Utvikling av lokalt posisjoneringssystem

TTK8 - Konstruksjon av innebygde systemer

Skrevet av
Morten Jansrud

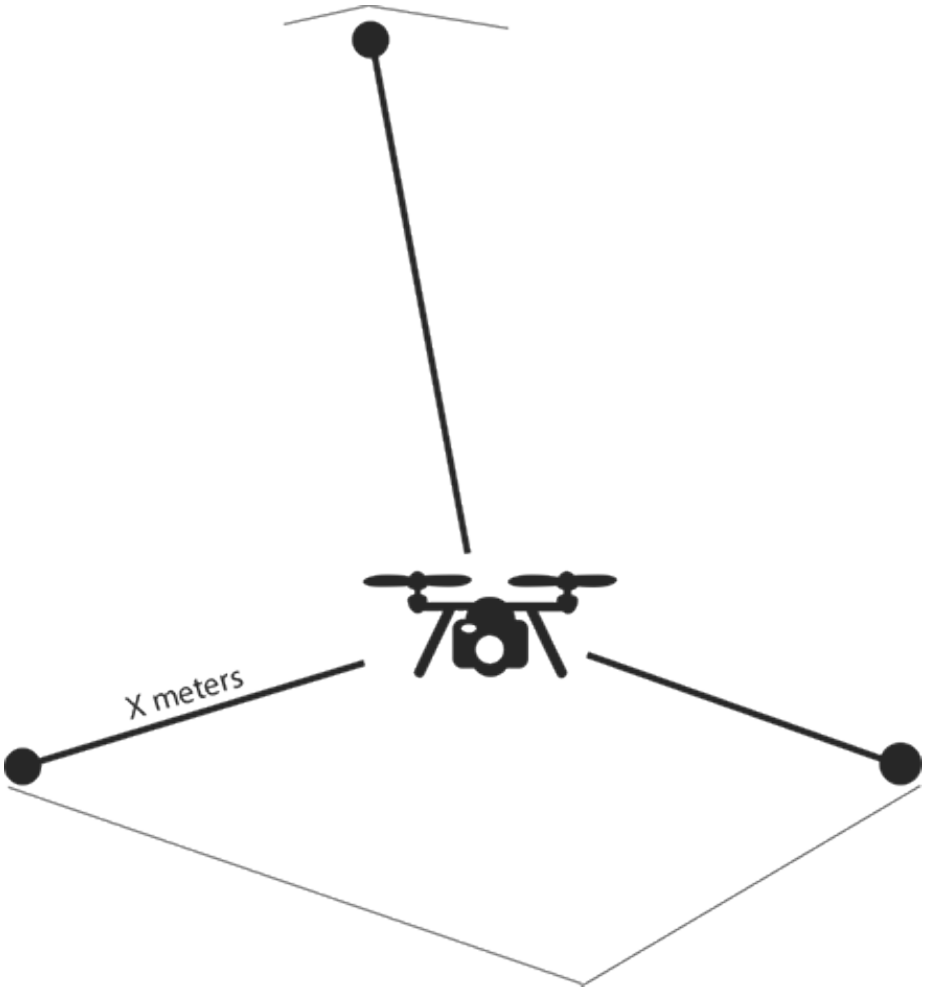
Institutt for Teknisk Kybernetikk
NTNU, Norge
November 2017

Innhold

1 Sammendrag

Droner tar stadig over flere operasjoner i samfunnet; de er kostnadsbesparende, effektive og nøyaktige. Det finnes fortsatt omgivelser som gjør automatisering av operasjoner vanskelig, blant annet fordi kritiske sensorer som posisjonering svikter. De aller fleste droner er utstyrt med GPS (Global Positioning System), men dette systemet er utilgjengelig i blant annet tunneler, gruver og bygninger av betong.

Denne rapporten tar for seg utvikling av et lavbudsjett posisjoneringssystem bestående av lokale avstandsmålere som kan erstatte GPS i områder hvor teknologien ikke er tilgjengelig.

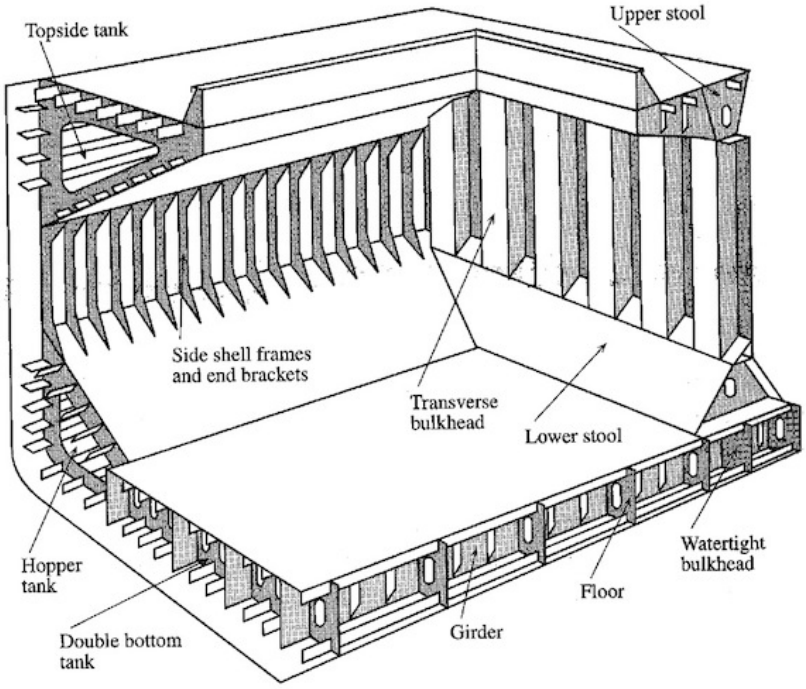


Figur 1: Simpel skisse over systemet

2 Introduksjon

2.1 Bakgrunn og motivasjon

Professor Tor Arne Johansen og PhD student Kristian Klausen jobber med utvikling av droner som skal kunne benyttes til inspisering av store tanker, i første omgang på skip. I tankene finnes det bjelker langs veggene hvor skjøtene må inspiseres for skader. I den forbindelse ønsket de et lokalt posisjoneringssystem som enkelt kan implementeres.



Figur 2: Potensiell tank som må inspiseres. Hentet fra [3].

2.2 Beskrivelse

Posisjoneringssystemet som ble utviklet består av $4xDW1000$ tådløse transceivere med ultra bredt frekvensbånd ($> 500MHz$) som kan sende store mengder data energieffektivt over korte avstander [1]. Radioene benyttes i dette tilfellet primært til å bestemme avstandene mellom nodene i nettverket, men under

oppstart og konfigurering av systemet sendes det meldinger for å samle informasjon og finne posisjonen til dronen. For å forenkle oppgaven er tre av nodene stasjonære og kun en bevegelig. Programvare kjøres på Arduino Pro og Pro Mini mikrokontrollere. Detaljert spesifikasjon og systembeskrivelse uttypes i kapittel 4 og 5.

2.3 Mål

Et lavbudsjett system som presist kan bestemme posisjonen til et objekt i bevegelse relativt til et definert koordinatsystem.

2.4 Omfang

I utgangspunktet var målet å lage *en* avstandsmåler mellom to noder i et nettverk basert på DW1000 radiosendere/mottakere. Det viste seg at det allerede eksisterte en driver for Arduino, noe som gjorde det mulig å utvide omfanget på prosjektet til posisjonering.

Det innebærer å koble sammen $4xDW1000$ med $4xArduinoPro$, finne og lagre avstanden mellom samtlige noder i nettverket og sende denne informasjonen til det bevegelige objektet.

Utregninger av posisjon er foreløpig i det 2-dimensjonale X-Y planet, men basert på informasjonen tilgjengelig skal det være mulig å bestemme Z posisjonen til objektet dersom samtlige avstander mellom det bevegelige objektet og ankernodene økes / minskes.

2.5 Strukturen på rapporten

Kapittel 3 inneholder teorien bak avstandsmåling og hvordan man kan finne posisjonenen til dronen ved bruk av to forskjellige metoder, kapittel 4 inneholder funksjonell spesfikasjon og detaljert beskrivelse av systemdesign (både hardware og software). Kapittel 5 beskriver hvordan system er implementert, kapittel 6 og 7 lister opp og diskuterer resultater.

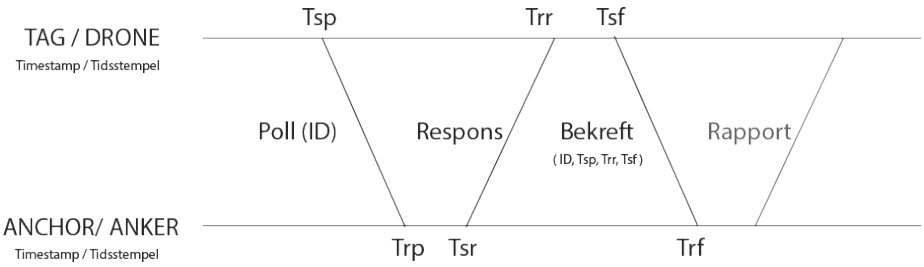
3 Teori

3.1 Avstandsmåling

For å måle nøyaktig avstand benyttes Decawave TWR, selskapets toveis avstandsprotokoll. I dette tilfellet vil dronen registrere et tidsstempel (Tsp) og starte en prosess som består av tre meldinger. Nodene registrerer tidsstempel både ved sending og mottak. Når dronen har mottatt rapporten med samtlige tidsstempel kan avstanden registreres

$$ToF = \frac{(T_{rr} - T_{sp}) - (T_{sr} - T_{rp}) + (T_{rf} - T_{sr}) - (T_{sf} - T_{rr})}{4}$$

$$Distance = ToF * speedoflight$$

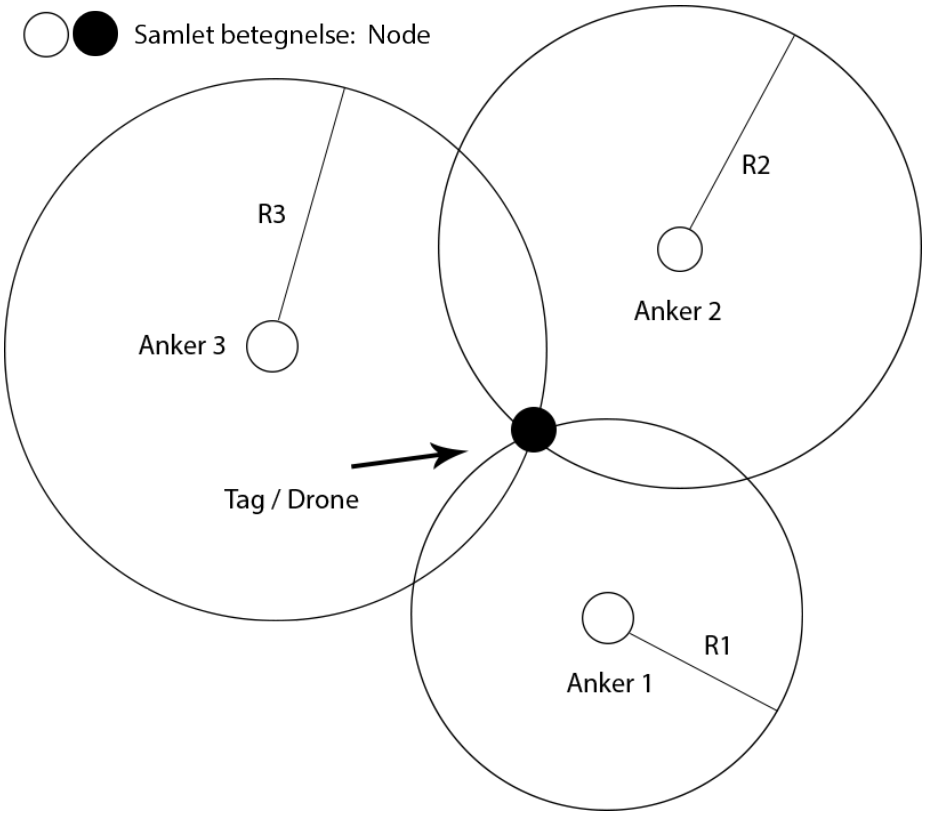


Figur 3: Decawave TWR, utgangspunkt fra [4].

3.2 Posisjonering

3.2.1 Uten å vite posisjonen til anker-nodene

Dersom man ikke har oppgitt eller definert posisjonen til node $1, 2$ og 3 som vist på figur 4 kan man allikevel finne relativ posisjon til dronen ved hjelp av kun radiusen/avstandsmålingen $R1, R2$ og $R3$. Avstandene vil kun danne et punkt i X-Y planet hvor sirklene krysser. Ulempen med framgangsmåten er at man hverken får et koordinatsystem eller kan regne ut Z verdien til dronen.



Figur 4: Relativ posisjonering vha avstandene R1, R2 og R3.

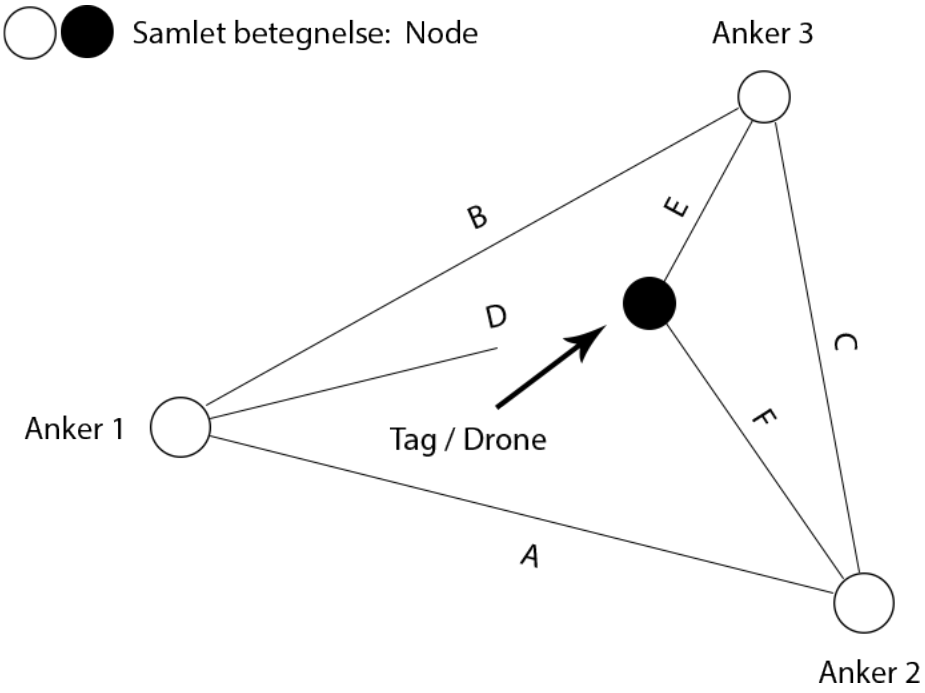
3.2.2 Med avstander mellom samtlige noder

Med avstander mellom samtlige noder tilgjengelig kan man enkelt definere et koordinatsystem for systemet. I denne oppgaven definerers $Anker1$ som nullpunkt $[0, 0, 0]$. Da blir posisjonen til $Anker2$ $[A, 0, 0]$ og $Anker3$ kan regnes ut fra

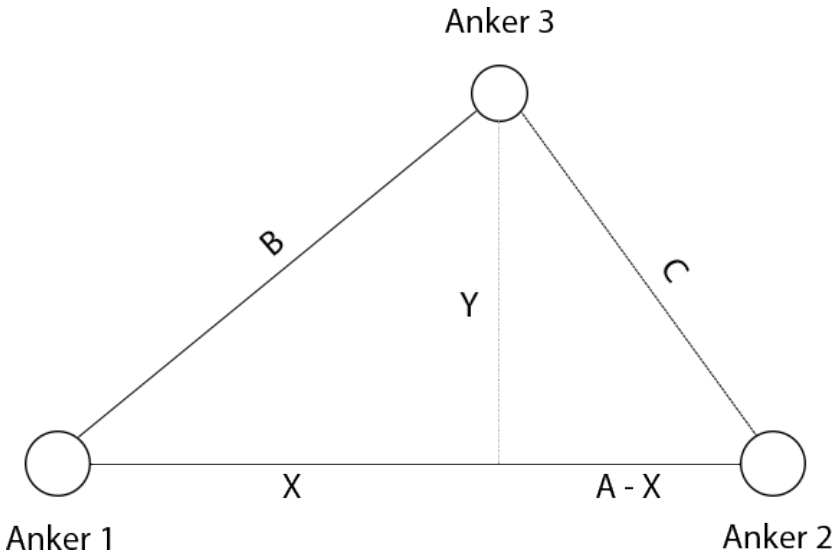
$$B^2 = X^2 + H^2$$
$$C^2 = (A - X)^2 + H^2$$

som resulterer i

$$X = \frac{A^2 + B^2 - C^2}{2A}$$
$$Y = +- \frac{\sqrt{-A^4 + 2A^2(B^2 + C^2) - (B^2 - C^2)^2}}{2A}$$



Figur 5: Posisjonering vha avstander mellom samtlige noder.



Figur 6: Benyttes for utregning av posisjonen til anker 3

Dette er en svært enkelt måte å regne ut posisjonen til nodene. Mer avanserte metoder finnes og burde brukes dersom prosjektet skal kommersialiseres. I dette emnet ble det meste av tiden brukt på å utvikle et system hvor avstandene mellom samtlige noder ble tilgjengelig for dronen.

4 Arkitektur

4.1 Funksjonell spesifikasjon

Systemet må tilfredsstille følgende krav:

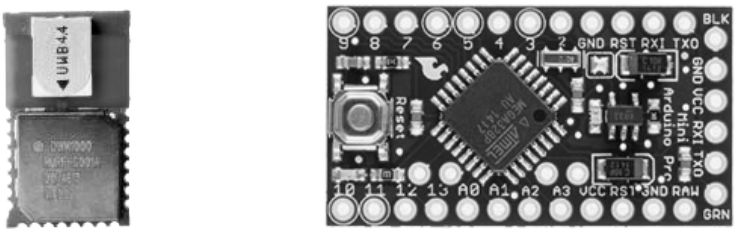
1. Bestemme posisjonen til dronen relativt til et definert koordinatsystem. For at dette skal fungere må følgende systemer være på plass:
   (a) Ha kontroll på hvilke signaler som tilhører hver node (ID)
   (b) Sende meldinger i mellom noder
   (c) Benytte meldingssystemet til å måle avstanden fra en node til en annen
   (d) Definere en node som enten tag/drone eller anker
   (e) Lage et system hvor alle noder har en x,y,z posisjon
   (f) Dronen må kunne motta avstander mellom samtlige anker
   (g) Regne ut posisjon basert på avstander

4.2 Maskinvare

Maskinvaren består av $4xDW1000$ tranceivers med en $IEEE802.15.4-2011$ chip som gjør avstandsmåling mulig med en presisjon på inntil 10cm innendørs selv om noden beveger seg i en hastighet på $5m/s$ [2]. Sensorene har et ultra bredt frekvensbånd som gjør at man kan f.eks sende meldinger med avstander over nettverket, de har høy energieffektivitet og klarer kommunikasjon opp mot 290m. Valget av radio falt på $DW1000$ fordi NTNU allerede hadde flere sensorer liggende.

Av mikrokontrollere sto valget mellom Bluepill og Arduino Pro. Felles krav var 3.3V SPI signal til DW1000 sensorene.

1. Bluepill med SM32
   (a) 64 kB Flash memory
   (b) 20 kB SRAM memory
   (c) 72 MHz 32-bit ARM CPU



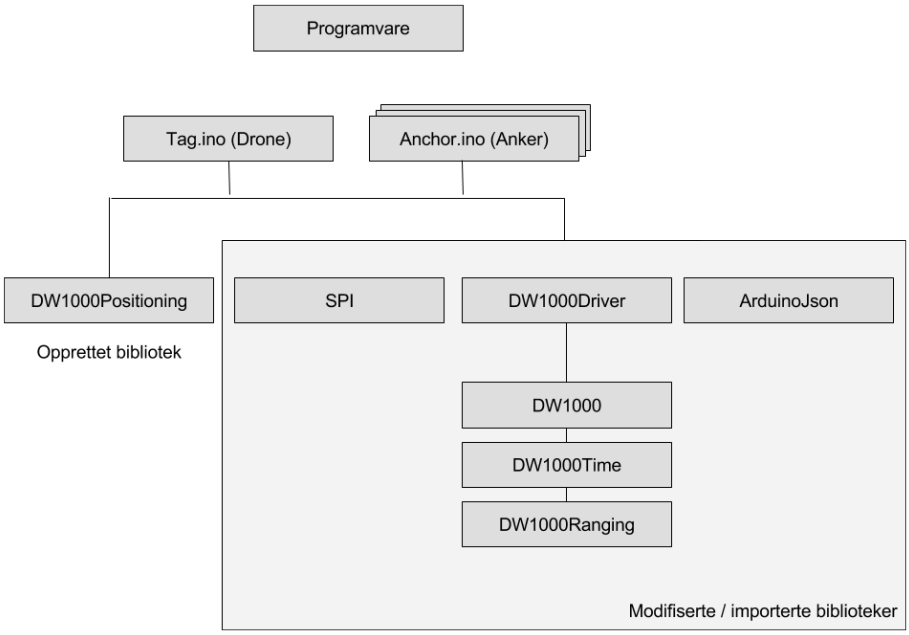
Figur 7: Oversikt over hardware, DW1000 t.v. og Arduini Pro Mini t.h.

2. Arduino Pro og Arduino Pro Mini med ATmega328
   (a) 32 kB Flash memory
   (b) 2 kB SRAM memory
   (c) 8 MHz 8-bit AVR CPU

Basert på spesifikasjonen til brettene skulle man tru at Bluepill med SM32 var den soleklare vinneren, men etter oppsett og testing viste det seg å være vanskelig å kompilere SM32 driveren til DW1000. Det ble funnet en allerede utviklet driver for Arduino Pro, og man byttet mikrokontroller for å komme kjappere i gang.

4.3 Programvare

Koden er bygget på toppen av DW1000 Arduino driveren/biblioteket [5] og splitter koden i tre moduler. To moduler er spesifikt laget for henholdsvis dronen og ankerne mens den tredje (DW1000Positioning) er et tilpasset bibliotek som benyttes av begge. Oppsettet er laget slik at andre enkelt kan ta i bruk systemet. En enkel oversikt vises i figur 8.

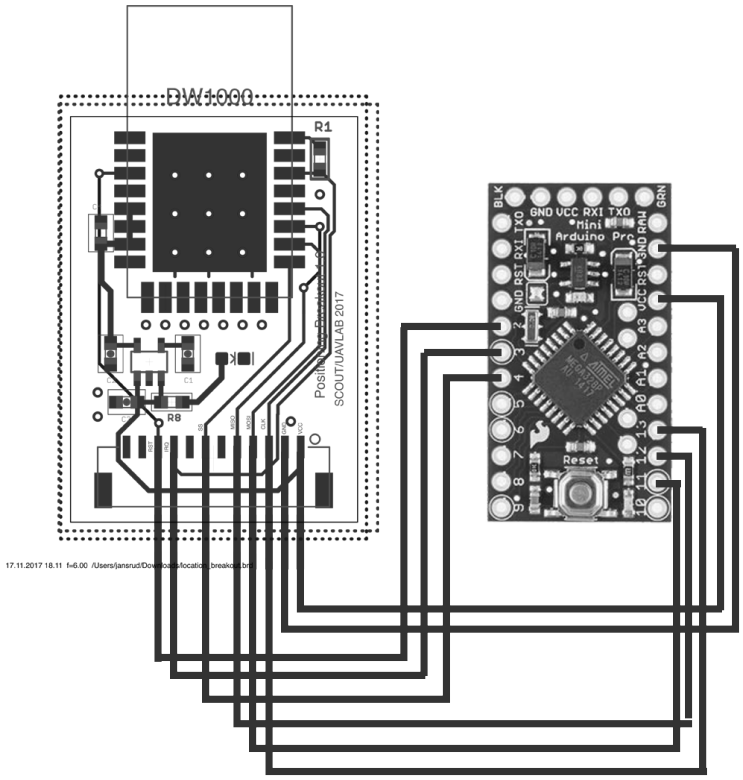


Figur 8: Overordnet oversikt over programvaren.

5 Implementasjon

5.1 Kobling av komponenter

En annen student ved NTNU, Simon Kristoffer Relling Berg, hadde allerede tegnet, bestilt og loddet PCBer for DW1000. Basert på dataark for DW1000 PCB .sch filene og en oversikt over Arduino utgangene koblet jeg mikrokontrolleren og DW1000 sammen som vist i figur 9. Prosessen ble gjentatt til 4x DW1000 var loddet på PCB og koblet til en Arduino. Under prosjektet har jeg kun hatt tilgang på 3x mikrokontrollere, men systemet skal i teorien fungere med alle nodene (4x) tilkoblet. For å forsyne tranceiverne ble en ekstern 5V 2A koblet direkte til sensorene. Dette var den eneste strømforsyningen tilgjengelig, noe som førte til at testingen ble gjennomført over svært korte avstander, som vist i figur 12.



Figur 9: Koblingsskjema mellom Arduino Pro og DW1000 PCB

For å unngå permanente løsninger i prototypen ble det loddet opp koblinger for breadboard og i hovedsak benyttet «jumper wires» som vist i figur 12.

5.2 Funksjonalitet

5.2.1 Melding- og avstandssystem

Funksjonaliteten for å sende data og skaffe avstand mellom to noder var allerede implementert i driveren.

Det som manglet var et system som holdt oversikt over posisjonen og avstand til et kluster av forskjellige sensorer. Valget ble å implementere en liste med objekter som inneholdt informasjon om hver node.

$$Devices = [Drone, Anker1, Anker2, Anker3]$$

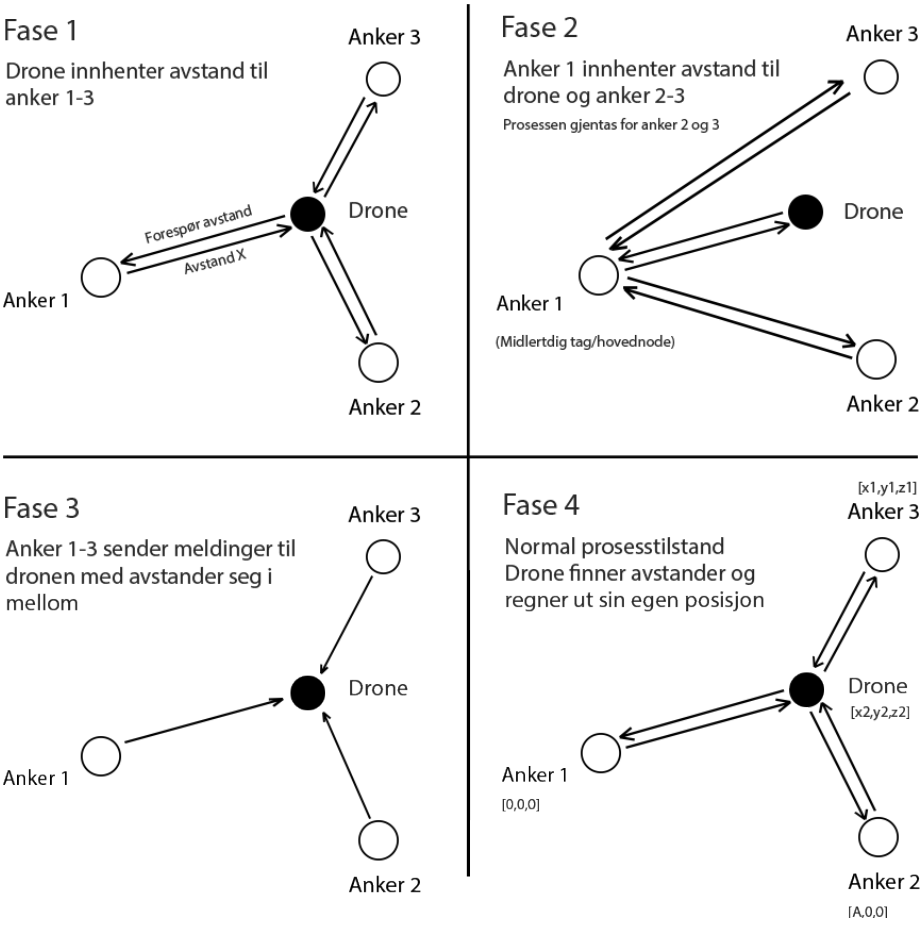
Hvert objekt består av

```
struct _Node{
    bool active;
    uint8_t address = 0;
    struct {
        float x = 0.0;
        float y = 0.0;
        float z = 0.0;
    } position;
    struct {
        float distance = 0.0;
    } distances[_NUM_DEVICES];
};
```

Ved å velge en slik løsning kan man enkelt hente ut avstanden mellom hver enkelt node. F.eks kan man hente ut avstanden fra dronen til node 2 ved å benytte kommandoen

```
Devices[0].distances[2].distance;
```

Det største problemet med oppgaven var å gjøre dronen bevisst over avstandene mellom de forskjellige ankrene. På samme tidspunkt kan det kun eksistere en tag som innehenter avstander til alle de andre nodene. I tillegg kan systemet ikke sende vanlige meldinger samtidig som det innhenter avstander. Løsningen ble å lage en konfigurasjonsfase. I konfigurasjonsfasen går nodene på rundgang over hvem som innhenter avstander til resten. Deretter går systemet inn i en meldingsfase hvor alle avstandene blir sendt over til dronen.



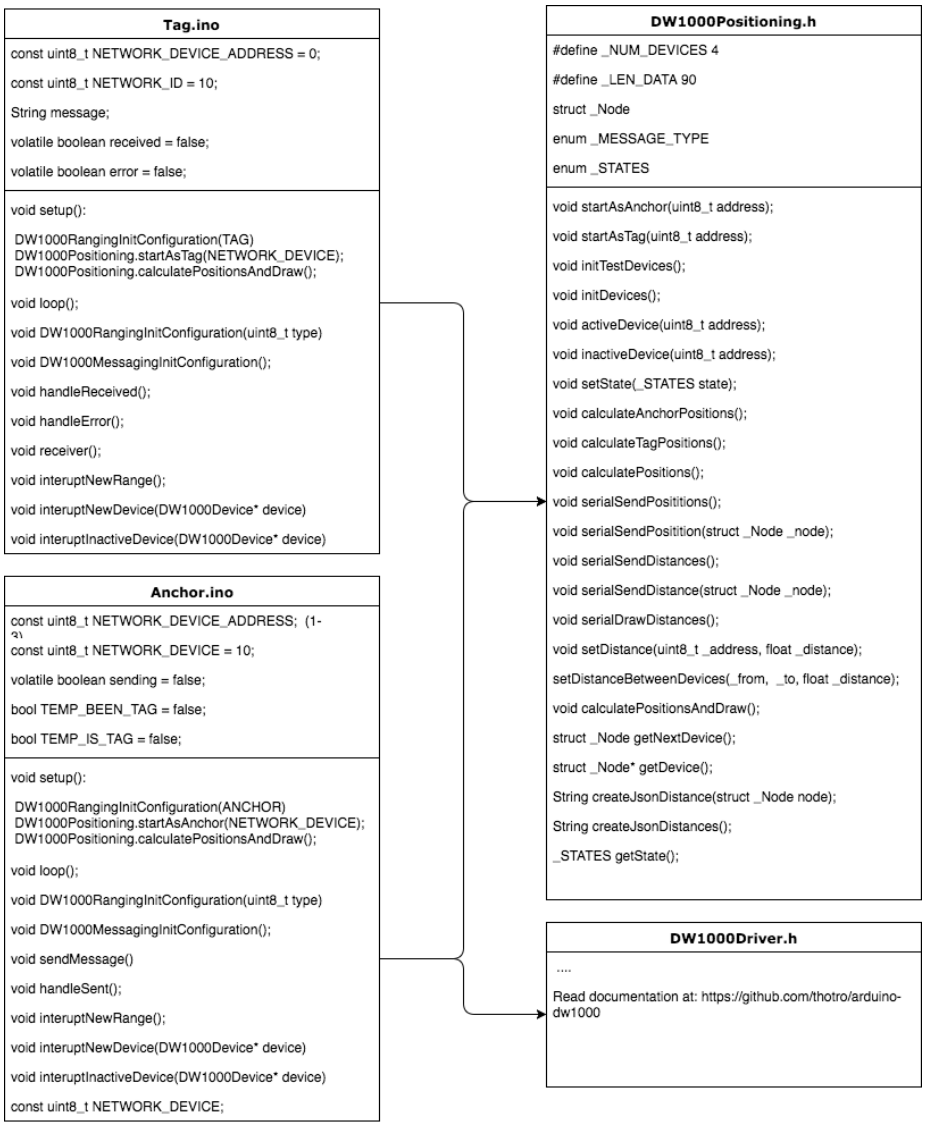
Figur 10: Forklaring om konfigurasjonsfasen

For å gjøre det enklere å sende og forstå meldinger i mellom noder ble det implementert en Json parser. Json er en standardisert måte å enkode og dekode objekter til en string som kan sendes over nettverk. Ulempen med løsningen er at man sender svært mange unødvendige tegn, noe som kan føre til problemer på et sanntidssystem. Mer informasjon diskuteres i kapittel 7.

5.2.2 Posisjonering

Når man allerede har et solid system med oversikt over avstandene mellom alle nodene i nettverket kan man benytte informasjonen for å regne ut posisjonen i henhold til algoritmen utledet i kapittel 3.

5.2.3 Klassediagram



Figur 11: Klassediagram av programvare

5.3 Stegvis bruksanvisning

For å innstallere programvaren, gjennomfør følgende steg:

1. Installer Arduino IDE fra https://www.arduino.cc

2. Åpne en terminal og naviger til Sketchbook

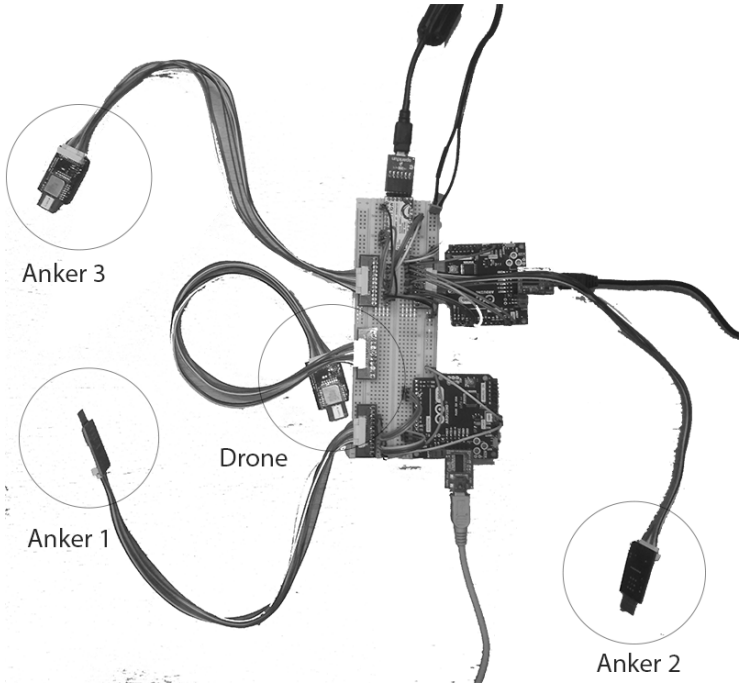
```
cd /your-sketchbook-folder/
```

3. Skaff filene og legg til i Sketchbook

```
git clone
https://github.com/mjansrud/Prosjektemne.git
```

4. Åpne Tag.ino og Anchor.ino
5. Bytt til korrekt NETWORK_DEVICE_ADDRESS. Tag/drone skal ha addresse 0, mens anker-nodene skal ha adresse 1-3 som vist i skisse 7.
6. Last opp koden på mikrokontrollere med påkoblet DW1000.

5.4 Bilde av komplett system



Figur 12: Komplett system bestående av $4xDW1000$ og $3xArduinoPro$. Den siste DW1000 sensoren er ikke aktiv.

6 Resultater

Ved testing mellom to noder med en faktisk rekkevidde på X meter ble følgende resultater registrert :

6.1 Avstandsmålinger

Tabell 1: 0.5m

Avstand *m*	**Styrke** *db*
1.00	-70.66
0.98	-69.95
1.52	-70.65
1.74	-70.40
0.97	-70.54
0.91	-70.13
0.91	-70.73
0.89	-70.24
0.94	-70.28
1.53	-70.37
1.52	-70.65
1.27	-70.07
0.97	-70.44
0.94	-70.14
0.92	-70.18
0.945	-70.32

Tabell 2: 1m

Avstand *m*	**Styrke** *db*
1.55	-75.23
1.49	-75.43
1.53	-75.20
1.56	-74.90
1.56	-74.77
1.55	-75.60
1.56	-75.83
1.56	-74.96
1.54	-74.69
1.54	-75.18
1.54	-75.27
1.56	-74.83
1.56	-75.53
1.59	-75.01
1.55	-75.42
1.44	-75.09

Tabell 3: 2m

Avstand *m*	**Styrke** *db*
2.45	-82.43
2.40	-81.75
2.51	-81.43
2.51	-81.38
2.51	-81.47
2.84	-84.12
2.46	-83.12
2.46	-82.85
2.46	-82.84
2.34	-80.15
2.50	-80.15
2.50	-81.26
2.50	-81.28
2.48	-81.95
2.48	-81.74
2.49	-81.91

Som man ser på resulatene har DW1000 sensorene dersom man benytter Arduino driveren et standardavvik på ca $0.5m$ uavhengig av lengde på målingen. Avviket ble tatt høyde for i implementeringen.

7 Diskusjon

Basert på resulatene har målingene et standardavvik på ca 0.5m. Avviket har blitt tatt høyde for i implementeringen, men rapporten dekker ikke hvor nøyaktige målingene er over lengre avstander hvor det er sannsynlig at avviket varierer i større grad. Et annet problem er at resultatene varierer en del fra måling til måling. En løsning kan være å ta gjennomsnittet av de siste X målingene for å få et mer stabilt resultat. Et annet alternativ er å lage en modell som finner en sannsynlighet for at en måling er korrekt eller feil, og forkaste usannsynlige målinger.

For å sende forståelige meldinger i mellom nodene ble det benyttet en JSON parser som sender svært mange unødvendige tegn i mellom nodene.

$$\{device : 2, range : 0.91, power : -70.13\}$$

I dette eksempelet kan både { , : og variablene fjernes. En mer komprimert melding kunne vært "20.91-70.13", men denne formen er ikke standardisert og er vanskeligere å dekode. JSON biblioteket ble benyttet for å gjøre sending og mottak enkelt og kommunikasjonen er vellykket i de aller fleste tilfeller.

Systemet er utviklet med en konfigurasjonsfase hvor alle nodene finner avstandene til hverandre før informasjonen sendes til dronen. Problemet er at hardware må aktiveres ca på samme tidspunkt for at alle enhetene er sykroniserte og konfigurasjonen skal fungere. Nåværende system er upraktisk og vil ikke fungere optimalt i en komersiell setting.

Det er i tillegg vanskelig å feilsøke hver enkelt node dersom man ikke har koblet mikrokontrollere til en PC med seriell debugging. Det burde utvikles et system hvor hver enkelt node kan melde fra om feil, f.eks i en situasjon hvor noden ikke har greid å innhente avstand til et annet anker. Forslag til system er LED lampe, skjerm eller feilmeldinger sendt over trådløst nettverk. Et problem med det siste forslaget er at noden må fungere på nettverket. En kombinasjon av forslagene kan være det beste alternativet.

8 Konklusjon

Det er fullt mulig å lage et lavbudsjett og presist posisjoneringssystem basert på DW1000 sensorer og billige mikrokontrollere. I konfigurasjonsfasen må hver node innhente avstander fra resten på rundgang. Basert på nåværende system må nodene aktiveres nogenlunde likt for at dette skal fungere, noe som er upraktisk i felt. Et system hvor dronen aktiverer konfigurasjonsfasen burde utvikles.

Bibliografi

[1] Wikipedia ultra wideband. https://en.wikipedia.org/wiki/Ultra-wideband, 2017.

[2] Decawave. Produktinformasjon decawave scensor dw1000. https://www.decawave.com/products/dw1000, 2017.

[3] Marinewiki. Iacs - safer and cleaner shipping. http://www.marinewiki.org/index.php?title=File:Single$_skin_bulkcarrier-typical_structure.jpgfilelinks$, 2011.

[4] Sewio. Informasjon om decawave twr (two way ranging protocol. https://www.sewio.net/uwb-sniffer/analyzing-decawave-two-way-ranging-twr/, 2017.

[5] T. Trojer. A library that offers functionality to use decawave's dw1000 chips/modules with arduino. https://github.com/thotro/arduino-dw1000, 2017.