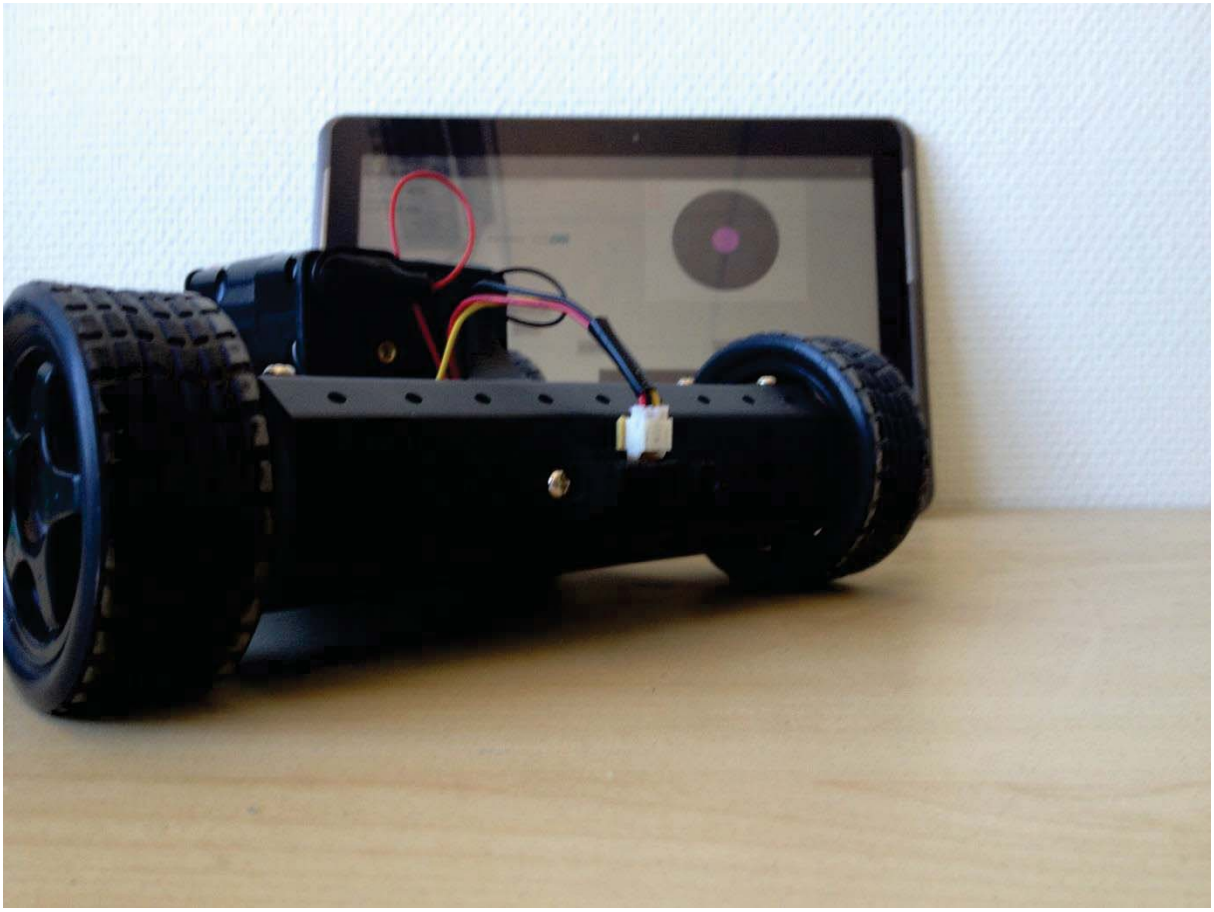
CHALMERS



Halvautonom radiostyrd bil

med Android applikation

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

DAVID WALL

FILIP TIDEMAN

Institutionen för Signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Förord

Detta är ett examensarbete utfört av David Wall och Filip Tideman som läser Mekatronikingenjörsprogrammet, 180hp, vid Chalmers Universitet. Examensarbetet omfattas av 15 högskolepoäng och sträcker sig över en läsperiod, vilket innefattar tio veckor, och görs vid institutionen Signaler och System. Arbetet är utfört hos företaget Broccoli AB, som har sitt kontor i centrala Göteborg, och har tillhandahållit största delen av handledandet samt allt material som arbetet krävt. Ett tack går ut till företaget och dess VD, Björn Bergholm, både för det finansiella stödet samt en väldigt bra idéspruta. Ett tack går även ut till företagets anställda Tobias Olsson som vart med och hjälp oss under utvecklandet av Android-applikationen.

David Wall & Filip Tideman

Sammanfattning

Broccoli Engineering AB är ett konsultbolag verksamt inom fordonsbranschen. Då närvaron på mässor och liknande är frekvent så presenterades en idé att skapa en prototyp av en bil som eventuellt kan medverka på dessa, både som intresseväckare men också som möjlig demonstrator av funktioner. Detta mynnade ut i att en radiostyrd bil konstruerades, som kommunicerar över blåtand och styrs med hjälp av en utvecklad Android-applikation. Arbetet har ägt rum på Broccolis kontor i centrala Göteborg, och material, elektronik och verktyg har antingen beställts eller redan funnits på plats.

Under konstruktionen av en radiostyrd bil ställs man inför en uppsjö av valmöjligheter, och denna rapport behandlar jämförande och argumentation för val av mikrokontroller, trådlös kommunikation, motorstyrning och programmeringslösningar. En ARM-processor kom att användas som mikrokontroller som programmerades i CrossWorks i språket C, och kunde med hjälp av sina in och utgångar ta in och behandla data samt skicka ut data till olika periferienheter. Android-applikationen utvecklades i Java i open source-miljön Eclipse, och tillåter föraren att styra bilen genom olika knappar och kommandon i applikationen. De största funktionerna som lyckades implementeras var att bilen, vid tappad kontakt med föraren, kan autonomt köra tillbaka till platsen där räckvidd tidigare existerat och själv etablera en ny uppkoppling mot föraren. Detta ska illustrera hur människa och teknik kan samarbeta för att åstadkomma en bättre och smidigare upplevelse för användaren. Hela arbetet resulterade i en tillfredsställande prototyp där bil både kan köras manuellt genom mänsklig interaktion men även helt autonomt utifrån ett mönster som ritas upp av föraren. Däremot fick viss funktionalitet som planerats under projektets tidiga skede förkastas, då implementering av viss funktionalitet drog ut på tiden mer än planerat. Denna funktionalitet omnämns i slutet av rapporten med tankar hur dessa skulle kunna lösas ifall vidareutveckling av prototypen är av framtida intresse.

Summary

Broccoli Engineering AB is a consulting firm that offer services towards mainly the automotive industry. Broccoli is a frequent visitor to different fairs so an idea was presented to create a prototype of a car that possibly could be a part of their booth, both as an eye catcher but also as a demonstrator. This led to the creation of a radio controlled car that communicates using Bluetooth and controlled using a developed Android application. The work took place at Broccoli's office in Gothenburg city center and all the material, electrical components and tools were either already in house or ordered online on the expense of Broccoli.

During the creation of a radio controlled car you are faced with many different options, and this thesis covers comparison and justification for the choosing of a micro controller, wireless communication, motor controller and programming solutions. An ARM processor became the micro controller of choice and was programmed in C language in CrossWorks, and by using the in and out puts one is able to send data to peripheral units. The Android application was developed in Java language using the open source environment Eclipse, and allows the driver to control the car using buttons and commands in the application. One of the most important features that was implemented was the car, upon loosing wireless connection with the driver, was able to autonomously return to where connection previously existed, and automatically once again establish connection with the driver. This to illustrate how co-operation between man and technology is able to create a better and more optimized experience for the user. The project resulted in a satisfactory prototype where the car can be driven both manually by a driver but also completely autonomous by following a pattern that the driver can draw in the application. Unfortunately the development of some functionality took longer than expected, and because of this some features never got developed due to lack of time. These features are mentioned at the end of the thesis where thoughts and ideas on how to implement these are presented, if it would be of interest to further develop the prototype.

Innehållsförteckning

Beteckningar

PWM - Pulse Width Modulator

ADC - Analog to Digital Converter

TIM - Timer

ARM - En processorarkitektur som återfinns i de flesta inbäddade mikrokontrollers

USART - Universal Synchronous/Asynchronous Receiver/Transmitter

MAC-Address - Media Access Controll-Address

WiFi - Wireless Fidelity, ett trådlöst protokoll

RSSI - Received Signal Strength Indicator

ISR - Interrupt Service Routine

PIC - Peripheral Interface Controller

GRPR – Golden Receiver Power Range

1. Inledning

Radiostyrda bilar är någonting som funnits i flera decennier i ett flertal olika utformningar. Länge har det använts som en leksak för barn men på senare år även i tävlingsform. Tanken med detta projekt är att skapa en bil i mer modern tappning med utökad funktionalitet som även ska förmedla en känsla av autonomi, genom att förarens manuella inmatning kompletteras med en programmerad intelligens.

1.1 Bakgrund

Broccoli AB är mjuk- och hårdvarukonsulter med främsta fokus på fordonsbranschen. Med detta i åtanke presenterades en idé om en kontrollerbar bil i miniatyr, bestyckad med bland annat mikrokontroller och givare för att möjliggöra implementation av olika funktioner och även upprätta en trådlös kommunikation mellan bilen och dess förare. Denna bil är ingenting som specifikt efterfrågats utav företaget, men den kan komma att användas i mäss-sammanhang för att locka besökare och möjligtvis även för att demonstrera viss programmerbar funktionalitet som dagens bilar har, till exempel förhindrande av kollision.

1.2 Syfte

Projektet går ut på att inom givna tidsramar skapa en bil i miniatyr som ska kunna manövreras med hjälp av trådlös kommunikation genom en Android-applikation som studenterna själva ska utveckla. En mikrokontroller ska även programmeras för att kunna ta in signaler från givare och modulen för trådlös kommunikation, samt skicka ut styrsignaler för motorstyrning. Slutresultatet ska vara en bil som ska kunna kontrolleras via den framtagna applikationen där även olika säkerhetsnivåer ska finnas tillgängligt för föraren, som bestämmer hur pass nära ett objekt bilen får köra. Detta ska illustrera hur samarbetet mellan människa och teknologi kan skapa en synergieffekt som får med förarens säkerhet som huvudprioritet.

Projektet ska även innefatta en autonom del där bilen ska kunna manövrera utan hjälp av en förare. Den funktionalitet som ligger till grund för den autonoma delen är att om och när bilen åker utanför räckvidden för den trådlösa kommunikationen, kommer bilen på egen hand köra tillbaka till den senaste platsen där kommunikation funnits, och ifall detta på nytt kan etableras ska detta skötas per automatik.

Slutligen har arbetet även som funktion att ge en chans för studenterna att visa upp kompetens inom såväl programmering av mikroprocessor och Android-applikation samt förståelse och konstruktion av elschema och allmän elektronikkunskap.

1.3 Avgränsningar

Arbetet kommer att sträcka sig över tio jobbveckor där den största tiden kommer att tillbringas på Broccolis kontor. Efter diskussioner med handlare beslutades det att gällande elektriska komponenter så kommer halvfabrikat att användas, då fokus skulle ligga på själva framtagandet av funktionalitet och inte skapandet av kretsar som redan finns färdigt i ett chip. Analogt med detta tänk så användes även ett färdigt robotkit där chassi och motorer medföljde, åter igen för att få en kortare startsträcka. Projektets budget preciserades aldrig men rymde användandet av halvfabrikat, ändå fast man ur ett ekonomiskt perspektiv hade kunnat komma undan billigare genom att konstruera behövt material från grunden..

Då projektet kan byggas vidare i näst intill all oändlighet är det viktigt att ha en klar uppfattning om när arbetet anses vara klart, och därav skapades en kravspecifikation som presenteras i punkten nedan som ger ett tydligt ramverk för vad som det färdiga arbetet ska innefatta.

1.4 Kravspecifikation

För att ha ett tydligt målsnöre att sträva mot så sattes en kravspecifikation upp som listar grundläggande krav samt extra funktionalitet som önskades finnas tillgängligt på prototypen vid arbetets avslut.

Grundkrav:

- Trådlös tvåvägskommunikation mellan förare och bil
- Android-applikation med styrning av bil
- Förhindrande av kollision
- Autonomt köra tillbaka inom räckvidd för trådlösa kommunikationen då den tappats

Extra funktionalitet:

- En mer avancerad styrning av bil i applikation
- Färddata från bil till förare
- Överlämning av styrning mellan två förare
- Mappning av omgivning med hjälp av sensorer

2. Teknisk bakgrund

2.1 Radiostyrd bil

Redan på 1940-talet började man experimentera med bilar i miniatyrstorlek, detta tack vare nitrometan-baserade motorer som introducerades under det decenniet. Då användes en tråd mellan bil och förare för att etablera kommunikation, men det dröjde ytterligare 20 år innan det kom att närma sig det vi idag tänker på när man pratar om en radiostyrd bil. 1966 var året då det Italienska företaget El-Gi (Elettronica Giocattoli) släppte den första radiostyrda bilen för konsumenter, och var en kopia av Ferraris 250LM i skala 1:12.

Det som gjorde det möjligt för den radiostyrda bilen att släppas just under detta tidsvarv var tack vare transistorernas intåg på marknaden, vilket gjorde att man nu inte bara kunde skicka en signal innehållandes en av två tillstånd, på eller av (även känt som bang-bang), utan kunna anpassa signalen som skulle skickas till flera olika nivåer. Detta kallas även proportionell styrning och kan tack vare den bakomliggande regleralgoritmen rampa styrsignaler upp eller ner, och i förlängningen gjorde detta att den radiostyrda bilen vi känner idag kunde realiseras. Den regleralgoritm som används är en väldigt enkel sådan som tar hänsyn till är- och börvärdet för att räkna ut felet, och sedan även förstärkningen, för att skapa denna upp- och ner-rampning. Nu kunde man alltså välja att köra bilen i olika hastigheter och även svänga olika skarpt beroende på förarens inmatning.

Spolar man fram tiden ytterligare, ungefär till mitten av 70-talet, anländer man vid tiden då den radiostyrda bilen inte bara längre använde sig av förbränningsmotorer, utan även elektriska. Det dröjde dock ytterligare innan den elektriska radiostyrda bilen blev en seriös utmanare, då batterierna som användes var antingen stora och tunga blybatterier eller utbytbara engångsbatterier. Det var när nickel-kadmium batteriet började användas i mer elektroniska applikationer som den elektriska radiostyrda bilen blev erkänd som ett alternativ till den klassiska radiostyrda bilen. Sedan detta årtioende har enbart mindre förändringar tillkommit och i sin helhet har den radiostyrda bilen mer eller mindre sett ut likadan ut i nästan 40 år (13).

2.2 Mikrokontroller

En mikrokontroller är en integrerad krets där en processor, minne och in- och ut-portar alla finns på en och samma bricka. Dess huvudsakliga syfte är vara inbyggt i ett större system och fungera som själva hjärnan, som styr beteendet hos systemet. Dessa återfinns i väldigt många appareter som vi i vår vardag kommer i kontakt med, till exempel som tvättmaskiner, bilar, kontroller, leksaker, medicinsk apparatur med mera.

En mikrokontroller funktionalitet grundar sig på avbrott och hantering utav dessa. När en specifik händelse inträffar kan man med hjälp av denna avbrottshantering avbryta nuvarande exekvering för att gå in i en avbrottsrutin, så kallad ISR, och utföra önskad uppgift beroende på vilken händelse som triggade avbrottet. På detta sätt kan man enkelt skapa en programloop som ligger och körs tills en händelse inträffar, till exempel en mottagen insignal från en givare, och därefter utföra en önskad sekvens. En annan variant av detta är timers, som precis som avbrotthanteraren tillåter programmet att hoppa in i en önskad del eller sekvens, men istället för att vara händelsestyrd är den beroende av en vald tidsfrekvens. Med timers kan man till exempel skicka ut signaler med ett önskat intervall oberoende av vart programmet befinner sig (10).

2.3 WiFi

WiFi är en trådlös teknik som baserar sig på 802.11 specifikationerna och möjliggör trådlös kommunikation över några specifika frekvensband, där det absolut vanligaste bandet ligger på 2.4 GHz. WiFi protokollet tillåter alla enheter med ett trådlöst nätverkskort att koppla upp sig till mot ett trådlöst LAN, Local Area Network. Detta lokala nätverk innebär att information kan skickas och hämtas från enheter uppkopplade mot detta nätverk, och kommunikationen är därmed upprättad. Då WiFi inte kräver någon fysisk inkoppling är det vanligt är att detta nätverk är skyddat av en typ av kryptering som enbart tillåter använder med rätt säkerhetsfras åtkomst till nätverket. Väl uppkopplad mot nätverket finns det stora möjligheter för delning av diverse information, som till exempel skrivare eller internet-delning. WiFi har tack vare den numera billiga teknologin blivit en standard bland de flesta bärbara enheter såsom laptops, telefoner, kameror, läsplattor med mera och är under ständig utveckling med nya funktioner och stöd för högre överföringshastigheter (9).

2.4 Blåtand

Blåtand är, precis som WiFi, en trådlös teknik som togs fram främst med intresse av att på ett kostnadseffektivt sätt kunna koppla samman enheter utan att använda en sladd. I likhet med WiFi används frekvensbandet 2.4GHz , men nyttjar kortvågiga radiofrekvenser vilket innebär att, precis som namnet implicerar, räckvidden är mera begränsad. Tanken är att enkelt kunna sätta upp en, ofta temporär, trådlös kommunikation för överföring av olika typer av data. Till skillnad från WiFi så fungerar blåtandens uppkopling efter en master-slave struktur, där enhet som gör förfrågan om att kopplas mot en annan enhet blir master, och enheten som blev tillfrågad blir slave. När en förfrågan om uppkoppling gjorts efterfrågas vanligtvis en PIN-kod och vid godkännande av denna kod så paras enheterna ihop och överföring av olika typer av data är nu möjlig (11).

2.5 Den autonoma bilen

En autonom bil är en bil som kan manövrera i en miljö utan mänsklig inmatning. Genom användandet av olika typer av sensorer och tekniska lösningar, som till exempel radar, optiska sensorer och GPS, så kan bilen avgöra hur den ska navigera sig fram till angivet mål. Arbetet med autonoma bilar har pågått sedan 1980-talet, men det är inte förrän senaste åren som teknik och lagstiftning gjort det möjligt för dessa bilar att testas i en riktig förarmiljö, på öppna vägar tillsammans med mänskliga bilförare. År 2013 hade tre stater i USA lagstiftat om tillåtande av testande av autonoma bilar på offentliga vägar, och en fjärde stat som tillät detta under förutsättning att en människa sitter i bilen hela tiden under körningen (14).

Försvarsmakten i USA annonserade att år 2004 skulle en årlig tävling påbörjas, kallat “The Grand Challenge”, där bland annat forskarteam, ingenjörer och företag har möjlighet att vinna en årligen ökande summa pengar, som år 2004 låg på en miljon amerikanska dollar. Kriteriet för vinst är att en helt autonom bil ska lyckas ta sig igenom en offroad-bana och först in i mål, dock ett kriterium som under årets lopp förändrats till bland annat fokus på navigering i storstadsmiljö och underhåll av robotar i farliga miljöer. Anledningen till detta incitament är att USA satt upp som mål att en tredjedel av deras markgående militära styrkor ska vara helt autonoma till 2015. Detta har även lett till att fler människor har involverats i utvecklingen av den autonoma bilen och enligt prognoser kan vi förväntas oss en kraftig ökning av förarlösa bilar inom de närmaste åren (15).

3. Metod

Vid projektets början skapades en planeringsrapport som tog upp mål, krav och tidsplanering för de kommande tio veckorna. Under första veckan sattes mål upp att undersöka och beställa komponenter som skulle behövas för bilen, kommunikationen och styrningen, samt sätta upp de olika programmeringsmiljöerna så de stod klart tills att komponenterna fanns till hands. Nästkommande veckor kom fokus att ligga på att lära sig hantera mikrokontrollern och bli bekväm i utvecklingsmiljön, detta genom att följa grundläggande guider och titta på en del färdiga exempel för inspiration. Ett JTAG-kort användes för att ladda över programvaran på mikrokontrollern, och för att underlätta test av olika enklare program så löddes ett par lysdioder och DIP-hållare fast på ett labbkort, för att kunna fästa mikrokontrollern och testa bland annat signaler på in och utgångar och AD-omvandling. AD-omvandlingen testades till exempel genom att mata ut IR-sensorns utsignal, mellan 0.3 och 3.0 V, till en lysdiod som då lyste olika starkt beroende på avståndet som sensorn fångade upp.

När all grundläggande funktionalitet var på plats så monterades bilens chassi ihop, labbkortet placerades i bilens kärna och kopplades sedan upp mot ett nätaggregat för att testa motorerna. För motorstyrningen användes ett chip med två H-bryggor mellan PWM-utgångarna på mikrokontrollern och motorerna för att möjliggöra hjulens rotation åt båda håll. Alla de elektriska kringkomponenterna som krävdes för de olika enheterna valdes utefter datablad som specificerar storlek och typ för olika applikationer.

I femte veckan ansågs bilen vara klar för testkörning, och dess tidigare stationära tillstånd uppkopplad mot ett nätaggregat byttes mot en batterihållare med sex stycken AA batterier så bilen nu kunde köras på mark. En enkel Android-applikation hade tagits fram med fyra pilar som kunde köra bilen rakt fram, höger, vänster samt bakåt, som möjliggjorde test av till exempel anti-krock systemet och verifiera hur vida värden från databladet överensstämde med verkligheten. En hel del buggar, både i Android-applikationen såsom bilens beteende, upptäcktes under första testkörningarna men benades så småningom ut, mycket tack vare internetforum men även försök och misslyckanden ledde slutligen till tillfredställande lösningar.

Under de sista veckorna var bilen körklar och den grundläggande funktionalitet som sats upp som krav i projektplaneringen uppfyllda. Fokus lades på att utveckla autonom funktionalitet och bättra på körupplevelsen, vilket resulterade i att de fyra pilarna som användes för manövrering byttes ut mot en virtuell joystick samt en ritfunktion lades till där föraren kunde rita ett mönster som bilen skulle härma. Funktionaliteten finns närmare beskriven i ett kapitel längre ner där beskrivning, tankegångar och misslyckanden återfinns.

Under hela arbetets gång fördes en loggbok veckovis där olika försök, närmaste framtidens planer och milstolpar dokumenterades. Detta visade sig vara väldigt behändigt, både för rapportskrivandet men även för att lättare plocka upp bollen efter helgen eller ett längre uppehåll i och med lov.

4. Val av trådlös kommunikation

För att kunna manövrera bilen så behövdes någon form av trådlös kommunikation mellan förare, som använder sig av en Android-platta, och bil. De två alternativen som valdes att undersökas var WiFi och blåtand. Anledningen till att just dessa två studerades var att de båda är en industristandard ute på företag, vilket också inkluderar arbetsplatsen där jobbet utfördes, och i och med det en god erfarenhet att ha arbetat med.

Då kunskapen kring mer ingående skillnader mellan de båda protokollen var någorlunda begränsade gjordes till och börja med en lättare kravspecifikation för den trådlösa kommunikationen som listade följande krav:

- Åtminstone ~10 meters räckvidd
- Stabil dataöverföring
- Rimlig prissättning på behövda komponenter
- Strömsnål

Vid undersökandet av räckvidden för de båda protokollen fann man att WiFi utan tvekan uppfyller det efterfrågade kravet. I fallet blåtand dock, beror det på vilken typ av klass som skulle användas, då klass 1 stödjer upp till 100 meter, klass 2 upp till tio meter, och klass 3 enbart upp till en. Med denna information visade det sig vara gångbart med både WiFi och blåtand, bara blåtandsklassen togs till hänsyn vid val av komponent (9) (11).

Gällande dataöverföring så är kravet på överföringshastighet låg, då ingen tyngre information kommer att skickas utan enklare styrsignaler samt lite positioneringsdata. Däremot är kravet på att kommunikationen ska vara kontinuerlig och stabil av mycket större vikt, då ett kortare avbrott i kommunikation kan misstolkas som att bilen kört utanför räckvidden för kommunikationen. Vad som funnits vid undersökande av de två är att stabiliteten är väldigt snarlik när det handlar om områden där förbindelsen har en bra signalstyrka, och de båda alternativen är fortfarande gångbara (12).

De två sista kraven som ställts visade sig vara det avgörande i beslutsfattandet kring den trådlösa kommunikationen. Hos de leverantörer som företaget använder sig av låg WiFi-komponenterna på ungefär det två- eller tre-dubbla priset av blåtands-komponenterna, vilket till största anledningen är på grund av att WiFi erbjuder starkare och större utbud av funktioner. Detta är dock på bekostnad av större strömförbrukning, vilket var ett av de kraven som ansågs viktigast då enklare AA-batterier skulle användas och det fanns därför en önskan om att hålla strömförbrukningen så låg som möjligt (6). Med dessa slutliga argument valdes blåtand framför WiFi då det visade sig bäst lämpat för projektet.

5. Urvalsprocess för material

Vägen till att skapa en radiostyrd bil är inte enspårig och uppenbar, det finns väldigt många olika vägar att gå och möjligheterna är långt ifrån begränsade. Nedan följer tankegångar kring val av de olika komponenter som krävdes för realisation av den radiostyrda bilen.

5.1 Bilens fundament

Initialt var tanken att köpa en redan färdig radiostyrd bil, för att montera isär den, och sedan nyttja vissa delar av den till projektet. Svårigheten med detta var att avgöra hur pass kompatibla dessa delar skulle vara med de andra komponenterna som skulle användas, och på grund av begränsad information kring detta, samt en strävan efter att göra så mycket som möjligt från grunden, så förkastades idéen. Med hjälp av sidan rsb.se kunde tips från medlemmar som tidigare gjort liknande projekt användas i beslutsfasen vilket, i kombination med diskussion med handledare, ledde till att en halv-färdig robotsats valdes som grund för projektet. Denna robotsats innehöll chassi, hjul och motorer med tillhörande skruvar och sladdar. Slutligen behövdes ett labbkort, som Broccoli tillhandahöll, för att kunna löda på de elektriska komponenterna (16).

5.2 Mikrokontroller

De tre varianterna olika mikrokontroller som tittades på var ARM, PIC och ett Arduino-kort (med tillhörande AVR-processor). PIC-kontrollern kändes som en självklar kandidat då det arbetats med flitigt i laborationer och undervisning. Däremot var både ARM och Arduino någonting helt nytt, men valdes som potentiella kandidater då de används i mycket större grad både på Broccoli och ute i arbetslivet. Detta sågs inte bara som en bra utmaning utan också som ett bra argument för att välja dessa framför PIC, och därav förkastades den kontrollern. Vid diskussioner med handledare och erfarna ingenjörer på företaget angående de två resterande alternativen listades för- respektive nackdelar med dem för att få en bättre överblick över vilken som skulle användas i slutändan.

<u>ARM</u>

- Generellt sett stort minne
- Snabb
- Billig
- Större kod
- Omfattande, på bekostnad av mer komplex

Arduino

- Enkel att komma igång med
- Mycket färdig kod/funktioner att tillgå
- Mer än bara en processor (viss annan hårdvara medföljer, beroende på kort)
- Långsammare
- Dyrare

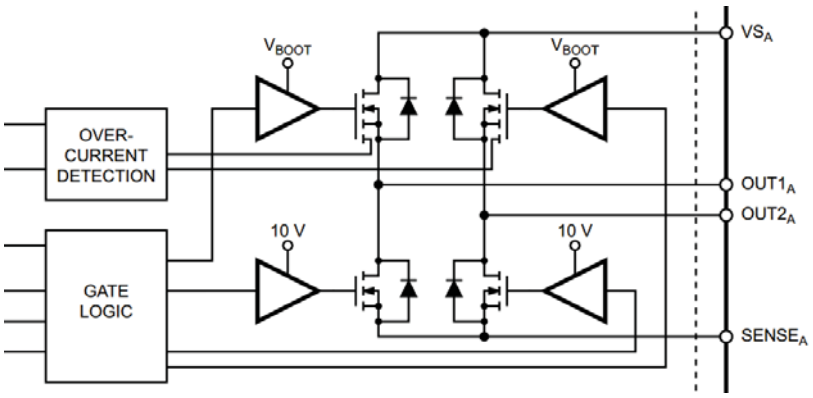
Utifrån dessa kriterier valdes ARM-processorn att gå vidare med för utveckling, med det tyngsta argumentet att den helt enkelt är mer använd ute i arbetslivet och samtidigt inte heller kommer vara för komplett, då förhoppningen på projektet var att i så lång utsträckning som möjligt skapa något på egen hand.

5.3 Elektriska kringkomponenter

Utöver mikrokontrollern så behövdes även kringkomponenter för att bland annat få rätt spänningsmatning till och från de olika komponenterna på labbkortet. Medföljande i det tidigare omnämnda robotkitet var bland annat en batterihållare för sex stycken AA batterier, som var för sig ligger på 1,5 V vilket kan leverera en maximal totalspänning på 9 V med seriekopplade batterier. I arbetets slutskede valdes dock att gå upp till att använda åtta stycken AA-batterier för att få ut större matning till motorerna och därav kunna nå högre hastighet, med en levererad maximalspänning på 12 V. Mikrokontrollern skulle dock enbart matas med 5 V och därför behövdes en spänningsregulator användas tillsammans med kondensatorer för att reglera spänningen efter önskad storlek. Spänningsregulatorn som valdes var LM7805C då den uppfyllde efterfrågade krav på både in och utmatning av spänning. För att försäkra oss om att spänningen ska vara stabil så användes även två kondensatorer i storleken 0,33 uF, mellan input (12 V) och output (5 V), och 1 uF mellan GND och input. Val av dessa gjordes direkt utifrån databladen för de olika enheterna där storlek och typ kunde utläsas beroende på användningsområde (5).

5.4 Motorstyrning

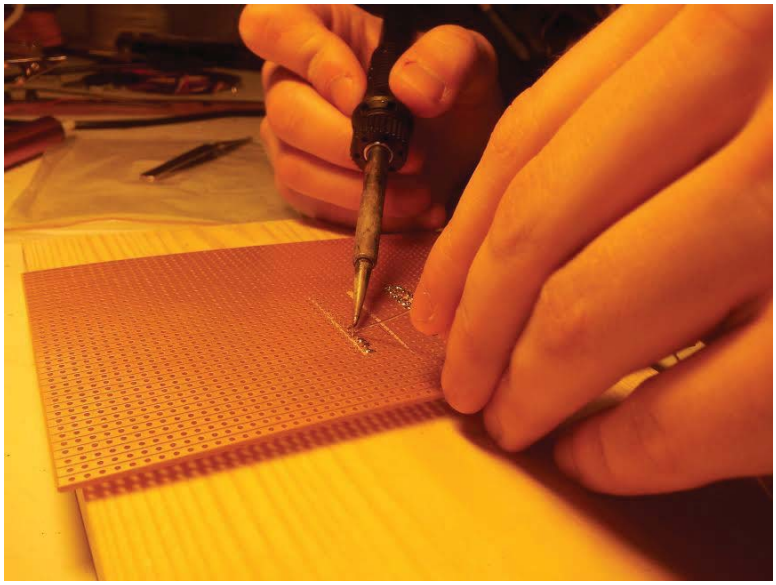
För att kontrollera motorerna används PWM-utgångarna på mikrokontrollern, där fyra stycken finns till förfogande. Dessa matar ut en justerbar spänning på de ben som finns specificerade i mikrokontrollerns datablad. Problematiken kring detta var att kunna styra bilens fyra motorer i båda riktningar, fram och bak, vilket kräver någon typ av elektrisk krets för att ändra strömmens riktning. Vid konsultation med handledare framkom att användet av en H-brygga skulle vara bäst lämpat för arbetet. Vidare behövdes heller inte fyra bryggor som man kan tro vid en första anblick, utan enbart två på grund av att de motorer som sitter på samma sida om bilen aldrig behöver kunna snurra åt olika riktningar. Med bristande utrymme på labbkortet så inhandlades ett färdigt chip, L6206N DIP, som har två fulla H-bryggor med färdiga in och utgångar som löddes fast och åter igen användes medföljande datablad för att välja lämplig storlek på resistorer och kondensatorer. För att få all elektronik att fungera användes kopplingsschemat för motorstyrning, som återfanns i databladet (8).



Figur 1. Överblick av H-brygga

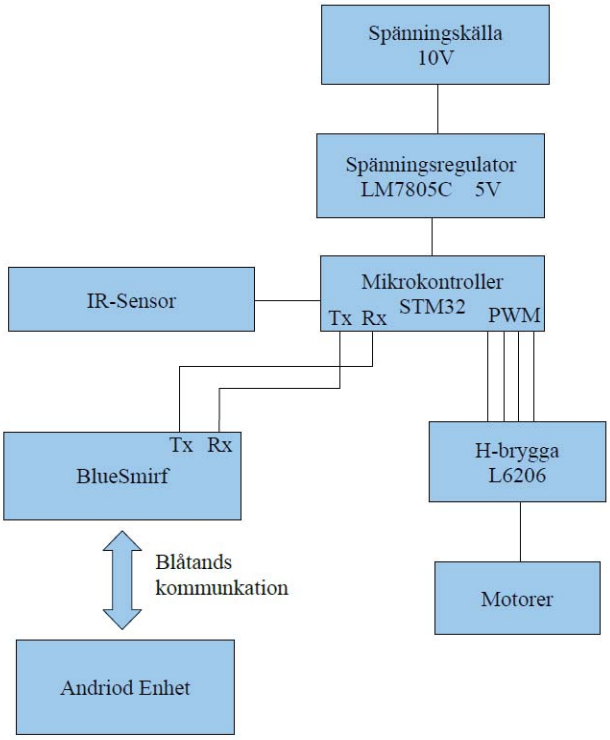
6. Utformning av labbkort

När komponenter valts ut och beställts efter så löddes de på eftersom på labbkortet. Labbkortet som användes är uppbyggt av rader med konduktivt material och hål fördelade för att passa DIP-komponenter, som har ett visst mellanrum mellan varje ben. För att göra plats åt de komponenter som använts krävdes det att avgränsningar på kortet skapades. Detta gjordes genom att med fil skrapa bort det ledande materialet där man önskar att elektricitet inte ska ledas, och kan på så sätt skapa avdelningar för olika komponenter på samma rad.



Figur 2. Lödning av komponenter

De komponenterna som slutligen användes för projektet presenteras i ett enklare kopplingsschema nedan. Under arbetets första fem veckor användes ett nätaggregat för att strömsätta komponenterna, men ersattes när bilens funktionalitet skulle testas då detta kräver en mobil strömkälla, som blev åtta stycken uppladdningsbara AA-batterier.



Figur 3. Blockschema över labbkort

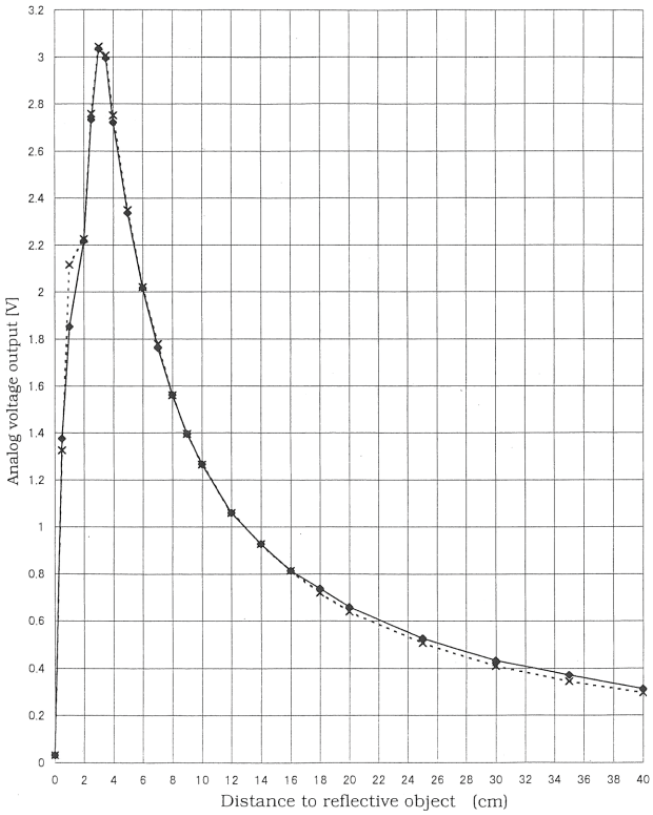
7. Funktionalitet

Funktionaliteten utgör den absolut största delen av arbetet, både med avseende på arbetets verkshöjd men även tidsåtgången. Under framtagandet och implementerande av funktionerna nedan gjordes löpande tester för att säkerhetsställa att de fungerar som tänkt. Dessa tester redogörs under varje funktion tillsammans med de lyckade och mindre lyckade försöken.

7.1 Förhindrade av kollision

Tidigt i projektet fanns en önskan om att kunna förhindra kollision med omgivning med hjälp av någon typ av sensor. Utifrån tidigare resonemang valdes IR-sensor för detektion av omgivning. Sensorn som inhandlades har en räckvidd mellan 4 till 30 cm som beroende på avstånd ger en analog utsignal mellan ungefär 0.3 till 3 V som grafen nedan illustrerar som återfanns i databladet (1).

Funktionen som sedan togs fram har tre olika lägen som presenteras till föraren i applikationen: ingen säkerhet, låg säkerhetsnivå och hög säkerhetsnivå. Ingen säkerhet betyder, precis som det låter, att bilen ignorerar avläsningarna från IR-sensorn och kan därmed krocka. Låg säkerhetsnivå innebär att bilen inte kan närma sig mer än 10 cm från en vägg, ställd vinkelrät mot sensorn, och hög säkerhetsnivå innebär att avståndet mellan vägg och sensor inte kan understiga 25 cm. Efter tester då bilen fick köra mot olika hinder för att sedan mäta avståndet där bilen stannade, insågs det att diagramet (figur 4) inte stämmer överrens med verkligheten. Detta på grund av att bilen har en hastighet som är klockanpassad samt att det även finns en bromsträcka som är beroende av hastigheten. Vid maximal hastighet mättes det fram att bromssträckan var upp emot 3 cm och antagandet gjordes att bromsträckan är lineär. Detta togs i hänsyn för att optimera denna funktion håller kraven för kravspecifikationen.



Figur 4. Diagram över spänning & distans till objekt

För att implementera anti-krock funktionen i bilen så krävs det att man behandlar signalen som kommer från IR-sensorn. Denna signal är en analog spänning som kan med hjälp av

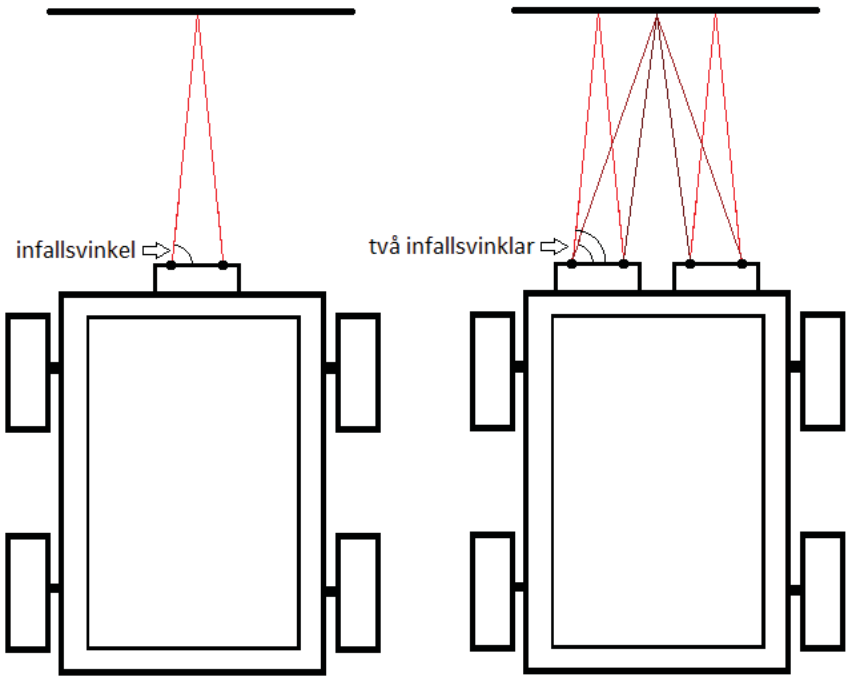
mikrochipets inbyggda ADC tolkas som en digital signal för att kunna skapa önskade intervaller för de olika säkerhetsnivåerna. ADC ger ett 12-bitars värde som då kan variera mellan 0 och 4096, där ett högt värde innebär att sensorn är nära ett objekt och ett lågt att avståndet till objektet är större. Funktionen fungerar sedan så att om ett digitalt värde överstiger en angiven gräns, det vill säga bilen åker tillräckligt nära ett objekt, så tvingas motorns styrsignaler ner till 0 och det går inte längre att köra framåt.

Problematiken med denna lösning är att IR-sensorn har en ytterst begränsad synvinkel, som illustrerat i figuren nedan, vilket innebär att ifall bilen kommer mot ett objekt från en snäv vinkel kommer sensorn ge utslag på felaktigt avstånd, eller i vissa fall inget alls. Med hjälp av användandet av tangens kan man enkelt ta fram vilken ingångsvinkel som är den maximala för de två olika säkerhetsnivåerna för att bilen ej ska krocka i en vägg.

$$Max\ ingångsvinkel\ = \tan^{-1}(\frac{\frac{bilens\ bredd}{2}}{säkerhetsnivå})$$

Som ett exempel så används låg säkerhetsnivå, 10 cm, med bilens halva bredd på 10 cm, vilket ger en max ingångsvinkel på 45° om en krock ska undvikas.

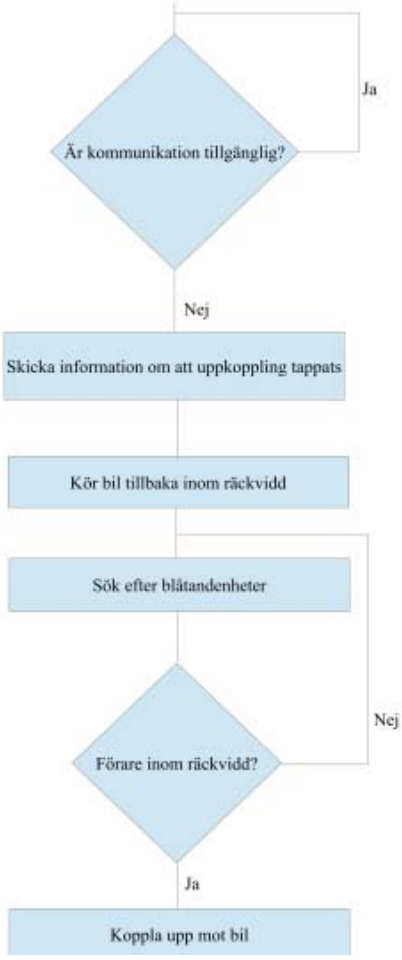
Under utvecklandet av denna funktion har förstås tankar gått hur man ska kunna istället för medvetet hålla sig från en viss vinkel vid kollision, implementera en lösning där bilen aldrig krockar med vägg. De två alternativen som kommit upp är att man antingen installerar servomotor där sensorn placeras och sveper fram och tillbaka mellan två ändlägen, och kan då identifiera objekt som befinner sig snett till höger eller vänster om bilen. En annan påtänkt lösning vore att använda två sensorer som fästes vid höger- respektive vänster-kant på bilens front för att kunna detektera objekt vid bilens sidor. Problematiken med detta är dock att de två sensorerna kan störa varandra. Detta på grund av att båda avger ljus från en punkt och fångas upp av en annan, och kan då beräkna avståndet till ett objekt beroende på ljusets infallsvinkel, men fångas ljus upp av en den andra sensorn kommer infallsvinkeln bli mycket större och objektet kommer att tolkas som mycket närmare än vad det egentligen är (4).



Figur 5. Skillnaden mellan en och två IR-sensorer

7.2 Autonomt hålla sig kvar inom räckvidd för blåtand

Funktionen som sågs som projektets viktigaste var att om bilen åkte utanför räckvidd för föraren, skulle en förprogrammerad algoritm ta kontroll över bilen för att ta den tillbaka till den plats där trådlös kommunikation tidigare existerat. Varför denna ansågs som den viktigaste var för att förmedla en känsla av att människa, tillsammans med tekniken, kan tillsammans hjälpas åt för att underlätta och förbättra till exempel redskap eller fordon. Att utveckla denna funktion visade sig bli mer komplicerat än vad först väntat, då den befintliga utrustningen som arbetades med saknade vissa väsentliga funktioner, så de olika tillvägagångsätten som prövades för att slutligen nå en tillfredställande funktion redovisas nedan.



Figur 6. Flödeschema över hålla sig inom räckvidd

7.2.1 Vid tappad kommunikation

Beroende på vilket landskap och miljö bilen navigerar i, har blåtandsmodulen lite olika räddvidd. Ett ungefärligt värde på 10 meter är att förväntas, men inte alltid. Den första varianten på denna funktion byggdes därför inte på hur långt bilen färdats, då räckvidden som sagt kan variera, utan väntade på att blåtandskommunikationen var bruten för att sedan låta en förprogrammerad algoritm ta hand om att navigera sig tillbaka mot föraren. Först och främst måste då en tappad kontakt identifieras.

Blåtandsmodulen som används har ett “Command Mode” som tillåter användaren att komma åt blåtandskonfigurationen. Manualen (2) berättar hur åtkomst till denna konfiguration görs möjlig genom att skicka strängen “$$$” genom en blåtandsterminal från en uppkopplad enhet, vilket måste göras inom 60 sekunder från att modulen slagits igång. Som ett bekräftande får man svaret “CMD” och sedan är modulen mottaglig för konfigurering av till exempel baud rate, namn, synlighet, pin kod, med mera. Det som var av ytterst relevans för denna funktion var att modulen skulle skicka ut en sträng på USART, till mikrokontrollern, omedelbart efter en frånkoppling upptäckts. Genom denna information skulle man då aktivera algoritmen som tar bilen tillbaka inom räckvidd.

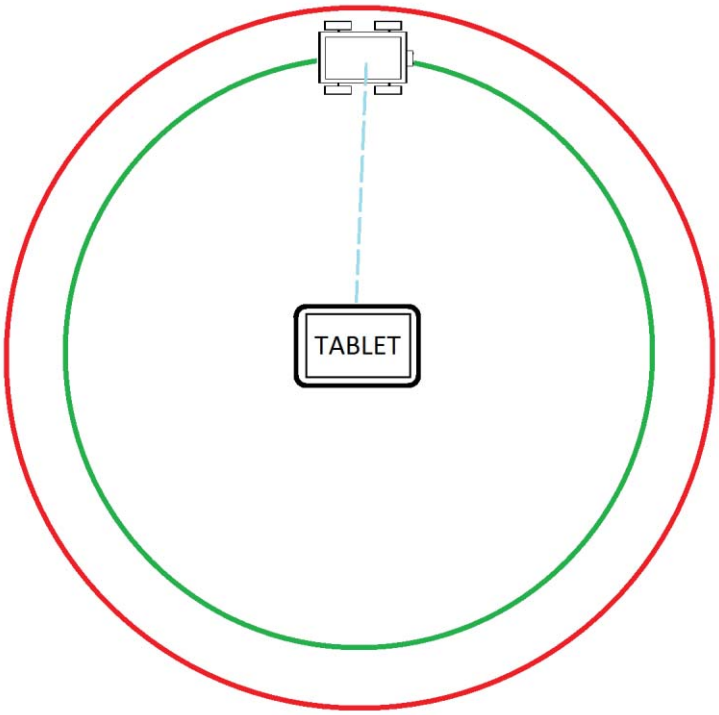
Problematiken med denna lösning upptäcktes vid testkörningen. Precis som flödesschemat beskriver så undersöks det om kommunikation finns tillgänglig, och då den inte är det ska information skickas om att ett avbrott i kommunikationen skett. Den informationen önskar vara omedelbar, alltså ske precis när föraren inte längre kan skicka styrsignaler till bilen. Vid olika tester implementerades ljud för att vet när bilen har tappad kommunikation för att lättare

indikera att kommunikationen är bruten. Dock bildades det en fördröjning på ungefär 12 sekunder mellan att det inte gick att styra bilen tills ljudet indikerade att kommunikationen är bruten. Slutsatserna som drog av dessa tester var att Androids blåtandsklass har en inbyggd fördröjning, då enheten försöker återupprätta uppkoppling mot senaste uppkopplad enhet. Då bilen kört ur räckvidd så är dessa 12 sekunder bara dötid, då uppkoppling inte kommer kunna etableras på nytt, och föraren kan inte heller skicka några signaler. När denna tid förflutit så skickar blåtandsmodulen ut information på USART att kommunikation tappats, och algoritmen där bilen letar sig tillbaka till föraren inleds.

Ett annat problem som visade sig vid testkörningen var att även sökningen har en fast tid på cirka 10 sekunder, vilket är tiden det tar för att fullända sökningen. Detta innebär att även fast bilen hittats under sökningens första sekunder måste ändå hela sökningen slutföras, vilket skapar en ytterligare en fördröjning för uppkopplandet mot föraren. Dessa två fördröjningar skapande en flaskhals för att funktionen skulle fungera som önskat, och därav förkastades idén och alternativa lösningar började studeras .

7.2.2 Avläsning av RSSI

RSSI betyder Received Signal Strenght Indication och är mått på styrkan hos en mottagen signal, vilket mäts i dBm. RSSI använder sig av GRPR, Golden Receiver Power Range, vilket anses som det optimala avståndet mellan två kommunicerande enheter, och RSSI mäter hur stor avvikelsen är från GRPR. Detta kan då ge användaren en fingervisning om avståndet mellan två enheter, eller åtminstonde ett jämförande värde för att avgöra om enheterna närmat sig varandra eller inte. Genom användandet av detta värde kan man skapa en ungefärlig radie runt föraren där bilen ska hålla sig inom, en sorts maxgräns för hur långt bilen kan färdas från föraren utan att tappa kommunikation, för att inte upprepa tidigare problem. Tanken var att med en frekvens på några gånger per sekund läsa av detta värde för att se om gränsvärdet överstigits, och ifall så ska algoritmen som tar bilen tillbaka till räckvidden exekveras.



Figur 7. Räckvidden för blåtandskommunikationen

Vid försök till implementering av denna funktion fanns att det API, den färdiga uppsättningen av funktionsanrop, som användes inte hade stöd för avläsning av RSSI när en uppkoppling var etablerad, utan enbart när man sökte efter närliggande blåtandsenheter. Detta innebar att RSSI enbart skulle kunna läsas

genom att först koppla ner enheterna, göra en fullständig sökning av enheter och på så sätt erhålla RSSI, för att sedan koppla upp på nytt. Detta var dock ingen godtagbar lösning då, som tidigare upptäckts, varje sökning tar tio sekunder. I senare version av Android API finns stöd för avläsning även under uppkopplat tillstånd, dessvärre var uppgradering av läsplattans operativsystem var inte möjlig då läsplattan i fråga, Galaxy Tab 2, var för gammal.

Då alternativet att läsa av RSSI genom Android inte visade såg hållbart så började möjligheter undersökas att göra det omvända, att låta blåtandsmodulen i bilen läsa av RSSI mot läsplattan. Detta sågs som en möjlighet då blåtandsmodulens medföljande datablad innehåller detaljer hur man kan få åtkomst till en så kallad "Link Quality" som ska returnera ett hexadecimalt värde mellan 0 och ff. För att få åtkomst till "Link Quality" värden behövdes det att blåtandsmodulen var i "Command mode" under användning. På grund av att läsa RSSI genom blåtandsmodulen samt även att skicka data mellan Android-enheten och sortera data i mikrokontrollen behövdes det att alla dess klockor på vadera processor skulle synka, vilket gjorde att det blev många krockar av data och omöjlig att använda sig utav.

7.2.3 Mätning av insignalsfrekvens

Efter många misslyckade och mindre lyckade försök med att få denna funktionalitet på plats så hittades en lösning som visade sig fungera precis som önskat. Vetskapen om hur ofta information skickas mellan de två kommunicerande enheterna tillsammans med avläsning på hur många av dessa signaler som nått fram blev nyckeln.

Den valda frekvensen på sändning och mottagning av data är tio gånger per sekund och mikrokontrollern läser av två gånger per sekund för att se om någon data har mottagits. Om ingen data har kunnat tas emot under längre än 1.5 sekunder, så anses bilen vara utanför räckvidd för att kunna köra bilen. För att bilen någorlunda skall kunna hitta tillbaka till räckvidd spelas det 100 gamla styrsignalerna upp i motssats håll, vilket körs under cirkus fem sekunder. Vid denna idé av lösningen är alltid bilen och Android-enheten uppkopplade mot varandra.

7.3 Autonom körning utefter handritat mönster

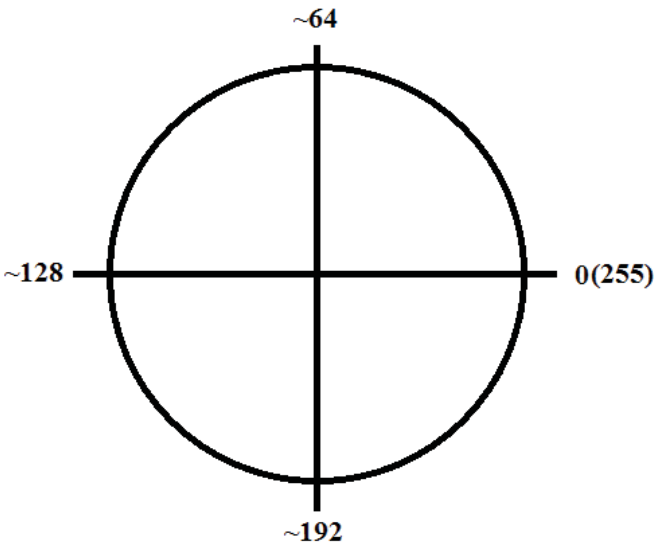
Vid brist av att bilen saknar en del autonomt beteende infördes en funktion när bilen ska härma ett handritat mönster från Android-enheten. Till exempel om en cirkel ritats på läsplattan skulle bilen köra en cirkel i verkligheten. Funktionen fungerar så att vid ett givet intervall så samplas punkter från det uppritade mönstret och sparar dessa koordinater i en datasträng samt beräknar längd och vinkeländringen mellan punkterna. Programmet tolkar vinklar utifrån enhetscirkeln, där ett sträck rakt uppåt blir 90° och rakt höger blir 0°. Längden mellan de olika samplade punkterna beräknas med hjälp av de sparade koordinaterna och avståndsformeln. När användararen har ritat mönstret och klickar på spara, så skickas alla vinklar och längder till bilen för uppspelning av figuren.

Då bilens inte hade någon uppfattning om vilket håll den stod riktad mot så gjordes det att ett sträck rakt uppåt tolkades som rakt fram. För att ta reda på hur mycket bilen skulle svänga vid

olika inmatning gjordes tester på hur fort bilen kunde rotera runt sin egen axel vid en viss motorhastighet. Ett tidtagarur och ett tiotal försök gav resultatet 3.40 sekunder för en full rotation runt egen axel. Detta innebar att om bilen skulle svänga till exempel 50 grader, behövde motorerna köras med den givna hastigheten i 0.47 sekunder, vilket kunde mätas genom en timer.

Problematiken med denna typ av lösning av funktionen var först och främst att bilen korrigerade varje liten gradskillnad, vilket vid tester visade sig inte vara bra spegling av figuren. Om en användares intention är att dra ett rakt sträck är det sällan som ett spikrakt sträck lyckas dras, utan flera gradskillnader kommer uppstå vid varje sampel. Därför infördes ett så kallat "acceptans-intervall" som gör det möjligt för bilen att tolka ett sträck som användaren försökt dra rakt men inte lyckats, som ett rakt sträck. Detta genom att ignorera mindre gradändringar, som vid tester visade sig vara sju grader åt vardera håll, så ett intervall på totalt 14 grader.

Vidare upptäcktes ett annat stort problem vid testkörningarna, drogs ett sträck rakt åt höger så kan gradtalet pendla mellan cirka 10 grader och 350, vilket gör att bilen snurrar runt nästan ett helt varv istället för att göra en liten korrigering åt ett håll. Även det tidigare omnämnda intervallet blir felaktigt just i denna övergång då skillnaderna alltid kommer bli större än 14 grader vid enhetcirkelns början. Försök att först identifiera "varning-zon" påbörjades för att vid denna zon kolla av mer värden för att få en uppfattning om användarens egentliga intention. Detta visade sig vara väldigt svårt att implementera och buggar smittade av sig på andra delar i ritfunktionen vid försök till implementering. Den funktion som slutligen blev tillfredställande skapades genom att omvandla grader, som sparas som en integer, till en character för att skapa denna cirkel som vid intervallets slut börjar om från noll, precis som enhetscirkeln. Detta visade sig fungera utmärkt då en beräkning av acceptans-intervallet från till exempel 2 grader nu fungerade både uppåt och neråt. Figur nummer 8 illustrerar hur värdena blir vid användandet av datatypen character.



Figur 8. Enhetescirkel för datatypen character

7.4 Styrning av bilen

En bra kontroll av bilen är fundamentalt för att kunna leverera en bra körupplevelse för föraren.

7.4.1 Styrning av bilen med styrkors

Det första som implementerades i applikationen efter en fungerande blåtandskommunikation upprättats mellan bil och läsplatta var en enklare styrning med hjälp av fyra pilar. Styrkorset är uppbyggt med fyra knappar som är riktade framåt, bakåt, höger och vänster och vid ett tryck på någon av dessa knappar skulle bilen köra åt respektive håll. Eftersom det fanns bara fyra olika håll bilen kunde förflytta sig på kunde man ge varje styrsignal ett unikt värde och på så sätt blev det enklare att sortera och packa upp dem packet som skickades mellan blåtand kommunikationen.

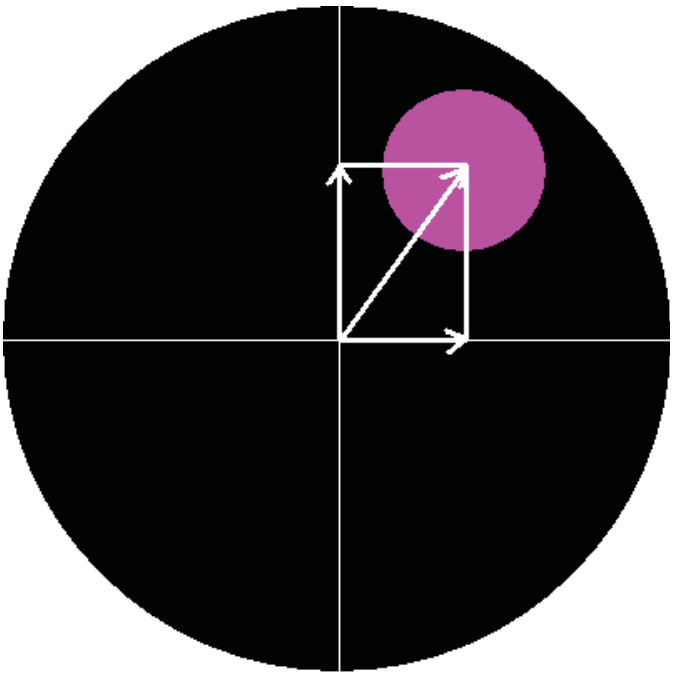
Figur 9. Bild på applikationens styrkors

För att inte slita sönder motorerna på bilen implementerades en upprampning av styrsignalen.

På grund av att styrningen hade bara fyra olika lägen att köra efter blev styrningen väldig stel och tråkig och idéer som att programmera en virtuell joystick skulle ge en bättre upplevelse till styrning.

7.4.2 Styrning av bilen med virtuell joystick

För att få en mer noggrann precision av styrning implementerades en virtuell joystick istället för ett styrkors och på sätt kunde bilen få en radie vid svängning och inte bara roteras i sidled på plats. Genom att bygga ett eget koordinatsystem inom ett begränsad area kan man med hjälp av olika algoritmer styra bilen på ett smart sätt.



Figur 10. Joysticken för Android applikationen

Joystickens rörelse är begränsad inom en cirkel som delats upp i fyra stycken kvadranter, precis som enhetscirkeln, och beroende på vart joysticken befinner sig används olika algoritmer för framtagande av styrsignal. Hastigheten på bilen är direkt beroende av triangelns hypotenusa, vars längd beräknas med hjälp av det etablerade koordinatsystemet och Pythagoras sats. Hur stor hastighet de olika hjulen ska ha, det vill säga hur

mycket bilen ska svänga, beror enkom på vilken kvadrant som joysticken befinner sig i. Är joysticken till exempel i första kvadranten så kommer det vänstra hjulens hastighet vara direkt beroende av hypotenusan, och det högra hjulens hastighet räkans fram genom att subtrahera hypotenusan med basen samt vice versa för alla andra kvadranter. Denna variant av styrning ger en mycket mer exakt navigering av bilen.

8. Programvara

För att skapa en bil som kan kontrolleras av en förare behövs både en kontroll, som i detta fall är en programmerad applikation som körs från en läsplatta, tillsammans med en bil vars beteende är programmerat i mikrokontrollern.

8.1 Bilens beteende

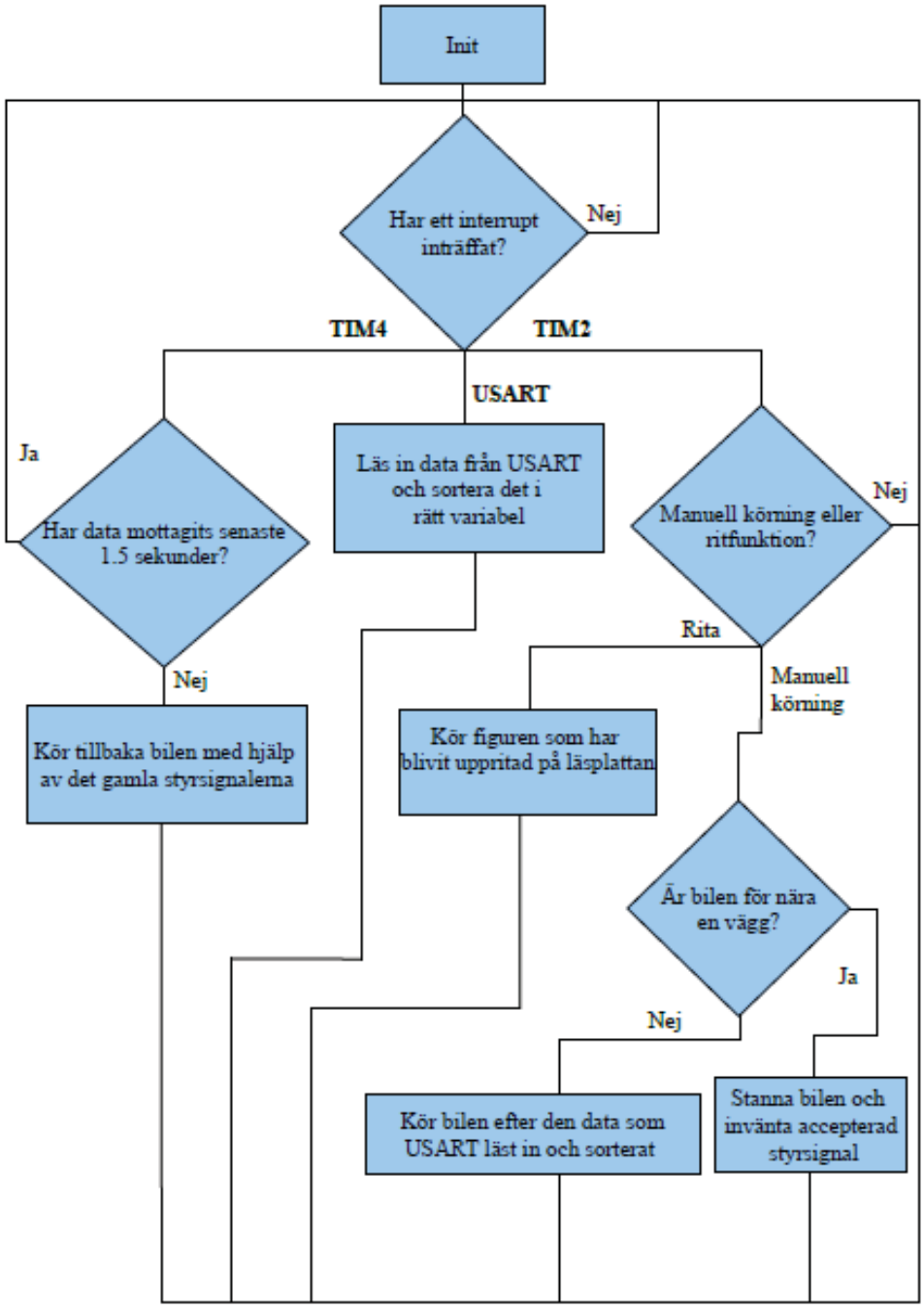
Hela bilens beteende är beroende av händelser och timer. En händelse fungerar så att när något specifikt inträffar, så utförs en specifik funktion, och timer fungerar som en klocka som man kan sätta till olika intervall då funktioner ska exekveras. Dessa två måste initieras för att ges olika prioritet, om det skulle vara så att två händelser råkar inträffa samtidigt. När den fullständiga initieringen är klar så ligger programmet i main-funktionen i väntan på att en händelse eller timer ska inträffa.

USART-händelsen inträffas så fort någon typ av data finns tillgänglig på mikrokontrollerns ben. Eftersom olika typer av värden kommer in på benet, som till exempel antal sampels eller säkerhetsavstånd, togs en meddelandehantering fram som först identifierar vilken typ av data som mottagits för att sedan placera och sortera data för vidare hantering. Tal mellan 121 och 127 har definierats som olika typer av data, som tabellen nedan redogör. Efter identifiering har tagits emot kommer sedan ett värde mellan 0 till 100 tas in som berättar om storleken hos data.

Unika tal	Typ av data
121	Joystickens placering i X-led
122	Joystickens placering i Y-led
123	Avstånd från joystickens mitt
124	Säkerhetsavstånd
125	Antal sampels i ritfunktion
126	Graderskillnad mellan två aktuella sampels i ritfunktion
127	Längd mellan två aktuella sampels i ritfunktion

Figur 11. Unika tal och typ av identifiering

Om till exempel talet 124 tas emot på USART, så identifierar meddelandehanteringen det som säkerhetsavstånd och vet nu att storleken på nästkommande data kommer att ligga mellan 0 till 2, där 0 innebär ingen säkerhetsnivå, 1 innebär låg och 2 innebär hög. Så samma sätt fungerar de andra datatyperna, men nästkommande värde kan variera mellan större spann, som till exempel avståndet från joystickens mitt vilket ligger mellan ett intervall på 0 till 100.



Figur 12. Flödesschema över bilens program

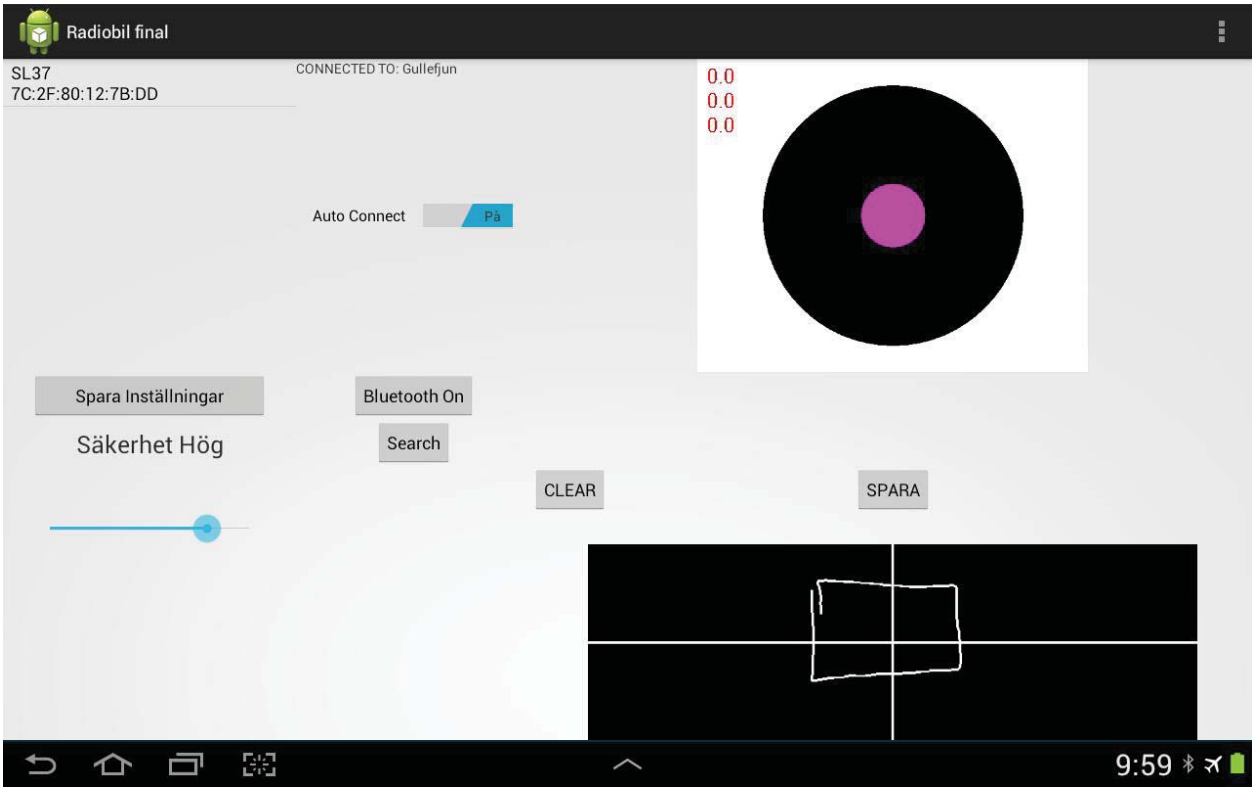
TIM2, den timer som ser efter bilens styrning, inträffar med en frekvens på 50 gånger per sekund. Först ut i denna programdel kollar programmet av om styrningen av bilen är beroende av antingen ett handritat mönster eller manuell körning med den virtuella joysticken. Vid ett handritat mönster i Android-applikationen inväntas först en färdig figur i ritområdet, för att

sedan invänta ett knapptryck som överlåter bilens kontroll helt till mjukvaran och bilen kör enligt det mönster som användaren ritat i applikationen. Vid användning av manuell styrning kollar programmet av om någon säkerhetsnivå är satt, och sedan om säkerhetsnivån är redan överskriden eller ej. Om bilen indikerar att bilen står vid en vägg, eller närmat sig en, så stannar motorerna och inväntar en accepterad styrsignal, vilket i detta fall är att backa bilen.

TIM4 anropas två gånger per sekund för att känna av om någon data har mottagits från USART. Om ingen data tagits emot inom detta intervall startar en räknare. Om ingen mer data mottagits under totalt 1.5 sekunder så tolkar programmet det som att kommunikationen har tappats och bilens kontroll tas över av mjukvaran och kör autonomt tillbaka bilen inom räckvidd.

8.2 Mjukvara i Android-enheten

Applikationen som har utvecklats till Android-enheten har inte något fokus på design utan all tanke är kring funktionalitet. Grunden i programmet är att data skickas ut från applikationen med en frekvens på 10 gånger per sekund genom användandet av en timer. Bortsett från timer så inväntas händelser från användaren, genom knapptryck eller pekrörelser, och beroende på vad som utförs kan olika algoritmer exekveras eller data sparas och skickas för användning. Vid till exempel tryck på "Search"-knappen så kollas först om blåtanden i Android-enheten är påslagen, för att sedan genomföra en sökning av alla närliggande enheter. Vid funnen blåtandsenhet sparas dess MAC-adress i en array för eventuell användning senare (7).



Figur 13. Överblick på Android applikationen

9. Vidareutveckling

Då bilens befintliga skick är en väldigt bra plattform för vidareutveckling, presenteras nedan idéer som diskuterades under arbetets gång och som hade vart intressanta att implementera om mer tid skulle läggas på utvecklandet.

9.1 GPS och kompass

Genom implementation av en GPS-modul och kompass i bilen skulle man kunna få information om bilens färdriktning samt koordinater om vart bilen befinner sig. Vidare skulle man kunna använda denna information för att placera ut vägpunkter, det vill säga koordinater, man önskar att bilen skulle ta sig till. Då GPS-modulerna enbart kan garantera en noggrannhet på några meter så är denna funktion någonting som i sådana fall enbart skulle kunna användas i miljöer där ytorna är mer öppna, som utomhus eller i större lokaler. För inom användande skulle man behöva använda sig av ett flertal stationära sändare för positionering och navigering.

Önskvärt till denna funktion vore en integrering med till exempel Google Maps, där användaren kan själv placera ut vägpunkter genom att klicka på en karta och låta bilen orientera sig efter dessa. En förutsättning för att denna navigering skulle fungera är också en vidareutveckling av kollisions-förhindrandet, då bilen med stor sannorlikhet kommer att komma i kontakt med hinder på väg mot de utplacerade vägpunkterna. En funktion där bilen kan följa en vägg eller objekt parallellt, samtidigt som att bilen på ett smart sätt väljer vilket håll den väljer att gå om objektet, skulle göra färden mot vägpunkten så smidig och kort som möjligt.

9.2 Mappning av omgivning

En idé som fanns med i den grundplaneringen var att kunna scanna av bilens omgivning och spara den i arrayer för att på så sätt skapa en ungefärlig bild av vad som finns i bilens närhet. Detta skulle vara genomförbart genom tidigare nämnd idé - att placera IR-sensorn på en servomotor - och genom denna svepande rörelse som servomotorn tillåter kan man läsa av vad som befinner sig inom räckvidden för rörelsen.

Gällande implementering och programmering av denna funktion är det viktigt att man vet med vilken frekvens man läser in varje värde samt servomotorns position vid avläsningen, så man kan bilda sig en uppfattning om vart objektet befinner sig. Även viktigt är att spara in arrayerna i rätt ordning, så att man vid en svepning från vänster till höger sparar in värdena från ett håll och det omvända när svepningen går från höger till vänster. Ett enkelt exempel är hur en låda framför bilen som kan ge följande utslag:

0 0 0 0 322 337 350 336 320 0 0 0

Där alla nummer skiljt från noll indikerar på att sensorn fångat upp någonting på ett visst avstånd, och nollorna är då sensorn inte fångat upp någonting alls. Med hjälp av dessa siffervärden, vetskapen om sensorns position vid avläsning samt lite matematik kan man då beräkna bredden på lådan, och grunden för en enklare identifiering är lagd. Vidare kan man till exempel identifiera en vägg eller åtminstone ett objekt som bilen står snedställd mot, vilket kunde ge ett utslag enligt nedan:

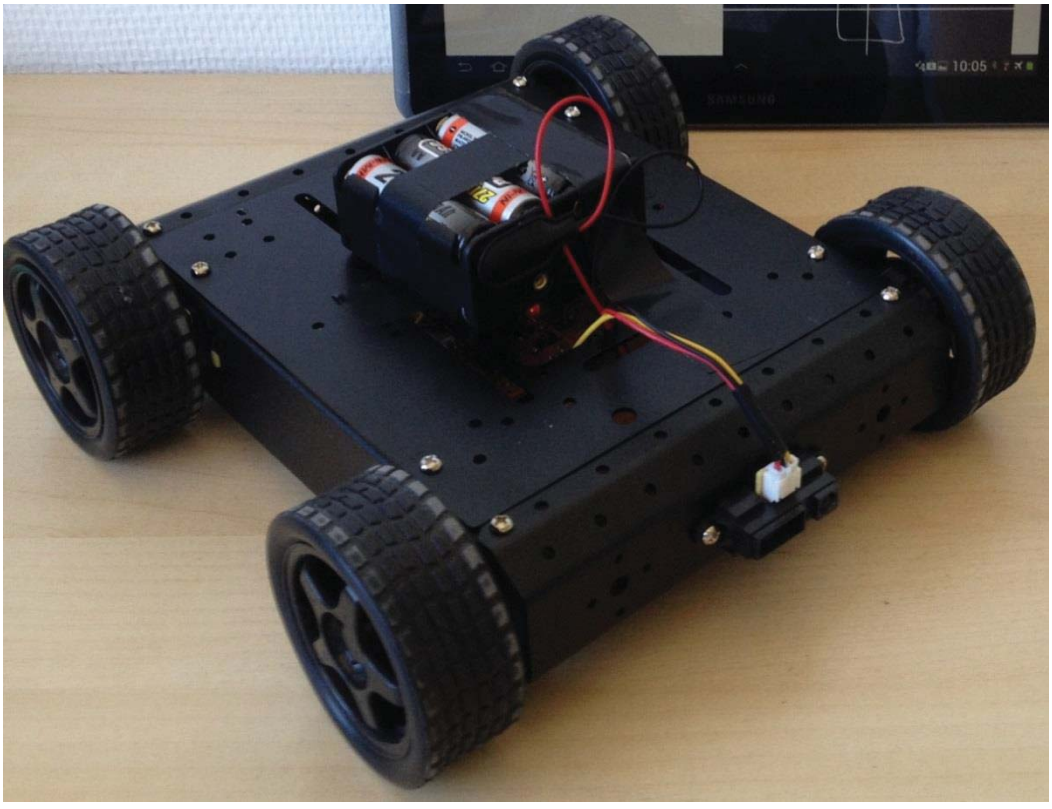
80 111 147 189 235 287 336 391

Om bilen är stillastående och ett ökande som ovan matas in indikerar på att ett objekt närmar sig sensorn när den sveper åt ett håll, vilket skulle då kunna vara en vägg. Genom att kunna ställa bilen parallellt mot denna vägg och låta den köra rakt fram tills väggen svänger kan man då skapa ett typ av koordinatsystem, där bilen sparar data där svängar gjorts och på så sätt mappa upp en enklare bild av bilens omgivning (4).

10. Resultat

Under de tio veckorna som arbetet sträckte sig över har största delen av tiden gått åt till att programmera och implementera funktionalitet, vilket även var det förväntade då det i planeringsrapporten dedikerats mest tid för det. Något som var svårt var i den inledande fasen när olika tidsfönster skulle sättas upp för olika delmoment. Då erfarenhet och kunskaper kring arbetet till en början var väldigt begränsade så höftades kraftigt och det gjorde det svårt att hålla sig inom de ramar som hade satts upp. I samband med planeringsrapporten skapades även en enkel kravspecifikation för att redogöra vilka funktioner som skulle finnas tillgängliga på den färdiga bilen, samt extra funktionalitet om tid skulle finnas över vid målgång. Det visade sig bra att planera för framtida funktionalitet då bilens grunder stod klara redan vecka sex. Den största avsaknad då var en känsla av autonomi, att bilen själv kunde köra utan någon förares interaktion. Därför valdes i samråd med handledare att lägga fokus på en funktion som kunde förmedla denna känsla, och därav kom ritfunktionen till där föraren fick rita ett mönster som bilen skulle köra efter.

Om arbetet skulle göras om idag, med den nyvunna kunskapen, hade det nog lagts upp på ett snarlikt vis. Den råder däremot ingen tvekan om att den största missen som gjorts genom arbetsgången är att det generellt sätt ägnades för lite tid åt förstudie och planering, utan att man istället hoppade in i någonting i hopp om att det skulle fungera och att det sedan visar sig att en veckas jobb bara är att slänga då lösningen inte visar sig vara gångbar. Det är då framförallt problemet med att läsa av RSSI som blev den största tidstjuven, vilket kunde ha undvikits med bättre planering.



Figur 14 - Bild på den färdiga bilen

11. Slutsats

Vid uppkomsten av idéen att tillverka en radiostyrd bil var syftet rätt oklart. Efter hand kom så småningom tankar på hur mycket som utvecklas idag, som är menat för människor, ofta kommer tillsammans med någon typ av artificiell intelligens som finns till för att hjälpa människan i användandet. Detta projekts syfte formulerades då med önskan om att lyckas illustrera detta med hjälp av en radiostyrd bil. Detta kändes både aktuellt då autonoma bilar är på frammarsch, men även för att företaget där arbetet skulle utföras har sitt huvudsakliga fokus på bilbranschen. Genom ett enklare anti-krock system illustreras hur människa och teknik kan sammarbeta för att nå en högre nivå av säkerhet och körupplevelse. Ritfunktionen som utvecklades blev ett enkelt sätt att beskriva hur man som användare kan ge instruktioner till en maskin för utförandet av en uppgift. Detta är en trend som antas växa mer och mer i samhället, inte bara för bekvämlighetens skulle men även för att kunna till exempel hjälpa till i sammanhang som anses farliga. Oundvikligt är att potentialen finns i de flesta områden, där människan kan gagnas av teknik som får stå för de delar där vi människor inte kan utmana dem.

Referenser

1. Sharp IR-sensor datablad
http://www.sharp.co.jp/products/device/doc/opto/gp2y0a41sk_e.pdf (Acc 2014-06-04)

2. Bluetooth Mate Silver (BlueSmirf) datablad
http://www.electrokit.com/productFile/download/3072 (Acc 2014-04-03)

3. STM32-H103 användarmanual
http://www.electrokit.com/productFile/download/3037 (Acc 2014-03-28)

4. Society of Robots, Sensors - Sharp IR range sensor
http://www.societyofrobots.com/sensors_sharpirrange.shtml (Acc 2014-05-05)

5. LM7805C spänningsregulator datablad
http://www.fairchildsemi.com/ds/LM/LM7805.pdf (Acc 2014-03-21)

6. Energy Consumption Analysis for Bluetooth, WiFi and Cellular Networks
http://nesl.ee.ucla.edu/fw/documents/reports/2007/poweranalysis.pdf (Acc 2014-03-25)

7. Android Development - Guide till programmera inom Android
http://developer.android.com/guide/index.html (Acc 2014-03-31 - 2014-06-05)

8. L6206 full H-brygga datablad
http://www.st.com/web/en/resource/technical/document/datasheet/CD00002346.pdf (Acc 2014-04-09)

9. WiFi övergripande information
http://en.wikipedia.org/wiki/Wi-Fi (Acc 2014-05-26)

10. Mikrokontroller övergripande information
http://en.wikipedia.org/wiki/Microcontroller (Acc 2014-05-25)

11. Blåtand övergripande information
http://en.wikipedia.org/wiki/Bluetooth (Acc 2014-05-26)

12. An Analysis of Bluetooth Technology, Master Thesis
http://btu.se/fou/cuppsats.nsf/all/bf95c4a61bb329d9c12575d600449bdb/$file/final%20thesis%20after%20removing%20comments.pdf (Acc 2014-03-24)

13. Radiostyrd bil övergripande information
http://en.wikipedia.org/wiki/Radio-controlled_car (Acc 2014-05-27)

14. Autonoma bilar övergripande information
http://en.wikipedia.org/wiki/Autonomous_car (Acc 2014-05-28)

15. DARPA Grand Challenge
http://en.wikipedia.org/wiki/DARPA_Grand_Challenge (Acc 2014-05-28)

16. Radiostyrda Bilars forumdel ”Projekt, tips & tricks”
http://rsb.se/arena/forumdisplay.php?57-Projekt-tips-amp-tricks (Acc 2014-03-17)

17. Bekkelien Master Thesis

http://tam.unige.ch/assets/documents/masters/bekkelien/Bekkelien_Master_Thesis.pdf (Acc 2014-04-14)

Bilagor

A1. Källkod

```
#include <__cross_studio_io.h>

#include "stm32f10x_conf.h"
// #define TIM1_IRQHandler 0x000000B4
/* Private define ------------------------------------------------------------*/
void MSGHandler();
#define TxBufferSize1   (countof(TxBuffer1) - 1)
#define TxBufferSize2   (countof(TxBuffer2) - 1)
#define RxBufferSize1   TxBufferSize2
#define RxBufferSize2   TxBufferSize1
#define VANSTER_FRAM 3
#define HOGER_FRAM 1
#define VANSTER_BAK 4
#define HOGER_BAK 2
#define X_STYR 121
#define Y_STYR 122
#define C_STYR 123
#define SAKERHET 124
#define BUFFER_SIZE 125
#define GRAD 126
#define LANGD 127
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
char temptemp=0;
char rup=0;
char gup=1;
char ritningpag=0;
int reading=0;
char readagain=0;
int sakerhetvarde=0;
int sakerhetavstand=9999;
int adomvn=0;
char raknare=0;
signed char oldStyrsignal[10];
int i=0,k=0,b=0,x=0,y=0;
signed char inlasning[]={0,0,0,0};
int OStyrsignal=0;
char BTStatus[12];
int ramp=0;
unsigned int ramp2 = 0;
int fram=0;
int framdis=1;
int bakdis=1;
unsigned char upgrader=0;
unsigned char upl=0;
```

```
int bak=0;
signed char test=0;
int stop=1;
int disconnected=0;
int connected = 0;
char BTRealStatus = 1;
signed char xory=0;
signed char xvalue=0;
signed char Oxvalue[100];
signed char yvalue=0;
signed char Oyvalue[100];
signed char hyper=0;
signed char Ohyper[100];
signed int grader[50];
char langder[50];
int Size=0;
char iBeep=1;
char enkorning2=0;



/* Private functions ---------------------------------------------------------*/
void init();
void delay_ms(u16 ms);
int main(void) // MAIN FUNKTION
{
init();
ADC();
while (1) // VÄNTAR PÅ INTERRUPT
{
}
}
void init()
{
RCC_72_COnfiguration();
RCC_Configuration();
GPIO_Configuration();
ADC_Configuration();
DMA_Configuration();
TIM_Configuration();
EXTI0_Config();
USART_Configuration();
NVIC_Configuration();




}
/*-------------------------------------------------------------------------------*/
void TIM2_IRQHandler(void) // RITFUNKTION ELLER MANUELL KÖRNING
```

```
{
 int tempx=0;
 int tempy=0;
 int tempc=0;
 signed char temp=0;
 signed char temp1=0;
 unsigned char internul=0;
 unsigned char internuh=0;
 unsigned char internel=0;
 unsigned char interneh=0;
 unsigned char tempgnu=0;
 unsigned char tempgne=0;
 unsigned char dgrader=0;



  if (sakerhetvarde == 0{ // OLIKA SÄKERHETS NIVÅER
    sakerhetavstand = 9999;
  }
  if (sakerhetvarde == 1){
    sakerhetavstand = (hyper*-1)+1289;
  }
  if (sakerhetvarde == 2){
    sakerhetavstand = (hyper*-1)+743;
  }
 adomvn=ADC(); // AD-OMVANDLING


 if (adomvn>sakerhetavstand) // STANNAR BILEN OM AD STOR
 {
   PWM(HOGER_FRAM,0);
   PWM(VANSTER_FRAM,0);
   if (hyper > 0 ){
     hyper=0;
   }
 }
 oldStyrsignal[raknare]= temp = yvalue;
 raknare++;
 if (raknare > 10 ){
   raknare=0;
 }
  for (char de=0;de<10;de++){
   if (temp < 0){
     temp1=oldStyrsignal[de];
     if(temp1 > 0){
        hyper=0;
     }
   }
    else if (temp > 0){
    temp1=oldStyrsignal[de];
```

```
    if(temp1 < 0){
        hyper=0;
     }
    }
  }
  if ( ritningpag == 1 && rup < Size){ // RITA FUNKTIONEN

    tempgnu = (grader[rup]*71)/100; // GÖR OM GRADER TILL CHAR
    tempgne = (grader[gup]*71)/100;
    internul=tempgnu-5; // INTERVALL +-5
    internuh=tempgnu+5;
    internel=tempgne-5;
    interneh=tempgne+5;
    dgrader=tempgnu-tempgne;

    if (dgrader > 127){  // ÄR SVÄNGEN STÖRRE ÄN 180 GRADER?
      PWM(HOGER_FRAM, 0);
      PWM(VANSTER_FRAM, 0);
      PWM(HOGER_BAK, 0);
      PWM(VANSTER_BAK, 0);
      for ( unsigned char dgrader1 = 255-dgrader ; dgrader1 > 0 ; dgrader1--){ // HÖGER
       PWM(HOGER_FRAM, 50);
       PWM(VANSTER_BAK, 50);
       delay_ms(16);
      }
    }
    if (dgrader < 127){ // ÄR SVÄNGEN MINDRE ÄN 180 GRADER?
      PWM(HOGER_FRAM, 0);
      PWM(VANSTER_FRAM, 0);
      PWM(HOGER_BAK, 0);
      PWM(VANSTER_BAK, 0);
      for ( unsigned char dgrader2 = dgrader ; dgrader2 > 0 ; dgrader2--){ // VÄNSTER
       PWM(HOGER_BAK, 50);
       PWM(VANSTER_FRAM, 50);
       delay_ms(16);
      }
    }
    if ( (internel > internul && internel < tempgnu)  || (interneh < internuh && interneh >
tempgnu) || langder[rup] > 0){ // ??
      PWM(HOGER_FRAM, 0);
      PWM(VANSTER_FRAM, 0);
      PWM(HOGER_BAK, 0);
      PWM(VANSTER_BAK, 0);
      for ( ; langder[rup] > 0 ; langder[rup]--){
       PWM(HOGER_FRAM, 50);
       PWM(VANSTER_FRAM, 50);
       delay_ms(17);
      }
    }
```

```
    rup++;
    gup=rup+1;
  }
  if ( ritningpag == 1 && gup == Size ){
    PWM(HOGER_FRAM, 0);
    PWM(VANSTER_FRAM, 0);
    PWM(HOGER_BAK, 0);
    PWM(VANSTER_BAK, 0);
    ritningpag = 0;
    rup = 0;
    gup = 1;
    upgrader=0;
    upl=0;

  }
  if (hyper > 0 && ritningpag == 0 ){ // MANUELL RITNING
   if( xvalue > 0 && yvalue < 0){ // första kvadranten
     PWM(HOGER_FRAM, hyper-xvalue);
     PWM(VANSTER_FRAM, hyper);
   }
   if( xvalue < 0 && yvalue < 0){  // andra kvadranten
     tempx=xvalue*-1;
     PWM(HOGER_FRAM, hyper);
     PWM(VANSTER_FRAM, hyper-tempx);
   }
  }
  if (hyper < 0  && ritningpag == 0  ){
   tempc=hyper*-1;
   if (xvalue < 0 && yvalue > 0){ //tredje kvadranten
     tempx=xvalue*-1;
     PWM(HOGER_BAK, tempc);
     PWM(VANSTER_BAK, tempc-tempx);
   }
   if (xvalue > 0 && yvalue > 0){ //fjärde kvadranten
     PWM(HOGER_BAK, tempc-xvalue);
     PWM(VANSTER_BAK, tempc);
   }
  }
  if (hyper == 0  && ritningpag == 0  ){ // STANNAR BILEN
    PWM(HOGER_FRAM, 0);
    PWM(VANSTER_FRAM, 0);
    PWM(HOGER_BAK, 0);
    PWM(VANSTER_BAK, 0);
  }
TIM_ClearITPendingBit(TIM2,TIM_IT_Update);
}
void USART1_IRQHandler(void)
{
if (USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
{
```

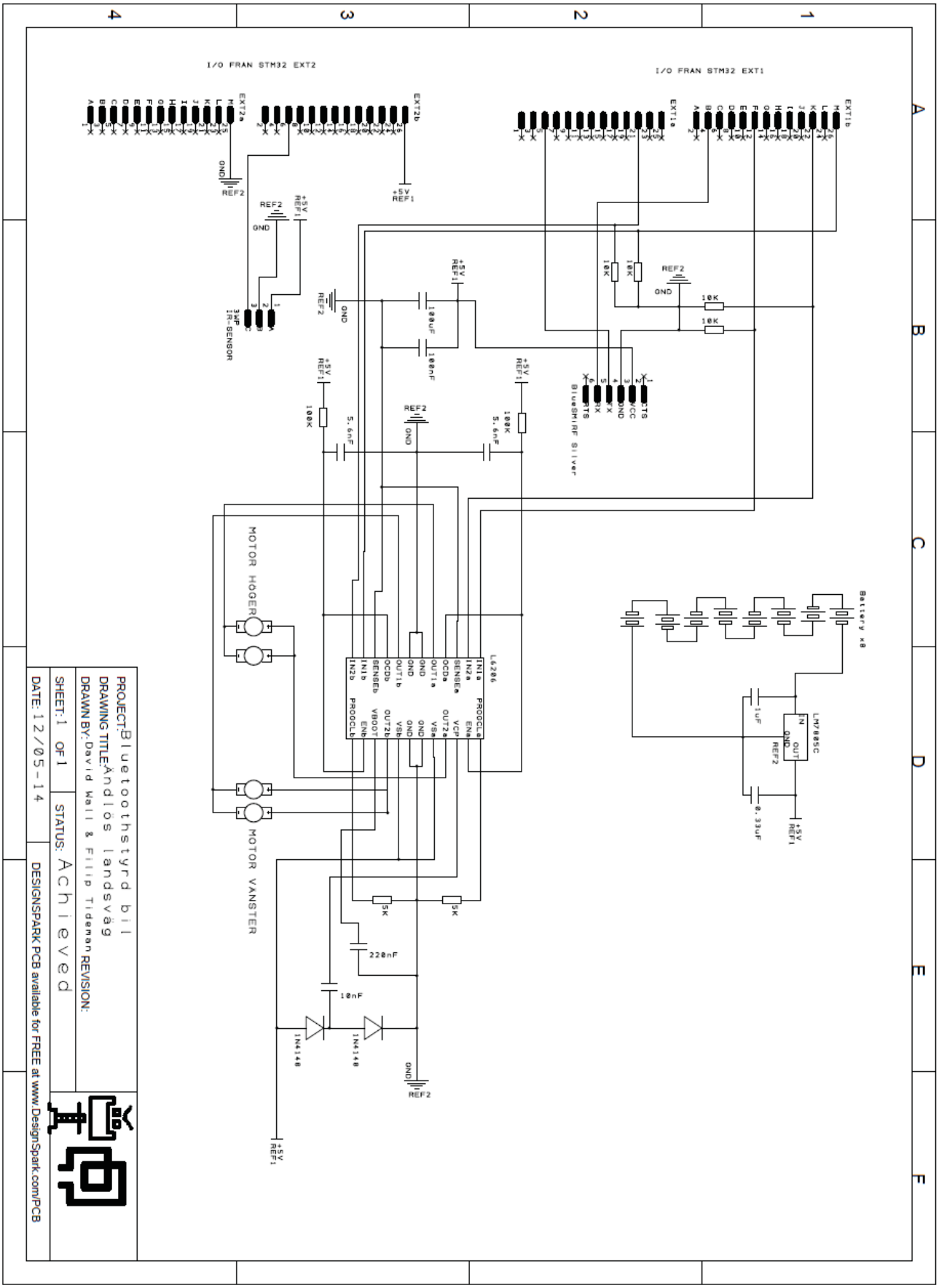
```
inlasning[i]=USART_ReceiveData(USART1);
reading=0;
enkorning2=1;

if (xory == X_STYR){ // SORTERAR OLIKA VÄRDEN I RÄTT VARIABEL
  xvalue=inlasning[i];
  Oxvalue[x]=xvalue;
  xory='\0';
  x++;
  connected=1;
  if (x>=100){
    x=0;
  }
}
if (xory == Y_STYR){
  yvalue=inlasning[i];
  Oyvalue[y]=yvalue;
  xory='\0';
  y++;
  if (y>=100){
    y=0;
  }
  connected=1;
}
if (xory == C_STYR){
  hyper=inlasning[i];
  Ohyper[b]=hyper;
  b++;
  xory='\0';
  connected=1;
  if (b>=100){
    b=0;
  }
}
if (xory == BUFFER_SIZE){
    Size = inlasning[i];
    xory='\0';

}
if (xory == GRAD){
   grader[upgrader]=inlasning[i];
   grader[upgrader]=grader[upgrader]*4;
   upgrader++;
   xory='\0';
   if ( upgrader >= Size ){
      ritningpag=1;
      upgrader=0;
   }
}
if (xory == LANGD){
```

```
      langder[upl++]=inlasning[i];
      xory='\0';
      if ( upl >= Size ){
         ritningpag=1;
         upl=0;
       }
     }
     if (xory == SAKERHET){
      sakerhetvarde=inlasning[i];
      connected=1;
      xory='\0';
     }

     if  (inlasning[i] == X_STYR || inlasning[i] == Y_STYR || inlasning[i] == C_STYR
         || inlasning[i] == SAKERHET || inlasning[i] == BUFFER_SIZE || inlasning[i] ==
GRAD  || inlasning[i] == LANGD   ) // KOLLAR AV DET UNIKA TALET
     {
      xory=inlasning[i];
      connected=1;
     }
     if (inlasning[i] == 'B' && BTRealStatus == 1)
     {
      k=0;
      BTRealStatus = 0;
      connected=1;
     }
    /* if (inlasning[i] >  4)
     {
      BTStatus[k] = inlasning[i];
      connected=1;
     }*/

     i++;
     k++;
     if(i>=4)
     {
      i=0;
     }
     if (BTStatus[2] == 'C' &&  k>9)
     {
        k=0;
        BTRealStatus = 1;
        connected=1;
     }
     if (k >12 && BTStatus[2] == 'D')
     {
      k=0;
      BTRealStatus = 1;
      connected=0;
     }
```

```
      if (k >12 ){
        k=0;
      }
}
}
void TIM4_IRQHandler(void)
{
  int rampav=0;
  char bb=0;
  char enkorning=1;
  signed char tempp=0;
  signed char temptemp=0,temptempx=0,temptemph=0;
  if(USART_GetITStatus(USART1, USART_IT_RXNE) == RESET){ // FÖR TILLBAKA
BILEN VID TAPPAD KOMMUNIKATION
    reading++;
    tempp=hyper;
    if(reading >= 3 && connected ==1 && enkorning2 == 1){
      temptemp=b;
      while(1){
        if (enkorning==1){
          PWM(HOGER_FRAM, 0);
          PWM(VANSTER_FRAM, 0);
          PWM(HOGER_BAK, 0);
          PWM(VANSTER_BAK, 0);
          enkorning=0;
          Delay_us(10000);
          hyper=xvalue=yvalue=0;
        }
        if (tempp > 0 ){
          if( Oxvalue[x] > 0 && Oyvalue[y] < 0){ // första kvadranten
            PWM(HOGER_BAK, Ohyper[b]);
            PWM(VANSTER_BAK, (Ohyper[b])-(Oxvalue[x]));
          }
          if( Oxvalue[x] < 0 && Oyvalue[y] < 0){  // andra kvadranten
            temptempx=Oxvalue[x]*-1;
            PWM(HOGER_BAK, (Ohyper[b])-(temptempx));
            PWM(VANSTER_BAK, (Ohyper[b]));
          }
          b--;
          y--;
          x--;
        }
        if (tempp < 0 ){
          temptemph=Ohyper[b]*-1;

          if (Oxvalue[x] < 0 && Oyvalue[y] > 0){ //tredje kvadranten
            temptempx=Oxvalue[x]*-1;
            PWM(HOGER_FRAM, temptemph-temptempx);
            PWM(VANSTER_FRAM, temptemph);
          }
```

```
        if (Oxvalue[x] > 0 && Oyvalue[y] > 0){ //fjärde kvadranten
          PWM(HOGER_FRAM, temptemph);
          PWM(VANSTER_FRAM, temptemph-Oxvalue[x]);
        }
        b--;
        x--;
        y--;
      }
      if (b<0){
        b=99;
      }
      if (y<0){
        y=99;
      }
      if (x<0){
        x=99;
      }
      delay_ms(100);
      if (temptemp == b){
        enkorning2=0;
        break;

      }
    }
  }
 }
 TIM_ClearITPendingBit(TIM4,TIM_IT_Update);
}
void delay_ms(u16 ms)
{
    volatile u16 i, j;
    for(; ms>0; ms--)
          for (i = 0; i < 500; i++)
                for (j = 0; j < 5; j++);

}
```

A2. Elschema


I/O FRAN STM32 EXT2
I/O FRAN STM32 EXT1
MOTOR HÖGER
MOTOR VÄNSTER
L6206
LM7805C
Battery x8
PROJECT: Bluetoothstyrd bil
DRAWING TITLE: Andlös landsväg
DRAWN BY: David Wall & Filip Tidenan
REVISION:
SHEET: 1 OF 1
STATUS: Achieved
DATE: 12/05-14
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB