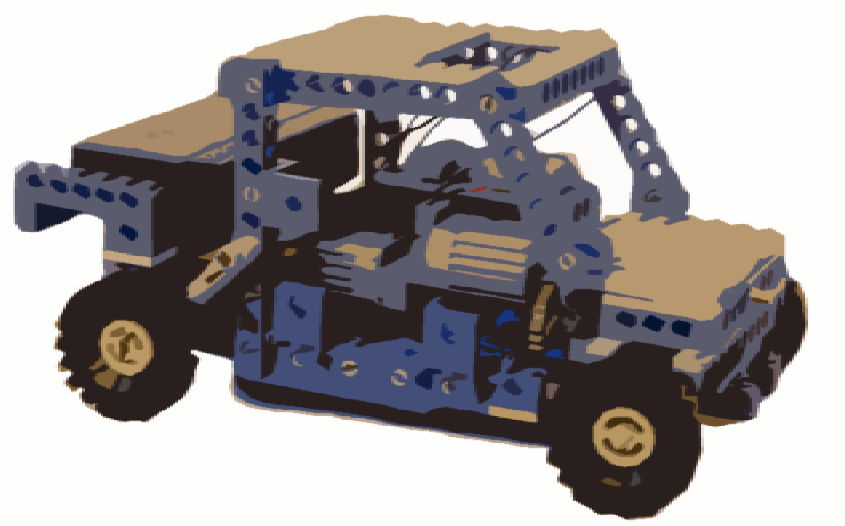
Rapport Radiostyrd LEGO – bil

Digitala Projekt
2002-05-13

Kristoffer Karlsson
Magnus Mathiasson
Tobias Åkesson



1. Innehållsförtäckning

2. Inledning

2.1. Bakgrund

Kommunikation i bilen sägs vara ett tillväxtområde för inbyggda system. Datorsystem byggs in i bilar för att kommunicera med verkstäder och informationscentraler. Fel i bilen rapporteras direkt till verkstaden. Är du vilsen visar fordonet närmaste vägen via kartor hämtade on-line från en central informationsbank. Telematik kallas denna integration mellan fordon och digital kommunikation.

Detta projekt behandlar telematik i dess enklaste form nämligen digital kommunikation mellan en radiostyrd LEGO - bil och PC.

2.2. Mål

Projektet innefattar konstruktion av mekanik, hårdvara och mjukvara. Resultatet är en fungerande prototyp. I detta fall har målet varit att bygga en digital motsvarighet till en radiostyrd bil. Bilen skall enkelt kunna styras från en vanlig PC. Även fordonet konstrueras med hänsyn till strömförbrukning, kopplingar mellan hårdvara och mekanik mm. De mest grundläggande kraven samlas i en kravspecifikation. I mån av tid kunde även en del extra funktioner läggas till.

2.3. Kravspecifikation

- Bilen skall styras med hjälp av tangentbord via en dator
- Bilen skall kunna acceleraras, bromsas och svänga
- Kommunikationen med bilen sker seriellt via radiosändare och mottagare.
- I mån av tid skall någon form av felkorrigering göras som kompensering för fel i överföringen.

3. Genomförande

Utöver det som nämns som rena krav måste flera andra designval göras för att kunna konstruera önskad prototyp. Detta kapitel innehåller detaljer kring konstruktionen samt en redogörelse för hur arbetet fortskred och de problem som uppstod (Kap. 3.6). Detta avsnitt är beskrivande, kretsschema och programkod återfinns i Appendix A: samt Referens [1], [2] och [3].

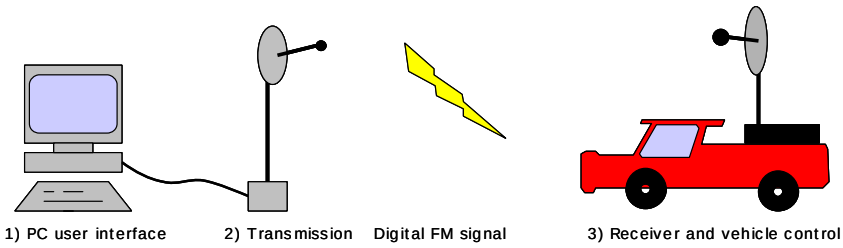


Figure 1: Ingående delsystem

Systemet består av de tre delarna PC, Sändare och Fordon. PC delen består endast av mjukvara och finns beskriven i kap. 3.1. Sändaren finns beskriven i 3.2. Den del som innehåller flest delar är LEGO – bilen. Denna innehåller både mjukvara och hårdvara. Mjukvaran beskrivs i 3.4 och hårdvaran i 3.3.

3.1. Styrgränssnitt PC

Styrningen av bilen sker via ett gränssnitt på en vanlig PC med Windows. Programmet är utvecklat i Java och kommunicerar med bilen via serieporten. Programmet har en enkel utformning där bilen styrs med piltangenterna.

- Pil Upp: Acceleration
- Pil Ned: Inbromsning eller Back
- Pil Vänster: Sväng hjul vänster
- PilHöger: Sväng hjul höger

Om inga tangenter rörs återgår bilen till stillastående och ställer hjulen rakt fram. Styrsignalen skickas tio gånger per sekund till bilen. Informationen packas i en byte med fyra bitars upplösning på styrning respektive drivning. För att kompensera för eventuella fel i överföringen skickas samma data flera gånger. Detta utnyttjas i mottagaren för att reducera sannolikheten för bitfel.

D1	D2	D3	D4	S1	S2	S3	S4
7	6	5	4	3	2	1	0

Paket som skickas till mottagare. Fyra bitar (tvåkomplement) för drivning och fyra för styrning.

Källkod för gränssnittet finns i referens [1].

3.2. Sändarkrets

Sändarkretsen är mycket enkel och består av två olika komponenter. Syftet med kretsen är att ta en insignal på COM-porten och skicka denna med hjälp av en radiosändare till en mottagare på bilen. För att kunna göra detta krävs att RS-232 12V-nivå först omvandlas TTL / CMOS 5V-nivå, detta görs med kretsen

MAX232. Efter omvandling kopplas signalen direkt till radiosändaren (nf401) som endast kräver en datasignal samt en ”Transmit Enable”-signal för att fungera

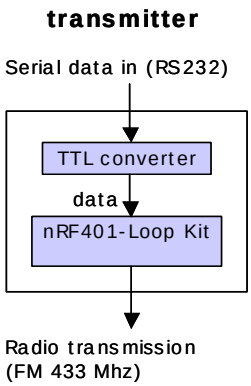


Figure 2: Blockschema Sändare

Då konstruktionsarbetet inleddes fanns en plan att kommunicera i båda riktningarna sändarkretsen har stöd för detta men möjligheten används ej i nuvarande konstruktion.

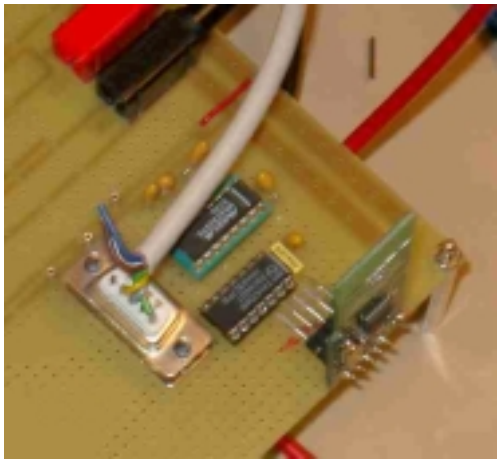


Figure 3: Sändarenhet

3.3. Hårdvara Bil

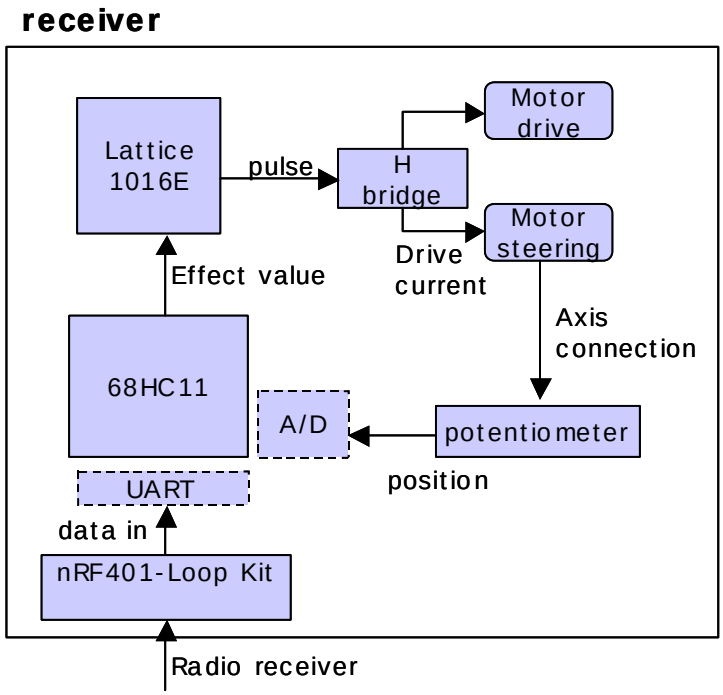


Figure 4: Blockschema mottagare (LEGO - bil)

3.3.1. 68HC11

Motorola 68Hc11 är en µProcessor vilken agerar central styrenhet på bilen. HC11:an har portar med vilka den kan läsa eller skriva till andra enheter. I vårt fall är en av dessa portar kopplad till den programmerbara logiken (Se 3.3.2). HC11:an har även en inbyggd A/D omvandlare vilken här kopplas till en potentiometer (se 3.3.5).

3.3.2. Lattice 1016E

För att ge rätt pulslängder till h-bryggan används programmerbar logik.
Lattice 10016E är en sådan krets och har ca 2000 programmerbara grindar. Kretsen programmeras med VHDL och klockas med samma klocka som HC11 d.v.s. 2 Mhz. Mjukvaran till denna krets beskrivs i 3.4.2

3.3.3. H-brygga

H-bryggan används för effektreglering till motorerna. Den styrsignal som skickas ut från logikkretsen är en fyrkantspuls med 5V. Denna signal är svag och kan inte själv driva någon motor. H-bryggan tar denna insignal och matas dessutom från en kraftigare strömkälla. I detta fall ett batteri på 6V. H-bryggan ger matningsspänning till motorn när styrsignalen är hög. Vid låg signal flyter ingen ström genom motorn. På detta sätt kan effekten regleras.

3.3.4. Transciever nRF401-Loop Kit

Detta är samma krets som sitter på sändarsidan (Se 3.2). Kretsen är på denna sida inställd i mottagarläge. Den signal som avgör iom kretsen är i sändar eller mottagarläge har kopplats till HC11:an så att det finns möjlighet att ställa om kretsen i sändarläge. Detta används dock ej i nuvarande konstruktion.

3.3.5. Potentiometer

Utan återkoppling från styraxeln hade mjukvaran behövt ge exakta pulser till styrmotorn. Hade effekten blivit det minsta fel skulle axelns position bli felaktig. Genom att avläsa positionen på axeln kan återkoppling ske och axelns position kan regleras med en enkel regulator.

Potentiometern kopplas till axeln för att ge önskad återkoppling. Genom att avläsa spänningsfallet över potentiometern kan en ”positionsangivelse” mätas. Denna position skickas via A/D omvandlaren till µProcessorn. Potentiometerns vred fästes på en LEGO – axel som kopplades till styraxeln via kugghjul. Fixering

skedde genom att bygga in potentiometern i en liten LEGO-låda bredvid styrmotorn.

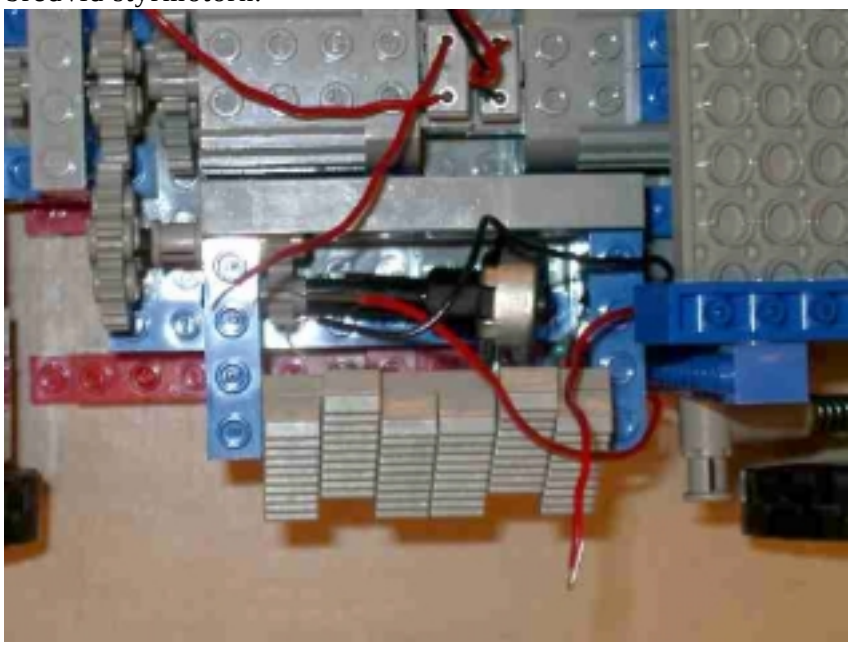


Figure 5: Bild på potentiometer

Hur har vi byggt in denna o varför? För att ge feedback om styraxelns position

3.4. Mjukvara Bil

3.4.1. Mjukvara HC11

Bilens µProcessor kör mjukvara för styrning av motorer och mottagning av data. Motorerna styrdes till en början direkt från HC11:an via en H-brygga. Denna konstruktion ersattes senare med ett mellansteg i form av programmerbar logik (se 3.4.2). Detta innebär att programvaran i HC11:an endast tar emot data från sändaren och behandlar denna för att ge styrsignaler till logik – kretsen.
Huvuduppgifterna kan kort beskrivas som:

- Ta emot data från transciever
- Felkontroll av data, förkasta om felaktig. Felkontroll sker genom att X antal bytes i följd måste vara identiska.
- Ta emot data från A / D omvandlare om styraxelns position.
- Extrahera data ur korrekta paket.
- Drivning till hjulen baseras direkt på styrsignal.
- Utslaget för styrningen beräknas med en enkel P – regulator.

- Önskad motoreffekt kodas och läggs på en 8–bitars port kopplad till Latticekretsen.

Programmet i sin helhet återfinns i referens [2].
Programmet blev mycket kompakt då nödvändiga periferienheter finns inbyggda på processor. Varken extern UART eller A/D behövde användas. HC11:ans kompakta programkod gör att styrprogrammet bara tar upp 400 byte programminne och ca 40 byte RAM.

3.4.2. Mjukvara Lattice 1016E

Latticekretsen är programmerad att skapa pulser till h-bryggan. Genom att ge full spänning under en viss tid av en pulslängd kan effekten till motorn kontrolleras.
Indata till kretsen är en byte kodad enligt:

FF/RW	D1	D2	D3	LFT/RHT	S1	S2	S3

Då h-bryggan ställs på olika sätt beroende på om motorn skall driva framåt eller bakåt kodas detta med en bit. D1 - D3 är tre-bitas positivt tal som anger effekten. 0 ger ingen effekt och 7 (111) ger full effekt. Detsamma gäller styrning.

Latticekretsen omvandlar denna data till pulser. Om styrningen får indata 1 (001) ger detta en puls som är hög 1/7 del av den totala periodtiden (2ms).

Källkoden finns i referens [3].

3.5. Konstruktion LEGO - bil

Vid formgivningen av bilen eftersträvades en någorlunda likhet med en riktig bil. Den fick ett Jeep-liknande utseende med ett flak längst bak för batteriet. All elektronik inklusive motorerna placerades inne i kupén. Konstruktionsmaterialet, LEGO Technic, visade sig vara mycket flexibelt och gjorde det lätt att testa olika mekaniska lösningar.

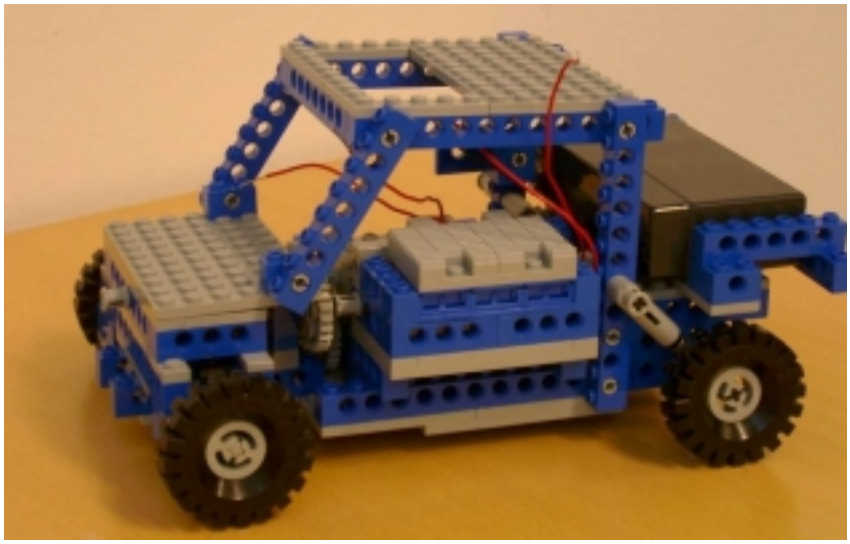


Figure 6: Bil från vänster

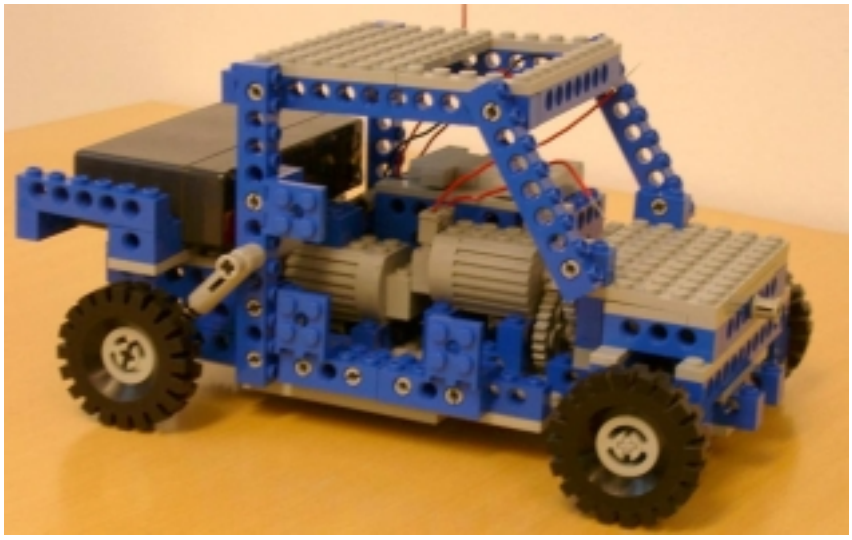


Figure 7: Bil från höger

I kravspecifikationen ingår att bilen ska styras enligt rattprincipen, dvs genom att låta en motor vinkla framhjulen. Detta gör att mekaniken blir en aning mer komplicerad än den skulle blivit med stridsvagnsprincipen. Dels måste hjulen naturligtvis vara vridbara och dels krävs kraftöverföring från styrmotorn till hjulens vridanordning. Motorn som används är en vanlig LEGO-motor. För att få tillräcklig kraft och lagom hastighet i vridningen krävs ordentlig utväxling. Kraftöverföringen består därför av ett antal kugghjul och till sist en kuggstång monterad på den ledbara ställning som hjulen sitter på.

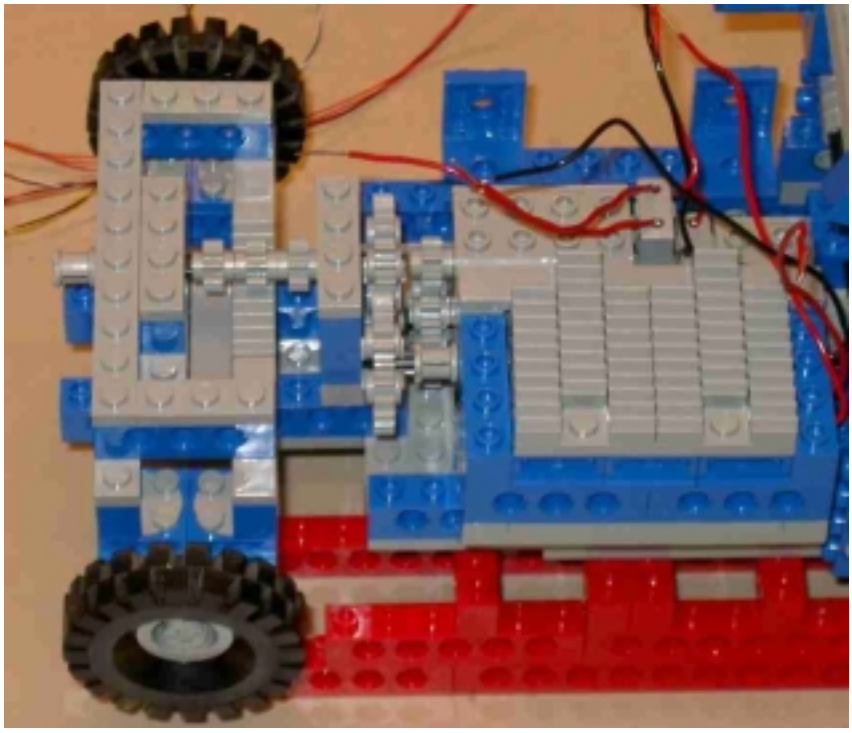


Figure 8: Kraftöverföring för styrningen

Bilen innehåller även en potentiometer. Att få denna att vridas av styrmotorn innebar en viss utmaning. Potentiometern är placerad i en liten låda av LEGO och kopplad till motorn med en utväxling som gör att i stort sett hela mätområdet utnyttjas.

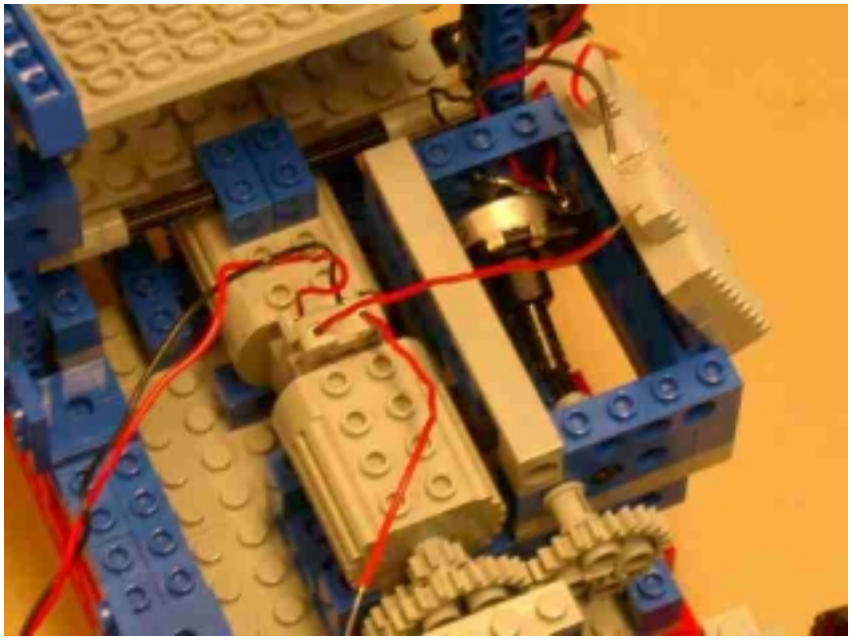


Figure 9: Potentiometer

För bilens drivning används en likadan LEGO-motor som för styrningen. Bilen är bakhjulsdriven och även kraftöverföringen till bakhjulen har viss utväxling, dock inte lika mycket som den till styrningen. Eftersom bilen saknar växlar, gällde det vid konstruktionen av utväxlingen att hitta en lämplig kompromiss mellan snabbhet och styrka. Kraftöverföringen till bakhjulen innehåller även en differential för att ge bilen bästa möjliga väghållning.

3.6. Arbetsgång

Projektet startade med öppen diskussion om lösningen. Huvuddragen bestämdes snabbt. Frågan om pulsning till motorerna skulle ske med programmerbar logik eller mjukvara återstod. En första lösning skulle byggas i mjukvara. Komponenter valdes och ett detaljerat kretsschema konstruerades (Se kap. 7). Efter godkänd kravspecifikation och ritning påbörjades konstruktionen av elektroniken. Parallellt med detta utvecklades användargränssnittet och mekaniken. I detta skede upptäcktes att det troligen krävdes en återkoppling från styrningen för att få tillräcklig precision. Lösningen blev att en potentiometer lades till konstruktionen. Arbetet flöt på utan några större komplikationer förutom en bränd radiokrets.

Efter några veckor var elektroniken, mekaniken och användargränssnittet färdiga och det var dags att börja programmera HC11:an. Några testprogram skrevs för att testa utmatning till portarna och inmatning från A/D-omvandlaren. Snart fanns ett program som kunde ge bakhjulen önskad hastighet och vrida framhjulen till önskad vinkel, dock utan kommunikation med PC:n. Problem uppstod när styrmotorn inte orkade vrida runt den ganska tröga potentiometern. Detta problem visade sig lyckligtvis bero på strömbegränsningen i spänningsaggregatet och löstes genom att koppla in batteriet. Ännu så länge hade elektroniken fungerat utan problem. Även radiokommunikationen fungerade förvånansvärt bra. Viss problematik uppstod dock med att få HC11:an interrupts att reagera på inkommande data. Vidare visade sig indata skilja sig markant från den data som skickades från PC:n. Detta visade sig bero på elektroniken. Däribland glappkontakt i sändare och mottagare. Mycket sent i projektet upptäcktes även att jorden spelade in. Med bilens jord kopplad till spänningskubens jord fungerade sändaren medan den inte fungerade med batteriets jordpunkt ensamt.

Nästa uppgift blev att ordna pulsningen till H-bryggorna. Hittils användes olika mjukvaruvarianter av pulsning. Den första metoden byggde helt enkelt på en snabb loop i mjukvara som

skapade pulserna. Denna metod visade sig bli ojämn och tog upp mycket processorkraft. Den andra metoden blev att skapa pulser med hjälp av en intern ”Output-compare” funktion. Denna funktion möjliggör avbrott vid förutbestämda tider. Denna metod är teoretiskt väl lämpad för pulsning. Diverse timingproblem och problem med att alltid hinna med att exekvera avbrotten gjorde att metoden aldrig blev riktigt funktionsduglig. Den gamla idén om programmerbar logik väcktes till liv. Arbetade skedde parallellt med de båda lösningarna och valet föll på logikkretsen då denna fungerade först.

Nästa steg var att eliminera datapaket som blivit felaktiga på grund av störningar och annat. En första tanke föll på felkontrollerande koder som CRC-32 men insåg att samma effekt kunde uppnås genom att upprepa paketet ett antal gånger.

Det var nu dags att för första gången placera kretskortet i bilen och driva det med batteriet. Genast uppstod ett nytt problem, nämligen att få 5 Volt till kretskortet från ett 6 Volts- batteri. Ett första försök gjordes med en spänningsregulator. Ett problem med denna är att önskad matningsspänning för en utspänning på 5.2V är 8-9V. Batteriets 6V gjorde att utspänningen blev 4.8V vilken skulle kunna vara för lågt i vissa fall. Därför provades även olika varianter av spänningsdelning.

De sista dagarna ägnades åt mystiska störningar i radioöverföringen. Skulden lades på sviktande jordplan och jordslingor. Med ”Sladdrift” fungerade konstruktionen problemfritt.

4. Resultat

4.1. Slutsatser

Målen i kravspecifikationen uppnåddes tidigt och därmed kan projektet anses vara lyckat. Arbetet förlöpte mycket smidigt och de problem som uppstod löstes snabbt.

Extrauppgiften som angavs i kravspecifikationen, dvs att korrigera felaktig data som skickats, löstes inte. Detta visade sig vara onödigt med tanke på att paketen skickas så ofta att det räcker att kontrollera dem och kasta bort dem som visar vara felaktiga. På detta vis korrigeras felaktigheter genom att skicka redundant data.

4.2. Förbättringsförslag?

Vid slutfasen av projektet fungerar delarna mycket bra tillsammans. Ett sista irritationsmoment var den mycket höga graden av fel i dataöverföringen. Denna del ligger öppen för förbättringar i form av en bättre skärmad och skyddad konstruktion med mindre störningar.

En annan del som kan optimeras är strömförbrukningen. Detta problem har inte adresserats i denna prototyp då effektförbrukningen i motorerna ansågs dominerande. Den dubbla hårdvaran i form av Logik och µProcessor kan ge upphov till onödig strömförbrukning. Likaså användes en mycket ineffektiv spänningsdelning för att få rätt matningsspänning till elektroniken. En konstruktion med en spänningsregulator skulle t.ex. ge minskad effektförbrukning.

Användargränssnittet var mycket enkelt och innehöll endast basala funktioner. Det finns här möjlighet till uppfräschning även om funktionaliteten ansågs fullt tillräcklig.

I övrigt är gruppen mycket nöjd med konstruktionen och anser att projektet var mycket professionellt utfört och med gott resultat som följd.

5. Sammanfattning

Vi valde att bygga en digital radiostyrd bil som styrs från en PC. Detta gav oss utmaningar inom digital konstruktion och programmering men ledde oss även in på områden som radio, seriell kommunikationon, analog elektronik och LEGO-konstruktion.

Vi strävade efter att få bilen att fungera ungefär som en vanlig radiostyrd bil. Den har därför styrning enligt rattprincipen, dvs genom att låta en motor vinkla framhjulen. För manövreringen används piltangenterna på PC:n, ungefär som när man spelar ett bilspel.

Projektet kan delas in i följande delar:

- Styrgränssnitt
- Sändarkrets
- Bilens hårdvara
- Bilens mjukvara
- Bilens mekanik

Styrgänssnittet består av ett Java-program som körs på PC:n. Det har ett enkelt GUI och styrs som sagt med piltangenterna.

Programmet skickar paket med information om hur bilen ska bete sig till sändarkretsen. Denna är kopplad till serieporten på PC:n och vidarebefordrar paketen via radio.

Bilens hårdvara bygger på enchipsdatorn HC11. Paketen tages emot via en radiokrets och tolkas i mjukvaran. Mjukvaran har även tillgång till framhjulens aktuella vinkel genom att en potentiometer är kopplad till HC11:ans A/D-omvandlaringång. Utifrån denna information avgörs vilka hastigheter motorerna ska ha. För att omvandla hastighetsvärdena till analoga spänningsnivåer till motorn använder vi en programmerbar logikkrets och H-bryggor.

Mekaniken består främst av utväxling mellan motorerna och de delar som ska drivas.

Vi uppnådde de mål vi hade föresatt oss i kravspecifikationen och anser därmed att projektet lyckades. Visserligen stötte vi på många problem under arbetets gång men inga som var värre än att de gick att lösa. Avslutningsvis bör sägas projektet var en intressant erfarenhet som på ett inspirerande sätt knöt samman många av de kurser vi tidigare läst på LTH.

6. Referens

[1] T.Åkesson : Källkod för grafiskt gränssnitt
www.it.lth.se/it/courses/Digp
/sammanfattning/2002/lp-4/grupp21/InputReader.java

[2] T.Åkesson: Källkod för styrning med HC11
www.it.lth.se/it/courses/Digp
/sammanfattning/2002/lp-4/grupp21/CarControlLattice.c

[3] K.Karlsson: Källkod för pulsning med Lattice 1016E
www.it.lth.se/it/courses/Digp
/sammanfattning/2002/lp-4/grupp21/h_bridge.vhd

[4] S.Nyman: Bygg och programmera med enchipsdatorn 68HC11
www.it.lth.se /it/courses/Digp/PDF_files/hc11/ByggProg.pdf

[5] Sammanfattning och länkar till referenser
www.it.lth.se/it/courses/Digp
/sammanfattning/2002/lp-4/grupp21/index.html

7. Appendix A: Kretsschema