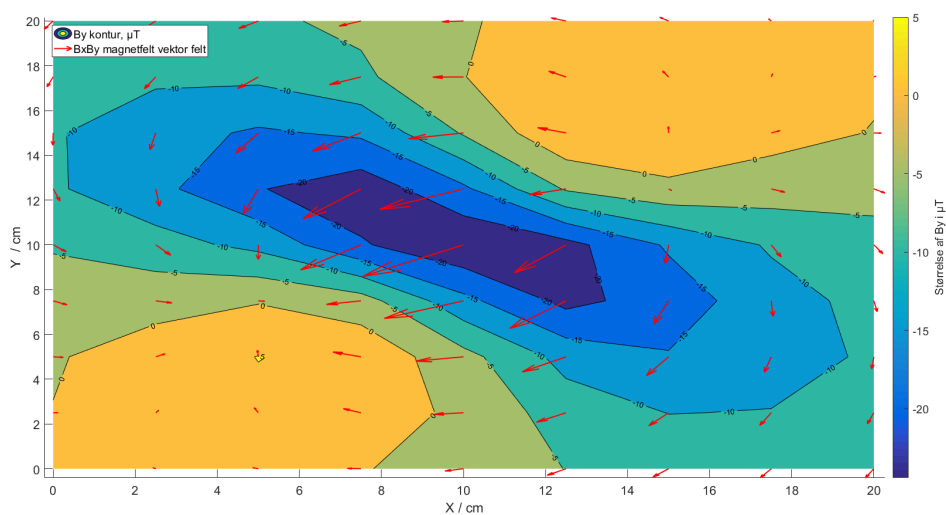
Figur 44: Konturplot for B_x for den lille magnet placeret fladt under bordet



Figur 45: Konturplot for B_y for den lille magnet placeret fladt under bordet



Figur 46: Konturplot for B_x for den lille magnet placeret på højkant under bordet



Figur 47: Konturplot for B_y for den lille magnet placeret på højkant under bordet

E. Tabeller for undersøgte komponenter

Tabel 4: Tabel over undersøgte μ -controller. Den grønmarkerede er den valgte

μ -controller	Interface	Spænding	Flash	Ram	Hz	Processor	Pins
Teensy LC	UART, SPI, I2C	3.3 V - 5V	62kB	8kB	48MHz	ARM M0+ 32 bit	27
Teensy 3.2	UART, SPI, I2C	3.3 V - 5V	256kB	64kB	72MHz	ARM M4 32 bit	27
Nucleo L031K6	UART, I2C, SPI	3.3V - 5V	32 kB	8 kB	32 MHz	ARM M0+ 32 bit	30
Arduino Micro	UART, I2C, SPI	5 V	32 kB	2.5 kB	16 MHz	ATmega32u4 8 bit	32
Pro Trinket	UART, I2C, SPI	3V / 5V	32 kB	2 kB	12 MHz	ATmega328 8/16 bit	18

Tabel 5: Tabel over undersøgte accelerometre. Det valgt er markeret med grøn

Accelerometer	Teknologi	Range	Opløsning	Forsyning	Strømforbrug	Interface	Akser
FXLS8471Q NXP	Ukendt	$\pm 2 / \pm 4 / \pm 8$	0.244 mg/LSB at ± 2	1.62 V - 3.6 V	3 mA boot, 10 μ A	SPI / I2C	3
ADXL335 Sparkfun	Kapacitiv	± 3 g	300 mV/g	1.8 V - 3.6 V	350 μ A	Analog	3
MMA8451 Adafruit	Kapacitiv	$\pm 2 / \pm 4 / \pm 8$	0.244 mg/LSB at ± 2	1.95 V - 3.6 V	1 mA boot, 10 μ A	I2C	3
MXR9500MZ	Termisk gas	± 1.5 g	500 mV/g	2.7 V - 3.6 V	4.2 mA	Analog	3
LIS331DLH	MEMS?	$\pm 2 / \pm 4 / \pm 8$	1mg/LSB at ± 2	2.16 V - 3.6 V	<250 μ A	I2C / SPI	3

Tabel 6: Tabel over de undersøgte magnetometre til instrumentet. Den grønne linje viser det valgte magnetometer

Magnetometer	Akser	Teknologi	Interval	Opløsning	Interface	Strømforbrug	Spænding
MAG3110 NXP digital	3	Magnetic Transducer	$\pm 1000 \mu\text{T}$	$0.1 \mu\text{T}$	I2C	$< 100 \mu\text{A}$	1.95 V - 3.6 V
HMC5883L	3	Magnetoresistiv	$\pm 800 \mu\text{T}$	$0.2 \mu\text{T}$	I2C	$100 \mu\text{A}$	2.16 V - 3.6 V
LIS3MDL STM	3	MEMS	$\pm 400 \mu\text{T}$ $\pm 800 \mu\text{T}$ $\pm 1200 \mu\text{T}$ $\pm 1600 \mu\text{T}$	$0.015 \mu\text{T}$ $0.03 \mu\text{T}$ $0.044 \mu\text{T}$ $0.06 \mu\text{T}$	I2C/SPI	$< 270 \mu\text{A}$	1.9 V - 3.6 V
Allegro A1319 analog	1	Hall Effekt			Analog		3.3 V
MEMSIC MMC3416xPJ	3	Magnetoresistiv	$\pm 1600 \mu\text{T}$	$0.2 \mu\text{T}$	Digital	$140 \mu\text{A}$	1.62 V - 3.6 V
Bartington MAG650	3	Fluxgate	$\pm 100 \mu\text{T}$	$25\text{mV}/\mu\text{T}$	Analog	2.5 mA	3 V - 4 V

Litteratur

- [6] David K. Cheng. *Field and Wave Electromagnetics*. 2nd Edition. Pearson, 1989.
- [7] “eNote 19 - Symmetriske Matricer”. I: (). DTU Matematik 1 (01005) materiale. URL: <http://01005.mat.dtu.dk/materialer/enoter/afsnit/NUID34-tn19/>.
- [8] Jacob Freedman. *Handbook of Modern Sensors - Physics, Designs, and Applications*. 4th Edition. Springer, 2010.
- [21] Hugh D. Young og Robert A. Freedman. *University Physics with Modern Physics - Volume One*. 13th Edition. Pearson, 2011.
- [22] Hugh D. Young og Robert A. Freedman. *University Physics with Modern Physics - Volume Two*. 13th Edition. Pearson, 2011.

Datablade

- [1] *0.5W Solar Panel*. Specificationer for solpanelet. Seedstudio. URL: http://wiki.seeed.cc/0.5w_Solar_Panel_55x70/.
- [4] *AN4509 - Tilt measurement using a low 3-axis accelerometer*. DocID026445 Rev 1. STMicroelectronics. Jun. 2014. URL: http://www.st.com/content/ccc/resource/technical/document/application_note/d2/d6/22/4f/b9/8f/45/59/DM00119046.pdf/files/DM00119046.pdf/jcr:content/translations/en.DM00119046.pdf.
- [5] *AN4602 - Application note - LIS3MDL*. DocID027014 Rev 1. STMicroelectronics. Dec. 2014. URL: http://www.st.com/content/ccc/resource/technical/document/application_note/69/d7/05/fe/a9/3e/4b/ea/DM00136626.pdf/files/DM00136626.pdf/jcr:content/translations/en.DM00136626.pdf.
- [9] Limor Fried. *Adafruit MMA8451Q Accelerometer Breakout*. Adafruit Industries. Jul. 2014. URL: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-mma8451-accelerometer-breakout.pdf>.
- [10] *Lipo Rider Pro*. Specificationer for LiPo Rider. Seedstudio. URL: http://wiki.seeed.cc/Lipo_Rider_Pro/.
- [11] *LIS3MDL - Digital Output Magnetic Sensor...* DocID024204 Rev 6. Maj 2017. URL: <http://www.st.com/content/ccc/resource/technical/document/datasheet/54/2a/85/76/e3/97/42/18/DM00075867.pdf/files/DM00075867.pdf/jcr:content/translations/en.DM00075867.pdf>.
- [12] *MMA8451Q 3-Axis 14-bit Digital Accelerometer*. MMA8451Q Rev. 8.1. Freescale Semiconductor. Okt. 2013. URL: <https://cdn-shop.adafruit.com/datasheets/MMA8451Q-1.pdf>.

- [13] *Modelbyggeri-batteripakke (LiPo) E-flite 3.7 V 150 mAh 45 C Minium*. Batterispecifikationer. URL: https://www.conradelektronik.dk/?websale8=conrad-dk&pi=758790&ws_tp1=cp&ref=PLA_Energi&subref=758790&utm_source=PLA_Energi&utm_medium=PLAfeed&utm_campaign=PLA_Energi_feed&utm_content=758790&gclid=CjwKEAjlw9_jJBRCXycSarr3csWcSJABthk07k2Vn14lFX7dqS39lHLBum7XQsPIa3m524XkTGoXucBoCj0_wcB.
- [14] *SPIRIT1*. DocID022758 Rev 10. STMicroelectronics. Okt. 2016. URL: <http://www.st.com/content/ccc/resource/technical/document/datasheet/68/6c/7b/ec/b2/6b/49/16/DM00047607.pdf/files/DM00047607.pdf/jcr:content/translations/en.DM00047607.pdf>.
- [16] *SPSGRF Sub-GHz low power programmable RF transceiver modules*. DocID027664 Rev 5. STMicroelectronics. Okt. 2016. URL: <http://www.st.com/content/ccc/resource/technical/document/datasheet/f9/a3/aa/67/de/86/4b/62/DM00171035.pdf/files/DM00171035.pdf/jcr:content/translations/en.DM00171035.pdf>.
- [17] *Teensy LC - Low Cost*. Overview af Teensy LC. PJRC. URL: <https://www.pjrc.com/teensy/teensyLC.html>.
- [20] *Wiring 128X64 OLED*. Note om brug af SSD1306 skærmen. URL: <https://learn.adafruit.com/monochrome-oled-breakouts/wiring-1-dot-3-128x64>.

Kode biblioteker

- [2] *Adafruit GFX.h*. Adafruit Industries. URL: <https://github.com/adafruit/Adafruit-GFX-Library>.
- [3] *Adafruit SSD1306 OLED Driver*. Adafruit Industries. URL: https://github.com/adafruit/Adafruit_SSD1306.
- [15] *SPIRIT1 Design Kit - STSW CONNECT009*. Vers. 2.2.1. Ikke et bibliotek, men et program. STMicroelectronics. URL: http://www.st.com/content/st_com/en/products/embedded-software/evaluation-tool-software/stsw-connect009.html.
- [18] K. Townsend. *Adafruit MMA8451.h*. Adafruit Industries. URL: https://github.com/adafruit/Adafruit_MMA8451_Library.
- [19] K. Townsend. *Adafruit Sensor.h*. Adafruit Industries. URL: https://github.com/adafruit/Adafruit_Sensor.

XI. Kode

A. C++ kode

Listing 1: Arduino(C++) kode til modtager enheden

```

1  /*****
2  *
3  *   Magnetmaaler modtager
4  *   Ananda Nielsen
5  *
6  *****/
7
8  #include <SPI.h>
9  #include <math.h>
10 #include <spirit1.h>
11 #include <Wire.h>
12 #include <Adafruit_GFX.h>
13 #include <Adafruit_SSD1306.h>
14
15 spirit1 wireless;
16
17 int w_length = 0;
18 uint8_t data[65];
19
20 float pitch, roll;
21
22 float mag_x, mag_y, mag_z, acc_x, acc_y, acc_z, true_x, true_y, true_z;
23 float avg_x = 0;
24 float avg_y = 0;
25 float avg_z = 0;
26
27 #define OLED_MOSI 11
28 #define OLED_CLK 13
29 #define OLED_DC 17
30 #define OLED_CS 10
31 #define OLED_RESET 16
32 Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);
33
34 #define NUMFLAKES 10

```

```
35 #define XPOS 0
36 #define YPOS 1
37 #define DELTAY 2
38
39 #if (SSD1306_LCDHEIGHT != 64)
40 #error("Height incorrect, please fix Adafruit_SSD1306.h!");
41 #endif
42
43 #define RX_INTERRUPT_PIN 15
44
45 #define BUTTON_PIN 21
46
47 byte RX = false;
48
49 bool OUTPUT_DATA = false;
50
51 int count = 0;
52
53 void print_data() {
54     OUTPUT_DATA = true;
55     //OUTPUT_DATA = !OUTPUT_DATA;
56 }
57
58 void RX_available() {
59     RX = true;
60 }
61
62 void setup(void) {
63     //Serial communication setup
64     Serial.begin(57600);
65     delay(1000);
66     Serial.println("Wireless Reciever!\n");
67
68
69     pinMode(RX_INTERRUPT_PIN, INPUT_PULLUP);
70     attachInterrupt(digitalPinToInterrupt(RX_INTERRUPT_PIN), RX_available, FALLING);
71     pinMode(BUTTON_PIN, INPUT_PULLDOWN);
72     attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), print_data, HIGH);
73
```

```
74 SPI1.begin();
75 delay(100);
76 wireless.initSpirit(RECEIVER);
77 delay(1000);
78 Serial.println("Flushing RX FIFO");
79 wireless.commandStrobe(CMD_FLUSH_RX_FIFO);
80 delay(1000);
81 wireless.printState();
82
83
84 // by default, we'll generate the high voltage from the 3.3v line internally! (
neat!)
85 display.begin(SSD1306_SWITCHCAPVCC);
86 display.display();
87 delay(1000);
88
89 // Clear the buffer.
90 display.clearDisplay();
91
92 display.setTextSize(1);
93 display.setTextColor(WHITE);
94 display.setCursor(10,15);
95 display.println("Magnetometer");
96 display.display();
97 delay(1000);
98
99 display.clearDisplay();
100
101 }
102
103 //udregn roll og pitch
104 void calcPitchRoll() {
105     pitch = atan( acc_x / sqrt( pow(acc_y, 2) + pow(acc_z, 2) ) ); // * 180/PI ;
106     roll = atan( acc_y / sqrt( pow(acc_x, 2) + pow(acc_z, 2) ) ); // * 180/PI;
107 }
108
109 //oversaet data fra spirit1
110 void decode_data(void){
111     uint8_t temp_data[4];
```



```
112 memcpy(&acc_x , data , 4);
113
114 for(int i = 0; i < 4; i++) {
115     temp_data[i] = data[i+4];
116 }
117 memcpy(&acc_y , temp_data , 4);
118
119 for(int i = 0; i < 4; i++) {
120     temp_data[i] = data[i+8];
121 }
122 memcpy(&acc_z , temp_data , 4);
123
124 for(int i = 0; i < 4; i++) {
125     temp_data[i] = data[i+12];
126 }
127 memcpy(&mag_x , temp_data , 4);
128
129 for(int i = 0; i < 4; i++) {
130     temp_data[i] = data[i+16];
131 }
132 memcpy(&mag_y , temp_data , 4);
133
134 for(int i = 0; i < 4; i++) {
135     temp_data[i] = data[i+20];
136 }
137 memcpy(&mag_z , temp_data , 4);
138
139 }
140
141 //Print paa skaerm
142 void screen_data() {
143     display.setTextSize(1);
144     display.setCursor(0,0);
145     display.println("Magnetic Field");
146     display.print("X: "); display.print(true_x*pow(10,5),3); display.println(" nT");
147     display.print("Y: "); display.print(true_y*pow(10,5),3); display.println(" nT");
148     display.print("Z: "); display.print(true_z*pow(10,5),3); display.println(" nT");
149     display.println();
150     display.println("Orientation");
```

```

151   display.print("Roll: "); display.println(roll*180/PI,5);
152   display.print("Pitch: "); display.println(pitch*180/PI,5);
153   display.display();
154 }
155
156 // funktion der kalibrer data
157 void calibrate_data() {
158     acc_x = 1.011089 * acc_x + 0.014918;
159     acc_y = 1.008408 * acc_y - 0.013774;
160     acc_z = 1.006251 * acc_z - 0.010126;
161
162     mag_x = 0.973036 * mag_x - 0.0119326;
163     mag_y = 1.017132 * mag_y + 0.0912794;
164     mag_z = 0.978729 * mag_z - 0.3547089;
165
166     float temp = mag_x;
167     mag_x = -mag_y;
168     mag_y = temp;
169 }
170
171 //funktion til at rotere vektorene
172 void rotation() {
173     calcPitchRoll();
174     true_x = cos(pitch)*mag_x - sin(pitch)*sin(roll)*mag_y - sin(pitch)*cos(roll)*
        mag_z;
175     true_y = cos(roll)*mag_y - sin(roll)*mag_z;
176     true_z = sin(pitch)*mag_x + cos(pitch)*sin(roll)*mag_y + cos(pitch)*cos(roll)*
        mag_z;
177 }
178
179
180 void loop() {
181
182     //bliv i RX tilstand
183     if(wireless.getState() == STATE_READY){
184         wireless.commandStrobe(CMD_RX);
185     }
186
187     w_length = wireless.readRegister(SPIRIT_LINEAR_FIFO_STATUSRX) & 0x7F;

```

```
188
189 //hele pakken er modtaget
190 if(w.length == 24 ){
191     wireless.empty_RX_FIFO(data);
192     RX = false;
193     decode_data();
194     calibrate_data();
195     rotation();
196     if(OUTPUT_DATA&&0){
197         OUTPUT_DATA = true;
198         count++;
199         avg_x += true_x;
200         avg_y += true_y;
201         avg_z += true_z;
202         if(count == 50) {
203             avg_x /= 50;
204             avg_y /= 50;
205             avg_z /= 50;
206             Serial.print(avg_x,20); Serial.print(",\t");
207             Serial.print(avg_y,20); Serial.print(",\t");
208             Serial.print(avg_z,20); Serial.println();
209             avg_x = 0;
210             avg_y = 0;
211             avg_z = 0;
212             count = 0;
213             OUTPUT_DATA = false;
214         }
215     }
216 // print al data naar det modtages
217 if(1){
218     Serial.print(acc_x,15); Serial.print(",\t");
219     Serial.print(acc_y,15); Serial.print(",\t");
220     Serial.print(acc_z,15); Serial.print(",\t");
221     Serial.print(mag_x,20); Serial.print(",\t");
222     Serial.print(mag_y,20); Serial.print(",\t");
223     Serial.print(mag_z,20); Serial.print(",\t");
224     Serial.print(true_x,20); Serial.print(",\t");
225     Serial.print(true_y,20); Serial.print(",\t");
226     Serial.print(true_z,20);
```

```

227     Serial.println();
228     count++;
229 }
230
231 else {
232 }
233 display.clearDisplay();
234 screen_data();
235 }
236 else if( w_length > 24 ){
237     wireless.commandStrobe(CMD_FLUSH_RX_FIFO);
238     Serial.println("FIFO flushed");
239 }
240 }

```

Listing 2: Arduino(C++) kode til måle-/sendeenheden

```

1  /*****
2  *
3  *   Magnetmaaler sender og sensor styring
4  *   Ananda Nielsen
5  *
6  *****/
7
8  // hardware/functional packages
9  #include <Wire.h>
10 #include <SPI.h>
11 #include <math.h>
12
13 // sensor packages
14 #include <Adafruit_MMA8451.h>
15 #include <Adafruit_Sensor.h>
16 #include <lis3mdl.h>
17 #include <spirit1.h>
18
19 SPISettings SPI1TEST(2000000, MSBFIRST, SPI_MODE3);
20
21 #define MAGDRDY 21
22 #define ACCDRDY 22
23 #define TX_trans 15

```

```
24
25 //sensor objects
26 spirit1 wireless;
27 lis3mdl mag;
28 Adafruit_MMA8451 mma = Adafruit_MMA8451();
29
30 SPISettings SPIRIT1(1000000, MSBFIRST, SPI_MODE0);
31
32 uint8_t data_2_transmit[24];
33
34 int count = 0;
35
36 byte MAG_RDY = false;
37
38 byte ACC_RDY = false;
39
40 byte TX = false;
41
42 byte MAG_RDY_2_SEND = false;
43
44 byte ACC_RDY_2_SEND = false;
45
46 void ACC_available() {
47     ACC_RDY = true;
48 }
49
50 void MAG_available() {
51     MAG_RDY = true;
52 }
53
54 void TX_Done() {
55     TX = true;
56 }
57
58 void setup(void) {
59     //Serial communication setup
60     Serial.begin(57600);
61     pinMode(MAG_DRDY, INPUT_PULLDOWN);
62     attachInterrupt(digitalPinToInterrupt(MAG_DRDY), MAG_available, RISING);
```

```

63 pinMode(ACC_DRDY, INPUT_PULLDOWN);
64 attachInterrupt(digitalPinToInterrupt(ACC_DRDY), ACC_available, RISING);
65 pinMode(TX_trans, INPUT_PULLUP);
66 attachInterrupt(digitalPinToInterrupt(TX_trans), TX_Done, FALLING);
67
68 delay(1000);
69 Serial.println("All sensor instrument!\n");
70
71 //SPIRIT 1 setup
72 SPI1.begin();
73 delay(100);
74 wireless.initSpirit(TRANSMITTER);
75 Serial.println("SPIRIT1 Init done\n");
76
77 //MMA8451 sensor setup
78 if (!mma.begin()) {
79     Serial.println("Couldnt start MMA8451");
80 }
81 Serial.println("MMA8451 found!");
82
83 //LIS3MDL Setup
84 SPI1.begin();
85 mag.init_magnetometer();
86
87 delay(100);
88 Serial.println("Here we go!\n!");
89 }
90
91 void transmit_all_wireless() {
92     uint8_t temp_bytes[4];
93
94     memcpy(temp_bytes, &mma.x_g, 4);
95     for(int i = 0; i < 4; i++) {
96         data_2_transmit[i] = temp_bytes[i];
97     }
98
99     memcpy(temp_bytes, &mma.y_g, 4);
100    for(int i = 0; i < 4; i++) {
101        data_2_transmit[i+4] = temp_bytes[i];

```

```

102 }
103
104 memcpy(temp_bytes, &mma.z_g, 4);
105 for(int i = 0; i < 4; i++) {
106     data_2_transmit[i+8] = temp_bytes[i];
107 }
108
109 memcpy(temp_bytes, &mag.x, 4);
110 for(int i = 0; i < 4; i++) {
111     data_2_transmit[i+12] = temp_bytes[i];
112 }
113
114 memcpy(temp_bytes, &mag.y, 4);
115 for(int i = 0; i < 4; i++) {
116     data_2_transmit[i+16] = temp_bytes[i];
117 }
118
119 memcpy(temp_bytes, &mag.z, 4);
120 for(int i = 0; i < 4; i++) {
121     data_2_transmit[i+20] = temp_bytes[i];
122 }
123
124 wireless.transmit_data(data_2_transmit, 24);
125
126 }
127
128 void loop() {
129     count++;
130     if(count > 300000) {
131         mma.read();
132         Serial.println("MMA8451");
133         mag.getVectorField();
134         Serial.println("LIS3MDL");
135         rotation();
136         Serial.print("Pitch: "); Serial.print(pitch*180/M_PI);
137         Serial.print("\tRoll: "); Serial.println(roll*180/M_PI);
138         Serial.print("X: "); Serial.print(mag.x,4);
139         Serial.print("\tY: "); Serial.print(mag.y,4);
140         Serial.print("\tZ: "); Serial.println(mag.z,4);

```

```
141     Serial.print("X: "); Serial.print(true_x,4);
142     Serial.print("\tY: "); Serial.print(true_y,4);
143     Serial.print("\tZ: "); Serial.println(true_z,4);
144     count = 0;
145     Serial.println("TIMEOUT\n");
146 }
147
148 if(MAG_RDY){
149     Serial.println("mag int");
150     mag.getVectorField();
151     mma.read();
152     MAG_RDY_2_SEND = true;
153     MAG_RDY = false;
154 }
155
156 if(ACC_RDY) {
157     Serial.println("acc int");
158     mma.read();
159     ACC_RDY_2_SEND = true;
160     ACC_RDY = false;
161 }
162
163 if(MAG_RDY_2_SEND) {
164     Serial.println("Sending Data");
165     wireless.printState();
166     transmit_all_wireless();
167     ACC_RDY_2_SEND = false;
168     MAG_RDY_2_SEND = false;
169     count = 0;
170 }
171
172 if(TX) {
173     Serial.println("Data transmitted");
174     TX = false;
175     wireless.commandStrobe(CMD_READY);
176     wireless.printState();
177 }
178 }
```


Listing 3: C++ koden til SPSGRF(SPIRIT1) biblioteket

```

1  /*****
2  *
3  *   @author:   Ananda Nielsen
4  *
5  *   Driver for ST SPIRIT1 SPSGRF wireless chip
6  *   using SPI
7  *
8  *****/
9
10 #include "SPI.h"
11 #include <spirit1.h>
12
13
14 SPISettings SPIRIT1_SPI(2000000, MSBFIRST, SPI_MODE0);
15
16 /*****
17 * @brief   constructor
18 * @param   sensor id value
19 * @return  none
20 *****/
21 spirit1::spirit1(uint8_t sensor_id){
22     _sensorID = sensor_id;
23 }
24
25
26 /*****
27 * @brief   initialize the hardware and registers of the spirit 1
28 * @param   none
29 * @return  none
30 *****/
31 void spirit1::initSpirit( byte type ) {
32     //Serial.println(_sensorID, HEX);
33     pinMode(SPIRIT_CS, OUTPUT);
34     pinMode(SPIRIT_SDN, OUTPUT);
35     SPIRIT_CS_HIGH;
36     SPIRIT_ON;
37     delay(100);
38     int value = ( readRegister(SPIRIT_DEVICE_INFO1) | readRegister(

```

```

    SPIRIT_DEVICE_INFO0) );
39 //Serial.println(value);
40 if(_sensorID != value) {
41     Serial.println("SPIRIT1 Not Connected/Found on SPI!");
42 }
43 else{
44     Serial.println("Writing setup registers!\n");
45
46     //BASIC
47     writeRegisters(SPIRIT_PCKTCTRL4 , 0x00); //0 control 0 addr length
48     writeRegisters(SPIRIT_PCKTCTRL3 , 0x04); //basic pack normal rx 5 bit length
49     writeRegisters(SPIRIT_PCKTCTRL2 , 0x1E); // 3 preamble 4 sync fix length
50     writeRegisters(SPIRIT_PCKTCTRL1 , 0x30); //normal tx 0x07 CRC whitening
51     writeRegisters(SPIRIT_PCKTLEN1, 0x00);
52     writeRegisters(SPIRIT_PCKTLEN0, 0x18);
53     writeRegisters(SPIRIT_IF_OFFSET_ANA, 0x36);
54     writeRegisters(SPIRIT_SYNT3, 0x06);
55     writeRegisters(SPIRIT_SYNT2, 0x82);
56     writeRegisters(SPIRIT_SYNT1, 0x8F);
57     writeRegisters(SPIRIT_SYNT0, 0x59);
58     writeRegisters(SPIRIT_CHSPACE, 0x01);
59     writeRegisters(SPIRIT_IF_OFFSET_DIG, 0xAC);
60     writeRegisters(SPIRIT_PA_POWER8, 0x17);
61     writeRegisters(SPIRIT_MOD1, 0x93);
62     writeRegisters(SPIRIT_CHFLT, 0x13);
63     writeRegisters(SPIRIT_AFC2, 0xC8);
64     writeRegisters(SPIRIT_AGCTRL1, 0x62);
65     writeRegisters(SPIRIT_ANT_SELECT_CONF, 0x15);
66     writeRegisters(SPIRIT_PCKT_FLT_OPTIONS, 0x41);
67     writeRegisters(SPIRIT_PROTOCOL2, 0x42);
68     writeRegisters(SPIRIT_PROTOCOL1, 0x01);
69     writeRegisters(SPIRIT_PROTOCOL0, 0x0C);
70     writeRegisters(SPIRIT_SYNTH_CONFIG0, 0xA0);
71     writeRegisters(SPIRIT_VCO_CONFIG, 0x25);
72     writeRegisters(SPIRIT_VCO_CONFIG, 0x35);
73     writeRegisters(SPIRIT_PA_POWER8, 0x01); //max dBm for TX
74     writeRegisters(SPIRIT_PA_POWER7, 0x00);
75     writeRegisters(SPIRIT_PA_POWER6, 0x00);
76     writeRegisters(SPIRIT_PA_POWER5, 0x00);

```

```

77     writeRegisters(SPIRIT_PA_POWER4, 0x00);
78     writeRegisters(SPIRIT_PA_POWER3, 0x00);
79     writeRegisters(SPIRIT_PA_POWER2, 0x00);
80     writeRegisters(SPIRIT_PA_POWER1, 0x00);
81     writeRegisters(SPIRIT_PA_POWER0, 0x07);
82
83
84
85     if(type == TRANSMITTER) {
86         writeRegisters(SPIRIT_GPIO3_CONF, 0x03);
87         writeRegisters(SPIRIT_IRQ_MASK0, 0x04);
88         writeRegisters(SPIRIT_IRQ_MASK1, 0x00);
89         writeRegisters(SPIRIT_IRQ_MASK2, 0x00);
90         writeRegisters(SPIRIT_IRQ_MASK3, 0x00);
91     }
92
93     if(type == RECEIVER) {
94         writeRegisters(SPIRIT_GPIO3_CONF, 0x03);
95         writeRegisters(SPIRIT_IRQ_MASK0, 0x01);
96         writeRegisters(SPIRIT_IRQ_MASK1, 0x00);
97         writeRegisters(SPIRIT_IRQ_MASK2, 0x00);
98         writeRegisters(SPIRIT_IRQ_MASK3, 0x00);
99     }
100
101 }
102 }
103
104
105
106 /*****
107  * @brief send a command strobe
108  * @param command to send
109  * @return none
110  *****/
111 void spirit1::commandStrobe( uint8_t cmd ) {
112     SPI1.beginTransaction(SPIRIT1_SPI);
113     SPIRIT_CS_LOW;
114     delayMicroseconds(2);
115     SPI1.transfer(0x80);

```

```

116     SPI1.transfer(cmd);
117     SPIRIT_CS_HIGH;
118     SPI1.endTransaction();
119 }
120
121
122 /* *****
123 * @brief    read the value of a specific register
124 * @param    register address
125 * @return   value from register
126 ***** */
127 uint8_t spirit1::readRegister( uint8_t reg ) {
128     uint8_t read;
129     SPI1.beginTransaction(SPIRIT1_SPI);
130     SPIRIT_CS_LOW;
131     delayMicroseconds(2);
132     status[1] = SPI1.transfer(0x01);
133     status[0] = SPI1.transfer(reg);
134     read = SPI1.transfer(0);
135     SPIRIT_CS_HIGH;
136     SPI1.endTransaction();
137     return read;
138 }
139
140 /* *****
141 * @brief    empty the rx fifo
142 * @param    data array
143 * @return   the length of the fifo
144 ***** */
145 int spirit1::empty_RX_FIFO( uint8_t * data ) {
146     int length = 0;
147     length = (readRegister(SPIRIT_LINEAR_FIFO_STATUSRX) & 0x7F);
148
149     SPI1.beginTransaction(SPIRIT1_SPI);
150     SPIRIT_CS_LOW;
151     delayMicroseconds(2);
152     SPI1.transfer(0x01);
153     SPI1.transfer(SPIRIT_FIFO);
154     for( int i = 0; i < length; i++ ) {

```

```

155     data[i] = SPI1.transfer(0);
156 }
157 SPIRIT_CS_HIGH;
158 SPI1.endTransaction();
159
160 return length;
161 }
162
163
164 /*****
165  * @brief transmit data over wireless
166  * @param data to send
167  * @return none
168  *****/
169 void spirit1::transmit_data( uint8_t * data, int length ) {
170     commandStrobe(CMD_FLUSHTXFIFO);
171     SPI1.beginTransaction(SPIRIT1_SPI);
172     SPIRIT_CS_LOW;
173     delayMicroseconds(2);
174     SPI1.transfer(0x00);
175     SPI1.transfer(SPIRIT_FIFO);
176     for(int i = 0; i < length; i++ ) {
177         SPI1.transfer(data[i]);
178     }
179     SPIRIT_CS_HIGH;
180     SPI1.endTransaction();
181     if( getState() == STATE_READY ) {
182         commandStrobe(CMD_READY);
183         delay(1);
184     }
185     commandStrobe(CMD_TX);
186     delay(2);
187     while( readRegister(SPIRIT_LINEAR_FIFO_STATUSTX) > 0 )
188     {
189
190     }
191     commandStrobe(CMD_READY);
192 }
193

```

```

194
195 /*****
196 * @brief write the value to a register
197 * @param register address, value to write
198 * @return none
199 *****/
200 void spirit1::writeRegisters( uint8_t reg, uint8_t value ) {
201     SPI1.beginTransaction( SPIRIT1_SPI );
202     SPIRIT_CS_LOW;
203     delayMicroseconds(2);
204     status[1] = SPI1.transfer(0x00);
205     status[0] = SPI1.transfer(reg);
206     SPI1.transfer(value);
207     SPIRIT_CS_HIGH;
208     SPI1.endTransaction();
209 }
210
211 /*****
212 * @brief returns the current state
213 * @param none
214 * @return the current state (or 0)
215 *****/
216 uint8_t spirit1::getState() {
217     if( readRegister(SPIRIT_DEVICE_INFO1) == ID1 ){
218         return (( status[0] >> 1 )&0x7F);
219     }
220     else{
221         Serial.println("Not available or shutdown");
222     }
223     return 0;
224 }
225
226
227 void spirit1::printState() {
228     uint8_t state = getState();
229     switch(state) {
230         case STATE_STANDBY:
231             Serial.println("\nState = Standby\n");
232             break;

```

```

233     case STATE_READY:
234         Serial.println("\nState = Ready\n");
235         break;
236     case STATE_SLEEP:
237         Serial.println("\nState = Sleep\n");
238         break;
239     case STATE_LOCK:
240         Serial.println("\nState = Lock\n");
241         break;
242     case STATE_RX:
243         Serial.println("\nState = RX\n");
244         break;
245     case STATE_TX:
246         Serial.println("\nState = TX\n");
247         break;
248     case STATE_LOCKWON:
249         Serial.println("\nState = LOCKWON\n");
250         break;
251     default:
252         Serial.println("\nNO VALID STATE!\n");
253         break;
254
255 }
256 }

```

Listing 4: h fil til SPSGRF(SPIRIT1) biblioteket

```

1  /*****
2  *
3  *   @author:   Ananda Nielsen
4  *
5  *   Driver for ST SPIRIT1 SPSGRF wireless chip
6  *   using SPI
7  *
8  *****/
9
10 #ifndef _SPIRIT1_H_
11 #define _SPIRIT1_H_
12
13 #include <Arduino.h>

```

```

14 #include <SPI.h>
15
16 //used pins on teensy
17 #define SPIRIT_CS 6
18 #define SPIRIT_SDN 2
19
20 //digital pin control macros
21 #define SPIRIT_CS_LOW digitalWrite(SPIRIT_CS, LOW)
22 #define SPIRIT_CS_HIGH digitalWrite(SPIRIT_CS, HIGH)
23 #define SPIRIT_ON digitalWrite(SPIRIT_SDN, LOW)
24 #define SPIRIT_OFF digitalWrite(SPIRIT_SDN, HIGH)
25
26 //states
27 #define STATE_STANDBY 0x40
28 #define STATE_SLEEP 0x36
29 #define STATE_READY 0x03
30 #define STATE_LOCK 0x0F
31 #define STATE_RX 0x33
32 #define STATE_TX 0x5F
33 #define STATE_LOCKWON 0x13
34
35 //command strobes
36 #define CMD_TX 0x60
37 #define CMD_RX 0x61
38 #define CMD_READY 0x62
39 #define CMD_STANDBY 0x63
40 #define CMD_SLEEP 0x64
41 #define CMD_LOCKRX 0x65
42 #define CMD_LOCKTX 0x66
43 #define CMD_SABORT 0x67
44 #define CMD_LDC_RELOAD 0x68
45 #define CMD_SEQUENCE_UPDATE 0x69
46 #define CMD_AES_ENC 0x6A
47 #define CMD_AES_KEY 0x6B
48 #define CMD_AES_DEC 0x6C
49 #define CMD_AES_KEYDEC 0x6D
50 #define CMD_SRES 0x70
51 #define CMD_FLUSH_RX_FIFO 0x71
52 #define CMD_FLUSH_TX_FIFO 0x72

```



```

53
54 //general configuration registers
55 #define SPIRIT_ANA_FUNC_CONF1    0x00
56 #define SPIRIT_ANA_FUNC_CONF0    0x01
57 #define SPIRIT_GPIO3_CONF        0x02
58 #define SPIRIT_GPIO2_CONF        0x03
59 #define SPIRIT_GPIO1_CONF        0x04
60 #define SPIRIT_GPIO0_CONF        0x05
61 #define SPIRIT_MCU_CK_CONF        0x06
62 #define SPIRIT_XO_RCO_TEST        0xB4
63 #define SPIRIT_SYNTH_CONFIG0      0x9F
64 #define SPIRIT_SYNTH_CONFIG1      0x9E
65 #define SPIRIT_IF_OFFSET_ANA      0x07
66
67 //radio configuration registers analog block
68 #define SPIRIT_SYNT3              0x08
69 #define SPIRIT_SYNT2              0x09
70 #define SPIRIT_SYNT1              0x0A
71 #define SPIRIT_SYNT0              0x0B
72 #define SPIRIT_CHSPACE            0x0C
73 #define SPIRIT_IF_OFFSET_DIG      0x0D
74 #define SPIRIT_FC_OFFSET1         0x0E
75 #define SPIRIT_FC_OFFSET0         0x0F
76 #define SPIRIT_PA_POWER8          0x10
77 #define SPIRIT_PA_POWER7          0x11
78 #define SPIRIT_PA_POWER6          0x12
79 #define SPIRIT_PA_POWER5          0x13
80 #define SPIRIT_PA_POWER4          0x14
81 #define SPIRIT_PA_POWER3          0x15
82 #define SPIRIT_PA_POWER2          0x16
83 #define SPIRIT_PA_POWER1          0x17
84 #define SPIRIT_PA_POWER0          0x18
85
86 //radio configuration registers digital blocks
87 #define SPIRIT_MOD1               0x1A
88 #define SPIRIT_MOD0               0x1B
89 #define SPIRIT_FDEV0              0x1C
90 #define SPIRIT_CHFLT              0x1D
91 #define SPIRIT_AFC2               0x1E

```

```

92 #define SPIRIT_AFC1          0x1F
93 #define SPIRIT_AFC0          0x20
94 #define SPIRIT_RSSI_FLT      0x21
95 #define SPIRIT_RSSI_TH       0x22
96 #define SPIRIT_CLOCKREC      0x23
97 #define SPIRIT_AGCTRL2       0x24
98 #define SPIRIT_AGCTRL1       0x25
99 #define SPIRIT_AGCTRL0       0x26
100 #define SPIRIT_ANT_SELECT_CONF 0x27
101
102 //packet/protocol configuration registers
103 #define SPIRIT_PCKTCTRL4      0x30
104 #define SPIRIT_PCKTCTRL3      0x31
105 #define SPIRIT_PCKTCTRL2      0x32
106 #define SPIRIT_PCKTCTRL1      0x33
107 #define SPIRIT_PCKTLEN1       0x34
108 #define SPIRIT_PCKTLEN0       0x35
109 #define SPIRIT_SYNC4          0x36
110 #define SPIRIT_SYNC3          0x37
111 #define SPIRIT_SYNC2          0x38
112 #define SPIRIT_SYNC1          0x39
113 #define SPIRIT_QI             0x3A
114 #define SPIRIT_MBUS_PRMBL     0x3B
115 #define SPIRIT_MBUS_PSTMBL     0x3C
116 #define SPIRIT_MBUS_CTRL      0x3D
117 #define SPIRIT_FIFO_CONFIG3    0x3E
118 #define SPIRIT_FIFO_CONFIG2    0x3F
119 #define SPIRIT_FIFO_CONFIG1    0x40
120 #define SPIRIT_FIFO_CONFIG0    0x41
121 #define SPIRIT_PCKT_FLT_GOALS12 0x42
122 #define SPIRIT_PCKT_FLT_GOALS11 0x43
123 #define SPIRIT_PCKT_FLT_GOALS10 0x44
124 #define SPIRIT_PCKT_FLT_GOALS9 0x45
125 #define SPIRIT_PCKT_FLT_GOALS8 0x46
126 #define SPIRIT_PCKT_FLT_GOALS7 0x47
127 #define SPIRIT_PCKT_FLT_GOALS6 0x48
128 #define SPIRIT_PCKT_FLT_GOALS5 0x49
129 #define SPIRIT_PCKT_FLT_GOALS4 0x4A
130 #define SPIRIT_PCKT_FLT_GOALS3 0x4B

```

```

131 #define SPIRIT_PCKT_FLT_GOALS2 0x4C
132 #define SPIRIT_PCKT_FLT_GOALS1 0x4D
133 #define SPIRIT_PCKT_FLT_GOALS0 0x4E
134 #define SPIRIT_PCKT_FLT_OPTIONS 0x4F
135 #define SPIRIT_PROTOCOL2 0x50
136 #define SPIRIT_PROTOCOL1 0x51
137 #define SPIRIT_PROTOCOL0 0x52
138 #define SPIRIT_TIMERS5 0x53
139 #define SPIRIT_TIMERS4 0x54
140 #define SPIRIT_TIMERS3 0x55
141 #define SPIRIT_TIMERS2 0x56
142 #define SPIRIT_TIMERS1 0x57
143 #define SPIRIT_TIMERS0 0x58
144 #define SPIRIT_CSMA_CONFIG3 0x64
145 #define SPIRIT_CSMA_CONFIG2 0x65
146 #define SPIRIT_CSMA_CONFIG1 0x66
147 #define SPIRIT_CSMA_CONFIG0 0x67
148 #define SPIRIT_TX_CTRL_FIELD3 0x68
149 #define SPIRIT_TX_CTRL_FIELD2 0x69
150 #define SPIRIT_TX_CTRL_FIELD1 0x6A
151 #define SPIRIT_TX_CTRL_FIELD0 0x6B
152 #define SPIRIT_PM_CONFIG2 0xA4
153 #define SPIRIT_PM_CONFIG1 0xA5
154 #define SPIRIT_PM_CONFIG0 0xA6
155 #define SPIRIT_XO_RCO_CONFIG 0xA7
156 #define SPIRIT_TEST_SELECT 0xA8
157 #define SPIRIT_PM_TEST 0xB2
158 #define SPIRIT_CHNUM 0x6C
159 #define SPIRIT_VCO_CONFIG 0xA1
160 #define SPIRIT_RCO_VCO_CALIBR_IN2 0x6D
161 #define SPIRIT_RCO_VCO_CALIBR_IN1 0x6E
162 #define SPIRIT_RCO_VCO_CALIBR_IN0 0x6F
163 #define SPIRIT_AES_KEY_IN15 0x70
164 #define SPIRIT_AES_KEY_IN14 0x71
165 #define SPIRIT_AES_KEY_IN13 0x72
166 #define SPIRIT_AES_KEY_IN12 0x73
167 #define SPIRIT_AES_KEY_IN11 0x74
168 #define SPIRIT_AES_KEY_IN10 0x75
169 #define SPIRIT_AES_KEY_IN9 0x76

```

```

170 #define SPIRIT_AES_KEY_IN8      0x77
171 #define SPIRIT_AES_KEY_IN7      0x78
172 #define SPIRIT_AES_KEY_IN6      0x79
173 #define SPIRIT_AES_KEY_IN5      0x7A
174 #define SPIRIT_AES_KEY_IN4      0x7B
175 #define SPIRIT_AES_KEY_IN3      0x7C
176 #define SPIRIT_AES_KEY_IN2      0x7D
177 #define SPIRIT_AES_KEY_IN1      0x7E
178 #define SPIRIT_AES_KEY_IN0      0x7F
179 #define SPIRIT_AES_DATA_IN15    0x80
180 #define SPIRIT_AES_DATA_IN14    0x81
181 #define SPIRIT_AES_DATA_IN13    0x82
182 #define SPIRIT_AES_DATA_IN12    0x83
183 #define SPIRIT_AES_DATA_IN11    0x84
184 #define SPIRIT_AES_DATA_IN10    0x85
185 #define SPIRIT_AES_DATA_IN9     0x86
186 #define SPIRIT_AES_DATA_IN8     0x87
187 #define SPIRIT_AES_DATA_IN7     0x88
188 #define SPIRIT_AES_DATA_IN6     0x89
189 #define SPIRIT_AES_DATA_IN5     0x8A
190 #define SPIRIT_AES_DATA_IN4     0x8B
191 #define SPIRIT_AES_DATA_IN3     0x8C
192 #define SPIRIT_AES_DATA_IN2     0x8D
193 #define SPIRIT_AES_DATA_IN1     0x8E
194 #define SPIRIT_AES_DATA_IN0     0x8F
195 #define SPIRIT_IRQ_MASK3        0x90
196 #define SPIRIT_IRQ_MASK2        0x91
197 #define SPIRIT_IRQ_MASK1        0x92
198 #define SPIRIT_IRQ_MASK0        0x93
199 #define SPIRIT_DEM_CONFIG       0xA3
200 #define SPIRIT_PM_CONFIG        0xA4
201 #define SPIRIT_MC_STATE1        0xC0
202 #define SPIRIT_MC_STATE0        0xC1
203 #define SPIRIT_TX_PCKT_INFO     0xC2
204 #define SPIRIT_RX_PCKT_INFO     0xC3
205 #define SPIRIT_AFC_CORR         0xC4
206 #define SPIRIT_LINK_QUALIF2     0xC5
207 #define SPIRIT_LINK_QUALIF1     0xC6
208 #define SPIRIT_LINK_QUALIF0     0xC7

```

```

209 #define SPIRIT_RSSI_LEVEL      0xC8
210 #define SPIRIT_RX_PCKT_LEN1    0xC9
211 #define SPIRIT_RX_PCKT_LEN0    0xCA
212 #define SPIRIT_CRC_FIELD2      0xCB
213 #define SPIRIT_CRC_FIELD1      0xCC
214 #define SPIRIT_CRC_FIELD0      0xCD
215 #define SPIRIT_RX_CTRL_FIELD3  0xCE
216 #define SPIRIT_RX_CTRL_FIELD2  0xCF
217 #define SPIRIT_RX_CTRL_FIELD1  0x0D
218 #define SPIRIT_RX_CTRL_FIELD0  0xD1
219 #define SPIRIT_RX_ADDR_FIELD1  0xD2
220 #define SPIRIT_RX_ADDR_FIELD0  0xD3
221 #define SPIRIT_AES_DATA_OUT15   0xD4
222 #define SPIRIT_AES_DATA_OUT14   0xD5
223 #define SPIRIT_AES_DATA_OUT13   0xD6
224 #define SPIRIT_AES_DATA_OUT12   0xD7
225 #define SPIRIT_AES_DATA_OUT11   0xD8
226 #define SPIRIT_AES_DATA_OUT10   0xD9
227 #define SPIRIT_AES_DATA_OUT9    0xDA
228 #define SPIRIT_AES_DATA_OUT8    0xDB
229 #define SPIRIT_AES_DATA_OUT7    0xDC
230 #define SPIRIT_AES_DATA_OUT6    0xDD
231 #define SPIRIT_AES_DATA_OUT5    0xDE
232 #define SPIRIT_AES_DATA_OUT4    0xDF
233 #define SPIRIT_AES_DATA_OUT3    0xE0
234 #define SPIRIT_AES_DATA_OUT2    0xE1
235 #define SPIRIT_AES_DATA_OUT1    0xE2
236 #define SPIRIT_AES_DATA_OUT0    0xE3
237 #define SPIRIT_RCO_VCO_CALIBR_OUT1  0xE4
238 #define SPIRIT_RCO_VCO_CALIBR_OUT0  0xE5
239 #define SPIRIT_LINEAR_FIFO_STATUSTX 0xE6
240 #define SPIRIT_LINEAR_FIFO_STATUSRX 0xE7
241 #define SPIRIT_IRQ_STATUS3      0xFA
242 #define SPIRIT_IRQ_STATUS2      0xFB
243 #define SPIRIT_IRQ_STATUS1      0xFC
244 #define SPIRIT_IRQ_STATUS0      0xFD
245
246 //registers
247 #define SPIRIT_DEVICE_INFO1      0xF0

```

```

248 #define SPIRIT_DEVICE_INFO0      0xF1
249 #define SPIRIT_FIFO                0xFF
250
251 #define TRANSMITTER 0x01
252 #define RECEIVER 0x02
253
254 //device ID values
255 #define ID1                0x01
256 #define ID2                0x30
257
258 class spirit1
259 {
260     public:
261         uint8_t status[2];
262
263         spirit1(uint8_t id = ID1|ID2);
264
265         void writeRegisters( uint8_t reg , uint8_t value );
266         uint8_t readRegister( uint8_t reg );
267         void commandStrobe( uint8_t cmd );
268         void initSpirit( byte type );
269         uint8_t getState();
270         void printState();
271         int empty_RX_FIFO( uint8_t * data );
272         void transmit_data( uint8_t * data , int length);
273     private:
274         uint8_t _sensorID;
275 };
276
277
278 #endif

```

Listing 5: C++ koden til LIS3MDL biblioteket

```

1  /*****
2  *
3  *   @author:   Ananda Nielsen
4  *
5  *   Driver for ST LIS3MDL 3axis magnetometer
6  *   using SPI

```

```

7  *
8  *****/
9
10
11 #include <lis3mdl.h>
12 #include "SPI.h"
13
14 //global variables
15 SPISettings LIS3MDL_SPI(2000000, MSBFIRST, SPI_MODE3);
16
17 /**
18  * @brief    write one byte to a register
19  * @param    register adress and value to write
20  * @return   none
21  */
22 void lis3mdl::writeRegister(uint8_t reg, uint8_t value) {
23     SPI1.beginTransaction(LIS3MDL_SPI);
24     CS_LOW;
25     SPI1.transfer(reg);
26     SPI1.transfer(value);
27     CS_HIGH;
28     SPI1.endTransaction();
29 }
30
31
32 /**
33  * @brief    constructor
34  * @param    sensor id value
35  * @return   none
36  */
37 lis3mdl::lis3mdl(uint8_t sensor_id){
38     _sensorID = sensor_id;
39 }
40
41
42 /**
43  * @brief    Read the value of a register
44  * @param    reg is the register adress to read
45  * @return   value read from register

```

```

46 *****/
47 uint8_t lis3mdl::readRegister(uint8_t reg) {
48     uint8_t read_value = 0;
49     reg = reg | 0x80;
50     SPI1.beginTransaction(LIS3MDL_SPI);
51     CS_LOW;
52     SPI1.transfer(reg);
53     read_value = SPI1.transfer(0);
54     CS_HIGH;
55     SPI1.endTransaction();
56     return read_value;
57 }
58
59
60 /*****
61 * @brief      setup the ctrl registers on lis3mdl to high performance mode
62 * @param      none
63 * @return     none
64 *****/
65 void lis3mdl::init_magnetometer() {
66
67     pinMode(CS_SPI1, OUTPUT);
68     CS_HIGH;
69     if(_sensorID != readRegister(LIS3MDL_WHO_AM_I)) {
70         Serial.println("LIS3MDL Not Connected/Found on SPI!");
71     }
72     else{
73         writeRegister(LIS3MDL_CTRL_REG2, 0x04);
74         delay(10);
75         writeRegister(LIS3MDL_CTRL_REG2, 0x08);
76         delay(10);
77
78         current_fs_sens = FS_4_SENSITIVITY;
79         writeRegister(LIS3MDL_CTRL_REG1, CTRL_REG1_SETUP);
80         writeRegister(LIS3MDL_CTRL_REG2, CTRL_REG2_SETUP);
81         writeRegister(LIS3MDL_CTRL_REG3, CTRL_REG3_SETUP);
82         writeRegister(LIS3MDL_CTRL_REG4, CTRL_REG4_SETUP);
83         writeRegister(LIS3MDL_CTRL_REG5, CTRL_REG5_SETUP);
84         changeDataRate(20);

```



```

85     }
86 }
87
88
89 /*****
90 * @brief    check the status register to se if new data is availble
91 * @param    none
92 * @return   true if data is avaiible else false
93 *****/
94 bool lis3mdl::newDataAvailable() {
95     if( (readRegister(LIS3MDL_STATUS_REG) & 0x08 ) == 0x08)
96         return true;
97     else
98         return false;
99 }
100
101
102 /*****
103 * @brief    reads the magnetic field data registers
104 * @param    none
105 * @return   none
106 *****/
107 void lis3mdl::readDataRegisters() {
108     uint8_t reg = 0xC0 | LIS3MDL_OUT_X_L;
109     SPI1.beginTransaction(LIS3MDL_SPI);
110     CS_LOW;
111     SPI1.transfer(reg);
112     x_l = SPI1.transfer(0);
113     x_h = SPI1.transfer(0);
114     y_l = SPI1.transfer(0);
115     y_h = SPI1.transfer(0);
116     z_l = SPI1.transfer(0);
117     z_h = SPI1.transfer(0);
118     CS_HIGH;
119     SPI1.endTransaction();
120 }
121
122
123 /*****

```

```

124 * @brief    change the sensitivity of the magnetometer
125 * @param    the fullscale
126 * @return   none
127 *****/
128 void lis3mdl::changeSensitivity(int fullscale) {
129     if(fullscale == 4){
130         writeRegister(LIS3MDL_CTRL_REG2, 0x00);
131         current_fs_sens = FS_4_SENSITIVITY;
132     }
133     else if(fullscale == 8){
134         writeRegister(LIS3MDL_CTRL_REG2, 0x20);
135         current_fs_sens = FS_8_SENSITIVITY;
136     }
137     else if(fullscale == 12){
138         writeRegister(LIS3MDL_CTRL_REG2, 0x40);
139         current_fs_sens = FS_12_SENSITIVITY;
140     }
141     else if(fullscale == 16){
142         writeRegister(LIS3MDL_CTRL_REG2, 0x60);
143         current_fs_sens = FS_16_SENSITIVITY;
144     }
145 }
146
147
148 *****/
149 * @brief    change the operation mode (performance)
150 * @param    operating mode
151 * @return   none
152 *****/
153 void lis3mdl::changeOpMode(uint8_t om) {
154     uint8_t reg_cont = readRegister(LIS3MDL_CTRL_REG1) & 0x9F;
155     om = reg_cont | ((om << 5) & 0x60);
156     writeRegister(LIS3MDL_CTRL_REG1, om);
157     om = (om << 2) & 0x0C;
158     writeRegister(LIS3MDL_CTRL_REG4, om);
159 }
160
161 *****/
162 * @brief    change the output data rate

```

```

163 * @param    data_rate
164 * @return   none
165 *****/
166 void lis3mdl::changeDataRate(uint8_t data_rate) {
167     uint8_t reg_cont = readRegister(LIS3MDL_CTRL_REG1) & 0xE3;
168     if(data_rate == 625)
169         reg_cont |= 0b00000000;
170     else if(data_rate == 125)
171         reg_cont |= 0b00000100;
172     else if(data_rate == 25)
173         reg_cont |= 0b00001000;
174     else if(data_rate == 5)
175         reg_cont |= 0b00001100;
176     else if(data_rate == 10)
177         reg_cont |= 0b00010000;
178     else if(data_rate == 20)
179         reg_cont |= 0b00010100;
180     else if(data_rate == 40)
181         reg_cont |= 0b00011000;
182     else if(data_rate == 80)
183         reg_cont |= 0b00011100;
184     writeRegister(LIS3MDL_CTRL_REG1, reg_cont);
185 }
186
187
188 /*****
189 * @brief    reads the magnetic field data registers
190 * @param    none
191 * @return   none
192 *****/
193 void lis3mdl::readDataRegisters2() {
194     if(newDataAvailable()) {
195         SPI1.beginTransaction(LIS3MDL_SPI);
196         CSLOW;
197         x_l = readRegister(LIS3MDL_OUT_X_L);
198         x_h = readRegister(LIS3MDL_OUT_X_H);
199         y_l = readRegister(LIS3MDL_OUT_Y_L);
200         y_h = readRegister(LIS3MDL_OUT_Y_H);
201         z_l = readRegister(LIS3MDL_OUT_Z_L);

```

```

202         z_h = readRegister(LIS3MDL_OUT_Z_H);
203         CS_HIGH;
204         SPI1.endTransaction();
205     }
206 }
207
208
209 /*****
210  * @brief    convert the high low registers to the x,y,z components (float)
211  * @param    none
212  * @return   none
213  *****/
214 void lis3mdl::calculateMagneticVector() {
215
216     x16 = (x_h << 8) | x_l;
217     //x16 = x16 << 1;
218     y16 = (y_h << 8) | y_l;
219     //y16 = y16 << 1;
220     z16 = (z_h << 8) | z_l;
221     //z16 = z16 << 1;
222
223     x = (float) x16 / current_fs_sens;
224     y = (float) y16 / current_fs_sens;
225     z = (float) z16 / current_fs_sens;
226 }
227
228
229 /*****
230  * @brief    reads registers and converts values to vector field
231  * @param    none
232  * @return   none
233  *****/
234 void lis3mdl::getVectorField() {
235     if(newDataAvailable()) {
236         readDataRegisters();
237         calculateMagneticVector();
238     }
239     else
240         Serial.println("Data Not Available");

```

```

241 }
242
243
244 /******
245 * @brief    enable the temperature sensor on the lis3mdl
246 * @param    none
247 * @return   none
248 *****/
249 void lis3mdl::enableTemperature() {
250     uint8_t value = readRegister(LIS3MDL_CTRL_REG1);
251     value |= 0x80;
252     writeRegister(LIS3MDL_CTRL_REG1, value);
253 }
254
255
256 /******
257 * @brief    disable the temperature sensor on the lis3mdl
258 * @param    none
259 * @return   none
260 *****/
261 void lis3mdl::disableTemperature() {
262     uint8_t value = readRegister(LIS3MDL_CTRL_REG1);
263     value &= 0x7F;
264     writeRegister(LIS3MDL_CTRL_REG1, value);
265 }
266
267
268 /******
269 * @brief    update the values of the temperature sensor to the lis3mdl
270 * @param    none
271 * @return   none
272 *****/
273 void lis3mdl::readTemperature() {
274     temp_l = readRegister(LIS3MDL_TEMP_OUT_L);
275     temp_h = readRegister(LIS3MDL_TEMP_OUT_H);
276     temp16 = (temp_h << 8) | temp_l;
277     temp = (float) temp16 / TEMP_SENSITIVITY;
278 }

```

Listing 6: h fil til LIS3MDL biblioteket

```

1  /*****
2  *
3  *   @author:   Ananda Nielsen
4  *
5  *   Driver for ST LIS3MDL 3axis magnetometer
6  *   using SPI
7  *
8  *****/
9
10 #ifndef LIS3MDL_H_
11 #define LIS3MDL_H_
12
13 // includes
14 #include "SPI.h"
15 #include <Arduino.h>
16
17
18 //Teensy Pins
19 #define CS_SPI1 10
20
21 //Register addresses
22 #define LIS3MDL_OFFSET_X_REG_L 0x05
23 #define LIS3MDL_OFFSET_X_REG_H 0x06
24 #define LIS3MDL_OFFSET_Y_REG_L 0x07
25 #define LIS3MDL_OFFSET_Y_REG_H 0x08
26 #define LIS3MDL_OFFSET_Z_REG_L 0x09
27 #define LIS3MDL_OFFSET_Z_REG_H 0x0A
28 #define LIS3MDL_WHO_AM_I 0x0F           //devive fac number 0011 1101
29 #define LIS3MDL_CTRL_REG1 0x20
30 #define LIS3MDL_CTRL_REG2 0x21
31 #define LIS3MDL_CTRL_REG3 0x22
32 #define LIS3MDL_CTRL_REG4 0x23
33 #define LIS3MDL_CTRL_REG5 0x24
34 #define LIS3MDL_STATUS_REG 0x27
35 #define LIS3MDL_OUT_X_L 0x28
36 #define LIS3MDL_OUT_X_H 0x29
37 #define LIS3MDL_OUT_Y_L 0x2A
38 #define LIS3MDL_OUT_Y_H 0x2B

```

```

39 #define LIS3MDL_OUT_Z_L 0x2C
40 #define LIS3MDL_OUT_Z_H 0x2D
41 #define LIS3MDL_TEMP_OUT_L 0x2E
42 #define LIS3MDL_TEMP_OUT_H 0x2F
43 #define LIS3MDL_INT_CFG 0x30
44 #define LIS3MDL_INT_SRC 0x31
45 #define LIS3MDL_INT_THS_L 0x32
46 #define LIS3MDL_INT_THS_H 0x33
47
48 //offset values
49 //0000 0000 0101 0100
50 #define X_OFFSET_H 0x00
51 #define X_OFFSET_L 0x54
52 //1111 1101 1001 1010
53 #define Y_OFFSET_H 0xFD
54 #define Y_OFFSET_L 0x9A
55 //0000 1001 1011 0000
56 #define Z_OFFSET_H 0x09
57 #define Z_OFFSET_L 0xB0
58
59 //
60 #define LIS3MDL_NUM 0x3D
61 #define CS_LOW digitalWrite(CS_SPI1, LOW)
62 #define CS_HIGH digitalWrite(CS_SPI1, HIGH)
63
64 //fullscale sensitivity
65 #define FS_4_SENSITIVITY 6842
66 #define FS_8_SENSITIVITY 3421
67 #define FS_12_SENSITIVITY 2281
68 #define FS_16_SENSITIVITY 1711
69
70 //operating mode
71 #define LOW_POWER 0
72 #define MEDIUM_POWER 1
73 #define HIGH_POWER 2
74 #define ULTRA_HIGH_POWER 3
75
76
77 //tempeature sensitivity

```

```

78 #define TEMP_SENSITIVITY 8
79
80 //default setup (ultra high performance, continous mode, no temperature, block
    update, 5 Hz)
81 #define CTRL_REG1_SETUP 0x6C //0110 1100 temp off, xy ultra high, 5Hz, fast odr
    off, self test off
82 #define CTRL_REG2_SETUP 0x00 //0000 0000 +- 4 g
83 #define CTRL_REG3_SETUP 0x00 //0000 0000 no low power, spi 4 wire, continous-
    conversion
84 #define CTRL_REG4_SETUP 0x0C //0000 1100 z ultra high, LSb at low addresses
85 #define CTRL_REG5_SETUP 0x60 //0100 0000 fast read off, data not updated before it
    is read
86
87 //
88
89
90 class lis3mdl
91 {
92 public:
93     lis3mdl(uint8_t id = LIS3MDL_NUM);
94
95     byte x_l, x_h, y_l, y_h, z_l, z_h;
96     int16_t x16, y16, z16;
97     float x, y, z;
98     byte temp_l, temp_h;
99     int16_t temp16;
100     float temp;
101     int current_fs_sens;
102
103     void writeRegister(uint8_t reg, uint8_t value);
104     uint8_t readRegister(uint8_t reg);
105     void init_magnetometer();
106     void readDataRegisters();
107     void readDataRegisters2();
108     void calculateMagneticVector();
109     bool newDataAvailable();
110     void enableTemperature();
111     void disableTemperature();
112     void readTemperature();

```



```
113 void getVectorField();  
114 void changeSensitivity(int fullscale);  
115 void changeOpMode(uint8_t om);  
116 void changeDataRate(uint8_t data_rate);  
117  
118 private:  
119     uint8_t _sensorID;  
120 };  
121  
122 #endif
```

B. Matlab kode

Listing 7: Matlabkoden til køleskabsmagnetplot

```

1 clear ;
2 ax = 0
3 ay = 0
4 az = 0
5 sum = 0
6
7 [x,y,z] = meshgrid(0:5:50,0:5:50,0:0.00001:0);
8 [tBx, tBy, tBz] = textread('batman.txt', '%f %*c %f %*c %f' , 121);
9
10 Bx = vec2mat(tBx, 11)*100;
11 By = vec2mat(tBy, 11)*100;
12 Bz = vec2mat(tBz, 11)*100;
13
14 for j = [1,8,9,10,11]
15     for i = 1:11
16         ax = ax + Bx(j,i);
17         ay = ay + By(j,i);
18         az = az + Bz(j,i);
19         sum = sum + 1;
20     end
21 end
22
23 ax = ax/sum;
24 ay = ay/sum;
25 az = az/sum;
26
27
28 %figure(2)
29 %quiver(x,y,Bx,By)
30 %%
31 figure(1)
32 clf
33 hold on
34 set(gca,'fontsize',16)
35 %title('Magnetfelt mlt for firkantet køleskabsmagnet')
36 xlabel('X / cm')

```

```

37 ylabel('Y / cm')
38 [C,h]=contourf(x,y,Bz-az,'ShowText','on')%
39 cb = colorbar;
40 cb.Label.String = 'Strrelse af Bz i T';
41 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
42 legend('Bz kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
43 hold off
44
45 %%
46 figure(2)
47 clf
48 hold on
49 set(gca,'fontsize',16)
50 %title('Magnetfelt mlt for firkantet kleskabsmagnet')
51 xlabel('X / cm')
52 ylabel('Y / cm')
53 [C,h]=contourf(x,y,Bx-ax,'ShowText','on')%
54 cb = colorbar;
55 cb.Label.String = 'Strrelse af Bx i T';
56 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
57 legend('Bx kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
58 hold off
59 %%
60 figure(3)
61 clf
62 hold on
63 set(gca,'fontsize',16)
64 %title('Magnetfelt mlt for firkantet kleskabsmagnet')
65 xlabel('X / cm')
66 ylabel('Y / cm')
67 [C,h]=contourf(x,y,By-ay,'ShowText','on')%
68 cb = colorbar;
69 cb.Label.String = 'Strrelse af By i T';
70 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
71 legend('By kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
72 hold off

```

Listing 8: Matlab kode til jernvinkel magnetfelt plot

```

1 clear;

```

```

2 ax = 0
3 ay = 0
4 az = 0
5
6 shapex = [13.5, 30.5, 31.5, 28, 27, 17, 14, 13.5];
7 shapey = [23, 22, 53, 53.5, 26.3, 26.8, 23.8, 23];
8
9 [x,y,z] = meshgrid(0:5:50,0:5:70,0:0.00001:0);
10 [tBx, tBy, tBz] = textread('jernvinkel.txt', '%f %*c %f %*c %f' , 165);
11
12 Bx = vec2mat(tBx, 11)*100;
13 By = vec2mat(tBy, 11)*100;
14 Bz = vec2mat(tBz, 11)*100;
15
16 for j = 14:15
17     for i = 1:11
18         ax = ax + Bx(j,i);
19         ay = ay + By(j,i);
20         az = az + Bz(j,i);
21     end
22 end
23
24 ax = ax/(2*11);
25 ay = ay/(2*11);
26 az = az/(2*11);
27
28 %%
29 figure(1)
30 clf
31 hold on
32 set(gca, 'fontsize', 16)
33 %title('Magnetfelt mlt for jernvinkel')
34 xlabel('X / cm')
35 ylabel('Y / cm')
36 [C,h]=contourf(x,y,Bz-az, 'ShowText', 'on')%
37 cb = colorbar;
38 cb.Label.String = 'Strrelse af Bz i T';
39 quiver(x,y,Bx-ax,By-ay, 'red', 'LineWidth', 1.5)
40 plot(shapex, shapey, 'magenta', 'LineWidth', 2)

```

```

41 legend('Bz kontur', T, 'BxBy magnetfelt vektor felt', 'metal objekt outline', '
    Location', 'northwest')
42 %axis([0 50.5 0 70])
43 hold off
44
45 %%
46 figure(2)
47 clf
48 hold on
49 set(gca, 'fontsize', 16)
50 %title('Magnetfelt mlt for jernvinkel')
51 xlabel('X / cm')
52 ylabel('Y / cm')
53 [C,h]=contourf(x,y,Bx-ax, 'ShowText', 'on')%
54 cb = colorbar;
55 cb.Label.String = 'Strrelse af Bx i T';
56 quiver(x,y,Bx-ax,By-ay, 'red', 'LineWidth', 1.5)
57 plot(shapex, shapex, 'magenta', 'LineWidth', 2)
58 legend('Bx kontur', T, 'BxBy magnetfelt vektor felt', 'metal objekt outline', '
    Location', 'northwest')
59 %axis([0 50.5 0 70])
60 hold off
61
62 %%
63 figure(3)
64 clf
65 hold on
66 set(gca, 'fontsize', 16)
67 %title('Magnetfelt mlt for jernvinkel')
68 xlabel('X / cm')
69 ylabel('Y / cm')
70 [C,h]=contourf(x,y,By-ay, 'ShowText', 'on')%
71 cb = colorbar;
72 cb.Label.String = 'Strrelse af By i T';
73 quiver(x,y,Bx-ax,By-ay, 'red', 'LineWidth', 1.5)
74 plot(shapex, shapex, 'magenta', 'LineWidth', 2)
75 legend('By kontur', T, 'BxBy magnetfelt vektor felt', 'metal objekt outline', '
    Location', 'northwest')
76 %axis([0 50.5 0 70])

```

```
77 hold off
```

Listing 9: Matlab kode til magnetfelt plot for den lille magnet lagt fladt

```
1 clear ;
2 ax = 0
3 ay = 0
4 az = 0
5 sum = 0
6
7 [x,y,z] = meshgrid(0:2.5:30,0:2.5:30,0:0.00001:0);
8 [tBx, tBy, tBz] = textread('lillemagnet.txt', '%f %*c %f %*c %f' , 169);
9
10 Bx = vec2mat(tBx, 13)*100;
11 By = vec2mat(tBy, 13)*100;
12 Bz = vec2mat(tBz, 13)*100;
13
14 for j = 1:2
15     for i = 1:5
16         ax = ax + Bx(j,i);
17         ay = ay + By(j,i);
18         az = az + Bz(j,i);
19         sum = sum + 1;
20     end
21 end
22
23 ax = ax/sum;
24 ay = ay/sum;
25 az = az/sum;
26
27
28 %figure(2)
29 %quiver(x,y,Bx,By)
30 %%
31 figure(1)
32 clf
33 hold on
34 set(gca,'fontsize',16)
35 %title('Magnetfelt ml for lille rund magnet')
36 xlabel('X / cm')
```

```

37 ylabel('Y / cm')
38 [C,h]=contourf(x,y,Bz-az,'ShowText','on')%
39 cb = colorbar;
40 cb.Label.String = 'Strrelse af Bz i T';
41 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
42 legend('Bz kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
43 hold off
44
45 %%
46 figure(2)
47 clf
48 hold on
49 set(gca,'fontsize',16)
50 %title('Magnetfelt ml for lille rund magnet')
51 xlabel('X / cm')
52 ylabel('Y / cm')
53 [C,h]=contourf(x,y,Bx-ax,'ShowText','on')%
54 cb = colorbar;
55 cb.Label.String = 'Strrelse af Bx i T';
56 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
57 legend('Bx kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
58 hold off
59
60
61 %%
62 figure(3)
63 clf
64 hold on
65 set(gca,'fontsize',16)
66 %title('Magnetfelt ml for lille rund magnet')
67 xlabel('X / cm')
68 ylabel('Y / cm')
69 [C,h]=contourf(x,y,By-ay,'ShowText','on')%
70 cb = colorbar;
71 cb.Label.String = 'Strrelse af By i T';
72 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
73 legend('By kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
74 hold off

```

Listing 10: Matlab kode til magnetfelt plot for den lille magnet på højkant

```

1 clear;
2 ax = 0
3 ay = 0
4 az = 0
5 sum = 0
6
7 [x,y,z] = meshgrid(0:2.5:20,0:2.5:20,0:0.00001:0);
8 [tBx, tBy, tBz] = textread('lillemagnetdrejet.txt', '%f %*c %f %*c %f' , 81);
9
10 Bx = vec2mat(tBx, 9)*100;
11 By = vec2mat(tBy, 9)*100;
12 Bz = vec2mat(tBz, 9)*100;
13
14 for j = 1
15     for i = 1
16         ax = ax + Bx(j,i);
17         ay = ay + By(j,i);
18         az = az + Bz(j,i);
19         sum = sum + 1;
20     end
21 end
22
23 ax = ax/sum;
24 ay = ay/sum;
25 az = az/sum;
26
27
28 figure(2)
29 quiver(x,y,Bx,By)
30 %%
31 figure(1)
32 clf
33 hold on
34 set(gca,'fontsize',16)
35 %title('Lille rund magnet')
36 xlabel('X / cm')
37 ylabel('Y / cm')
38 [C,h]=contourf(x,y,Bz-az,'ShowText','on');

```



```

39 cb = colorbar;
40 cb.Label.String = 'Strrelse af Bz i T';
41 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
42 legend('Bz kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
43 hold off
44
45 %%
46 figure(2)
47 clf
48 hold on
49 set(gca,'fontsize',16)
50 %title('Lille rund magnet')
51 xlabel('X / cm')
52 ylabel('Y / cm')
53 [C,h]=contourf(x,y,Bx-ax,'ShowText','on');
54 cb = colorbar;
55 cb.Label.String = 'Strrelse af Bx i T';
56 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
57 legend('Bx kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
58 hold off
59
60 %%
61 figure(3)
62 clf
63 hold on
64 set(gca,'fontsize',16)
65 %title('Lille rund magnet')
66 xlabel('X / cm')
67 ylabel('Y / cm')
68 [C,h]=contourf(x,y,By-ay,'ShowText','on');
69 cb = colorbar;
70 cb.Label.String = 'Strrelse af By i T';
71 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)
72 legend('By kontur, T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
73 hold off

```

Listing 11: Matlab kode til magnetfelt plot for den lille skruetrækker

```

1 clear;
2 ax = 0

```

```

3 ay = 0
4 az = 0
5 sum = 0
6
7 [x,y,z] = meshgrid(0:2.5:25,0:2.5:20,0:0.00001:0);
8 [tBx, tBy, tBz] = textread('lilleskrue.txt', '%f %*c %f %*c %f' , 99);
9
10 Bx = vec2mat(tBx, 11)*100;
11 By = vec2mat(tBy, 11)*100;
12 Bz = vec2mat(tBz, 11)*100;
13
14 for j = 7:9
15     for i = 1:3
16         ax = ax + Bx(j,i);
17         ay = ay + By(j,i);
18         az = az + Bz(j,i);
19         sum = sum + 1;
20     end
21 end
22
23 ax = ax/sum;
24 ay = ay/sum;
25 az = az/sum;
26
27
28 %figure(2)
29 %quiver(x,y,Bx,By)
30 %%
31 figure(1)
32 clf
33 hold on
34 set(gca,'fontsize',16)
35 %title('Magnetfelt mlt for lille skruetrkker')
36 xlabel('X / cm', 'FontSize', 16)
37 ylabel('Y / cm', 'FontSize', 16)
38 [C,h]=contourf(x,y,Bz-az,'ShowText','on')%
39 cb = colorbar('FontSize', 16);
40 cb.Label.String = 'Strrelse af Bz i T';
41 quiver(x,y,Bx-ax,By-ay,'red','LineWidth',1.5)

```

```

42 legend('Bz kontur', T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
43 hold off
44 %%
45 figure(2)
46 clf
47 hold on
48 set(gca, 'fontsize', 16)
49 %title('Magnetfelt mlt for lille skruetrækker')
50 xlabel('X / cm', 'FontSize', 16)
51 ylabel('Y / cm', 'FontSize', 16)
52 [C,h]=contourf(x,y,Bx-ax, 'ShowText', 'on')%
53 cb = colorbar('FontSize', 16);
54 cb.Label.String = 'Strrelse af Bx i T';
55 quiver(x,y,Bx-ax,By-ay, 'red', 'LineWidth', 1.5)
56 legend('Bx kontur', T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
57 hold off
58 %%
59 figure(3)
60 clf
61 hold on
62 set(gca, 'fontsize', 16)
63 %title('Magnetfelt mlt for lille skruetrækker')
64 xlabel('X / cm', 'FontSize', 16)
65 ylabel('Y / cm', 'FontSize', 16)
66 [C,h]=contourf(x,y,By-ay, 'ShowText', 'on')%
67 cb = colorbar('FontSize', 16);
68 cb.Label.String = 'Strrelse af By i T';
69 quiver(x,y,Bx-ax,By-ay, 'red', 'LineWidth', 1.5)
70 legend('By kontur', T', 'BxBy magnetfelt vektor felt', 'Location', 'northwest')
71 hold off

```

Listing 12: Matlab koden for præcisionsmålingen med 2000 målinger

```

1 t = 0.05:0.05:103.1;
2 t = t';
3 ts = 1:1:103;
4 ts = ts';
5 [ax, ay, az, Bx, By, Bz, rBx, rBy, rBz] = textread('stationr2minv9.txt', '%f %f %f
    %f %f %f %f %f %f', 'delimiter', ',');
6

```

```

7 avgaccumBx = zeros(103,1);
8 avgBx = zeros(103,1);
9 avgaccumBy = zeros(103,1);
10 avgBy = zeros(103,1);
11 avgaccumBz = zeros(103,1);
12 avgBz = zeros(103,1);
13 B = zeros(2062,1);
14 avgaccumB = zeros(103,1);
15 avgB = zeros(103,1);
16
17 for i = 1:2062
18     B(i) = sqrt(Bx(i)^2 + By(i)^2 + Bz(i)^2);
19 end
20
21 for i = 1:103
22     avgBx(i) = mean(Bx(i*20-19:i*20));
23     avgaccumBx(i) = mean(Bx(1:i*20));
24     avgBy(i) = mean(By(i*20-19:i*20));
25     avgaccumBy(i) = mean(By(1:i*20));
26     avgBz(i) = mean(Bz(i*20-19:i*20));
27     avgaccumBz(i) = mean(Bz(1:i*20));
28     avgB(i) = mean(B(i*20-19:i*20));
29     avgaccumB(i) = mean(B(1:i*20));
30 end
31
32 figure(19)
33 clf
34 subplot(2,2,1);
35 hold on
36 axis([0 110 -12.5 -9.5]);
37 set(gca,'fontsize',16);
38 scatter(t,Bx*100,'o','Filled');
39 scatter(ts,avgBx*100,'o','Filled','m');
40 scatter(ts,avgaccumBx*100,'o','Filled');
41 plot([0,103.05],[mean(Bx),mean(Bx)]*100,'LineWidth',3);
42 plot([0,103.05],[mean(Bx)+std(Bx),mean(Bx)+std(Bx)]*100,'LineWidth',3);
43 plot([0,103.05],[mean(Bx)-std(Bx),mean(Bx)-std(Bx)]*100,'LineWidth',3);
44 title(['Bx, std = ' num2str(std(Bx)*100) ' uT, gennemsnit = ' num2str(mean(Bx)
    *100) ' uT']);

```

```

45 xlabel('t / s');
46 ylabel('Bx / uT');
47 hold off
48 subplot(2,2,2);
49 hold on
50 axis([0 110 20.5 23.5]);
51 set(gca,'fontsize',16);
52 scatter(t,By*100,'o','Filled');
53 scatter(ts,avgBy*100,'o','Filled','m');
54 scatter(ts,avgaccumBy*100,'o','Filled');
55 plot([0,103.05],[mean(By),mean(By)]*100,'LineWidth',3);
56 plot([0,103.05],[mean(By)+std(By),mean(By)+std(By)]*100,'LineWidth',3);
57 plot([0,103.05],[mean(By)-std(By),mean(By)-std(By)]*100,'LineWidth',3);
58 title(['By, std = ' num2str(std(By)*100) ' uT, gennemsnit = ' num2str(mean(By)
    *100) ' uT']);
59 xlabel('t / s');
60 ylabel('By / uT');
61 hold off
62 subplot(2,2,3);
63 hold on
64 axis([0 110 -43 -39]);
65 set(gca,'fontsize',16);
66 scatter(t,Bz*100,'o','Filled');
67 scatter(ts,avgBz*100,'o','Filled','m');
68 scatter(ts,avgaccumBz*100,'o','Filled');
69 plot([0,103.05],[mean(Bz),mean(Bz)]*100,'LineWidth',3);
70 plot([0,103.05],[mean(Bz)+std(Bz),mean(Bz)+std(Bz)]*100,'LineWidth',3);
71 plot([0,103.05],[mean(Bz)-std(Bz),mean(Bz)-std(Bz)]*100,'LineWidth',3);
72 title(['Bz, std = ' num2str(std(Bz)*100) ' uT, gennemsnit = ' num2str(mean(Bz)
    *100) ' uT']);
73 xlabel('t / s');
74 ylabel('Bz / uT');
75 hold off
76 subplot(2,2,4);
77 hold on
78 axis([0 110 46.5 49.5]);
79 set(gca,'fontsize',16);
80 scatter(t,B*100,'o','Filled');
81 scatter(ts,avgB*100,'o','Filled','m');

```

```

82 scatter(ts, avgaccumB*100, 'o', 'Filled');
83 plot([0, 103.05], [mean(B), mean(B)]*100, 'LineWidth', 3);
84 plot([0, 103.05], [mean(B)+std(B), mean(B)+std(B)]*100, 'LineWidth', 3);
85 plot([0, 103.05], [mean(B)-std(B), mean(B)-std(B)]*100, 'LineWidth', 3);
86 title(['|B|', std = ' num2str(std(B)*100) ' uT, gennemsnit = ' num2str(mean(B)*100)
      ' uT']);
87 xlabel('t / s');
88 ylabel('|B| / uT');
89 legend('Mlte vrdier', 'Gennemsnit efter hvert sekund', 'Akkumuleret gennemsnit
      hvert sekund', 'Gennemsnit', 'Gennemsnit + standard afvigelse', 'Gennemsnit -
      standard afvigelse');
90 hold off
91
92 format long g
93 std(ax)
94 std(ay)
95 std(az)

```

Listing 13: Matlab koden for præcisionsmålingen med 10000 målinger

```

1 t = 0.05:0.05:503.1;
2 t = t';
3 ts = 1:1:503;
4 ts = ts';
5 [ax, ay, az, Bx, By, Bz, rBx, rBy, rBz] = textread('stationr8minv2.txt', '%f %f %f
      %f %f %f %f %f %f', 'delimiter', ',');
6
7 avgaccumBx = zeros(503,1);
8 avgBx = zeros(503,1);
9 avgaccumBy = zeros(503,1);
10 avgBy = zeros(503,1);
11 avgaccumBz = zeros(503,1);
12 avgBz = zeros(503,1);
13 B = zeros(10062,1);
14 avgaccumB = zeros(503,1);
15 avgB = zeros(503,1);
16
17 for i = 1:10062
18     B(i) = sqrt(Bx(i)^2 + By(i)^2 + Bz(i)^2);
19 end

```

```

20
21 for i = 1:503
22     avgBx(i) = mean(Bx(i*20-19:i*20));
23     avgaccumBx(i) = mean(Bx(1:i*20));
24     avgBy(i) = mean(By(i*20-19:i*20));
25     avgaccumBy(i) = mean(By(1:i*20));
26     avgBz(i) = mean(Bz(i*20-19:i*20));
27     avgaccumBz(i) = mean(Bz(1:i*20));
28     avgB(i) = mean(B(i*20-19:i*20));
29     avgaccumB(i) = mean(B(1:i*20));
30 end
31
32 figure(34)
33 clf
34 subplot(2,2,1);
35 hold on
36 axis([0 530 -7 -4]);
37 set(gca,'fontsize',16);
38 scatter(t,Bx*100,'o','Filled');
39 scatter(ts,avgBx*100,'o','Filled','m');
40 scatter(ts,avgaccumBx*100,'o','Filled');
41 plot([0,503.05],[mean(Bx),mean(Bx)]*100,'LineWidth',3);
42 plot([0,503.05],[mean(Bx)+std(Bx),mean(Bx)+std(Bx)]*100,'LineWidth',3);
43 plot([0,503.05],[mean(Bx)-std(Bx),mean(Bx)-std(Bx)]*100,'LineWidth',3);
44 title(['Bx, std = ' num2str(std(Bx)*100) ' uT, gennemsnit = ' num2str(mean(Bx)
    *100) ' uT']);
45 xlabel('t / s');
46 ylabel('Bx / uT');
47 hold off
48 subplot(2,2,2);
49 hold on
50 axis([0 530 -11.5 -8.5]);
51 set(gca,'fontsize',16);
52 scatter(t,By*100,'o','Filled');
53 scatter(ts,avgBy*100,'o','Filled','m');
54 scatter(ts,avgaccumBy*100,'o','Filled');
55 plot([0,503.05],[mean(By),mean(By)]*100,'LineWidth',3);
56 plot([0,503.05],[mean(By)+std(By),mean(By)+std(By)]*100,'LineWidth',3);
57 plot([0,503.05],[mean(By)-std(By),mean(By)-std(By)]*100,'LineWidth',3);

```

```

58 title(['By, std = ' num2str(std(By)*100) ' uT, gennemsnit = ' num2str(mean(By)
    *100) ' uT']);
59 xlabel('t / s');
60 ylabel('By / uT');
61 hold off
62 subplot(2,2,3);
63 hold on
64 axis([0 530 -47 -42]);
65 set(gca,'fontsize',16);
66 scatter(t,Bz*100,'o','Filled');
67 scatter(ts,avgBz*100,'o','Filled','m');
68 scatter(ts,avgaccumBz*100,'o','Filled');
69 plot([0,503.05],[mean(Bz),mean(Bz)]*100,'LineWidth',3);
70 plot([0,503.05],[mean(Bz)+std(Bz),mean(Bz)+std(Bz)]*100,'LineWidth',3);
71 plot([0,503.05],[mean(Bz)-std(Bz),mean(Bz)-std(Bz)]*100,'LineWidth',3);
72 title(['Bz, std = ' num2str(std(Bz)*100) ' uT, gennemsnit = ' num2str(mean(Bz)
    *100) ' uT']);
73 xlabel('t / s');
74 ylabel('Bz / uT');
75 hold off
76 subplot(2,2,4);
77 hold on
78 axis([0 530 44 48]);
79 set(gca,'fontsize',16);
80 scatter(t,B*100,'o','Filled');
81 scatter(ts,avgB*100,'o','Filled','m');
82 scatter(ts,avgaccumB*100,'o','Filled');
83 plot([0,503.05],[mean(B),mean(B)]*100,'LineWidth',3);
84 plot([0,503.05],[mean(B)+std(B),mean(B)+std(B)]*100,'LineWidth',3);
85 plot([0,503.05],[mean(B)-std(B),mean(B)-std(B)]*100,'LineWidth',3);
86 title(['|B|, std = ' num2str(std(B)*100) ' uT, gennemsnit = ' num2str(mean(B)*100)
    ' uT']);
87 xlabel('t / s');
88 ylabel('|B| / uT');
89 legend('Mlte vrdier', 'Gennemsnit efter hvert sekund', 'Akkumuleret gennemsnit
    hvert sekund', 'Gennemsnit', 'Gennemsnit + standard afvigelse', 'Gennemsnit -
    standard afvigelse');
90 hold off
91

```



```

92 format long g
93 std(ax)
94 std(ay)
95 std(az)

```

Listing 14: Matlab koden for præcisionsmålingen med 9000 målinger

```

1 t = 0.05:0.05:486.9;
2 t = t';
3 ts = 1:1:486;
4 ts = ts';
5 [ax, ay, az, Bx, By, Bz, rBx, rBy, rBz] = textread('stationr8minv3.txt', '%f %f %f
    %f %f %f %f %f %f', 'delimiter', ',',');
6
7 avgaccumBx = zeros(486,1);
8 avgBx = zeros(486,1);
9 avgaccumBy = zeros(486,1);
10 avgBy = zeros(486,1);
11 avgaccumBz = zeros(486,1);
12 avgBz = zeros(486,1);
13 B = zeros(9738,1);
14 avgaccumB = zeros(486,1);
15 avgB = zeros(486,1);
16
17 for i = 1:9738
18     B(i) = sqrt(Bx(i)^2 + By(i)^2 + Bz(i)^2);
19 end
20
21 for i = 1:486
22     avgBx(i) = mean(Bx(i*20-19:i*20));
23     avgaccumBx(i) = mean(Bx(1:i*20));
24     avgBy(i) = mean(By(i*20-19:i*20));
25     avgaccumBy(i) = mean(By(1:i*20));
26     avgBz(i) = mean(Bz(i*20-19:i*20));
27     avgaccumBz(i) = mean(Bz(1:i*20));
28     avgB(i) = mean(B(i*20-19:i*20));
29     avgaccumB(i) = mean(B(1:i*20));
30 end
31
32 figure(12)

```

```

33 clf
34 subplot(2,2,1);
35 hold on
36 set(gca,'fontsize',16);
37 scatter(t,Bx*100,'o','Filled');
38 scatter(ts,avgBx*100,'o','Filled','m');
39 scatter(ts,avgaccumBx*100,'o','Filled');
40 plot([0,486.9],[mean(Bx),mean(Bx)]*100,'LineWidth',3);
41 plot([0,486.9],[mean(Bx)+std(Bx),mean(Bx)+std(Bx)]*100,'LineWidth',3);
42 plot([0,486.9],[mean(Bx)-std(Bx),mean(Bx)-std(Bx)]*100,'LineWidth',3);
43 %legend('Mlte vrdier', 'Gennemsnit efter hvert sekund', 'Akkumuleret gennemsnit
    hvert sekund', 'Gennemsnit', 'Gennemsnit + standard afvigelse', 'Gennemsnit -
    standard afvigelse');
44 title(['Bx, std = ' num2str(std(Bx)*100) ' uT, gennemsnit = ' num2str(mean(Bx)
    *100) ' uT']);
45 %title(['Bx' num2str(4)]);
46 xlabel('t / s');
47 ylabel('Bx / uT');
48 hold off
49 subplot(2,2,2);
50 hold on
51 set(gca,'fontsize',16);
52 scatter(t,By*100,'o','Filled');
53 scatter(ts,avgBy*100,'o','Filled','m');
54 scatter(ts,avgaccumBy*100,'o','Filled');
55 plot([0,486.9],[mean(By),mean(By)]*100,'LineWidth',3);
56 plot([0,486.9],[mean(By)+std(By),mean(By)+std(By)]*100,'LineWidth',3);
57 plot([0,486.9],[mean(By)-std(By),mean(By)-std(By)]*100,'LineWidth',3);
58 title(['By, std = ' num2str(std(By)*100) ' uT, gennemsnit = ' num2str(mean(By)
    *100) ' uT']);
59 xlabel('t / s');
60 ylabel('By / uT');
61 hold off
62 subplot(2,2,3);
63 hold on
64 set(gca,'fontsize',16);
65 scatter(t,Bz*100,'o','Filled');
66 scatter(ts,avgBz*100,'o','Filled','m');
67 scatter(ts,avgaccumBz*100,'o','Filled');

```

```

68 plot([0,486.9],[mean(Bz),mean(Bz)]*100,'LineWidth',3);
69 plot([0,486.9],[mean(Bz)+std(Bz),mean(Bz)+std(Bz)]*100,'LineWidth',3);
70 plot([0,486.9],[mean(Bz)-std(Bz),mean(Bz)-std(Bz)]*100,'LineWidth',3);
71 title(['Bz, std = ' num2str(std(Bz)*100) ' uT, gennemsnit = ' num2str(mean(Bz)
    *100) ' uT']);
72 xlabel('t / s');
73 ylabel('Bz / uT');
74 hold off
75 subplot(2,2,4);
76 hold on
77 set(gca,'fontsize',16);
78 scatter(t,B*100,'o','Filled');
79 scatter(ts,avgB*100,'o','Filled','m');
80 scatter(ts,avgaccumB*100,'o','Filled');
81 plot([0,486.9],[mean(B),mean(B)]*100,'LineWidth',3);
82 plot([0,486.9],[mean(B)+std(B),mean(B)+std(B)]*100,'LineWidth',3);
83 plot([0,486.9],[mean(B)-std(B),mean(B)-std(B)]*100,'LineWidth',3);
84 title(['|B|, std = ' num2str(std(B)*100) ' uT, gennemsnit = ' num2str(mean(B)*100)
    ' uT']);
85 xlabel('t / s');
86 ylabel('|B| / uT');
87 legend('Mlte vrdier', 'Gennemsnit efter hvert sekund', 'Akkumuleret gennemsnit
    efter hvert sekund', 'Gennemsnit', 'Gennemsnit + standard afvigelse', 'Gennemsnit -
    standard afvigelse');
88 hold off
89
90 format long g
91 std(ax)
92 std(ay)
93 std(az)

```

DTU Space, Danmarks Tekniske Universitet
Elektrovej bygning 328
2800 Kgs. Lyngby

www.space.dtu.dk