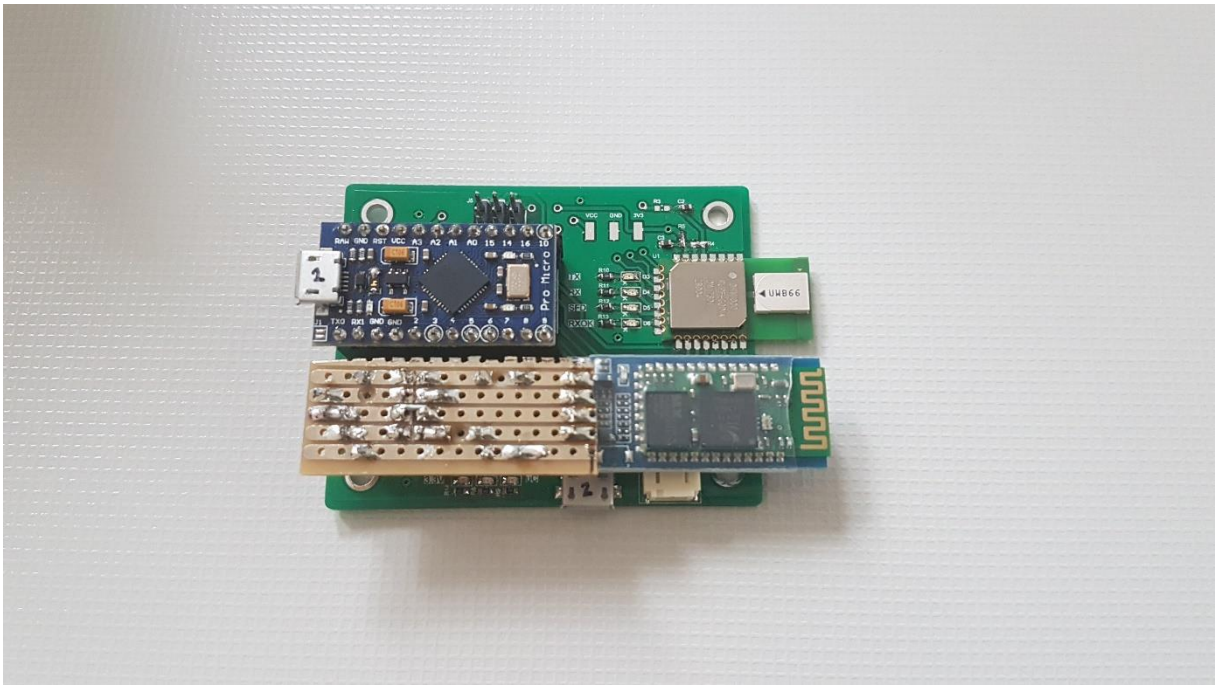
CHALMERS



Inomhuspositionering med UWB och Bluetooth

Studie av kommersiella produkter och konstruktion av eget system

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

Olof Norrby
Johan Friberg

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2018

FÖRORD

Mekatronikprogrammet.
Chalmers tekniska högskola.

Utbildningen omfattar ämnena Ellära, mekanik, programmering och reglerteknik.
Examensarbetet omfattar ämnena Ellära, programmering och reglerteknik.

Medverkande har varit Lärare på Chalmers och handledare från företaget Virtual Manufacturing.

Extra tack till Mikael Miettinen vår handledare på Virtual Manufacturing, Daniel Muhr och alla dem som hjälp oss på Virtual Manufacturing. Stor hjälp också från programansvarig på Chalmers, Veronica Olesen och till vår handledare Jian Yang professor på Chalmers.

SAMMANFATTNING

Dagens utveckling med fordon/ farkoster och robotar som skall kunna köra runt autonomt på en tillförlitlig nivå kräver stora krav på hur de navigerar och rör sig. Företaget Virtual Manufacturing har haft en önskan om att sätta sig in i denna utveckling, då med fokus på två typer av tekniker passande för just inomhuspositionering, då detta område fortfarande är under stor utveckling. Den ena tekniken är en av dem allra vanligaste radio-teknikerna som finns ute på marknaden och är Bluetooth low energy (BLE). Den andra tekniken som arbetet behandlar är Ultra wideband radio (UWB), dock är den sistnämnda inte så vanlig på marknaden ännu då den fortfarande är under utveckling. Med dessa två radio-tekniker har en studie tagits fram för att undersöka vilka kommersiella produkter som finns på marknaden idag och hur dessa står sig mot varandra. Vidare i arbetet behandlas en djupare del om just UWB-tekniken och börjar med byggandet av ett komplett system för inomhuspositionering med då fyra enheter som använts. Vidare arbetas matematiska modeller fram för att kunna implementeras i vardera enhet. Detta gör det möjligt att plocka fram koordinater för t.ex. en självkörande truck i en industrilokal. Positionen skall optimeras med avseende på den data som systemet ger och det kan realiseras med filter. Dessa filter fungerar som en slags regulator och ser till att de då efterbehandlade mätningarna blir mer exakta. Avslutningsvis görs en jämförelse med det system som byggts under projektet med studien om kommersiella produkter och ser hur det står sig mot marknaden.

ABSTRACT

The development of today's industry with vehicles and robots that are supposed to autonomously drive around on a reliable enough level sets great demands on how they are to navigate and move around. The company Virtual Manufacturing have the desire to embark on this development, then with focus on two types of technologies suitable for indoor positioning since this is still an area under greater development. One of the technologies we're using is one of the most common ones found on the market today, called Bluetooth low energy (BLE). The other technology that is treated is Ultra wideband radio (UWB), though the last named is not that common on the market yet since its still kind of a new technology there. With these to radio-technologies in mind, a study has been developed to investigate which commercial products are on the market today and how they stand against each other.

In addition, a more thorough part of the UWB-technology will be dealt with, starting with the construction of a complete indoor positioning system, with four units being built. Furthermore, mathematical models are being developed to later on be implemented into each built unit, this makes it possible to get coordinates from example a self-driving forklift in an industrial building. The position should be optimized regarding the data provided by the system and can be realized with filters. These filters work like kind of a regulator and makes sure the post-processed readings become more exact. In conclusion, a comparison is made between the system built under the project and the Commercial Product Study to show how it stands against the market.

INNEHÅLL

1. INLEDNING

1.1 BAKGRUND

Företaget Virtual Manufacturing är en leverantör av leanbaserade produktions- och logistikutvecklingstjänster och [1] arbetar bland annat med att utforska, testa och hitta användningsområden med ny teknik. Här kommer intresset in på inomhuspositionering och det är ett område som inte har fått så stor uppmärksamhet tidigare, därmed vill Virtual Manufacturing satsa på att utforska just detta område och se vad det finns för potential. Störst fokus är på teknikerna BLE och UWB.

1.2 SYFTE

Uppdraget som Virtual Manufacturing har beställt innefattar att först undersöka marknaden efter existerande tekniker av inomhuspositionering, då med tekniken UWB i centrum i och med att den tekniken kan leverera bäst precision. Därefter skall arbetet gå vidare till att testa ett färdigt Tracking-system och se hur vi med den data vi får in kan optimera position och hantera så kallad “drifting” som kan uppstå.

1.3 AVGRÄNSNINGAR

Då uppdraget har fokus på inomhuspositionering kommer tekniker för positionering utomhus ej att behandlas i någon större utsträckning. Det kommer ske huvudsakligen med teknikerna UWB och/eller BLE, en annan teknik som då inte heller blir aktuell för positionering är Wifi då denna teknik inte kan leverera särskilt bra precision.

1.4 PRECISERING AV FRÅGESTÄLLNING

- Vilka kommersiella modeller finns på marknaden?

- Hur står de sig mot varandra?

- Kan ett delvis open-source projekt mäta sig med kommersiella produkter?

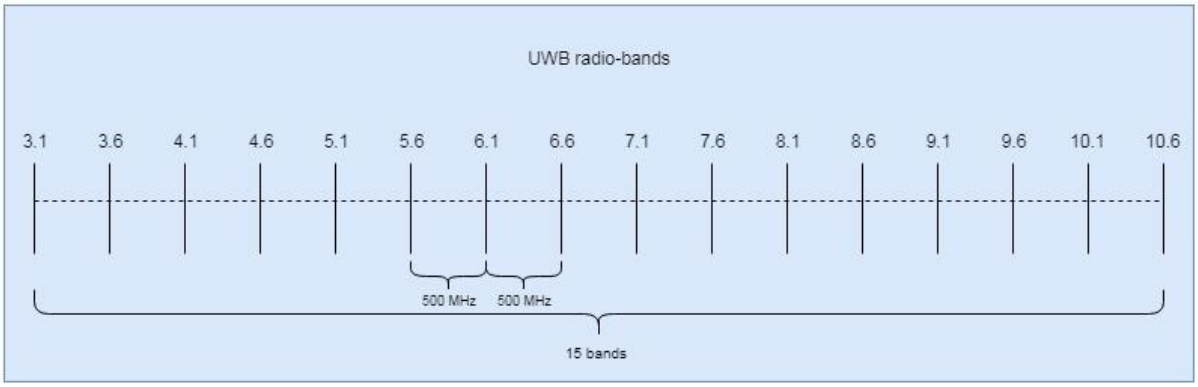
- Vad finns det för användningsområden för UWB?

- Hur skall precisering av position optimeras med hjälp av den data som erhålls.?

2. TEKNISK BAKGRUND

2.1 ULTRA WIDE BAND (UWB)

Ultra Wideband även kallat UWB, är en radioteknik som bygger på användandet av ett mycket brett radiobands spectrum. I vanliga fall används så kallade smalband, användning av radio över en specifik centerfrekvens, precis som radion i bilen. Där vill man skicka data över en specifik frekvens medans med UWB används ett mycket brett spektrum av radiofrekvenser. Standard är att frekvensbandet ligger på 500 MHz över frekvenser från 3.1 GHz till 10.6 GHz. (Se figur 2.1).



Figur 2.1, Beskriver de 15 frekvensbanden i UWB radio.

Då ser man snabbt att UWB kommer ligga över vanliga populära frekvenser som t.ex. 2.4 GHz som används av bland annat Bluetooth och radio för modellflygplan, men på grund av det breda spektrum som UWB ligger över behöver och får UWB ej ha hög effekt på signalen. Detta leder till att andra radiomottagare kommer se UWB-signalerna som brus och detta kan då filtreras bort [2]. En UWB mottagare kan sedan kombinera signalerna för att få en puls som kan användas vid beräkning.

Redan 1901 kom ideén fram med att använda UWB teknik [3], då med syftet att använda det för morsekod. Tekniken är därmed en gammal idé som egentligen inte fått större uppmärksamhet förrän UWB blev kommersiellt tillåtet av FCC år 2002. Därefter har tekniken börjat växa fram med en mängd användningsområden, t.ex. dataöverföring (likt trådlöst USB), avståndsmätning och radar.

2.2 BLUETOOTH LOW ENERGY (BLE)

Bluetooth Low Energy är en trådlös kommunikationsstandard utvecklad 2006 av Nokia. Den gick tidigare under namnet Wibree men standardiserades år 2010 som Bluetooth Low Energy (BLE) [4].

BLE använder samma frekvensband som Bluetooth, 2.4 GHz och konstruktioner som använder det kan därför använda samma antenn till båda varianterna.

Även BLE har använts till positionering, bland annat av Apple genom deras iBeacon-protokoll [5] där vilken iOS-enhet som helst med stöd för dataöverföring genom BLE kan konfigureras som ett Ankare eller en Mottagare. En av fördelarna med att använda BLE för positionering är att teknologin redan finns som standard i många kommersiella apparater, t.ex. smarta telefoner, surfplattor och datorer. Nackdelen är det relativt trånga frekvensbandet (ca 80 MHz, jämfört med UWB >500 MHz) vilket leder till en längre pulstid och en sämre träffsäkerhet när man uppskattar positionen.

2.3 ARDUINO

Arduino är en open-source plattform baserad kring Atmels mikrokontroller. De finns i flera olika varianter men i detta projekt används den Arduinokompatibla modellen Arduino Pro Micro.

2.3.1 ARDUINO PRO MICRO

Arduino Micro använder Atmels ATmega32u4 och finns i två olika varianter, 5 V och 3.3 V vilka har en frekvens på 16 MHz respektive 8 MHz. För detta examensarbete används 3.3 V-modellen eftersom andra komponenter i kretsen, t.ex. DWM1000-modulen, endast var kompatibla med 3.3 V. Den är även en av de fysiskt minsta modellerna på 3.3 x 1.8 cm.

2.3.2 ARDUINO IDE

Arduinos utvecklingsmiljö använder ett programspråk baserat på C/C++, där funktionerna och koden som skrivs i Arduinos IDE automatiskt konverteras till C/C++ innan den skickas till kompilatorn. Eftersom plattformen är open source så finns det ett stort community och en stor databas med privat kodade bibliotek för olika moduler och sensorer. I detta arbete används ett bibliotek för Decawaves DWM1000-modul gjort av Thomas Trojer.[6]

2.4 CAD OCH 3D-PRINTING

CAD i kombination med 3D-printing är ett snabbt och smidigt sätt att designa och konstruera prototyper av chassin och annan kringutrustning för att skydda elektroniken. För detta projekt har det använts en printer vid namn Flashforge Creator Pro för att printa ett chassi till Mottagaren och stativ samt hållare till alla Ankare. CAD-programmet som använts har varit Catia V5.

2.5 KALMAN-FILTER

Insamlad data från ett system kan regleras med olika metoder, att applicera ett filter kan öka precisionen på datan som givits. Alltså att inkommande data får efterbehandlas av ett filter. Kalman-filter som används i detta projekt är ett mycket kraftfullt filter / verktyg med styrkan i att ju fler källor som kan bidra till att precisera data desto effektivare kan filtret arbeta. Vanliga användningsområden för Kalman-filter är bland annat Ranging och Positioning [7], därmed ett mycket passande verktyg då man t.ex bygger självkörande robotar eller AGV:er.

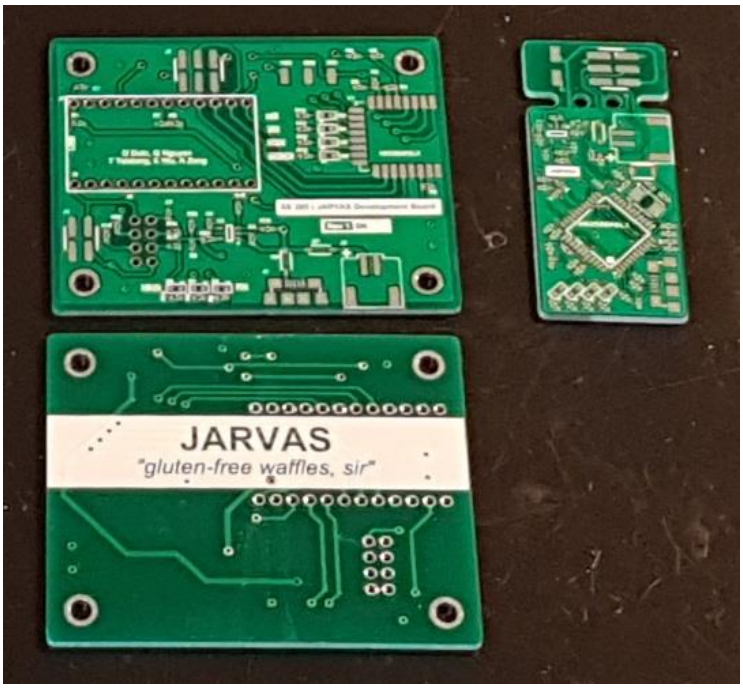
2.6 MATLAB

Matlab är ett programmeringsspråk med sina rötter i C [8] men med sina egna funktioner. Fördelarna med matlab är alla dess "verktygslådor" som går att ladda ned beroende på vad du jobbar med. Simulink är en av dem mer kända verktygen i matlab och kan t.ex. användas för att smidigt autogenerera arduino-kod utifrån ett system du byggt i simulink, men för detta krävs en liten insticksmodul. Matlab används främst för att utföra svåra matematiska beräkningar för att sedan ge möjligheten att presentera resultaten i exempelvis grafer.

2.7 JARVAS

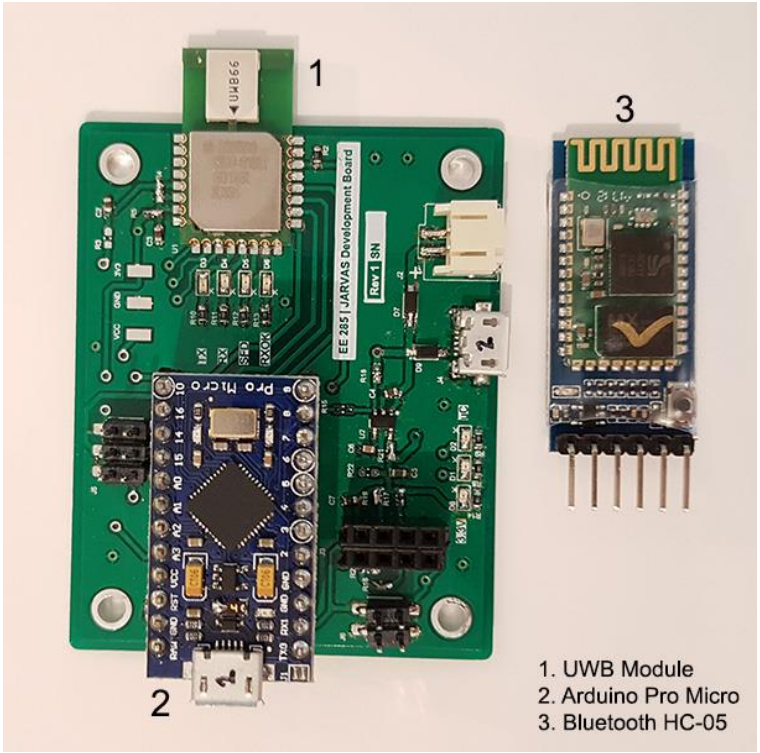
JARVAS är ett tidigare examensarbete inom inomhuspositionering skapat av D. Dutz, Q. Nguyen, T. Teisberg, E. Wu och A. Zeng vid Stanford University. Efter sitt arbete delade de med sig av projektets hård- och mjukvara som ett öppet projekt som alla kunde få tillgång till över internet.

Hårdvaran består av tre ankare och en mottagare som ska spåras. Alla ankare använder här samma hårdvara medan mottagare är en mycket mindre enhet som istället för att använda en Arduino har en egen mikrokontroller på PCB-kortet (Se figur 2.2).



Figur 2.2 PCB på Ankare (fram och baksida) t.v. och Tag (framsida) t.h

Hårdvarans huvudkomponenter är mikrokontrollern ATmega32u4 och UWB-modulen DWM1000 (se figur 2.3), där UWB-modulen sköter kommunikationen mellan mottagaren och systemets ankare för att bestämma avstånden mellan dessa. Dessa avstånd skickas sedan till mikrokontrollern där den filtreras och sedan beräknas mottagarens position. Positionen skickas sedan till en molntjänst via en WiFi-modul vid namn ESP8266 för att kunna sparas och visas upp grafiskt.



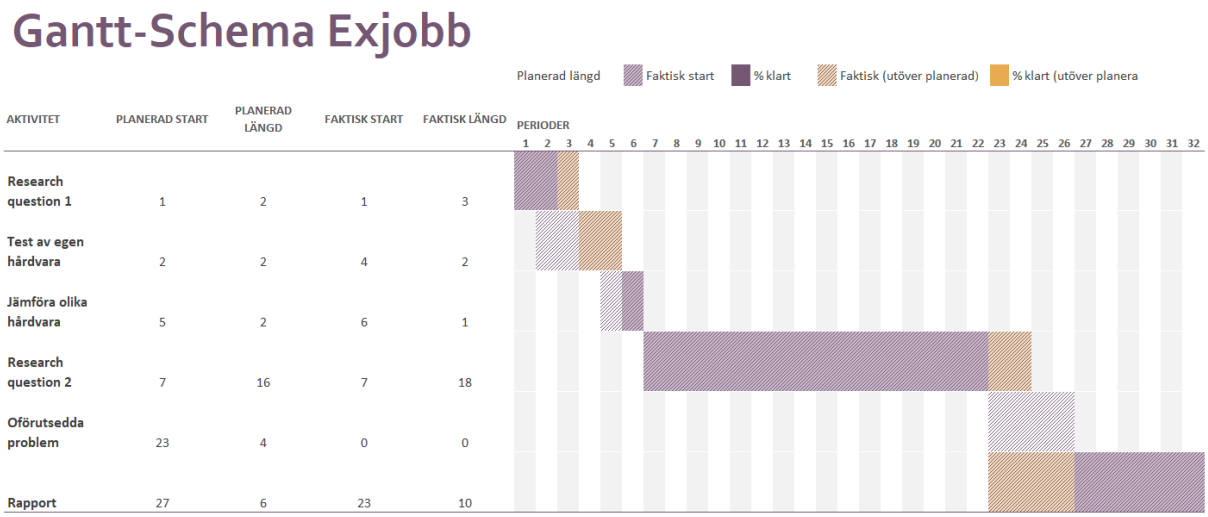
Figur 2.3 Bild med UWB modul, Arduino och Bluetooth chip

3. METOD

Projektet har bestått av flera olika steg enligt underrubrikerna nedan. Det första som gjordes var att läsa på om teknikerna UWB och BLE, samt skapa en referensram för vilken prestanda som ansågs vara troligt att uppnå genom att studera redan existerande kommersiella produkter på marknaden. Därefter påbörjades konstruktionen av det egna systemet baserat på hårdvara från ett open source-projekt[9].

3.1 PLANERINGSSCHEMA

En planeringsrapport togs fram i början av arbetet och då även ett Gantt-schema (se figur 3.1) för att uppskatta den tid som behövs för respektive del i projektet.



Figur 3.1 Gantt schema för planerad tidsåtgång

3.2 INHÄMTA INFORMATION OM TEKNIKERNA UWB OCH BLE.

Då UWB tekniken är en relativt ny teknik och har ännu inte riktigt växt fram som en vanlig standard gällande radio-tekniker är informationshämtning något svårare än för den vanligaste tekniken BLE. Studier från universitet [10] [11] och tekniska beskrivningar från tillverkare [12] [13] av UWB-moduler ger dem bästa redskapen för att förstå UWB. Gällande BLE finns mycket information att hämta då detta är den vanligaste radiotekniken för styrning och informationsöverföring på kortare distanser.

3.3 STUDIE AV KOMMERSIELLA PRODUKTER FÖR INDOOR TRACKING.

Inledningsvis påbörjades en studie av vilka kommersiella produkter som går att använda till inomhuspositionering, då med fokus på produkter med teknikerna UWB och BLE. Analys av några företag gav en bild av vilka områden studien skall täcka.

Viss information kunde ej hittas på tillverkarnas hemsidor och därmed var direktkontakt med företagen vad som kunde ge den information som behövdes för studien. Studien avslutas med jämförelse-grafer mellan dem företag som valts ut i studien.

3.4 BYGGA SYSTEMET

Systemets hårdvara baserades på ett open-source projekt vid namn JARVAS[9]. PCB-korten beställdes från Kina och komponenterna beställdes från Sverige och USA.
Komponenterna monteras med lödpasta och en lödpincett eftersom designen använde ytmonterade komponenter.

3.5 OPTIMERING AV POSITION VID INDOOR TRACKING

Då positioneringen grundar sig i avståndsmätning tillsammans med trilaterering finns det två faktorer som går att använda för att optimera positionen. Det första är att kalibrera korrekt, det avser att ta hänsyn till fördröjningar som uppkommer samt korrekt uppmätta positioner för vardera ankare som sedan programmeras in. Det andra som går att göra är att optimera den data som erhålls efter avståndsmätningarna. Då först gäller det positioneringsmetod, i detta fall används trilaterering med minsta kvadratmetoden.

3.6 TEST OCH JÄMFÖRELSE MED EGEN HÅRD-/MJUKVARA.

Systemet testas och jämförs med de egenskaper som beskrivs i studien av kommersiella produkter och ser hur det står sig gentemot dem. Viktiga aspekter här är priset och precisionen som kan uppnås då en del av syftet med arbetet har varit att se hur väl ett egenbyggt system står sig mot kommersiella system. Systemet testas i CLOS (Clear Line Of Sight) vid stillastående och vid förflyttning av mottagare.

4. STUDIE AV KOMMERSIELLA POSITIONERINGSSYSTEM

Studien baserades på ett fåtal punkter som ansågs vara relevanta för att kunna jämföra de olika kommersiella produkterna som finns på marknaden. Ett av kraven för att en produkt skulle kunna vara med i studien var att den skall bygga på en radioteknik som går att användas till inomhuspositionering. Några företag hade produkter med sådan radioteknik men föll bort på grund av att designen för just den produkten gjorde att den inte var lämpad för inomhuspositionering. De områden som studien jämför är följande:

- Typ av teknik (mjukvara/hårdvara).
- Frekvensband
- Precision vid positionering.
- Tänkt användningsområde/ möjliga användningsområden.
- Pris.
- Min/Max antal ankare och noder.
- Andra för/nackdelar t.ex. batteritid, vattentäthet, storlek, räckvidd

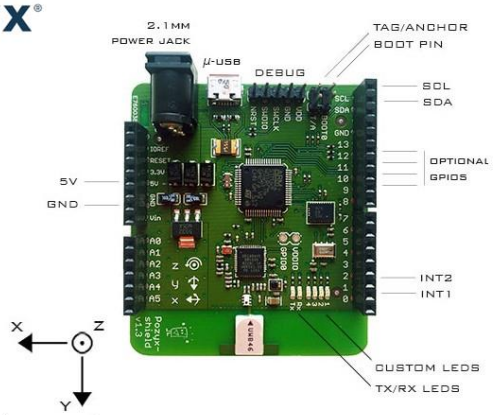
Studien består av en kort beskrivande bild för varje företag med information om just dessa områden för var och en av dem som kom med i studien. (Se figurer 4.1 till 4.6)

Figur 4.1 Ruuvi tag [14], teknisk jämförelse

Assembly & Production Flow

Pozyx

https://www.pozyx.io/



- Teknik
  - UWB
- Frekvensband
  - 6 kanaler mellan 3.5 GHz och 6.5 GHz
- Precision
  - Upp till 10 cm
- Användningsområden
  - Inomhuspositionering, motion tracking
- Pris
  - 599 Euro för en tag och fyra ankare
- Antal ankare och mottagare
  - Mellan 3 och 16 ankare, finns även möjlighet till multi tracking (flera tags).
- Annat
  - Möjlighet till både 2D och 3D-positionering. Rörelesesensorer.

Brings Value in Production.

VIRTUAL manufacturing.

Figur 4.2 Pozyx [15], teknisk jämförelse

Assembly & Production Flow

Estimote

https://estimote.com/

- Teknik
  - BLE kombinerat med UWB
- Frekvensband
  - Positionering använder 2.4 GHz till 2.48 GHz (Bluetooth Standard). Beacons använder 6 kanaler mellan 3.5 GHz och 6.5 GHz för avståndsmätning
- Precision
  - 1 till 4 meter beroende på avstånd och terräng
- Användningsområden
  - Avståndsmätning och X, Y-positionering
- Pris
  - 159 USD för 4 UWB beacons
- Antal ankare och mottagare
  - 4 Beacons
- Annat
  - Upp till 200 meter räckvidd. IPX4.

Brings Value in Production.

VIRTUAL manufacturing.

Figur 4.3 Estimote [16], teknisk jämförelse

Assembly & Production Flow

InfSoft

https://www.infsoft.com

- Teknik
  - UWB/BLE/GPS/Kamera
- Frekvensband
  - 2.4 GHz (BLE Standard), 3.1 GHz till 6.9 GHz (UWB)
- Precision
  - 10-30 cm
- Användningsområden
  - Tracking och dataöverföring
- Pris
  - Varierar beroende på valda funktioner. Noder börjar på 70 Euro/st.
- Antal ankare och mottagare
  - En tag och 3+ ankare (Nodes)
- Annat
  - Lång batteritid (~1 år). Liten storlek.

infsoft Locator Node

Brings Value in Production.

VIRTUAL manufacturing.

Figur 4.4 InfSoft [17], teknisk jämförelse

Assembly & Production Flow

Wittra

www.wittra.se

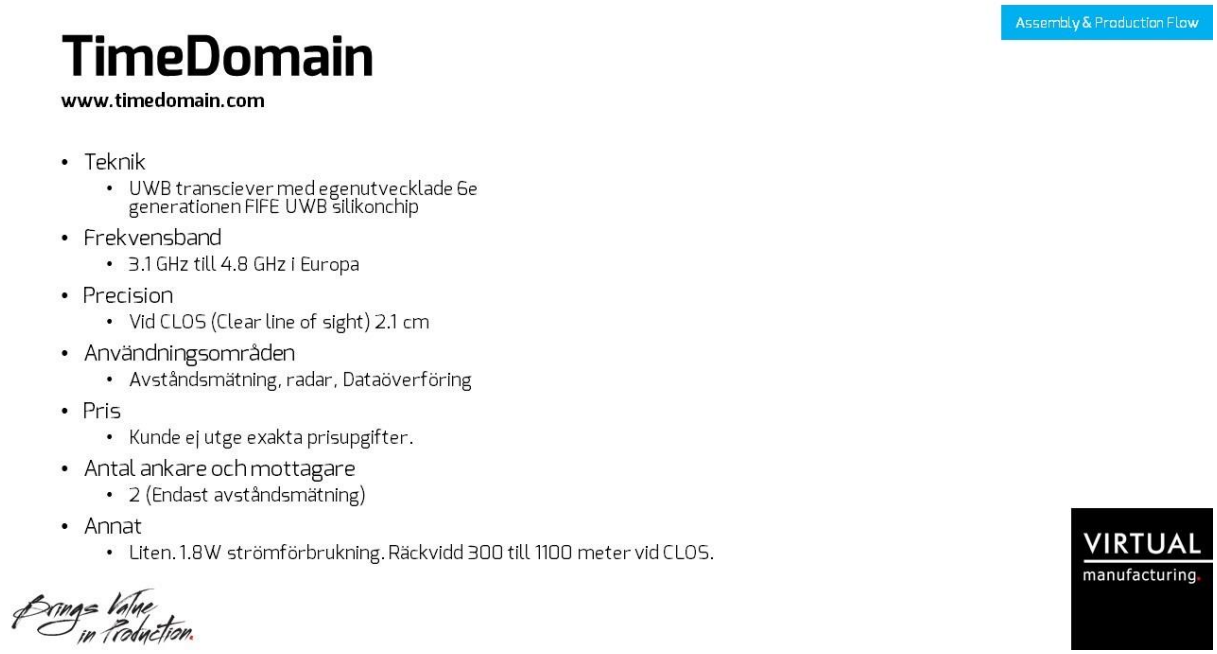
RED HERRING 100 WINNER N. AMERICA

- Teknik
  - ISM,BLE,GPS
- Frekvensband
  - EU 868 MHz, US 915 MHz
- Precision
  - 3-5 Meter
- Användningsområden
  - Tracking och Övervakning
- Pris
  - Tag 40 USD/st och Ankare 80 USD/st
- Antal ankare och mottagare
  - 10 000+ Noder
- Annat
  - Lång batteritid (minst 12 mån). Liten storlek. Modulär, Accelerometer

Brings Value in Production.

VIRTUAL manufacturing.

Figur 4.5 Wittra [18], teknisk jämförelse



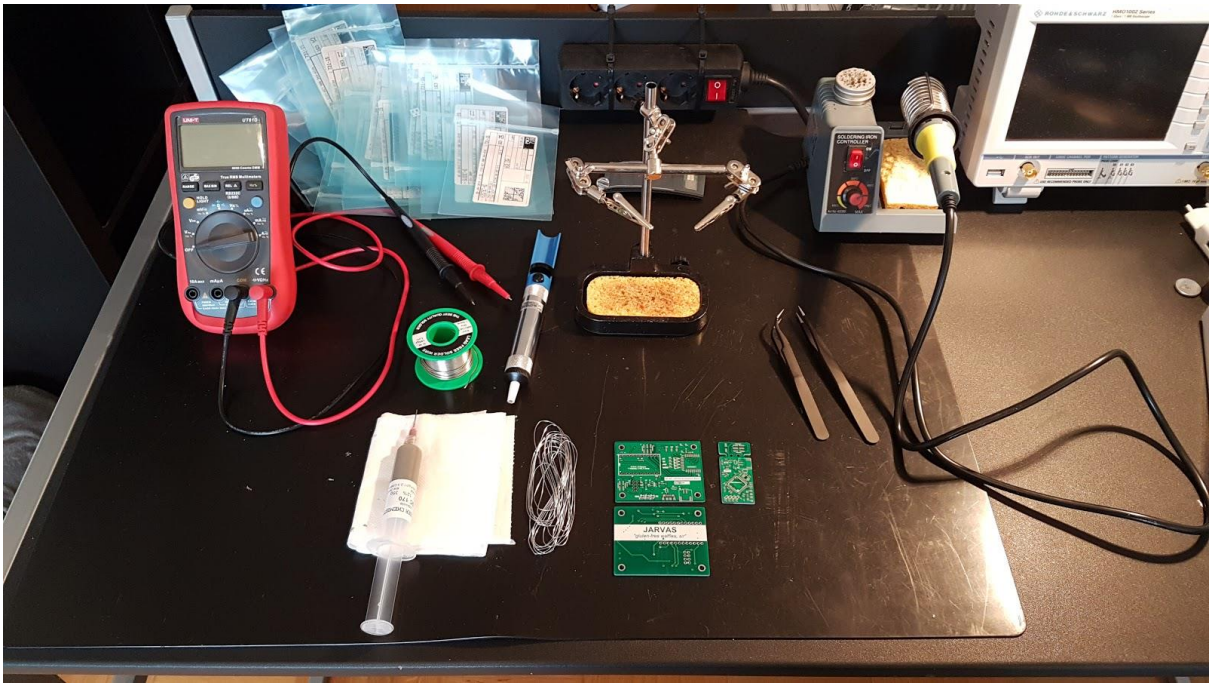
Figur 4.6 TimeDomain [19], teknisk jämförelse

5. GENOMFÖRANDE

5.1 HÅRDVARA

Till skillnad från det ursprungliga projektet (JARVAS) så använder alla fyra enheter här samma hårdvara, med undantaget att mottagaren har en inkopplad Bluetooth-modul för att kunna skicka sin position vidare till en Androidenhet, PC eller liknande för att spara informationen och en eventuell grafisk presentation. Även batterierna är annorlunda här då detta projekt använder batterier på ca 3000 MAh jämfört med de ursprungliga projektets 850 MAh.

Alla fyra enheter (mottagare och ankare) byggdes för hand med de verktyg som ses på bilden nedan (figur 5.1). Inledningsvis löddes komponenterna med vanligt lödtenn och lödkolv. Efter det att första enheten byggts klart användes lödpasta för de resterande enheterna. Alla komponenter är av typ SMD, ytmonterade komponenter i storleksklassen 0306. De SMD komponenter som använts på varje enhet är: resistorer, kondensatorer, lysdioder, spänningsomvandlare, USB-connector, Batteri-connector, schottky-dioder och connector-pins för debugging. Utöver dessa är det en Arduino och ett UWB-chip. Det är endast Arduinons connector-pins som är av hålmonterad typ.



Figur 5.1 Översiktsbild av arbetsstation samt de komponenter och verktyg som användes.

5.2 AVSTÅNDSMÄTNING

För att bestämma mottagarens koordinater behöver systemet först veta avståndet mellan mottagaren och respektive ankare. För att beräkna dessa avstånd använder vi "Two-way ranging", vilket betyder att mottagaren först skickar en puls till ankaret som svarar med att skicka en puls tillbaka. Detta ger oss en tid T_{round}för hela förloppet (se figur 5.2).

Med T_{round} och T_{reply} kända kan vi sedan beräkna den tid det tog för signalen att färdas mellan mottagare och ankare, ToF, vilket står för Time of Flight.

$$2 * ToF = T_{round} - T_{reply}$$

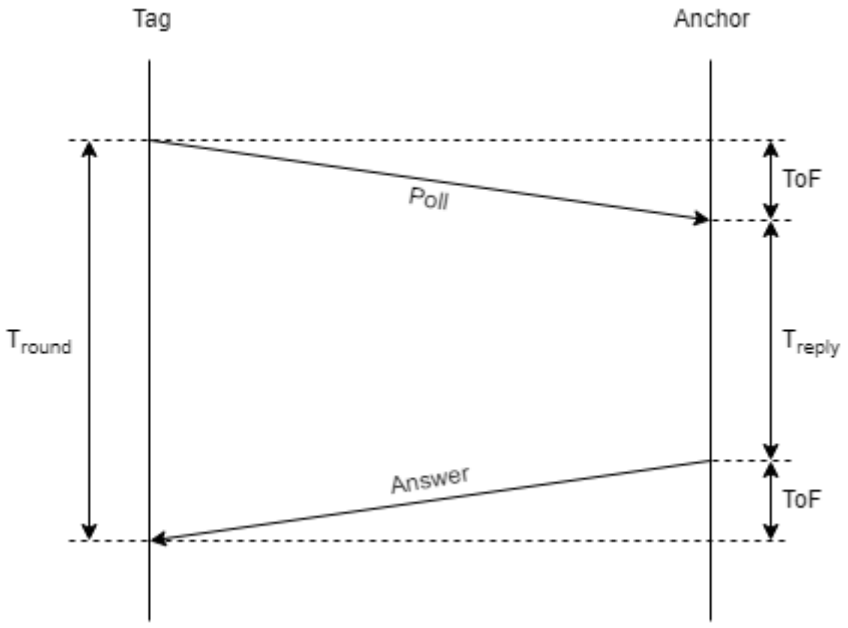
Vilket ger:

$$ToF = \frac{T_{round} - T_{reply}}{2}$$

Då signalerna färdas med ljusets hastighet kan vi med ToF nu beräkna avstånden

$$l = c * ToF * f_{klocka}$$

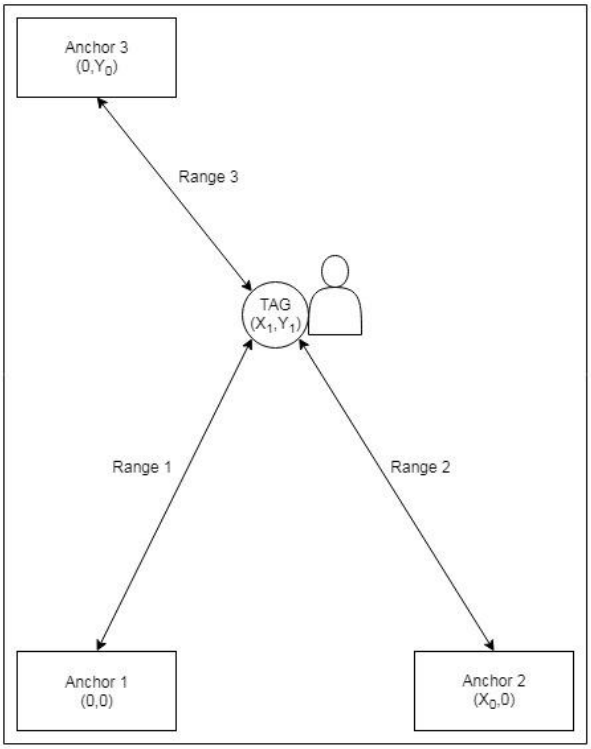
där l är avståndet mellan mottagaren och ett ankare, c är ljusets hastighet i vakuum och f_{klocka} är klockfrekvensen.



Figur 5.2 En grafisk representation av Two-way ranging.

5.3 TRILATERERING

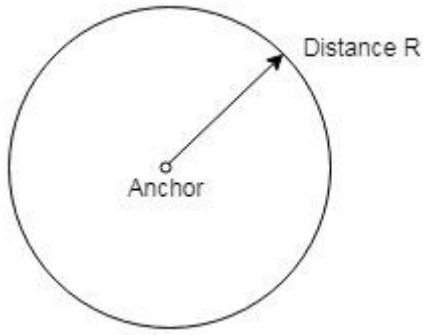
Det första som gjordes var en trilatererings-modell i form av ett linjärt [20] minsta-kvadratmetoden system. Först måste ett fysiskt koordinatsystem upprättas i det rum som man vill ha positionering i. Därefter sätts tre ankare ut i rummet (figur 5.3) och koordinaterna för vardera ankare mäts i förhållande till koordinatsystemet, där ett ankare är satt i origo. Därefter kan varje ankare skicka sitt uppmätta avstånd till mottagaren som därefter utför beräkningarna via minsta-kvadratmetoden och kan skicka vidare sina X och Y koordinater till önskad plattform.



Figur 5.3 Beskrivande bild av hur systemet sätts upp.

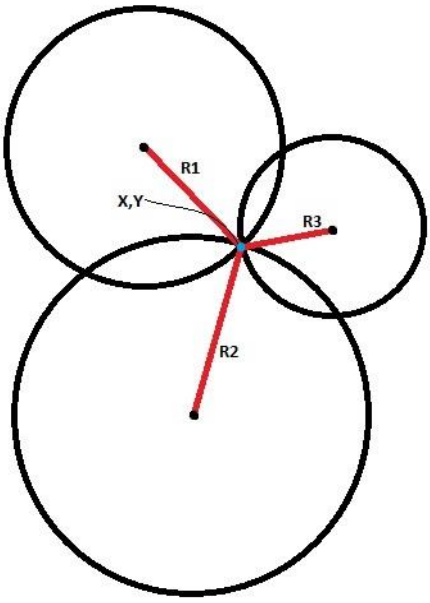
Själva trilatereringen beräknas utifrån en matematisk modell.
Vi tänker oss att mottagaren tagit emot en signal från ett ankare och beräknar därefter ett avstånd. Problemet nu är att avståndet mellan enheterna inte kan specificeras mot en riktning.

Därför kan modellen för endast avståndsmätning beskrivas som en cirkel där ankaren är i centrum och mottagaren är någonstans vid ytterradien (se figur 5.4).



Figur 5.4 Beskriver radien av en cirkel runt ett ankare

Som vi ser i figur 5.5 nedan kan en position utifrån tre uppmätta avstånd beräknas och det blir då där de tre cirklarna möts i samma punkt, det är detta som kallas för trilaterering.



Figur 5.5 Trilaterering mellan 3 ankare och en tag.
Tag är den blå pricken där cirklarna möts

Avståndet beräknas från vartdera ankaret utefter följande modell:

$$R1 = \sqrt{(x - x_1)^2 + (y - y_1)^2} \quad (1)$$
$$R2 = \sqrt{(x - x_2)^2 + (y - y_2)^2} \quad (2)$$
$$R3 = \sqrt{(x - x_3)^2 + (y - y_3)^2} \quad (3)$$

Där x och y är koordinaterna för mottagaren. (x_1, y_1), (x_2, y_2), (x_3, y_3), är kända koordinater för vardera ankare och $R1$, $R2$ samt $R3$ är det uppmätta avståndet. Om vi sedan kvadrerar båda sidor av ekv (1), (2) och (3) fås följande uttryck:

$$R1^2 = (x - x_1)^2 + (y - y_1)^2 = x^2 - 2x * x_1 + {x_1}^2 + y^2 - 2y * y_1 + {y_1}^2 \quad (4)$$
$$R2^2 = (x - x_2)^2 + (y - y_2)^2 = x^2 - 2x * x_2 + {x_2}^2 + y^2 - 2y * y_2 + {y_2}^2 \quad (5)$$
$$R3^2 = (x - x_3)^2 + (y - y_3)^2 = x^2 - 2x * x_3 + {x_3}^2 + y^2 - 2y * y_3 + {y_3}^2 \quad (6)$$

Då detta system blir olinjärt kan vi subtrahera ekvation (6) från (4) och (5) och då ser det ut på följande vis:

$$R1^2 - R3^2 = -2x(x_1 - x_3) + {x_1}^2 - {x_3}^2 - 2y(y_1 - y_3) + {y_1}^2 - {y_3}^2 \quad (7)$$
$$R2^2 - R3^2 = -2x(x_2 - x_3) + {x_2}^2 - {x_3}^2 - 2y(y_2 - y_3) + {y_2}^2 - {y_3}^2 \quad (8)$$

Det går nu att skapa två matriser av dessa två ekvationer, enligt formen:

$$A[x, y] = b$$

Där delmatriserna A och b kan skrivas som:

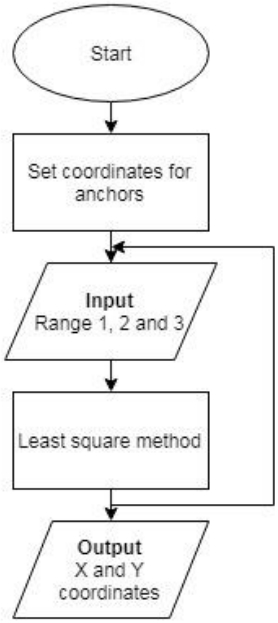
$$A = -2\begin{bmatrix} x_1 - x_3 & y_1 - y_3 \\ x_2 - x_3 & y_1 - y_3 \end{bmatrix}$$

$$b = \begin{bmatrix} R_1^2 - x_1^2 - y_1^2 - R_3^2 + x_3^2 + y_3^2 \\ R_2^2 - x_2^2 - y_2^2 - R_3^2 + x_3^2 + y_3^2 \end{bmatrix}$$

Positionen för mottagaren uttryckt i X och Y blir:

$$[x, y] = A^{-1} \cdot b$$

För att kunna implementera minsta kvadratmetoden i det byggda systemet gjordes först matematiska modeller i matlab som kunde testas för att sedan lättare kunna realiseras i Arduino-kod. Flödesschemat för minsta kvadratmetoden i figur nedan (se figur 5.6) beskriver koden som använts i både matlab och Arduino.



Figur 5.6 Flödesschema för ”Least square method”.

Nedan kan två utdrag av koden betraktas för att ge en tydligare bild av hur matematiken implementerats först in i matlab för att sedan skrivas om till Arduino kod. Första utdraget (se figur 5.7), är den del som är skrivet i matlab:

```
n=3;
e=ones(n,1);
d=e; x=e; y=e;

d(1)=3.74; d(2)=2.83; d(3)=2.24;
x(1)=0; x(2)=5; x(3)=4;
y(1)=0; y(2)=0; y(3)=4;

b=[d(1)^2 -x(1)^2 -y(1)^2 -d(3)^2 x(3)^2 y(3)^2
   d(2)^2 -x(2)^2 -y(2)^2 -d(3)^2 x(3)^2 y(3)^2];

A=-2*[x(1)-x(3) y(1)-y(3)
   x(2)-x(3) y(1)-y(3)];
p=A\b;
p=[sum(p(1,:))
  sum(p(2,:))];
X_pos=p(1);
Y_pos=p(2);
```

Figur 5.7 ”Least square method” i MatLab-kod

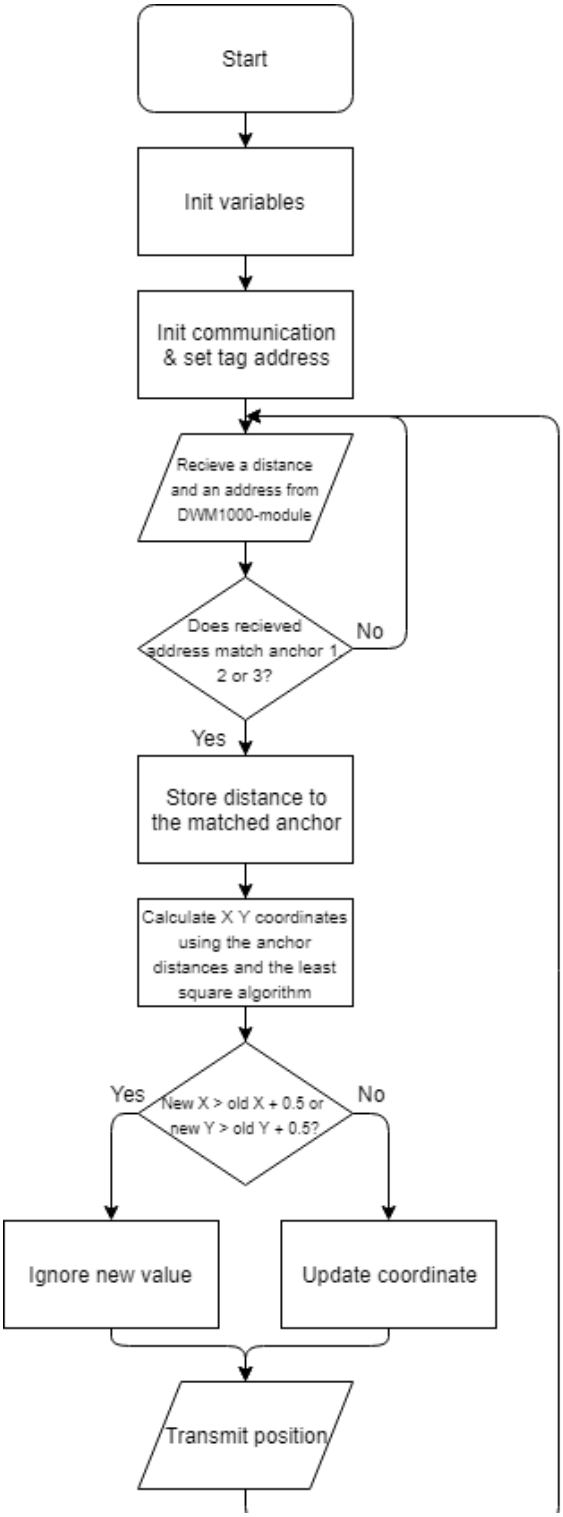
Figur 5.8 visar den översatta delen i Arduino-kod.

```
b[0] = pow(d[0],2) - pow(x[0],2) - pow(y[0],2) - pow(d[2],2) + pow(x[2],2) + pow(y[2],2);
b[1] = pow(d[1],2) - pow(x[1],2) - pow(y[1],2) - pow(d[2],2) + pow(x[2],2) + pow(y[2],2);

A[0][0] = -2*(x[0]-x[2]);
A[0][1] = -2*(y[0]-y[2]);
A[1][0] = -2*(x[1]-x[2]);
A[1][1] = -2*(y[1]-y[2]);

Adet = (A[0][0]*A[1][1] - A[0][1]*A[1][0]);

Ainv[0][0] = (1.0f/Adet)*A[1][1];
Ainv[0][1] = (1.0f/Adet)*(-A[0][1]);
Ainv[1][0] = (1.0f/Adet)*(-A[1][0]);
Ainv[1][1] = (1.0f/Adet)*A[0][0];

pos[0] = b[0]*Ainv[0][0]+b[1]*Ainv[0][1];
pos[1] = b[0]*Ainv[1][0]+b[1]*Ainv[1][1];
```

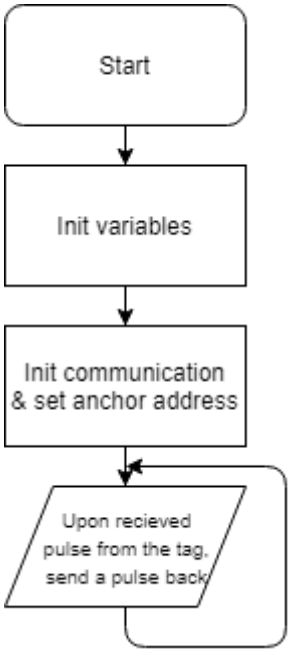
Figur 5.8 Samma funktion som innan, översatt till Arduino-kod

5.4 ARDUINO KOD

Arduinokoden kan representeras med två olika flödesscheman (se figur [5.9] och [5.10]). Figur 5.9 representerar mottagarens kod medan figur 5.10 representerar koden för ett ankare. Man ser här att det är mottagaren som gör alla större beräkningar för att bestämma positionen. Mottagaren kan även utföra diverse debug-funktioner, t.ex. genomsnittliga avstånd och min/max avvikelser, men eftersom dessa funktioner endast används i debugging-syfte så är de inte inkluderade i flödesschemat.



Figur 5.9 Flödesschema för mottagarens arduino-kod.



Figur 5.10 Flödesschema för arduino-kod i ett ankare.

5.5 KALIBRERING

För att få korrekta värden för avståndsmätning i detta projekt krävdes en viss kalibrering. I början av projektet, innan någon kalibrering gjorts, uppmättes fel på mellan 0.5 och 1 meter. Felet kunde förminskas avsevärt genom att lägga till kod för kalibrering i mottagarens arduinokod (se figur 5.11).

```
// If calibration has not been done, change calibration to "true" and show_average to the
// anchor unit you wish to calibrate. Record the average measured distance error at 1, 2 and 3
// meters and insert these values into their respective variables.
// Do this for all three anchors.

#define calibration false
#define show_average false // Either "false", 1, 3, or 4

#define calibration_unit1_1m 0.68
#define calibration_unit1_2m 0.73
#define calibration_unit1_3m 0.75
#define calibration_unit1 (calibration_unit1_1m+calibration_unit1_2m+calibration_unit1_3m)/3

#define calibration_unit3_1m 0.61
#define calibration_unit3_2m 0.77
#define calibration_unit3_3m 0.65
#define calibration_unit3 (calibration_unit3_1m+calibration_unit3_2m+calibration_unit3_3m)/3

#define calibration_unit4_1m 0.77
#define calibration_unit4_2m 0.87
#define calibration_unit4_3m 0.68
#define calibration_unit4 (calibration_unit4_1m+calibration_unit4_2m+calibration_unit4_3m)/3
```

Figur 5.11 Manuellt inställda variabler för kalibrering av systemet.

Här mäts genomsnittliga avståndsfel upp manuellt mellan mottagaren och alla tre ankare vid 1, 2 och 3 meters avstånd. Dessa fel skrivs in i sina respektive variabler och programmet räknar sedan ut det genomsnittliga felet. När variabeln *calibration* sedan är satt till "false" korrigerar programmet de uppmätta avstånden med det genomsnittliga värdet (se figur 5.12), vilket ger en mycket större noggrannhet.

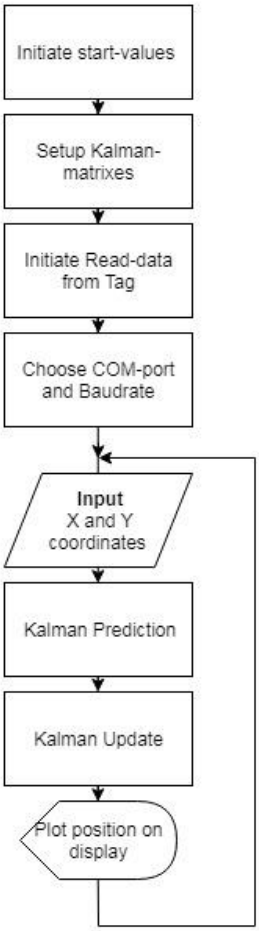
```
if(address == UNIT_1_ADDRESS){ // U1
  range1 = DW1000Ranging.getDistantDevice()->getRange();
  if(!calibration) {
    range1 -= calibration_unit1;
  }
  d[0] = range1;
}
else if(address == UNIT_3_ADDRESS){ // U3
  range3 = DW1000Ranging.getDistantDevice()->getRange();
  if(!calibration) {
    range3 -= calibration_unit3;
  }
  d[1] = range3;
}
else if(address == UNIT_4_ADDRESS){ // U4
  range4 = DW1000Ranging.getDistantDevice()->getRange();
  if(!calibration) {
    range4 -= calibration_unit4;
  }
  d[2] = range4;
}
```

Figur 5.12 Funktionen som utför kalibreringen.

5.6 KALMAN-FILTER

Inledningsvis måste en modell av systemet göras och sedan kan koden skrivas med rätta matriser utefter systemets uppskattade utseende [8]. Då det kan bli tunga beräkningar görs dessa beräkningar initialt i Matlab på en PC och om Arduino kan hantera beräkningarna själv kan det sedan översättas dit.

Då systemet är en enhet som skall röra sig med varierande hastighet och läge samt att den enda data som kan inhämtas är position, uppstår ett litet problem. Modellen är som sagt framtagen för något som rör sig och därför måste hastighetsförändringar approximeras då de inte fås som indata. Detta görs bland annat genom att införa ett uppskattat fel i hastighets- och accelerations-"mätningarna". Med rätt inställda parametrar kan därmed ett fungerande system för positionering skapas på endast positionsdata.



Figur 5.13 Flödesschemat beskriver kodens struktur som skrivits i matlab

5.6.1 FLÖDESSCHEMA MATLAB

Först initieras de startvärden som används igenom koden. Därefter ställs Kalman matriserna upp. (En djupare beskrivning av Kalman matriser och hur filtret sätts upp kommer efter beskrivningen av flödesschemat i figur 5.13).

Härefter behöver MatLab initieras för att kunna ta emot dataström från COM porten och i samband med detta sätts även baudraten för att kunna kommunicera med Arduinon i rätt hastighet.

Nu kan X och Y koordinaterna läsas in från Arduinon och köras genom Kalman filtret. Först utför filtret en uppskattning av position som därefter med hjälp av en viktad process uppdateras till att bestämma var mottagaren faktiskt befinner sig. Detta blir då filtrets bästa uppskattning av positionen.

Avslutningsvis plottas detta i ett fönster med utritade kanter som beskriver rummet som mottagare och ankare befinner sig i.

5.6.2 KONSTRUKTION AV KALMAN FILTRET

Ett Kalman filter bygger på principen av att det gissar sig fram vad som är korrekt position med avseende på den brusiga indata som fås och tidigare kända position, hastighet och acceleration. Som i detta arbete finns alla tre i både X- och Y-riktningen och det kommer behöva ställas upp matriser för att kunna hantera alla beräkningar. De matriser som behövs för filtret är: X, P, Q, H, R och F matriserna. Mycket viktigt här är att en bra bild av systemet skapas för att kunna utnyttja filtret på bästa sätt [21] [8].

Den första fasen i filtret kallas “Predict” och innebär två saker. Först att tillstånds-matrisen X uppdateras genom att ta senast kända värden och anpassas efter hur systemet ser ut genom att multipliceras med F matrisen:

$$X_k = [1\ 0\ 0\ 1\ 0\ 0]'$$

$$X_k = F \cdot X_{k-1}$$

Matrisen F (Eng “state transition funktion”) bygger på att accelerationen integrerats två gånger i både X- och Y-led:

$$a: 1$$
$$v: 1 + t$$
$$s: 1 + t + \frac{t^2}{2}$$

$$F = \begin{bmatrix} 1 & dt & \frac{dt^2}{2} & 0 & 0 & 0 \\ 0 & 1 & dt & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & dt & \frac{dt^2}{2} \\ 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Den andra delen i "Predict"-fasen är "state covariance / prediction error" matrisen P som också multipliceras med F för att anpassas efter systemet men här adderas ett brus Q för att ta hänsyn till felet i det uppskattade värdena.

$$P_k = F \cdot P_{k-1} \cdot F' + Q$$

$$P = \begin{bmatrix} p & 0 & 0 & 0 & 0 & 0 \\ 0 & p & 0 & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 \\ 0 & 0 & 0 & p & 0 & 0 \\ 0 & 0 & 0 & 0 & p & 0 \\ 0 & 0 & 0 & 0 & 0 & p \end{bmatrix}$$

$$Q = \begin{bmatrix} q & q & q & 0 & 0 & 0 \\ q & q & q & 0 & 0 & 0 \\ q & q & q & 0 & 0 & 0 \\ 0 & 0 & 0 & q & q & q \\ 0 & 0 & 0 & q & q & q \\ 0 & 0 & 0 & q & q & q \end{bmatrix}$$

Den andra fasen är "Update" som består av tre moment. Det första är att ta fram den scalärmatris K (även kallad "Kalman gain") som sedan avgör hur mycket vikt som läggs vid uppmätta värden eller uppskattade värden av position, hastighet och acceleration. Där värdena i matrisen K ligger mellan 0 och 1.

$$K = P_k \cdot H' \cdot (H \cdot P_k \cdot H' + R)$$

Här är H en anpassning-matris för att ta hänsyn till att det endast finns position som indata och R är en matris för bruset på mätningarna.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix}$$

Nu kan tillstånds-matrisen X uppdateras med hjälp av K och uppmätt data från Z matrisen och en ny position är nu uppskattad.

$$X_k = X_{k-1} + K \cdot (Z - H \cdot X_{k-1})$$

$$Z = [X_{pos}\ Y_{pos}]'$$

Avslutningsvis uppdateras matrisen P med hjälp av viktnings-matrisen K och ett nytt uppskattat P erhålls.

$$P_k = P_{k-1} - K \cdot H \cdot P_{k-1}$$

Nu kan nästa mätning göras och en ny position uppskattas av filtret.

5.7 SPIKFILTER

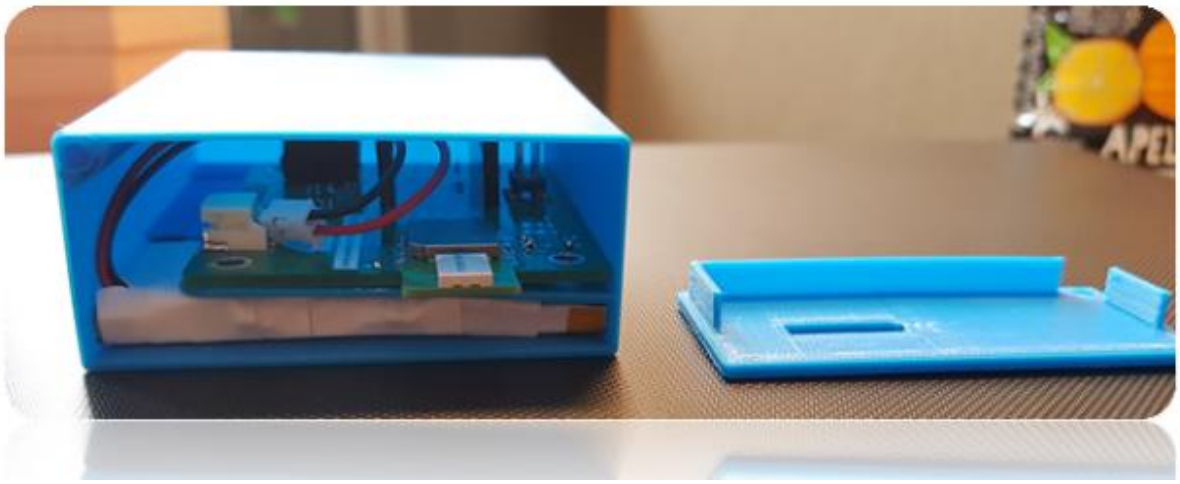
På grund av reflekterade signaler och andra störningar innehåller koden till mottagaren även ett spikfilter, vilket filtrerar bort plötsliga hopp i X- och Y-led. Dessa hopp är väldigt påtagliga, ofta flera kilometer, och påverkar endast systemet under en kort tid vilket gör dessa väldigt enkla att identifiera och ignorera. Detta görs av funktionen spikeFilter() (se figur 5.14), vilken jämför de nya beräknade X- och Y-koordinaterna med de senaste för att se hur långt mottagaren har rört sig sedan den senaste cykeln. Om mottagaren har rört sig mer än 0.5 meter längst antingen X- eller Y-axeln sedan senaste beräkningen så filtreras detta resultat ut, då detta ansågs vara en orimligt snabb förflyttning under en cykel på ungefär 0.2 sekunder.

```
void spikeFilter() {
  if(posFilterStart == false) {                    // posFilterStart is equal to "false" at startup.
    posx = pos[0];                                 // This allows the program to run one cycle
    posy = pos[1];                                 // with the filter disabled. Otherwise it wouldn't
  }                                                // accept any measured positions until the tag
  else if(posFilterStart == true) {                // was within 0.5 meters of the origin anchor,
    if(pos[0] < posx+0.5 && pos[0] > posx-0.5) {   // since the initial X and Y values are 0.
      posx = pos[0];
    }                                              // While active, the filter compares the new
    if(pos[1] < posy+0.5 && pos[1] > posy-0.5) {   // coordinates to the previous values.
      posy = pos[1];                               // If the tag is within 0.5 meters of the previous
    }                                              // measured value, the coordinate is accepted.
  }

  posFilterStart = true;                           // After the first cycle, the filter becomes active
}
```

Figur 5.14 Spikfilter-funktionen i mottagaren.

5.8 CAD CHASSI

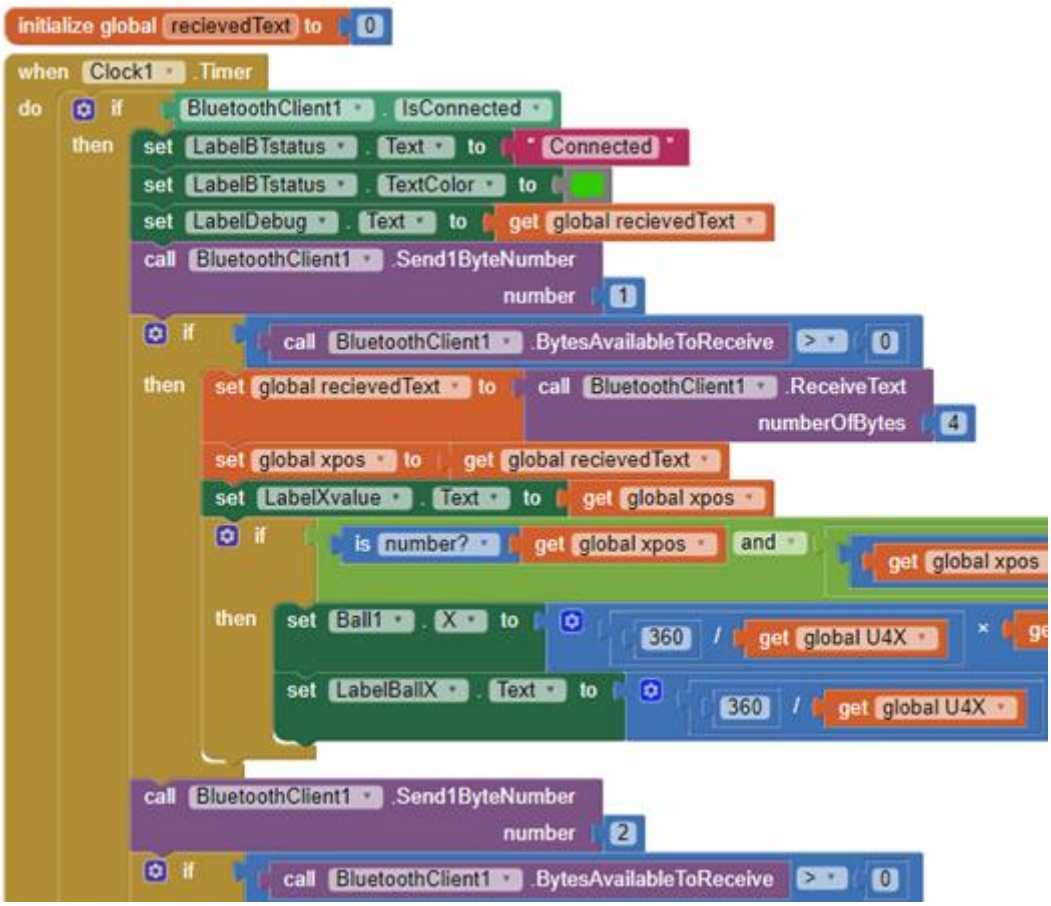
För att skydda mottagaren och för att underlätta test av systemet ritades det även upp ett chassi för mottagaren och stativ för alla ankare i Catia V5. Dessa printades sedan ut med en 3D-printer (se figur 5.15).



Figur 5.15 Det 3D-printade chassit med mottagaren och ett batteri inuti.

5.9 ANDROIDAPPLIKATION

Till projektet skapades en Androidapplikation för att visuellt kunna visa hur mottagaren rör sig i rummet. Mottagaren sänder sin position till Androidenheten via en inkopplad Bluetooth-modul, HC-05 [22] och applikationen skrevs med hjälp av MIT App Inventor 2 [23]. Detta använder ett blockbaserat programspråk, där figur 5.16 nedan visar ett kort utdrag ur applikationens kod.



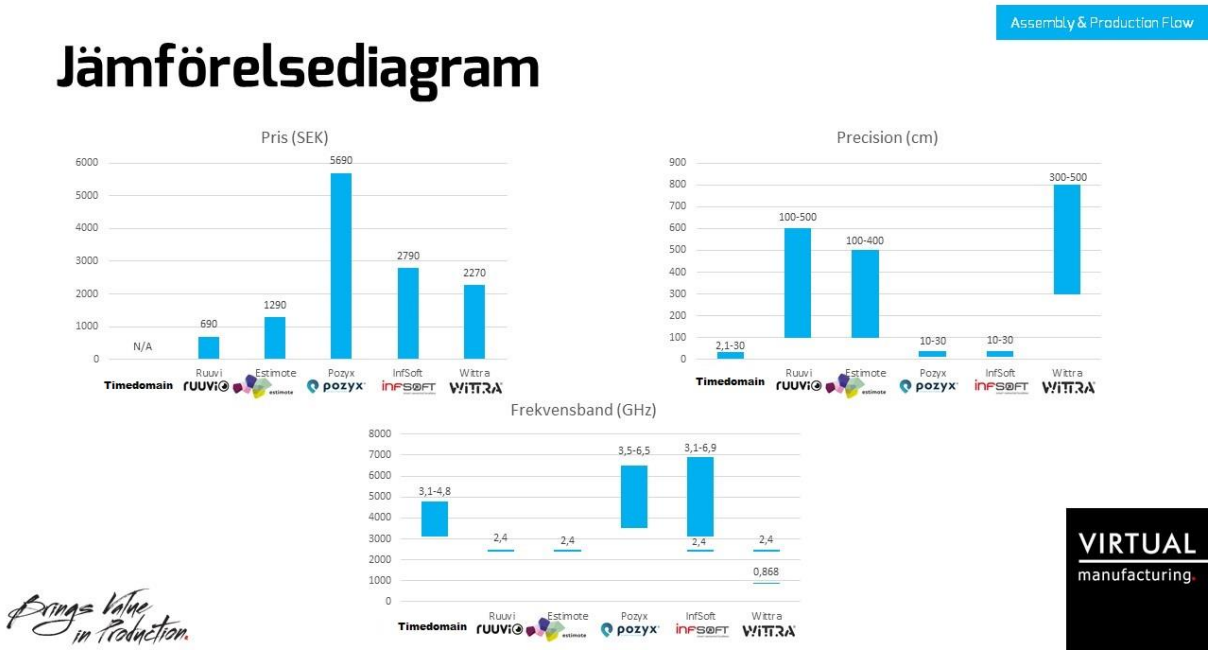
Figur 5.16 Ett urklipp ur applikationens kod. Skrivet i s.k. ”blockbaserat” programspråk.

6. RESULTAT

6.1 RESULTAT AV DEN KOMMERSIELLA STUDIEN

Resultatet av den genomförda studien visar att kommersiella positioneringssystem varierar något inom kategorierna pris och precision. Systemet med bäst rapporterade precision produceras av Time Domain, vilka marknadsför en träffsäkerhet på 2,1cm till 10cm. De andra UWB-systemen vi tittade på, tillverkade av Pozyx respektive Vittra rapporterade en träffsäkerhet på 10 till 30 cm.

Studien avslutades med tre jämförelse-grafer, sammanställda i figur 6.1, för att lättare kunna visa på skillnaderna i pris, position och frekvensband.



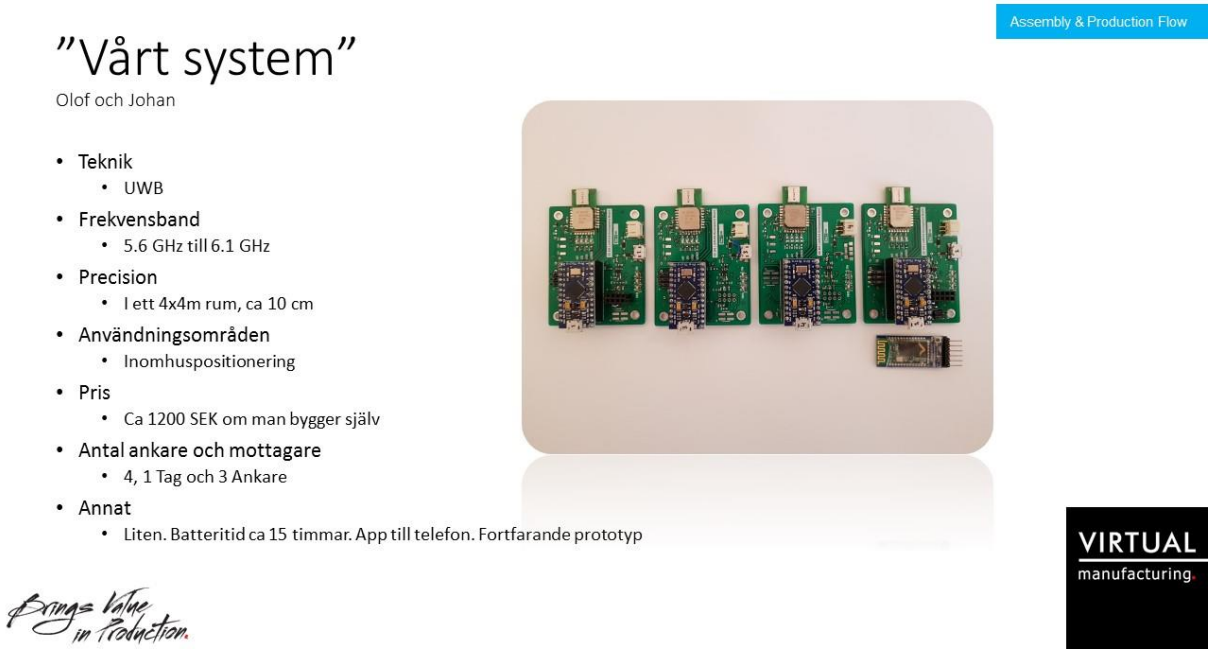
Figur 6.1 Grafer över pris, precision och frekvensband för jämförda produkter.

6.2 ANVÄNDNINGSOMRÅDEN

Det utfördes ingen djupare undersökning i möjliga användningsområden för UWB men de vanligaste kategorierna är: avståndsmätning, dataöverföring och radar.

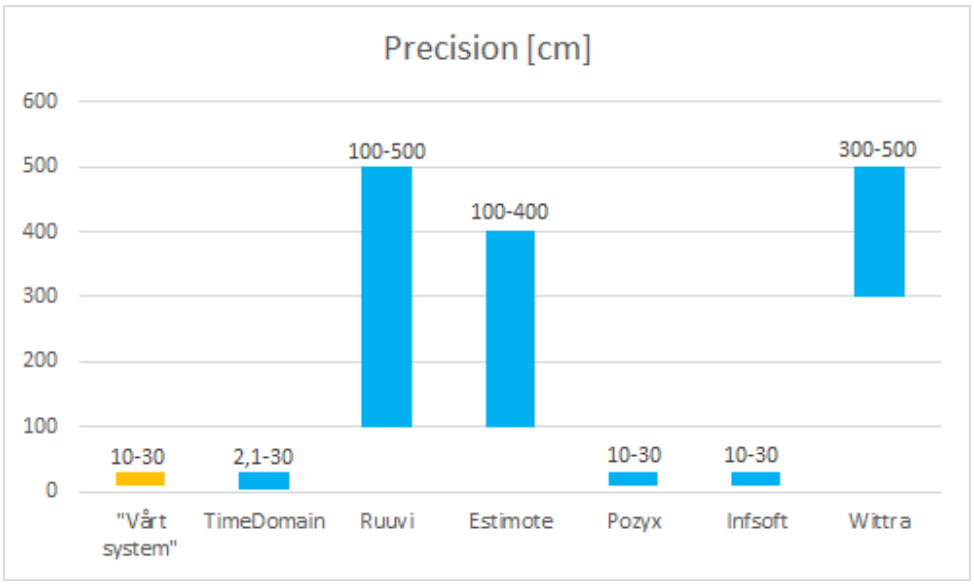
6.3 JÄMFÖRELSE MELLAN OPEN SOURCE OCH KOMMERSIELLA PRODUKTER

Enligt modellen för jämförelser mellan kommersiella produkter i kap 4 har en liknande beskrivande bild (figur 6.2) av systemet som byggdes i detta projekt skapats för jämförelse.

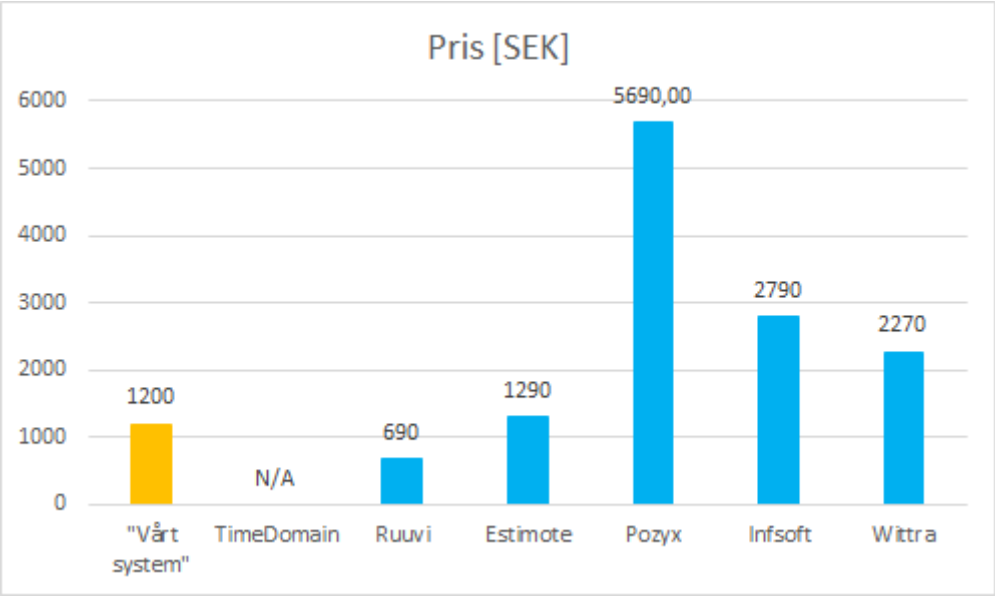


Figur 6.2 Detta projekt framställt som de jämförda kommersiella produkterna.

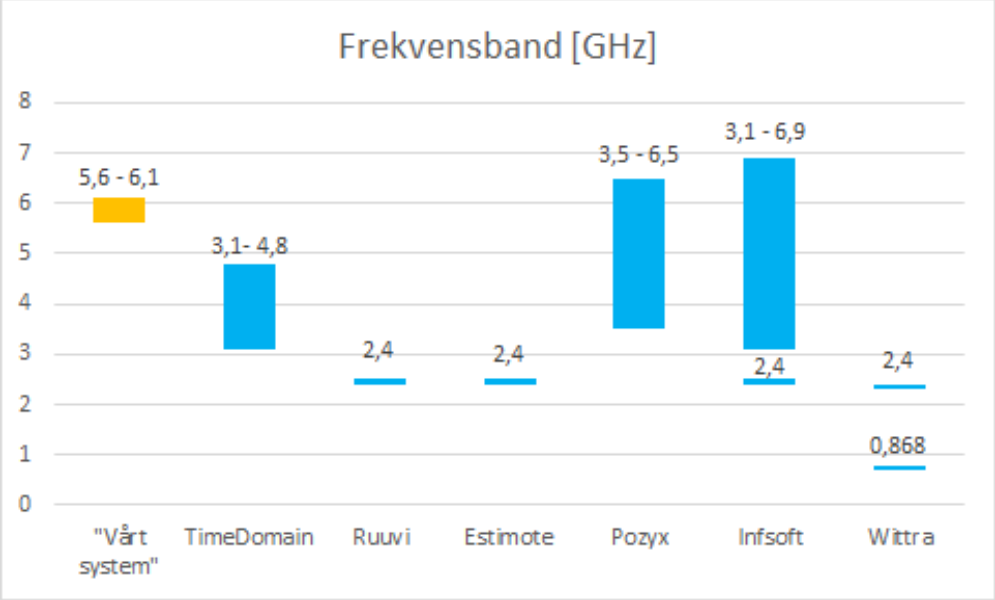
Nedan är tre jämförelsegrafer (figur 6.3, 6.4 och 6.5) som jämför projektets byggda system med de kommersiella varianterna. Som i kapitel 4 är de kommersiella varianterna med blå staplar och systemet som byggts i detta projekt i gula staplar.



Figur 6.3 Precision för detta projekt samt kommersiella produkter.



Figur 6.4 Pris för detta projekt samt kommersiella produkter.

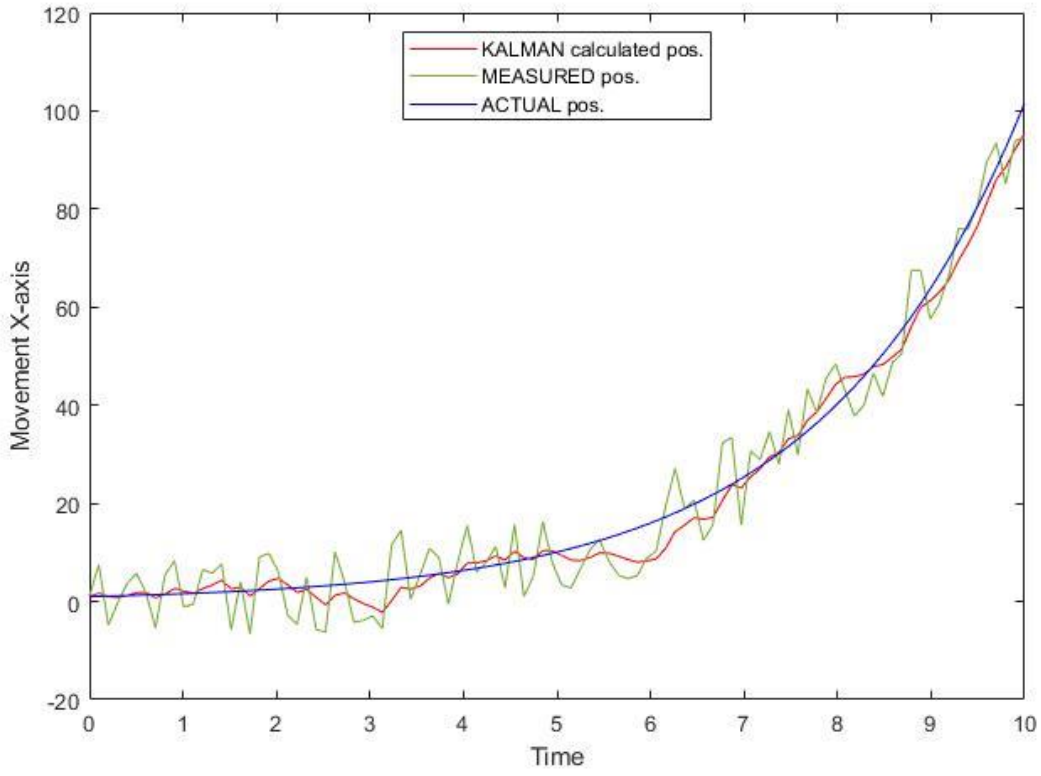


Figur 6.5 Frekvensband för detta projekt samt kommersiella produkter.

Ö

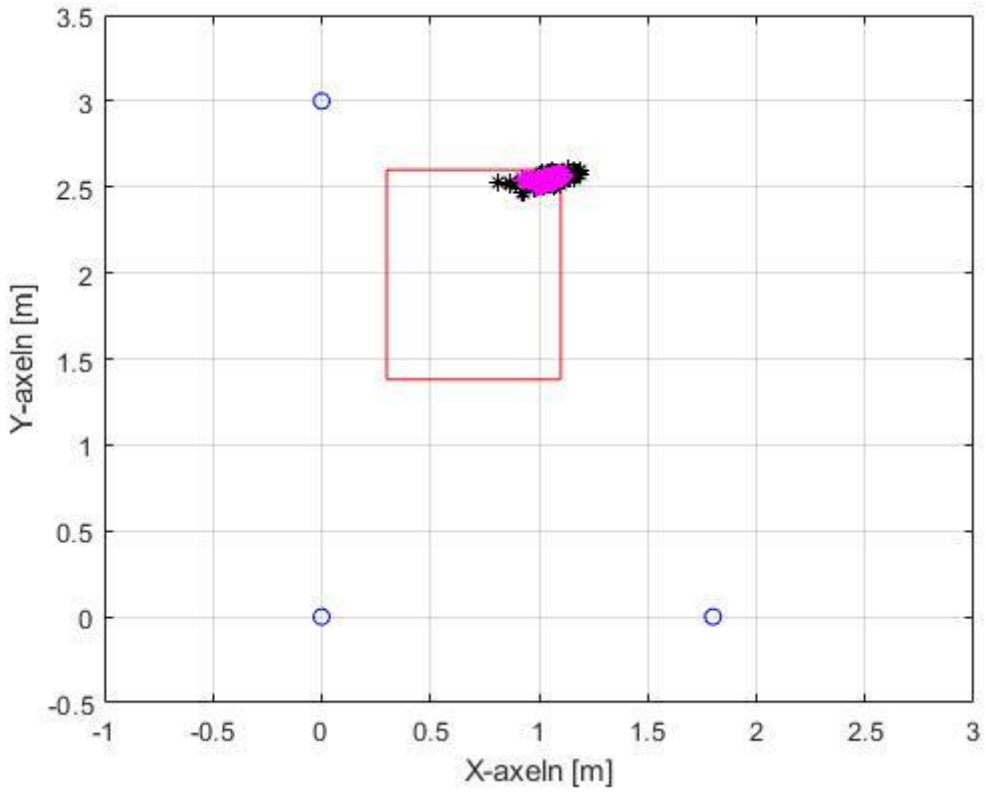
6.4 RESULTAT EFTER IMPLEMENTERING FILTER

Bilden nedan (se figur 6.6) visar hur Kalman-filtret arbetar. Fiktiva rörelser i X-led (Blå kurva) är den faktiska positionen i testet och slumpmässigt genererade mätvärden (Grön kurva) kan ses längs den riktiga kurvan och har en spridning på ca +- 10 LE. Slutligen kan ses att Kalman-filtret har använt dessa mätvärden och beräknar fram mer exakta positioner, hastigheter och accelerationer (Röd kurva) som följer den riktiga kurvan med lovande resultat. De korrigerade rörelserna blir mjukare än de "uppmätta" värdena.



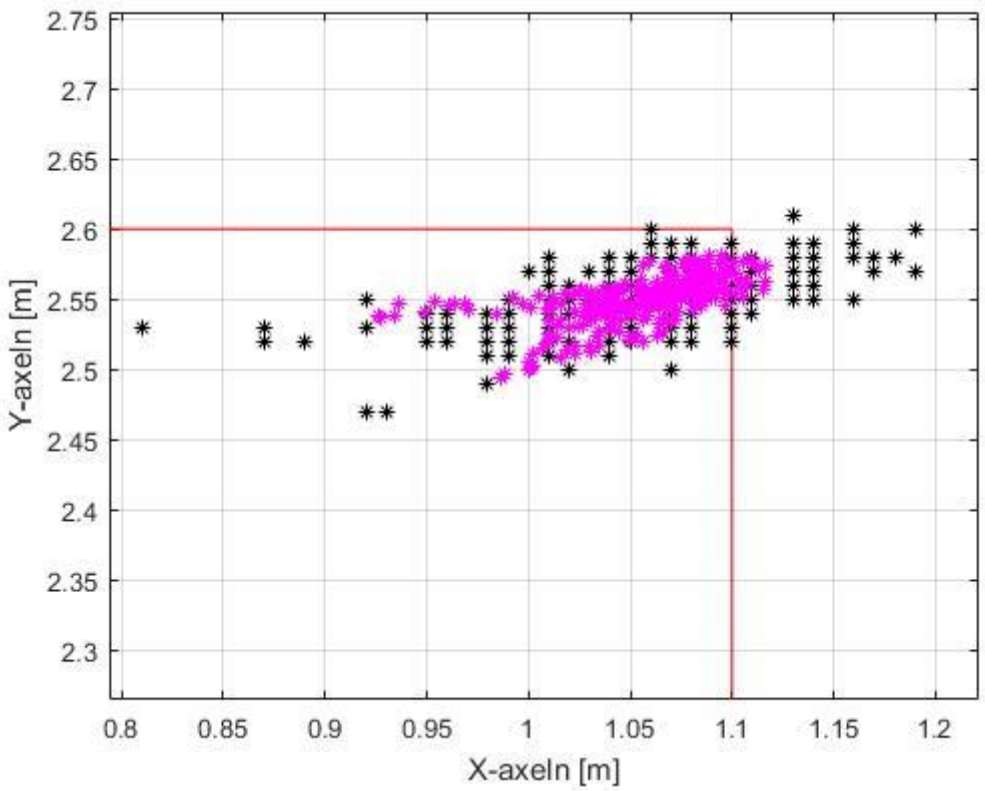
Figur 6.6 Kalmanfiltret när det behandlar en simulerad rörelse med störningar.

Systemet testades även i verkligheten i ett rum där de tre ankare placerades ut för att bilda ett koordinatsystem. Bilden nedan (se figur 6.7) visar hur uppmätta punkter (svarta) och korrigerad position (magenta) förhåller sig till varandra. Det är tydligt att Spik- och Kalman-filtret optimerar positionen för mottagaren. De blå cirklarna visar var ankaren är placerade och den röda rektangeln är en utritad yta som testet skulle förhålla sig till. 300 mätpunkter gjordes som sedan ritas ut och därefter får dessa samma punkter gå igenom Kalman-filtret och ritas
ovanpå.



Figur 6.7 Ett test av projektet där mottagaren stod stilla.

Inzoomad del av mät-klustret (figur 6.8) visar att mätfelet är mindre än tre decimeter och efter Kalman-filtrets korrigering minskas felet till under två decimeter.



Figur 6.8 Samma diagram som figur 6.7 men under förstoring.

7. SLUTSATS OCH DISKUSSION

Arbetet har varit både utmanande och mycket givande. Att sätta sig in i radioteknik har varit något relativt nytt för oss båda och eftersom UWB är tämligen nytt på marknaden idag finns inte riktigt lika information och lösningar som det gör till de mer vanliga teknikerna BLE och WiFi. Det var mycket roligt att få lära sig om just UWB tekniken och nu efter arbetet känns det som en ytterst aktuell teknik som sprider sig på marknaden. Det som gör den så aktuell är det som beskrivs i rapporten. Behovet av hög precision vid positionering och då framförallt inomhuspositionering då GPS inte fungerar där. Många företag satsar på olika typer av självkörande farkoster och det blir då som sagt mycket viktigt att dem har ett bra navigeringssystem.

I studien av kommersiella produkter hittade vi flera företag som beslutades att de inte skulle vara med. Det var på grund av utformningen av produkten som gjorde att deras användningsområden inte var anpassade för inomhuspositionering av något/någon som rör sig

i en lokal. Det kunde exempelvis vara så att de var designade för att tracka kroppsdelar för att implementeras i exempelvis film, spel eller liknande. Däremot har inte alla de företag vi tagit med i studien färdiga lösningar för just positionering, men deras produkter är möjliga att programmeras likt vårt bygge till att kunna utföra detta arbete och därför fick de komma med i studien. En vidare undersökning av användningsområden för UWB beslutades att tas bort då fokus istället lades på optimering av precision.

När själva byggandet av vardera enhet började stötte vi på problem. Bland annat att det är otroligt svårt att löda ytmonterade komponenter i storleken 0306 med en lödpenna och lödtenn. Detta ledde till att vi var tvungna att införskaffa lödpasta, vilket underlättade arbetet avsevärt. Så här i efterhand hade det nog varit smartare att beställa färdigmonterade kort, men det blir då såklart mycket dyrare och eftersom priset var en viktig punkt utifrån våran aspekt så gjordes allt av oss för hand. Ett annat problem som uppstod var de långa leveranstiderna från Kina och att det saknades komponenter som beställts från USA. Leveranstiderna är något man får leva med helt enkelt, då priset är dramatiskt mycket lägre men man får vänta längre. Däremot de komponenter som inte kom med var 0 ohms resistorer och det problemet löste vi enkelt genom att bara löda en “brygga” mellan de punkter på kortet där dessa skulle vara. Slutligen i byggprocessen så hade vi endast en enhet kvar att löda och det var själva mottagaren. Den skilde sig något i utseende gentemot ankarna men hade samma processor och funktion, bara något mindre. Den blev tyvärr fel-löd och beslutet togs att bygga ett fjärde ankare eftersom att varje ankare gick att programmera som mottagare.

Det ursprungliga Open-source projektet som hårdvaran bygger på hade rätt små batterier (850mah) i modellen medans vi använde mer än 3 gånger så stora batterier (3000mah). Anledningen till detta var att dem stora batterierna fanns tillgängliga för oss från ett annat system som vi kunde ta ifrån.

Vi valde även att utföra sista steget i arbetet, Kalman-filter, uteslutande i matlab på grund av bland annat tidsåtgång men även för att ett Kalman-filter kräver rätt tunga beräkningar och vi ville få ut bästa mätdata, därför gjordes dessa på en PC. Vidare arbete lämnas till test av filtret på Arduino eller annan mikroprocessor som går att använda på systemet.

Vad gäller systemets kalibrering så kunde en relativt enkel kalibreringsfunktion minska mätfelet ganska mycket, se kapitel 5.5. Ett kvarstående problem är dock att detta mätfel växer med ökat avstånd mellan mottagare och ankare, vilket DecaWave har försökt motverka genom att korrigera det uppmätta utifrån en avståndstabell.

Angående kalibreringen så skulle en eventuell framtida produkt även kunna implementera en funktion för automatisk kalibrering. Detta antingen genom applikationen eller med någon form av hårdvara på mottagaren. Även avstånden mellan ankare skulle kunna fås på detta sätt, men då skulle dessa inte bli lika exakta som när man mäter upp avstånd manuellt.

Spikfiltret i arduinokoden har också en svaghet, vilket är att om “hoppet” visar sig påverka positionen tillräckligt länge för att mottagaren skulle hinna röra sig mer än 0,5 meter skulle systemet låsa sig och inte registrera nya positioner förrän mottagaren återvänt till en accepterad position. Detta skulle dock kunna lösas i koden med en slags reset-funktion.

8. REFERENSER

[1] Virtual Manufacturing. (2018). Hemsida för Virtual Manufacturing. https://www.virtual.se/en/ [Accessed 2018-05-04].

[2] Adrio Communications Ltd. (2018). Ultra Wideband Technology Tutorial. http://www.radio-electronics.com/info/wireless/uwb/ultra-wideband-technology.php [2018-05-04].

[3] Jeffrey H, Reed. (2005). Introduction to Ultra Wideband Communication Systems http://www.informit.com/articles/article.aspx?p=384461 [Accessed 2018-05-04].

[4] Bluetooth (2018). Technology. https://www.bluetooth.com/buetooth-technology [Accessed 2018-06-11].

[5] Apple. (2018). iBeacon Developer. https://developer.apple.com/ibeacon/ [Accessed 2018-06-09].

[6] Thomas Trojer. (2018). Arduino-dw1000. https://github.com/thotro/arduino-dwm1000 [Accessed 2018-06-11].

[7] Ramsey Faragher. (2012). Understanding the basis of the Kalman filter. [online] available at: http://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf [Accessed 2018-07-16].

[8] Cantón Paterna, V. Calveras Augé, A. Paradells Aspas, J. (2017). A bluetooth low energy indoor positioning system with channel diversity, weighted trilateration and Kalman filtering. [online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5750706/ [Accessed 2018-07-16].

[9] Dddutz. (2018). Jarvas indoor positioning system. www.intructables.com/id/jarvas-indoor-positioning-system [Accessed 2018-06-11].

[10] Sahinoglu, Z., Gezici, S. & Güvenc, I (2008). Ultra-Wideband Positioning Systems: Theoretical Limits, Ranging Algorithms, and Protocols. Cambridge university Press. [online] available at: http://library.books24x7.com.proxy.lib.chalmers.se/toc.aspx?site=Y7V97&bookid=28785 [Accessed 2018-06-11].

[11] Johansson, H. Kornhill, J. (2015). Inomhuspositionering UWB. [online] Availabe at: http://studentarbeten.chalmers.se/publication/218878-inomhuspositionering-uwb [Accessed 2018-07-16].

[12] Pozyx. (2015). How does UWB work. https://www.pozyx.io/Documentation/how_does_uwb_work [Accessed 2018-07-16].

[13] Infsoft. (2018). Sensors: Ultra-Wideband. https://www.infsoft.com/technology/sensors/ultra-wideband [Accessed 2018-07-16].

[14] Ruuvi. (2018). Boost your business. www.ruuvi.com/press [Accessed 2018-07-16].

[15] Pozyx. (2018). Pozyx pins. [Online picture]. Available at: https://www.pozyx.io/assets/images/docs/pozyx_pins.jpg [Accessed 2018-07-16].

[16] Estimote. (2018). [Online picture]. Available at: https://estimote.com/assets/gfx/press/product/Proximity-Beacons-and-Estimote-App-Mint-Background-thumb.0cf4e24e.png [Accessed 2018-07-16].

[17] Infsoft. (2018). [Online picture]. Available at: https://cdn.infsoft.com/www/images/technology/hardware/locator-nodes/infsoft-Locator-Node-EN-fullsize.jpg [Accessed 2018-07-16].

[18] Wittra. (2018). Products. http://www.wittra.se/#products [Accessed 2018-07-16].

[19] Timedomain. (2018). Pulson-330. https://timedomain.com/products/pulson-330/ [Accessed 2018-07-16].

[20] Pozyx. (2015). How does positioning work. https://www.pozyx.io/Documentation/how_does_positioning_work [Accessed 2018-06-09].

[21] OpenPilot. DIY Drones. Wilkin.T. (2018). The Extended Kalman Filter: An Interactive Tutorial for Non-Experts. https://home.wlu.edu/~levys/kalman_tutorial/kalman_01.html [Accessed 2018-06-09].

[22] ITead Studio. (2010). HC-05 Bluetooth to Serial Port Module. https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf [Accessed 2018-06-09].

[23] Massachusetts Institute of Technology. (2018). MIT App Inventor 2. hai2.appinventor.mit.edu/ [Accessed 2018-04-03].