# PROSJEKTOPPGAVE

HØGSKOLEN I OSLO
OG AKERSHUS

Høgskolen i Oslo og Akershus

Fakultet for Teknologi, Kunst og Design

Ingeniørfag - Elektronikk og Informasjonsteknologi

Av Mark Strømme, Christophe Plaissy, Oda O. Nedrejord

Innleveringsdato: 21.04.2016

# Innholdsfortegnelse

# Forord

Denne rapporten er skrevet i forbindelse med et prosjektarbeid gitt av labingeniør i faget Digitale Systemer. Rapporten er skrevet for både labingeniør og forelesere i faget, men er også ment for andre studenter som har en viss kjennskap til elektronikk og informasjonsteknologi, samt framtidige ingeniører som leter etter løsninger som rapporten kan tilby.

Hensikten med rapporten er å gjøre det enkelt for utenforstående å sette seg inn i prosjektet. Det er meningen at man skal kunne gjennomføre et tilsvarende prosjekt, og å kunne bygge videre på det ved hjelp av innholdet i rapporten. I tillegg er den skrevet slik at labingeniør og forelesere lett skal kunne evaluere arbeidet vi har lagt ned i prosjektet.

Vi ønsker også å forklare leseren hvordan man kan gjennomføre et tilsvarende prosjekt, om tanken bak, utfordringer og videre arbeid med prosjektet. Samt rette klare paralleller til hvordan prosjektet er relevant for dagens samfunn.

Rapporten er skrevet av Mark Strømme, Christophe Plaissy og Oda Nedrejord

# Sammendrag

Vårt prosjektet går ut på å lage en bil som følger en bane, kjører rundt eventuelle hindringer på veien og reagerer med hensyn til lyskryss. Tidligere forsker har prøvd å oppnå det med lyssensorer, og feilet. Derfor ville vi løse denne problemstillingen ved hjelp av Blåtann.

For dette prosjektet brukte vi Lego som konstruksjonsmiddel for design av bilen og lyskryss. Vi brukte Lego Mindstorms sine sensorer og motorer, sammen med Lego NXT mikrokontroller, for å programmere funksjonene bilen skulle utføre, og vi brukte RobotC som programmeringsspråk. I tillegg brukte vi NXT-en som en mottaker for signalet fra lyskryss. Vi brukte Arduino kombinert med Adafruit sin 'Bluefruit EZ-Link' (Blåtan hardware) for å styre lyskrysset og sende signalene.

Ferdig produkt av vårt prosjekt dekket det meste av våre mål. Bilen greide å følge bane, å stoppe ved hindringer på veien og å reagere på lyskryss. Derimot klarte vi ikke å simulere en realistisk type kjøring, vi rakk heller ikke få den til å kjøre rundt hindringer.

Som konklusjon til prosjektet kan ikke bilen brukes samfunnsrelatert uten videre arbeid, men det er en fin byggestein for videreutvikling.

# I. Innledning

## I.1. Problemstilling og formål

Da vi først startet å jobbe med prosjektet, ønsket vi å lage en elektrisk og automatisk bil. Vi brukte en fargesensor for å lese av et lyskryss, avstandssensor for å lese av hindringer og lyssensor for å følge bane. Lyssensoren greide ikke å lese av fargene til lyskrysset. I kombinasjon av for svakt lys og for upresis lyssensor var dette ikke mulig. Vi bestemte oss for å bruke Blåtann som løsningen på dette. Dermed sto vi igjen med et mål om: *"Å lage en elektrisk automatisert bil som kjører etter en linje og reagerer på hindringer i veien, samt på lyssignaler sendt fra et lyskryss"*.

I denne rapporten blir Lego mikrokontroller ofte kalt for "NXT" og Arduino UNO mikrokontroller forkortet til "Arduino".

## I.2. Omfang og begrensninger

Siden en bil på en bilvei gjerne har mange potensielle farer å ta hensyn til, bestemte vi oss for at vi skulle implementere flere utfordringer underveis etter hvor raskt vi klarte å nå hovedmålet vårt. Dessverre tok både linjefølgingen til bilen og Blåtann-senderen så lang tid at vi i denne omgang ikke fikk implementert noe nytt. Allikevel er vi fornøyde med å ha forutsett dette, slik at vi faktisk nådde hovedmålet.

## I.3. Strategi for løsning

Da vi begynte planleggingen av prosjektgjennomføringen, var vi alle tre tidlig enige om hvordan vi skulle ta fatt på oppgaven. Vi ønsket å starte så enkelt som mulig; få bilen til å kjøre og få Blåtann til å sende signaler. Deretter kunne vi gå mer avansert til verks.

# II. Materialer og metoder

Da vi startet med prosjektet hadde vi flere valg i forhold til det vi hadde lært og jobbet med forrige semester, samt hva vi hadde av tilgjengelige ressurser på skolen. I og med at Lego var lettest tilgjengelig i form av byggeklosser, valgte vi da å bruke det. Delene vi brukte for design er fra Lego Technic og sensorene, motorene og NXT-en fra Lego Mindstorms. Arduino, sammen med lysdioder og motstander, var det den mest praktiske løsningen for bygging av lyskryss.

## II.1. Teori og bakgrunn

### II.1.a. Programmering

I dette prosjektet har vi fått bruk for mye kunnskap som vi har tilegnet oss i løpet av skoleåret. Dette er blant annet programmering i Arduino C, og praktisk bruk av Arduino. I tillegg til arrays, if- og while-løkker, lærte vi oss programmering i med Bluefruit, og RobotC programmeringsspråk. Vi lærte oss å bruke "Threads" i RobotC programmering for å få en bedre flyt i skriptet vårt.

### II.1.b. Oppkobling

Oppkobling av elektrisk hardware anså vi ikke som en stor utfordring, i og med at vi har praktisert mye av dette før. Vi hadde kunnskap om hvordan dette skulle kobles opp, som blant annet lysdioder/motstander til Arduino, og oppkobling av Bluefruit fantes det koblingsskjema til.

## II.2. Kravspesifikasjon

### II.2.a. Bil

#### II.2.a.i. Linjefølging

Bilen skal greie å holde seg på banen til enhver tid. Ved videreutvikling skal den følge linjen med en realistisk kjøring, dette innebærer at bilen kjører rett fram når det ikke er

i en sving, og ikke kjører ut av banen under eller etter en sving. Ut i fra dette forventer vi et avvik på ca.10%.

#### II.2.a.ii. Avstandsmåling

Bilen skal stanse tidsnok når den møter en hindring på veien. Ut i fra dette forventer vi ingen avvik da dette er en viktig del av prosjektet. Hvis det lar seg gjøre, skal bilen kjøre rundt hindringer på veien. Dette er ikke en hovedprioritet, men kan bearbeides om det er tilstrekkelig med tid.

#### II.2.a.iii. Signalbehandling

Både Arduino og NXT skal reagere raskt på lyskrysset. Arduino skal sende signaler via Blåtann så fort lyset i hovedgata skifter tilstand. NXT skal lese av Blåtann-signaler og oppdateres ut i fra den gjeldende situasjonen, men kun når den nærmer seg lyskrysset. Vi forventer et avvik på 0% da dette er et av hovedmålene i prosjektet vårt.

### II.2.b. Lyskryss

#### II.2.b.i. Lyskryssekvens

Lyskrysset skal reagere på en realistisk måte i forhold til signal fra knappen, som skal forestille en knapp for fotgjenger, eller sonar som er avstandssensoren til kryssende bil fra sidegate. Vi forventer et avvik på 0% da dette er et av hovedmålene i prosjektet vårt.

#### II.2.b.ii. Blåtann

Blåtann skal bruke tilstanden til lyskrysset for å sende gjeldende signal, kun fra hovedgaten kontinuerlig, slik at bilen rekker å lese i alle tilfeller. Vi forventer et avvik på 0% i forhold til dette.

# III. Systemløsning og resultater

## III.1. Systemløsning/konstruksjon

### III.1.a. Systemoversikt

Bilen er bygd på følgende måte: Bakhjulene sammen med bakre motorer bestemmer farten til bilen. Fremhjulene sammen med fremre motor bestemmer retning ved hjelp av et smart tannhjulsystem (j.f. VII.8.). Bilen svinger mot høyre så fort høyre sensor oppdager den svarte streken, og mot venstre i tilsvarende situasjoner med venstre sensor (j.f. VII.10.).

Arduino-lyskrysset har en standardtilstand der det er grønt lys i hovedgata som bilen kjører i, og rødt i både sidegata og fotgjengerfeltet. Når avstandsensoren i sidegata oppdager en bil, eller om en fotgjenger trykker på knappen vil lyset byttes etter kort tid fra grønt til rødt i hovedgata, og fra rødt til grønt i sidegata og fotgjengerfelt. Blåtann-signaler sendes i ut i fra hvilket lys som lyser i hovedegata (j.f. VI.5.).

### III.1.b. Detaljert løsning RobotC

Alle innebygde funksjoner vi brukte i RobotC er hentet fra biblioteket i Robomatter sin IDE. I RobotC bruker vi tasks. Tasks tilsvarer “Threads” i C. De er da uavhengige programbiter som kjører samtidig, deler minne og kan dermed kommunisere med hverandre. Vi valgte denne løsningen av forskjellige grunner. Trådene enkle å bruke og dokumentasjon på nettet er bred og lett å finne. Dette lar oss skrive et kortere og mer oversiktlig program. Til slutt var trådene, eller sagt på en annen måte “Threads” nytt for oss. Det var da spennende å lære om dette og eventuelt å kunne bruke det i andre situasjoner. Eksempelvis bruker vi en tråd til å lese verdiene fra sensorene og skrive de ut på skjermen. Dette gjøres i én while-løkke i stedet for å måtte skrives i hver eneste løkke programmet kommer inn i (j.f. VII.10., 12.). Det kan også nevnes at vi brukte en synkroniseringsfunksjon innebygd i RobotC til å få den ene bakre motoren å styre den andre. Dette forenklet selve språket i programmet.

### III.1.c. Detaljert løsning Arduino C

I Arduino C har vi brukt ferdige funksjoner som blant annet "pinMode" "digitalWrite" og "digitalRead" fra Arduino C biblioteket (Arduino, Udatert). I tillegg har vi laget egne funksjoner for sidegate, hovedgate, Blåtann og gående gate. Disse funksjonene kalles med en bestemt sekvens fra Void loop hver gang ultralyd-sensoren detekterer bevegelse, dersom bilen fra sidegaten er nærme nok eller når knappen trykkes. Vi har brukt arrays flere steder i koden for å spare plass, og for å gjøre programmet mer ryddig. I tillegg har vi brukt for-løkker, while-løkker og if-setninger alt ettersom hvor vi har ment det er mest praktisk og nyttig. Blåtann-funksjonen kalles hver gang det skjer lysforandringer i hovedgaten. I Blåtann-funksjonen skrives det ut en adresse med tre ulike signaler til seriell monitor, hvis lyset er grønt, sendes en adresse med signalet "1" flere ganger i sekundet, og tilsvarende "3" for gult og "2" for rødt til NXT-roboten.

Da vi skulle implementere Blåtann-programmet hadde lysdiodene en venteperiode flere steder i Void loop på eksempelvis to sekunder. Dersom vi hadde gjort det samme med Blåtann, ville den kun ha sendt ut ett signal med varighet på et halvt millisekund på en to-sekunders ventetid (delay). For at Blåtann skulle kunne sende flere signaler i sekundet, måtte vi ha en egen av-og-på sekvens for hver av de tre Blåtann-signalene, og med en en kort venteperiode for hver gang den sendte ut signaler, slik at det ikke ville påvirke resten av lyskrysset. Dermed måtte vi fjerne venteperioden til resten av lyskrysset, og heller bruke Blåtann-funksjonen sin venteperiode opp til flere ganger. Dette løste vi med en for-løkke. På denne måten fikk vi lyskrysset til å fungere godt, realistisk og samtidig sende flere signaler i sekundet til NXT-roboten.

## III.2. Systemprøving/simuleringer

### III.2.a. Bil

Ved å teste bilen, først på banen, uten Blåtann-kobling oppdaget vi en del problemer. Lego-sensorene var ikke presise. De første sensorene vi brukte var lyssensorer. De visste deres begrensinger på to forskjellige måter: verdiene sensorene leser av avhenger mye av

lyset i omgivelsen og avstand fra grunnflaten, noe som gjorde at det var vanskelig å teste. Derfor byttet vi fra lyssensorer til RGB (Red-Green-Blue) fargesensorer. Selv om lyset i rommet ikke påvirket disse sensorene var de likevel ustabile i forhold avstanden fra grunnflaten. Ombygging var dermed nødvendig for at sensorene ikke skulle hoppe ut av riktig leseområdet under kjøringen.

En annen utfordring gjaldt avstandssensorene. Etter testkjøringer så vi at hvis ikke sensoren var rettet parallelt med overflaten (gulvet eller hindring) den måler mot, vil den ikke lese avstandsverdien fram til veggen. Dette gjorde det vanskelig for oss å ha med funksjonen som kjørte rundt en hindring fordi sensoren leste ikke av riktig verdier, noe som gjorde at bilen ikke oppfylte kravene. Vi valgte da å la dette være, og ha med kun en "nødsituasjon-funksjon", som var å stanse når en hindring står på veien. Eksempelvis hvis et menneske eller dyr skulle gå ut i veien, skulle da bilen kunne bråstoppe.

Til slutt klarte vi ikke å inkludere en realistisk kjørestil: En funksjon som retter opp retningen til bilen kombinert med en funksjon som finner igjen banen ble testet. Det ga gode resultater i veldige presise forhold, men med en gang situasjonen ble forskjellig, oppstod det feil, og bilen kjørte ut av veien.

### III.2.b. Lyskryss

Vi ønsket at lyskrysset skulle virke så realistisk som mulig, og sjekket dermed hvordan lyskryssene ute i gaten oppførte seg. Vi la merke til at lyskryssene oppførte seg litt forskjellig fra sted til sted. Vi valgte en lyssekvens som ble et resultat av våre observasjoner.

Lyskrysset valgte vi å sette opp ved banelinjen. Med hovedlyset vendt i motsatt retning av kjøreretningen, og sidelys samt gåendelys vendt i motsatt retning av kjøreretningen til kryssende gate. Altså med en 90 graders vinkel fra hovedgaten.

Vi ønsket å bruke fargesensorer fra Lego Mindstorms til å detektere fargene i lyskrysset. Etter å ha snakket med andre som hadde prøvd lignende og i tillegg etter å ha testet dette, fant vi svakheter hos fargesensoren som gjorde at den ikke kunne lese fargene til lyskrysset godt nok. Det viste seg å være fordi ledlysene som vi brukte i lyskrysset sendte ut for svakt lys og at sensorene krevde en viss avstand fra grunnflaten, slik at fargesensoren ikke klarte å detektere fargene godt nok. I tillegg var fargesensoren mindre konsekvent da

den skulle lese verdier. Dermed måtte vi finne en annen løsning for å sende signaler fra lyskrysset til bilen. Derfor brukte vi Blåtann.

## III.2.c. Blåtann

Da vi først fikk Adafruit sin Bluefruit EZ-Link, ønsket vi å starte veldig enkelt. Lese bruksanvisning, begynne med å koble opp, teste signalene og kjøre enklest mulig programmer kun for å se om den fungerte som den skulle. Dette ble en stor utfordring for oss siden vi ikke hadde loddet før. Signalene var ustabile og det ble mye feilsøking før vi skjønte at vi måtte lodde Blåtann-modulet. Da loddingen var gjennomført, ble signalene fort stabile. Dermed kunne vi sette i gang med å teste hvordan man parer Bluefruit EZ-Link med en PC via Blåtann. Dette var veldig godt forklart i bruksanvisningen som fulgte med Bluefruit EZ-Link. Videre koblet vi Bluefruit til Arduino, og deretter paret vi Bluefruit med en annen PC som hadde Blåtann. Programmet sendte signaler fra Arduino til Bluefruit som igjen sendte signalene videre til PC via Blåtann.

Nå kunne vi begynne å søke etter informasjon om hvordan man sender Blåtann-signaler fra en mikrokontroller med et språk, til en annen med et noe annerledes språk. Altså fra Arduino C via Bluefruit EZ-Link til RobotC. Siden vi allerede hadde testet at man kunne sende signaler via Blåtann fra Arduino til PC, visste vi at det var mulig å skrive signaler gjennom seriell monitor, som var funksjoner vi allerede hadde fra Arduino-biblioteket(Arduino, Udatert).

Det som ikke var like enkelt, var å finne ut hvordan RobotC skulle kunne lese og forstå disse signalene. Vi begynte å se på hvilke typer lignende funksjoner vi kunne bruke i RobotC for å skrive og lese av signaler mellom Arduino C og RobotC. Biblioteket til RobotC (Litteraturliste) hadde utallige eksempler på hvordan man kunne sende og motta Blåtann-signaler, men kun mellom to like NXT-mikrokontrollere fra RobotC. Likevel forsøkte vi å bruke mange av de ulike Blåtann-funksjonene i RobotC til å motta Blåtann-signalene fra Bluefruit EZ-Link. Vi laget så små og enkle programmer som mulig for at det skulle bli enklere å feilsøke.

Vi klarte etter hvert å sende over Blåtann-signaler fra RobotC, til seriell monitor i Arduino C via Blåtann. Signalene ga ikke mening. Hver gang vi sendte, kom det en sekvens

med tall: ”10, 0, 128, 9, 0, 6, 1, 0, 0, 0, 0, 0”. Tallene i sekvensen som vist var ulike, men sekvensen var alltid lik. Det var vanskelig å forstå hvorfor det ble sendt en sekvens med tall, siden det vi hadde sendt fra RobotC, var tallet 1.

Vi brukte Google og Youtube mye for å finne eksempler på noe lignende som var gjort tidligere. Etter mange informasjonssøk og feilsøking, forsøkte vi igjen med en ny runde med informasjonssøk, da kom vi plutselig over en side fra MathWorks (Mathworks, Udatert). Nettsiden forklarte om hvordan RobotC leser signaler fra andre mikrokontrollere enn RobotC. Vi fant ut at den trengte en egen adresse eller telegram for å kunne lese signalet som ble sendt. Det var i tillegg en annen nettside (Sandler A., Udatert) som forklarte en del om telegrammet, men ikke nok til å finne ut hva hele telegrammet skulle inneholde. Men så innså vi at sekvensen med tall som vi tidligere fikk opp da vi sendte signalene fra RobotC til seriell monitor i Arduino C, var dette telegrammet med tall. Dermed sendte vi først tallet 1 fra RobotC til Arduino, så tallet 2, og til slutt tallet 3. Tallsekvensen viste seg å kun variere med “1”, “2” og “3” på en plass i sekvensen, ellers var tallsekvensen helt lik for tallene “1”, “2” og “3”, nå kunne vi bruke den samme tallsekvensen i Arduino-programmet og sende til RobotC. Mot slutten av prosjektarbeidet fant vi også en protokoll som forklarte de resterende plassen i adressen, viser til vedlegg VII7. Vi programmerte i RobotC slik at det kunne motta signaler for så å skrive de ut. Problemet var løst, RobotC klarte å lese alle tre signalene og skrev ut tallet 1 når vi sendte adressen med 1 osv. for tallene 2 og 3.

Da kommunikasjonen via Blåtann mellom NXT og Arduino-lyskrysset var i orden, kunne vi begynne å tenke på hvordan telegrammet med signalene skulle implementeres i lyskryss-programmet. Denne implementeringen er beskrevet tidligere i oppgavenen under programmering og detaljert løsning, Arduino C.

### III.2.d. Bane

Første utfordring var at underlaget ikke var helt jevnt. Flaten som bilen skulle kjøre på, bestod av to store hvite papirark som vi festet med teip. Første gang vi satte opp underlaget, flatet vi ikke ut banen, noe som gjorde at det ble luftbobler under baneflaten og skapte et bølgete underlag. Dette førte til at sensorene ikke alltid leste av riktige verdier i

forhold til den sorte teipen den skulle følge. Dette løste vi ved å stryke underlaget flatt før vi festet det sammen med teip.

Da vi satte opp baneflaten første gang brukte vi en 273x122 meter boks som arena. Når bilen da kjørte løypen så ville avstandssensoren som er festet fremst på bilen oppdage arenaveggen i svingene, noe som førte til at bilen stoppet i og med at programmet har en funksjon som stopper bilen når den kommer for nærme en hindring. Dette løste vi ved å ta underlaget til bilen ut av arenaen og legge den på bakken i gangen, som førte til at bilen kunne kjøre i svingene uten å møte på hinder, og løste samtidig bølge-problemet med papir underlag.

Da bilen kjørte over område med rød farge, så registrerte sensorene kun den røde fargen og sluttet å registrere den sorte streken, noe som førte at den kjørte ut av banen hvis de hadde for stor vinkel mot høyre eller venstre. Dette fikset vi ved å korrigere i programmet slik at sensorene fortsatte å følge den sorte streken samtidig som at de registrerte den røde sonen.

En annen utfordring når det gjaldt bilbanen var å lage svinger i bilbanen som ikke var for bråe slik at bilen kjørte ut av banen. Svingene var laget i form av buer, men var for bråe, noe som førte til at sensorene ikke rakk å lese av løypen og tilpasse seg etter den. Dette ble fikset ved utvide buene slik at sensorene rakk å lese av verdiene til underlaget og følge banen, samt ved å korrigere selve programmet: bilen skal rekke å følge banen.

# IV. Diskusjon

For å få bilen til å rette seg opp, prøvde vi å implementere en mer avansert funksjon som skulle oppnå dette. Dette innebar vansker og andre utfordringer. Det ble slik at hvis bilen kom ut av banen, ville den rette seg opp, akkurat som om den var på riktig spor. Vi prøvde da å lage en funksjon for at bilen kunne “finne igjen” veien. Skjønt dette fungerte bra i spesifikke tilfeller, gikk det ikke alltid, og det skjedde at bilen begynte å kjøre feil. Deretter prøvde vi med en RGB fargesensor til. Meningen var da at hvis sensoren på midten oppdaget den sorte streken, skulle bilen kjøre rett fram, hvis det ellers var en av de to på sidene skulle bilen kjøre mot gjeldende sensor. Men vi tror at NXT-en ikke klarte å håndtere så mange

beskjeder på en gang: vi måtte da nedgradere, og gå tilbake til programmet som vi visste fungerte.

# V. Konklusjon

Dette prosjektet var lærerikt og spennende. Hovedmålene ble nådd. Vi mener likevel at vi kunne ha tatt det lengre med mer tid, forbedring av kode og forsøke med nytt type utstyr.

## V.1. Bil

Videre kan det jobbes med mer avansert utstyr inne i selve Lego Mindstorms, som f.eks. NXT Cam v4 (en type kamera hovedsakelig til Lego NXT) og Lego Mindstorm sin nye mikrokontroller: EV3. Det ser ut som at NXT-en ikke er kraftig nok til å håndtere tre RGB fargesensorer samtidig. Vi har også lagt merke til at denne type sensoren og motoren (Lego Minstorms) ikke er altfor presis og kraftig. Endringer i forhold til dette kunne kanskje hjulpet i forsøket på å løse problemer ved både realistisk kjøring og avvik ved linjefølging, samt hjelpe oss (eller andre) å nå høyere mål, som f.eks. rundkjøring, valg av rute osv. Selv om hardware begrenset våre muligheter er det fortsatt forbedringspotensial i koden.

Dersom bilen brukes i mer kompliserte situasjoner, som f.eks. ekte trafikk, kan vi også tenke oss at kommunikasjonsmiddel vi brukte ikke passer lenger. Vi kunne forsøkt å bruke en annen kanal en Blåtann. Mulig, kunne vi brukt flere Blåtann-moduler samtidig eller sende mer informasjon til flere biler samtidig. Alt dette på en trygg måte. Det er relevant å tenke på hvordan arbeidet vårt kunne implementeres til realsistiske situasjoner, og i hvilken grad dette er mulig å brukes.

## V.2. Lyskryss

Videre arbeid for lyskrysset kunne eksempelvis vært å gjøre om på sidegate- og hovedgate-funksjonene. Disse to funksjonene er på mange måter svært like og kunne gjerne

blitt satt sammen til én funksjon. Dette ville gjort programmet mer ryddig og lettere å feilsøke. I tillegg kunne vi forbedret lyskryssets utseende for å få det til å se mer realistisk ut.

## V.3. Blåtann

Her kunne vi funnet en måte å bruke Blåtann uavhengig av resten av lyskrysset. Som en tråd som kjører samtidig med de andre funksjonene. Vi gjorde informasjonssøk om dette, og leste i flere forum at bruk av tråder ikke er mulig i Arduino C. Dette er noe man kunne gjort flere informasjonssøk for å lære om hvordan man kunne løst det på en annen måte.

# VI. Litteraturliste

Adafruit learning system.(2014). *Introducing Bluefruit EZ-Link Breakout*. Hentet 1.februar 2016 fra https://learn.adafruit.com/downloads/pdf/introducing-bluefruit-ez-link.pdf

Arduino. (Udatert). *Language Reference.* Hentet 1. Mars 2016 fra https://www.arduino.cc/en/Reference/HomePage

Bilder til lego bitene (LegoCarduino: Byggeinstruksjoner): http://www.nightsolo.net/lego/ & http://brickarchitect.com/book/

Ideer til oppbygging: http://www.nxtprograms.com/NXT2/race_car/steps.html

Lego Mindstorms.(2006) *LEGO® MINDSTORMS® NXT Direct Commands.* Hentet 3. Februar 2016 fra http://joanna.iwr.uni-heidelberg.de/projects/NXT_DAME/data/nxt_direct_command.pdf

Mathworks. (Udatert). *Communicating with Lego® Mindstorms® NXT brick over Bluetooth®.* Hentet 10. Mars 2016 fra http://se.mathworks.com/help/instrument/examples/communicating-with-the-lego-mindstorms-nxt-brick-over-Blåtann.html?s_tid=gn_loc_drop&requestedDomain=se.mathworks.com

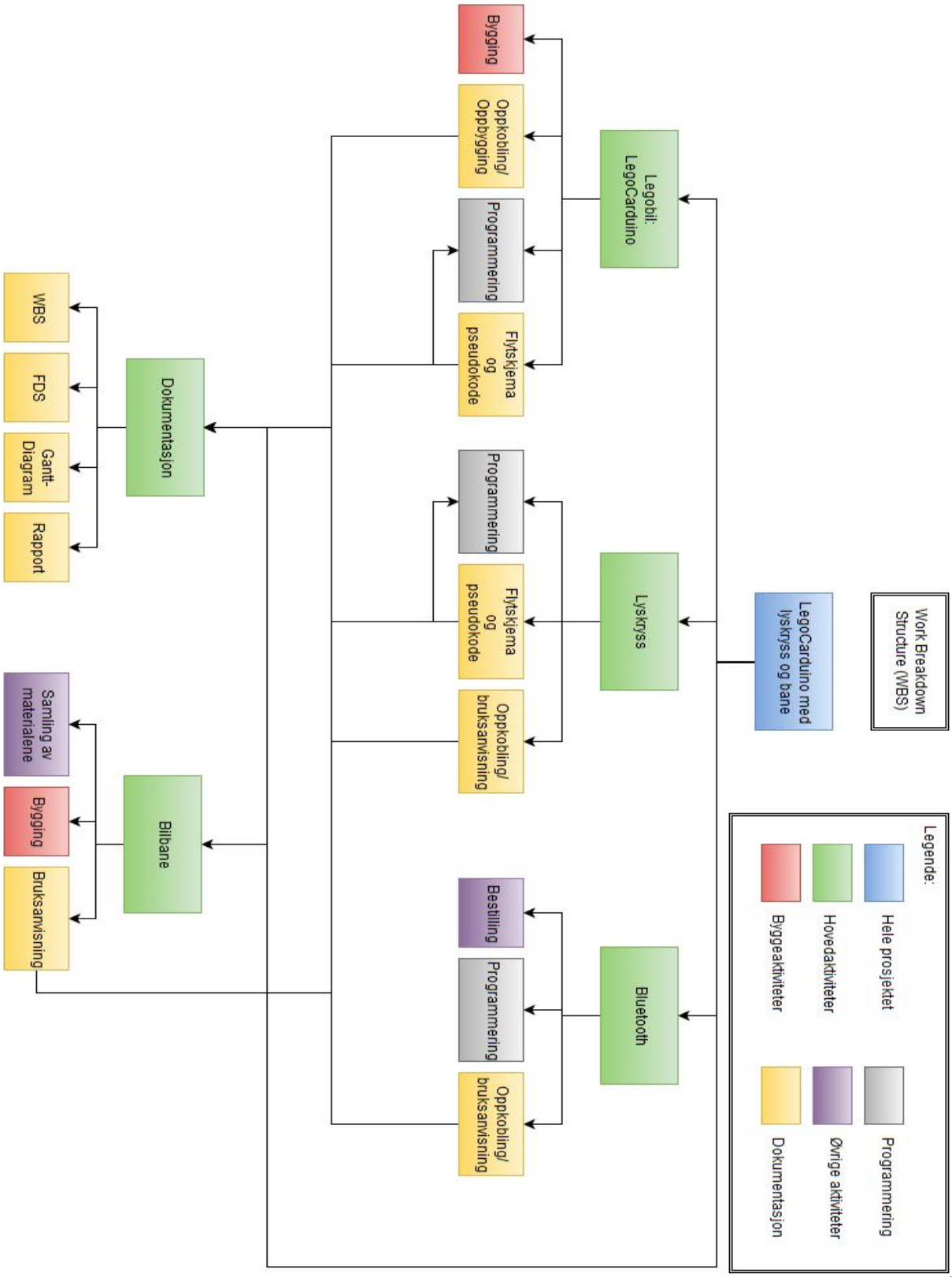
RobotC (2012) *NXT Functions Bluetooth*. Hentet 3. Februar 2016 fra http://www.robotc.net/wikiarchive/NXT

RobotC (2012) *NXT*. Hentet 3. Februar 2016 fra http://www.robotc.net/wikiarchive/NXT_Functions_Bluetooth

Sandler, A.(Udatert) What is a Lego NXT Bluetooth Telegram. Hentet 14. Februar 2016 fra http://www.robotappstore.com/Knowledge-Base/What-Is-a-NXT-Bluetooth-Telegram/24.html

# VII. Vedlegg

## VII.1. Work Breakdown Structure (WBS)



# VII.2. Functional Design Structure (FDS)

## VII.2.a. Legobilen

### VI.2.a.1. Bygging/Oppkobling

Beskrivelse:
Vi skal bygge opp bilen som skal brukes i dette prosjektet, ved hjelp av Lego Technics deler og Lego Mindstorm motorer og mikrokontroller.

Utstyrliste:

- Se 'Vedlegg'

Input Data:

- Lego deler

Output Data:

- Selve bilen

Ressurser:

- To ingeniørstudenter
- 25 timer (mange justeringer)

Utfordringer:

- Bygge opp en modell som eventuelt ikke finnes fra før, som er fin men fortsatt enkel å bruke/håndtere i forhold til prosjektet.
- Sensorene som er ustabile er en utfordring i seg selv: fargesensor f.eks. skal ha akkurat riktig avstand fra bakken for å kunne lese fargen.

Vedlegg:

- Heft "LegoCarduino: Byggeinstruksjoner"
- Heft "LegoCarduino: Bruksanvisning"

### VI.2.a.2. Programmering

Beskrivelse:

Vi skal få bilen til å kjøre etter en løype, samt få avstandssensor til å fungere slik at bilen kan stanse før å treffe hindringer (eller orientere seg rundt løypen).

Utstyrsliste:

- RobotC IDE (Integrated Development Environment) på datamaskin
- Lego deler
- Sort/rød teip
- Hindringer (boks)

Input Data:

- Legobilen
- Flytskjema
- Pseudokode
- Lyssensor-verdier
- Avstands-sensorverdier

Output Data:

- RobotC-programmet

Ressurser:

- En til to ingeniører (Chris)
- 50+ timer

Utfordringer:

- Få bilen til å kjøre etter den sorte teipen
- Finne tilbake til den hvis den hopper ut (og orientere seg rundt hindringer)
- Motta Blåtann-signaler

Vedlegg:

- Flytskjema
- Pseudokode
- Heft “Arduino C og Robot C kode”

# VII.2.b. Arduino lyskryss

### VI.2.b.1. Programmering

Beskrivelse:
Vi skal lage et lyskryss med Arduino C programmeringsspråk.

Utstyrsliste:

- Arduino IDE (Integrated Development Environment) på datamaskin
- Evt. dokumentasjon om Arduino C (programmeringsspråk, funksjoner, osv...)
- Arduino Uno mikrokontroller
- Oppkoblingsbrett & Ledninger
- 8 lysdioder
- 8 motstander på 220 Ohm

Input Data:

- Input data fra sensoren
- Input data fra knappen
- Flytskjema
- Pseudokode

Output Data:

- Output fra Arduino Uno mikrokontroller til de forskjellige pinnene (lys)
- Lyskryssets tilstand - Se "Blåtann (Arduino & Lego)"

Ressurser:

- Tre ingeniørstudenter
- 10+timer

Utfordringer:

- Forenkle programmet så mye som mulig (Den skal kunne leses og bli forstått av noen som ikke har jobbet med selve prosjektet, samt være effektivt)
- Inkludere nytt utstyr for oss i programmet (sensor, og deretter Blåtann-anlegg)

Vedlegg:

- Flytskjema
- Pseudokode
- Heft "Arduino C og Robot C kode"

### VI.2.b.2. Oppkobling

Beskrivelse:

Det skal lages et lyskryss med Arduino utstyr. (I parallell med programmeringen)

Utstyrsliste:

- Arduino Uno mikrokontroller
- Oppkoblingsbrett
- 8 lysdioder (3 grønne, 3 røde, 2 gulle)
- Ledninger
- 8 motstander på 220 Ohm
- 1 kondensator på 1µF/60V
- 1 avstand sensor
- 1 knapp
- 1 Blåtann-"anlegg" fra Adafruit "Bluefruit EZ-Link"

Input Date:

- Arduin C programmet

Ouput Data:

- Data fra sensoren
- Data fra knappen

Ressurser:

- En ingeniørstudent
- ~1 time

Utfordringer:

- Oppkoblingen til det nye utstyret (Sensor og Blåtann anlegg)

Vedlegg:

- Koblingsskjema
- Bruksanvisning

# VII.2.c. Blåtann (Arduino og NXT)

**Programmering av Blåtann**

Beskrivelse:

Lage Blåtann-signal som er lesebar av både Arduino UNO og NXT mikrokontroller. Få Arduino lyskryssignaler til å leses av Blåtann, og deretter sende signaler om lyskryssets tilstand till NXT som den kan lese av. Til slutt implementere programmet som sender og mottar Blåtann-signalene i både Arduino C og RobotC.

Utstyrsliste:

- Lego NXT mikrokontroller
- Arduino UNO mikrokontroller
- Adafruit 'Bluefruit EZ-Link'
- Kondensator 1µF/60V (oppkobling)
- Bruksanvisning for oppkobling av Bluefruit EZ-Link
- Flytskjema og Pseudokode

Input Data:

- RobotC programmet (NXT)
- Blåtann-signal
- Programmet til Arduino UNO
- Flytskjema
- Pseudokode

Output Data:

- Blåtann-signal

Ressurser:

- En til to ingeniørstudenter (ODA)
- 50+ timer

Utfordringer:

- Koble opp Blåtann og få den til å parre med en annen enhet med Blåtann f.eks. PC.
- Koble opp en Blåtann kommunikasjon mellom Arduino UNO og Lego NXT. Få NXT gjennom RobotC-språk til å forstå Blåtann-signalet Arduino UNO sender, eventuelt oversette Arduino-signalet til et signal RobotC kan forstå.

Vedlegg:

- Oppkobling/Bruksanvisning
- Pseudokode
- Hefte “Arduino C og Robot C kode”

## VII.2.d. Bilbane

**Oppsett av bane til Bilen**

Beskrivelse:
Lage en bane for bilen, hvor man kan teste bilens funksjoner

Utstyrsliste:
- Lego bilen (med Lego NXT mikrokontroller)
- Lyskryss (Se ‘Vedlegg’)
- Sort og rød teip til bane
- Arduino UNO med Bluefruit EZ-Link
- Hindring

Input Data:
- RobotC- og Arduino C-programmet
- Samling av materialene og bestilling

Output Data:
- Data fra sort teip
- Hindringer til sensor på bilen
- Blåtann-data fra Arduino

Ressurser:
- En til to ingeniører
- ~5 timer

Utfordringer:
Legge opp løypen slik at bilen greier å kjøre den, eller få bilen til å kjøre hvilken som helst bane.

Vedlegg:
- Heft “LegoCarduino: Bruksanvisning”
- Til lyskryss: Se koblingsskjema og FDS ‘Oppkobling’

# VII.2.e. Dokumentasjon

**Rapport**

Beskrivelse:

- Lage en oversiktlig rapport med god dokumentasjon som gjør at leseren kan utføre og videreutvikle tilsvarende prosjekt på en enklest mulig måte

Målet for denne oppgaven er:

- Hvem som helst skal enkelt kunne utføre tilsvarende prosjekt og jobbe videre med det.

Utstyrsliste:

- Rapport med dokumentasjon om bygging, oppkobling og programmering av bilbane, lyskryss, Blåtann og bil.

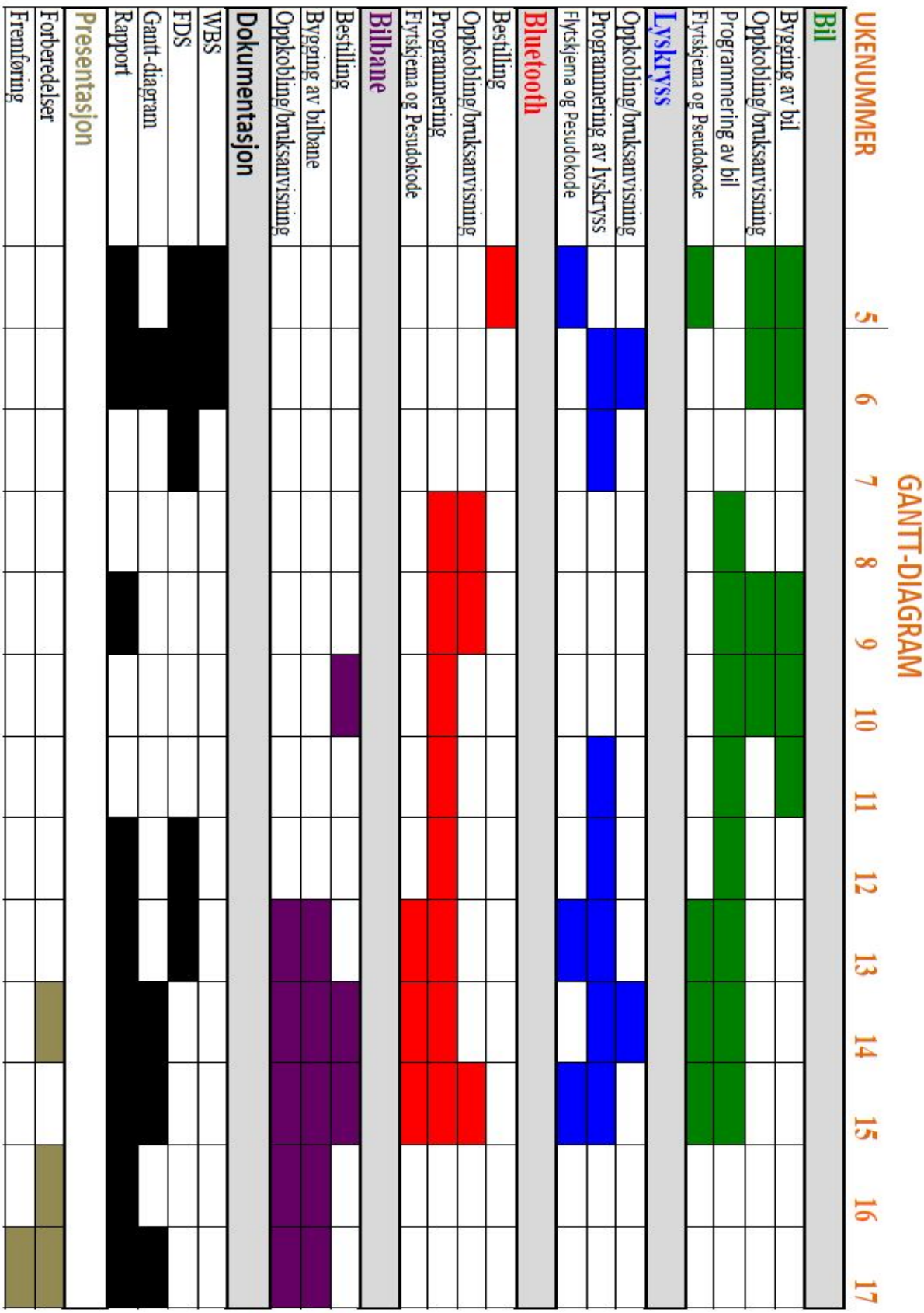
Utfordringer:

- Beskrive og forklare for noen som er helt ukjent med de forskjellige komponentene og mikrokontrollere i prosjektet.

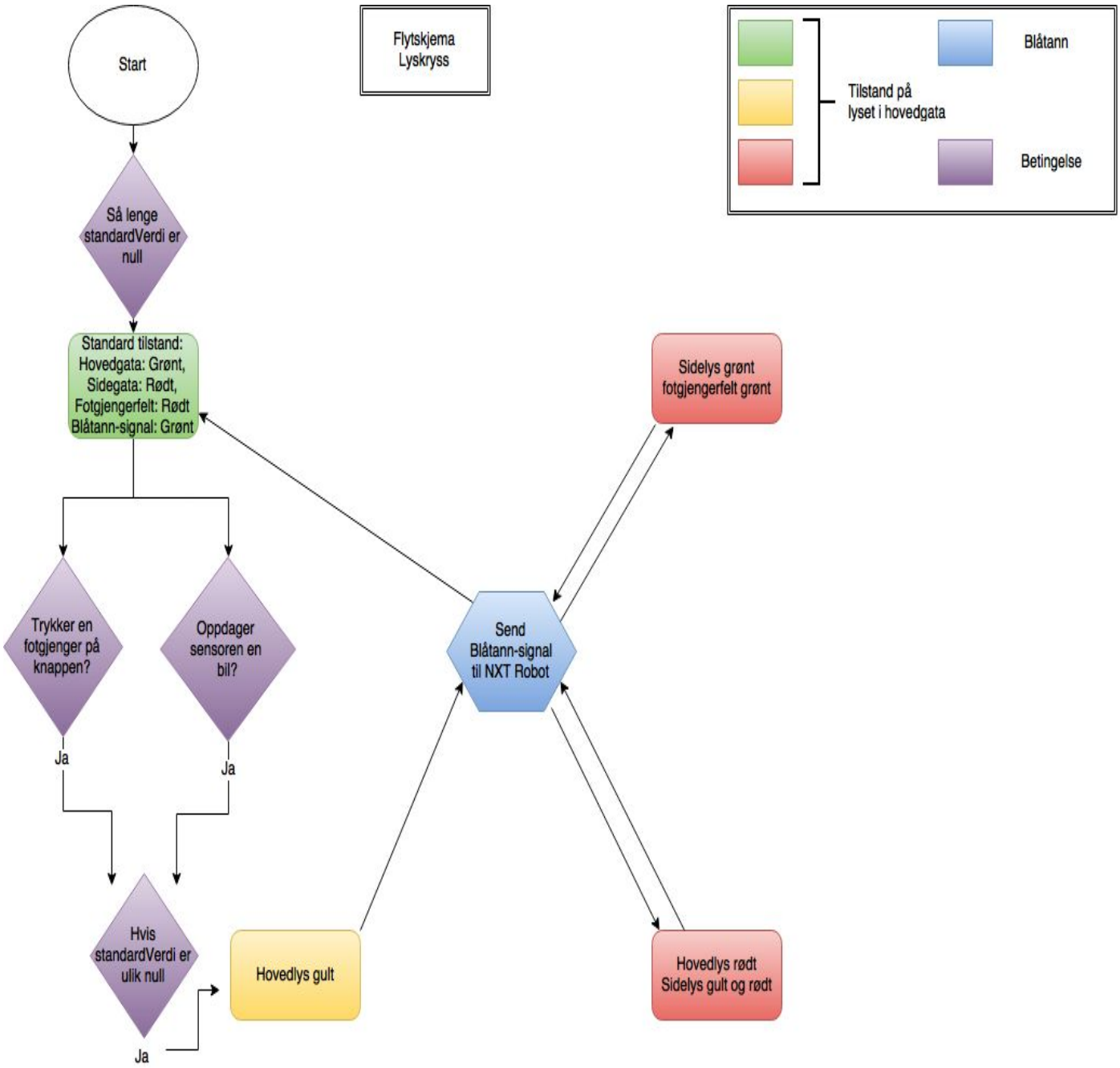
Vedlegg

- Heft “LegoCarduino: Bygningsinstruksjoner”
- Heft “LegoCarduino: Bruksanvisning”
- Heft “Arduino C og Robot C kode”
- Koblingsskjemaer og flytskjemaer
- FDS

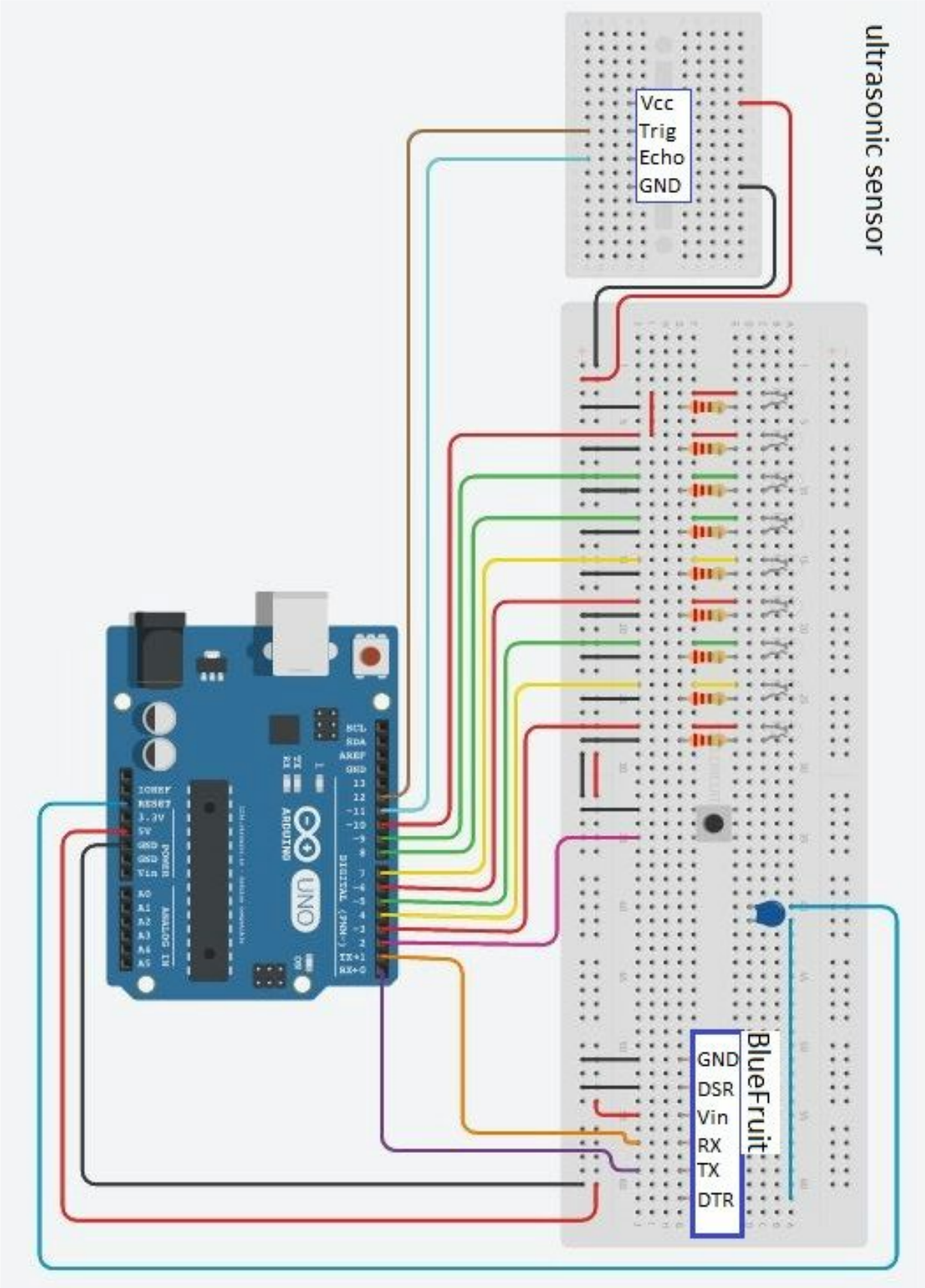
# VII.3. Gantt-diagram



# VII.4. Arduino lyskryss flytskjema



# VII.5. Arduino lyskryss koblingskjema



# VII.6. Arduino pseudokode (Arduino C)

## VII.6.a. Variabler

- Setter variabelnavn til alle porter

## VII.6.b. Setup

- Starter opp seriell kommunikasjon
- Bestemmer at alle porter til ledlys er utganger
- Bestemmer at echo-porten til sensor er inngang
- Bestemmer at trig-porten til sensor er utgang
- Bestemmer at knappen er pullup-inngang

## VII.6.c. Algoritme for styring av lyskryss

Hvis

Setter lyskrysset I standardtilstand:

- Hovedgate får grønt lys
- Sidegate får rødt lys
- Gående får rødt lys

Hvis sensoren detekterer et objekt eller knappen trykkes:

- Hovedgate får rødt lys
- Sidegate får grønt lys
- Gående får grønt lys

## VII.6.d. Utskrift

- Skriver til nxt-roboten via Blåtann når lyskrysset i hovedgaten bytter farge

## VII.7. Forklaring av Blåtann-adresse

{10, 0, 128, 9, 0, 6, 1, 0, 0, 0, 0, 0} - 12 plasser

Plass 1 og 2 Desimalt: 10,0 Heksadesimalt: A,0x00:
Bestemmer telegramlengde. Lengden forteller antall bytes, altså inneholder adressen 10 bytes, ekskludert de to første plassene som bestemmer antall bytes i adressen.

Plass 3: Desimalt: 128 Heksadesimalt: 0x80
Sier hvilken kommandotype som sendes heksadesimalt og oversettes til "direkte kommando, svar kreves ikke"(robotappstore.com).

Plass 4: Desimalt : 9 Heksadesimalt: 0x09

Plass 5: Desimalt: 0 Heksadesimalt: 0x00
Sier hvilket innboksnummer

Plass 6: Desimalt : 6 Heksadesimalt: 0x06
Sier hvilken meldingsstørrelse den har

Plass 7: Desimalt: 1(2, 3) Heksadesimalt: 0x01(0x02, 0x03)
Innholdet i meldingen som sendes

Plass 8: Desimalt: 0 Heksadesimalt: 0x00
Suksess

Plass 9: Desimalt: 0 Heksadesimalt: 0x00
Suksess

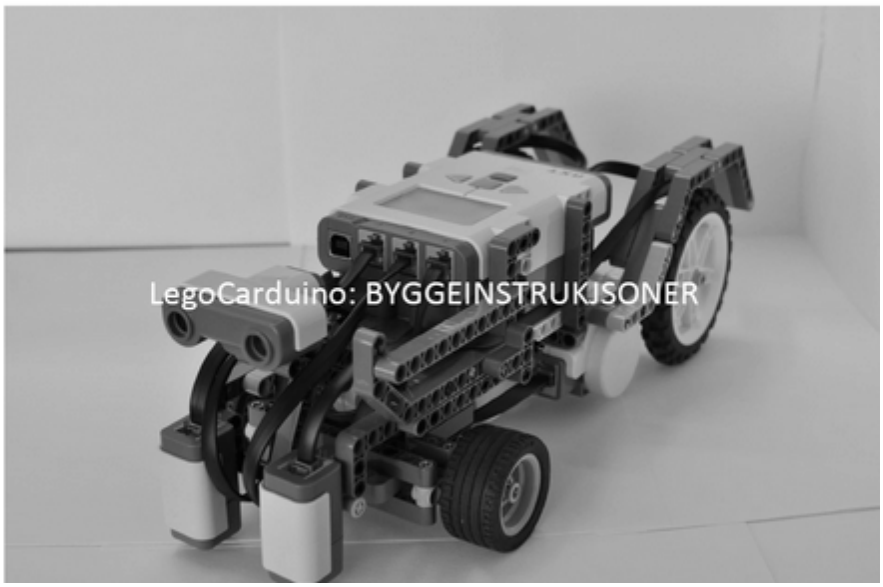
Plass 10: Desimalt: 0 Heksadesimalt: 0x00
Suksess

Plass 11: Desimalt: 0 Heksadesimalt: 0x00
Suksess

Plass 12: Desimalt: 0 Heksadesimalt: 0x00
Suksess

# VII.8. LegoCarduino Byggeinstruksjoner

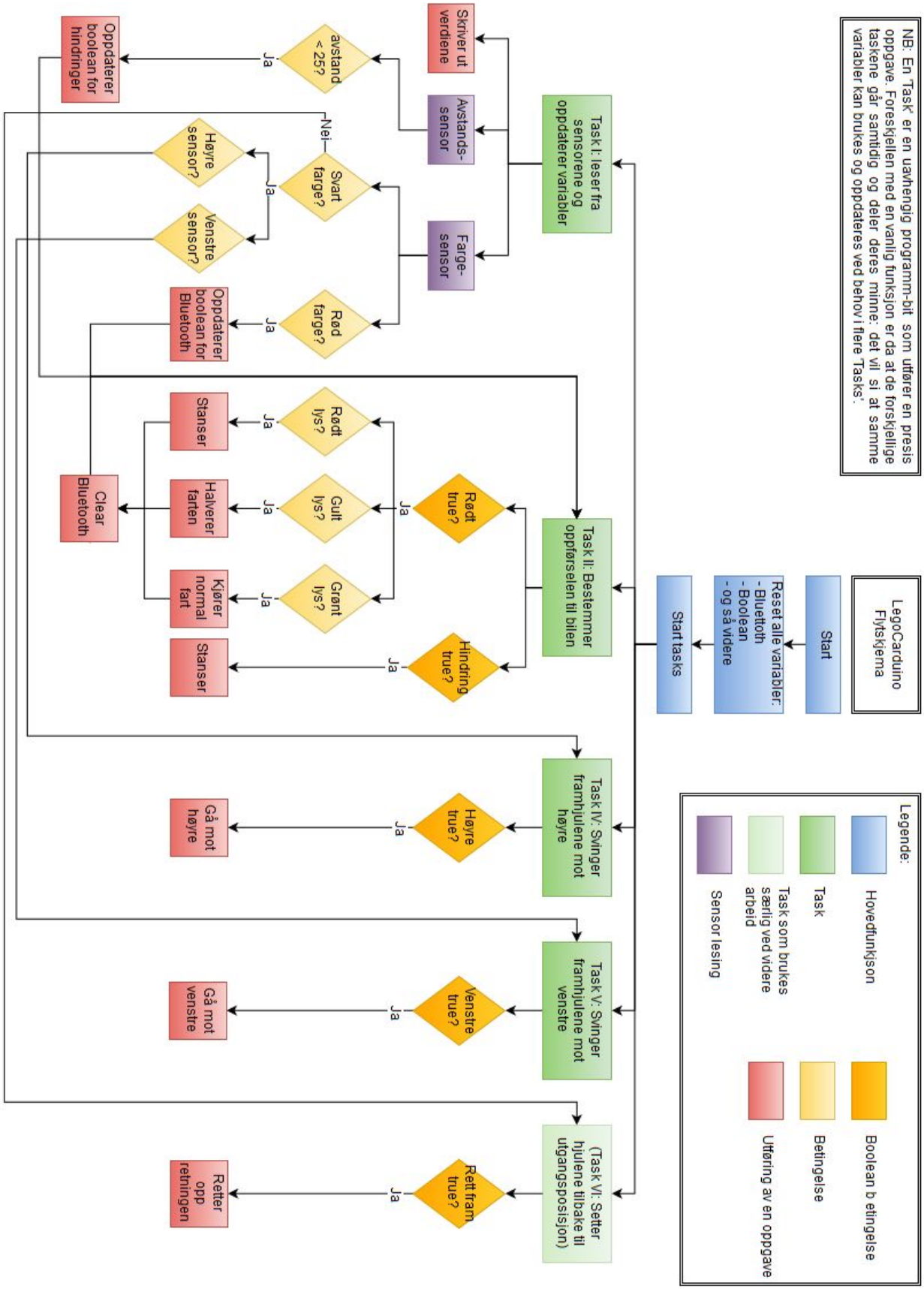
j.f. heftet “LegoCarduino: BYGGEINSTRUKJSONER”




# VII.9. LegoCarduino Bruksanvisning

j.f. heftet “LegoCarduino: BRUKSANVISNING”

# VII.10. Bilens flytskjema



# VII.11. Bilens pseudokode (RobotC)

## VII.11.a. Utgangene på Lego NXT mikrokontroller:

- S1 Ev. sonar (tilleggs avstandssensor)
- S2 Venstre RGB (fargesensor)
- S3 Hoved sonar (avstandssensor)
- S4 Høyre RGB (fargesensor)
- MotorA 'Retnings'motor
- MotorB Høyre motor
- MotorC Venstre motor

## VII.11.b. Variablene:

- Standard fart (kjøre- og svingefart)
- Hvor mange grader MotorA svinger før å stoppe
- Verdiene på svart og rød farge
- Variabel som skal motta Blåtann meldingen
- Alle boolean-variablene.

## VII.11.c. Funksjonene:

- En 'resetAll' funksjon for å sette alle variablene til deres standard verdi
- Skrive taskene:
    - En til å lese fra sensorene, skrive ønskede verdiene på skjermen og oppdatere boolean-variablene, Blåtann, avstand…
    - En til å bestemme retningen og lese fra Blåtann til bilen
    - En som svinger mot høyre, og en som svinger mot venstre
    - (En til å rette opp retningen)
- En main()

## VII.11.d. Main:

- Startsmelding
- Kalle opp resetAll funksjon
- Starte taskene

# VII.12. Kode

j.f. heftet “Arduino C og Robot C kode”