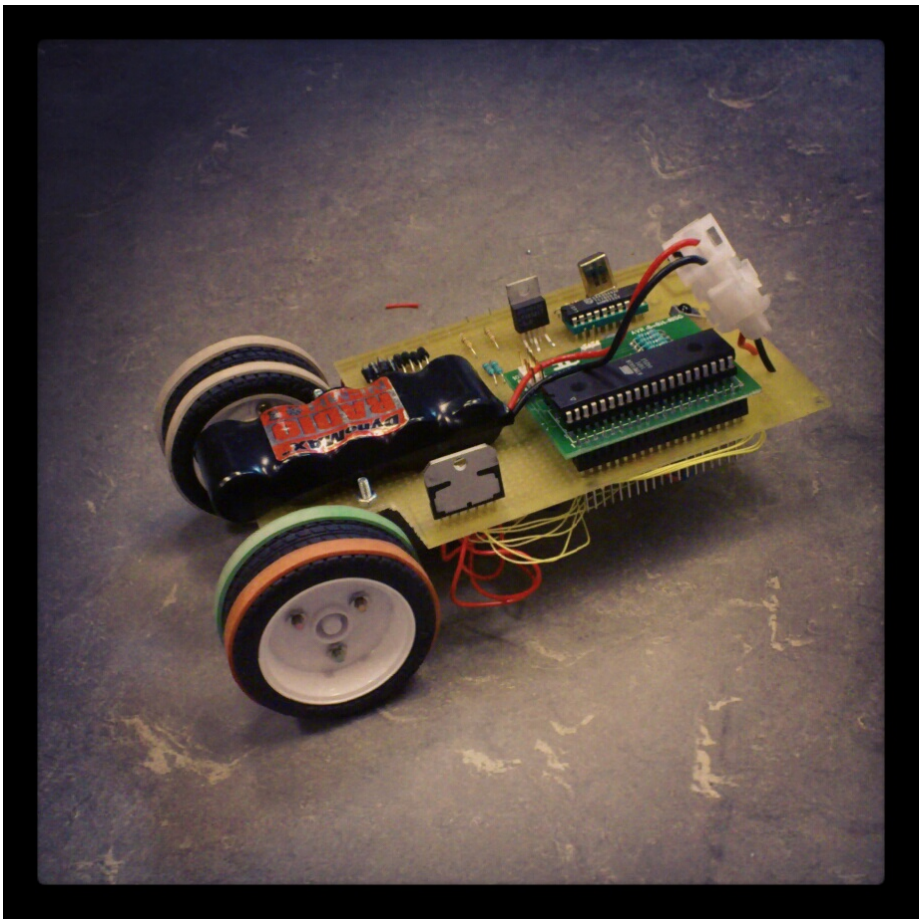
High Core Remote Car X8000 Cool

Ett projekt i kursen Digitala projekt EITF11

Grupp 13
Niklas Ohlsson
Anna Rengstedt
Per Vingå

Handledare: Bertil Lindvall



Abstract:

This project was made when taking the course Digitala projekt EITF11 at LTH. Its purpose is to give students a brief introduction to electronic and digital systems and to see the interaction between them. This is achieved by constructing some kind of electronic prototype. We chose to construct a small car that had the ability to listen commands given by a simple remote control. The commands were drive forwards, drive backwards, turn right and turn left and they were all handled by a software which we had to developed ourselves. During the construction of the prototype, different problems was encountered and taken care of when we reached them. This report gives you a more detailed picture of the entire project, a project which we all agree on was both fun and interesting.

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Syftet med kursen Digitala projekt är att illustrera industriellt utvecklingsarbete. Målet med projektuppgiften är att ta fram en prototyp för vidareutveckling med nödvändig dokumentation. Huvuddelen av kursen består i att konstruera, bygga och testa respektive konstruktion. Följande rapport beskriver konstruktionen av en liten bil som styrs av en fjärrkontroll.

Rapporten inleds med målsättning, kravspecifikation och en beskrivning hur projektet var tänkt att realiseras från en början. Sedan följer en beskrivning av hårdvara och mjukvara, resultat och problem under projektets gång. Bifogat som bilaga finns kopplingsschema och programkod.

1.2 Målsättning

Målet med projektet sattes från en tidig början till att utveckla en farkost som skulle kunna styras mobilt via en fjärrkontroll. Projektet knyter samman tre delar av tekniken:

- *Mekanik* som omfattar konstruktion av en fungerande fjärrstyrd bil. Bilen ska simulera beteendet av en verklig bil, dvs. den ska kunna köras fram, bak samt kunna vrida på de främre hjulen.
- *Elektronik* (hårdvara). Bilen skall förses med lämplig elektronik som möjliggör styrning av motorerna via trådlös kommunikation
- *Mjukvara* (programmering). Beteendet hos vår konstruktion skall vara programmerbar dvs. den skall styras med hjälp av mjukvara.

Kraven på bilens funktioner kan utläsas i rubriken kravspecifikation som följer nedan.

1.3 Kravspecifikation

Primära krav

- Bilen ska kunna gå framåt vid kommando.
- Bilen ska kunna svänga 45 grader åt höger eller vänster vid kommando.
- Det ska finnas en av- och påknapp.
- Bilen ska kunna bromsa.
- Samtliga kommandon ska styras från en IR-fjärrkontroll.

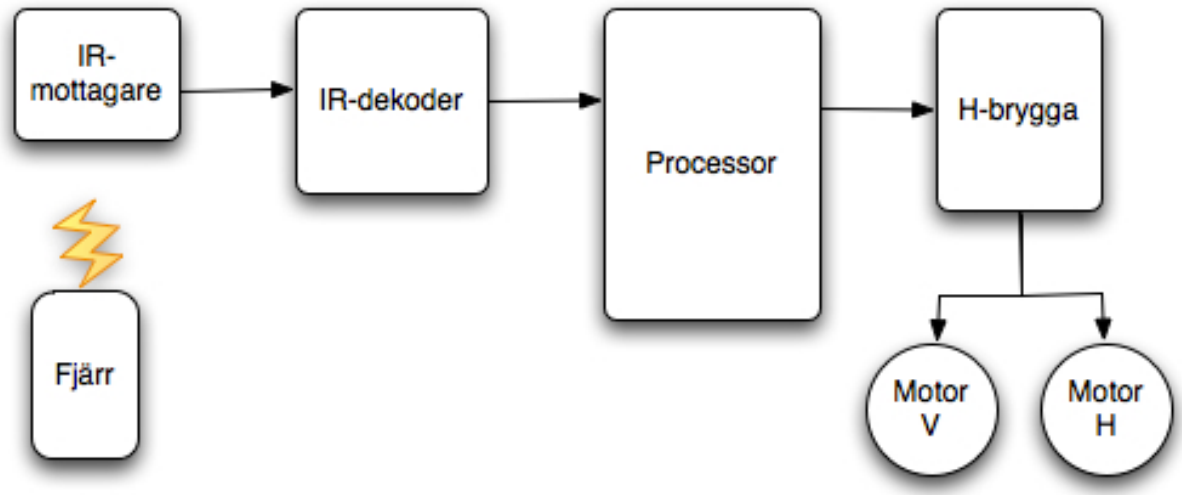
Sekundära krav

- Tidsfördröjning från knapptryck till reaktion måste understiga en sekund.
- Bilen ska kunna köra efter ett fördefinierat mönster.
- Bilen ska kunna hitta tillbaka samma väg den kom ifrån.

2. Genomförande

2.1 Hårdvara

Ett övergripande blockschema kan skådas nedan.



Följande komponenter ingår i bilens hårdvara:

2.1.1 Processor

Processorn som används är ATMega16, en 8-bitars mikroprocessor. Detta är en processor som innehåller många integrerade funktioner, så som intern klocka, A/D-omvandlare, 16KB programmerbart minne, avbrottsrutiner och programmerbara I/O-pinnar. Alla dessa funktioner styrs av en stor mängd så kallade styrregister som består av 8-bitar data per register.

ATMega16 och de flesta andra processorer i AVR-serien används främst på grund av sin låga kostnad och stora flexibilitet. De är också väldigt strömsnåla och relativt lätthanterliga. 32 av pinnarna på ATMega16 är I/O-pinnar. Dessa kan styras individuellt om de ska fungera som input eller output. 8 av dessa är också analoga ingångar med inbyggd A/D-omvandlare som kan användas vid behov.

2.1.2 IR-diod

För att kunna ta emot signaler från fjärrkontrollen behövs en IR-diod, i detta fall ELRIM-8608S. Denna diod kan användas till TV, multimedia, luftkonditionering samt till en mängd andra ändamål.

2.1.3 IR-dekoder

En IR-dekoder har till uppgift att detektera en signal från t.ex. en IR-diod och konvertera denna till en parallell signal som kan läsas av processorn. Dekodern som används heter SAA3049A och har två olika tillstånd. Skillnaden på dessa är att de detekterar olika typer av signaler. Fjärrkontrollen som används använder standardprotokollet RC5 vilket i fallet med den fjärrstyrda bilen att Mode = 0V.

2.1.4 H-brygga
En H-brygga används för att man ska kunna ge höga effekter med bara insignaler från en mikroprocessor. H-bryggan kan styra hastighet och riktning på två motorer oberoende av varandra. Signalen till H-bryggan styrs med två separata PWM-signaler (Pulse-Width Modulation) från processorn.

2.1.5 Spänningsregulator
Processorn kräver en stabil matningsspänning på 5V och till förfogande vid framtagandet av prototypen fanns det ett batteri på 6V, därför krävdes det en spänningsregulator som sänkte batteriets utsignal. Spänningsregulatorn som användes var en LP3855, som ger en stabil spänning på 5V, vilket var ett krav för att mikroprocessorn skulle fungera önskvärt.

2.1.6 Motorer
För att fordonet skulle kunna förflyttas framåt behövdes det någon form av motor som driver. För att kunna styra utan ett svänghjul användes det två DC-motorer. Dessa styrdes separat med hjälp av två H-bryggor och separata PWM-signaler från processorn. Detta för att kunna ge olika spänning till dem så fordonet blev styrbart.

2.2 Mjukvara
Mjukvaran till radiobilen skrevs i C och fördes över till processorn via AVR Studio. Se bilaga 2 för att ta del av den slutliga programkoden.

2.3 Arbetsgång
Det första momentet i arbetet var att rita ett kopplingsschema med hjälp av databladen för de olika komponenterna. Så snart detta var gjort, kopplade vi ihop samtliga delar genom att löda och vira. Under arbetets gång modifierade vi sedan vår konstruktion, då vi insåg att ytterliggare beståndsdelar krävdes.

Nästa steg var att få motorerna att reagera på fjärrkontrollens signaler och att bestämma vilka knappar som skulle signalera framåt, bakåt, höger, vänster och stopp. Med detta som grund byggde vi sedan upp programkoden i AVR Studio.

Det sista momentet innefattade att finslipa programkoden så att bilen skulle reagera på önskat sätt, gällande bland annat hastighet, avbrott och körsträcka.

3. Resultat
Följande avsnitt är avsett att beskriva hur arbetet har fortgått och vad vi gjorde för att komma fram till vår slutgiltiga prototyp.

Kopplingsschemat gjordes med vissa problem eftersom mycket kunskap saknades om vilka komponenter som behövdes och hur dessa skulle sammankopplas(se bilaga 1). En del datablad var bristfälliga, framförallt de till H-bryggan och IR-dekodern vilket försvårade arbetet. Ihopkopplandet av konstruktionen fortlöpte utan större problem men många omkopplingar har gjorts i och med att vi lärt oss mer och därmed insett hur komponenterna ska kopplas. Ovana vid att löda och vira gjorde att vissa fantasifulla lösningar fick tillämpas när vi märkte hur många kablar vi behövde.

Så snart vi förstått oss på hur vi skulle skriva kod för att få hjulen att snurra, uppstod ett problem då vi skulle bestämma vilka signaler fjärrkontrollen sände ut och hur dessa skulle tas omhand. Med hjälp av en logikpenna kunde vi se att många pinnar befann sig i ett ”inget-läge”, istället för low eller high. Lösningen var att koppla in ett antal pull-up motstånd, vilket genast gav resultat. Samtidigt såg vi i H-bryggans datablad att vi behövde ett antal kondensatorer, vilka genast förbättrade funktionen. När vi bestämt vilka knappar på fjärrkontrollen som skulle ge vilka funktioner så märkte vi att bilen reagerade både långsamt och inkonsekvent. Detta problem löstes direkt då vi fäste kondensatorer på motorerna vilka direkt dämpade störningarna.

Vidare till programkoden så insåg vi snart att vi behövde avbrott för att bilen skulle kunna ta emot mer än ett kommando. Det var svårt att förstå hur avbrotten fungerade och lång tid fick läggas på att lära sig detta. När avbrotten var programmerade kom nästa problem då det visade sig att de två hjulen snurrade olika snabbt. Vi tog därför med två timers för att kunna reglera hastigheten för respektive motor. Vi bestämde oss även för att låta bilen köra en viss tid vid varje knapptryckning, för att sedan stanna och ta emot ett nytt kommando. Genom att reglera denna tid, kunde vi se till att bilen svängde 45 grader och att den tog sig framåt önskad sträcka.

4. Slutsatser

4.1 Lärdomar

Lärdomarna efter slutfört projekt är många, inte minst eftersom ingen av oss innan projektets början visste särskilt mycket om elektronik eller hur man utformar mjukvara till en komponent. En viktig lärdom är att vi har lärt oss tyda datablad och ta till oss teknisk information.

Något som har varit vår filosofi under arbetets gång är att prova sig fram tills det blir rätt. Även om det har inneburit att vi har fått löda om, skriva om kod och byta ut komponenter, så är det med hjälp av detta vi har lyckats ta oss framåt. Särskilt motstånd och kondensatorer är något vi fått lägga till i efterhand.

Under arbetets gång har logikpennan varit till stor hjälp för att bland annat kontrollera pinnarnas läge och reaktion vid knapptryckningar. Även multimetern har kommit till användning.

4.2 Förbättringsförslag

Vår slutgiltiga radiobil uppfyllde samtliga primära krav i kravspecifikationen, med undantag från att det inte finns någon av- och påknapp. Man kan dock få bilen att stanna och i värsta fall är batteriet lätt att koppla ur.

Ett problem vid körning av bilen var att däcken hade dåligt grepp, vilket försvårade styrning. Detta löstes delvis genom att sätta gummiband runt däcken, men funktionen kan fortfarande förbättras. Vid låga hastigheter hade bilen svårt att ta sig framåt, vilket beror främst på de svaga motorerna. Starkare motorer skulle därför kunna göra bilen mer kraftfull.

5. Källförteckning

AVR Libc User Manual 1.6.7. Hämtat från
http://www.eit.lth.se/fileadmin/eit/courses/edi021/avrMlibcMuserM
manual/index.html

ATmega16 Datasheet. Hämtat från
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.
pdf

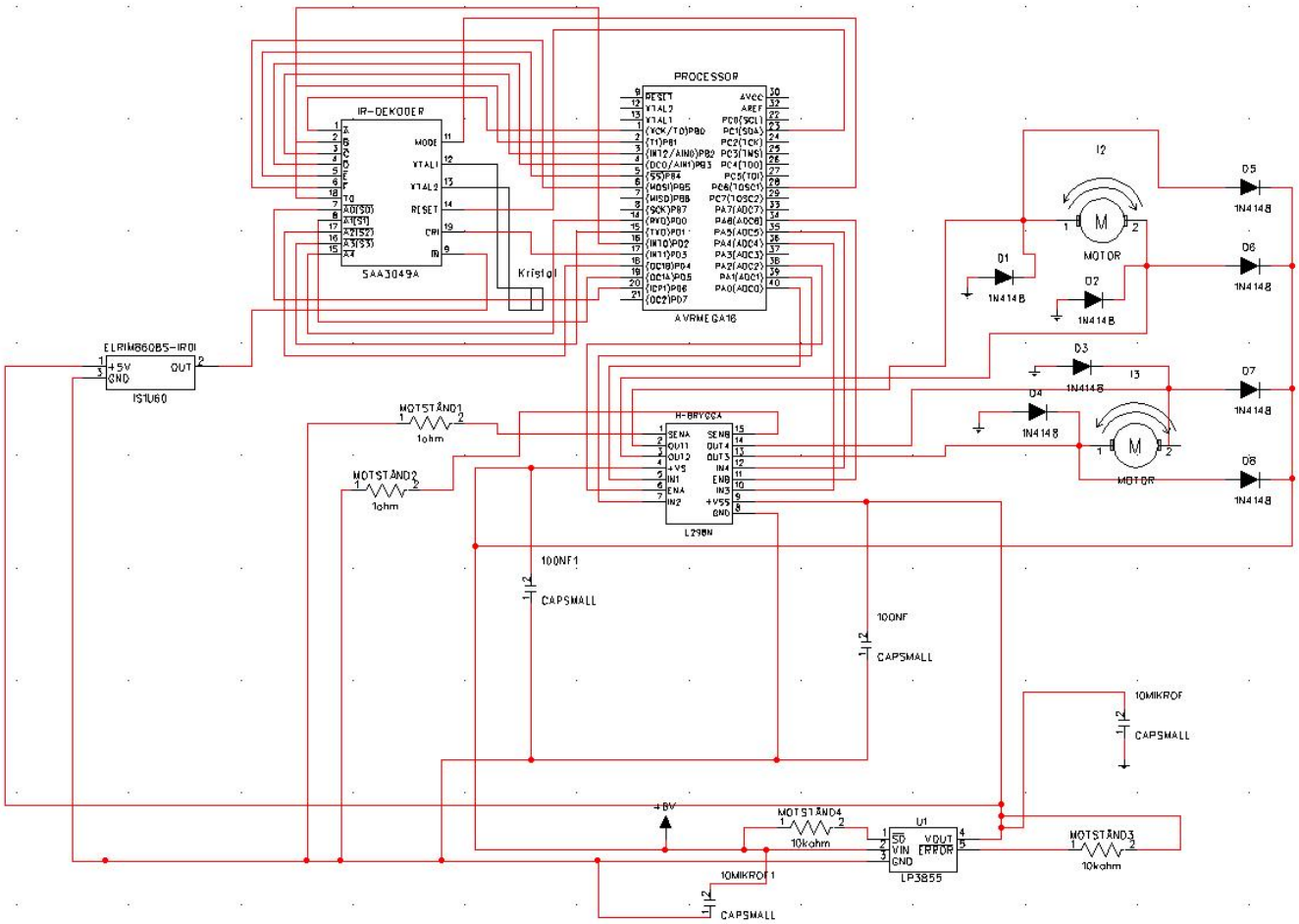
L298 Dual Full Bridge Driver Datasheet. Hämtat från
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Analog/linear/l298.pdf

SAA3049A Infrared Remote Control Decoder Datasheet. Hämtat från
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/Other/SAA3
049A.pdf

LP3852/LP3855 Fast Response Ultra Low Dropout Linear Regulators Datasheet. Hämtat
från
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Analog/voltage/lp3855.
pdf

6. Bilagor

6.1 Kopplingsschema



6.2 Källkod

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define forward 0xFD
#define backward 0xFF
#define left 0xFA
#define right 0xF8
#define stop 0xF9

char direction = 0xF9;
int drivetime = 0;

volatile unsigned int   count0 = 0;
volatile unsigned int   count1 = 0;
volatile unsigned int   count2 = 0;

/*
Sätter starttillståndet
*/
void initialize()
{

                DDRA = 0xFF;
                                                        //PORTA alla utsignaler
                DDRB = 0x00;
                                                        //PORTB alla insignaler
                DDRC = 0x42;
                                                        //PORTC alla insignaler, förutom PC6&PC1 - till
                                                        // MODE&RESET
                DDRD = 0x00;
                                                        //PORTD alla insignaler

                PORTC = 0xBF;
                                                        // sätter RESET till 1
                PORTC = 0xBD;
                                                        // RESET & MODE = 0

                MCUCR |= (1<<ISC10);
                                                        //INT1 OCH INT0 AKTIVERAD
                GICR |= (1<<INT1);
                                                        //INT1 OCH INT0 AKTIVERAD

                //TIMER1
                TCCR1B=(1<<WGM12)|(1<<CS10)|(1<<CS11);  //PRESCALER 64
                OCR1A=125;
                                                        //Räkna till (motsvarar 1ms)

                //TIMER2
                TCCR2 = (1<<WGM21)|(1<<CS22);
                //PRESCALER 64
                TCNT2 = 125;
                                                        //Räkna till (motsvarar 1ms)
                TIMSK = (1<<OCIE2)|(1<<OCIE1A);

                sei();                                  //Aktiverar globala interrupts
}


/*
Sätter PORTA till rätt värde beroende på vilken riktning som bestämts med fjärrkontrollen
*/
void command()
{

                switch(direction)
                {

                case stop:
                                                        // Knapp (6)
                                PORTA = 0xFF;
```

```
                                        break;
                        case backward:
                                                                                // Knapp (0)
                                        drivetime = 600;
                                        PORTA = 0xDD;
                                        break;
                        case forward:
                                                                                // Knapp (2)
                                        drivetime = 600;
                                        PORTA = 0xEE;
                                        break;
                        case left:
                                                                                // Knapp (5)
                                        drivetime = 170;
                                        PORTA = 0xDE;
                                        break;
                        case right:
                                                                                // Knapp (7)
                                        drivetime = 170;
                                        PORTA = 0xED;
                                        break;
                        }
}


/*
Slår på TIMER0
*/
void timeron()
{
                        TCCR0|=(1<<CS01)|(1<<CS00);
                        //PRESCALER 64
                        TIMSK|=(1<<TOIE0);
                        TCNT0=0;
                        sei();
}

/*
Stänger av TIMER0
*/
void timeroff()
{
                        TCCR0 = 0;
                        TIMSK = TIMSK & ~_BV(TOIE0);
}

/*
Avbrott för TIMER0
*/
ISR(TIMER0_OVF_vect)
{
                        count0++;
                        if(count0 > drivetime)
                        {
                                        PORTA = 0xFF;
                                        count0 = 0;
                                        timeroff();
                        }
}


/*
Avbrott för TIMER1 för reglering av hastighet (höger motor)
*/
ISR(TIMER1_COMPA_vect)
{
                        count1++;
                        if( count1 <6)
                        {
                                        PORTA = PORTA | _BV(PA2);
                        //EN PÅ
                        } else if (count1 <8)
                        {
```

```
                                        PORTA = PORTA & ~_BV(PA2);
                                        //EN AV
                        } else {
                                        count1 = 0;
                        }
}


/*
Avbrott för TIMER2 för reglering av hastighet (vänster motor)
*/
ISR(TIMER2_COMP_vect)
{
                        count2++;
                        if( count2 <4)
                        {
                                        PORTA = PORTA  | _BV(PA6);
                        //EN PÅ
                        } else if (count2 <7)
                        {
                                        PORTA = PORTA  & ~_BV(PA6);
                        //EN AV
                        } else {
                                        count2 = 0;
                        }
}

/*
Externt vbrott vid knapptryckning
*/
ISR(INT1_vect)
{
                        direction = PINB;
                        command();
                        timeron();
}

int main(void)
{
                        initialize();

                        while(1)
                        {
                        }
}
```