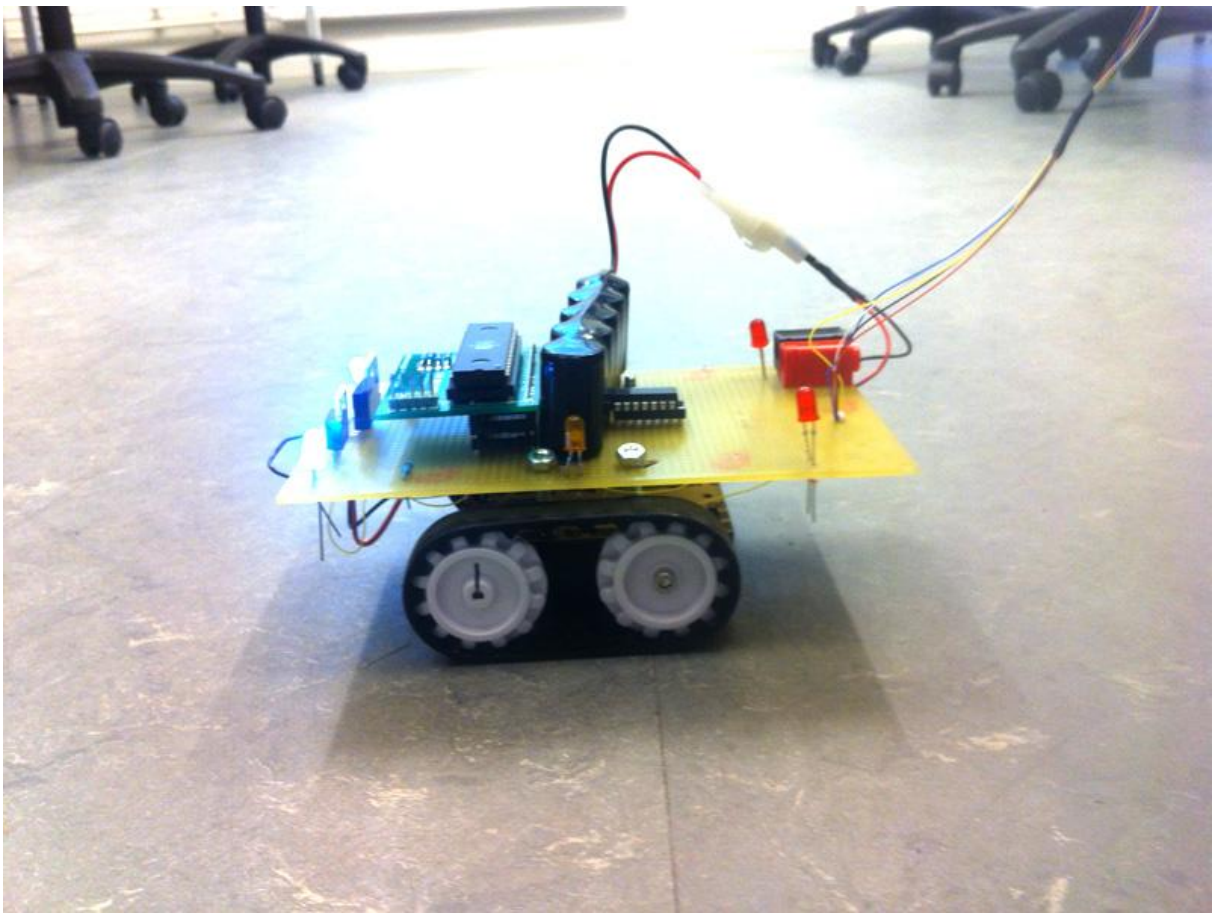
2013

Lunds Tekniska
Högskola

I-10
Grupp 11:
David Sundström
Max Schulz
Albert Lundberg
Handledare:
Bertil Lindvall



DIGITALA PROJEKT

The objective of the course Digital Project is letting a group of students construct an electronic unit. This group chose to construct a RC-vehicle which was to be controlled by a computer. The plan was to connect the computer to an own constructed control and then let the control, through radio transceivers, communicate commands to the car. The group never got the radio communication to work and had to solve this through wire communication instead. The report describes the working process and all the challenges the group met throughout the project.

Innehållsförteckning

Inledning

Digitala projekt är en kurs vars syfte är att ge en grupp studenter möjligheten att under sju veckors tid konstruera en fungerande elektronisk enhet. I detta ingår planering och konstruerande av själva hårdvaran samt planering och implementering av mjukvaran som utgörs av programkod.

I denna rapport redovisas hur gruppen gått tillväga vid byggandet av en radiostyrd bil. Det ursprungliga målet som gruppen satte upp, var att m.h.a. en dator styra den radiostyrda bilen. Detta skulle uppnås genom att konstruera två olika enheter. Dels en enhet som hade direkt kontakt med datorn, i denna rapport benämnd kontrollen, och dels en enhet som utgjordes av själva radiostyrda bilen, i denna rapport benämnd bilen. Kommunikationen mellan kontrollen och bilen skulle åstadkommas m.h.a. radiosändare, något som gruppen misslyckades med. Kommunikationen skapades istället av sladdar som kopplades mellan kontrollen och bilen.

Målsättning

Gruppens ursprungliga målsättning med projektet var att skapa en radiostyrd bil som skulle kunna styras m.h.a. en dator. Datorn skulle vara kopplad till en kontroll som bestod av dels en processor samt dels en radiosändare. Kontrollen skulle sedan i sin tur kommunicera med bilen via dess radiomottagare. Signalerna från kontrollen skulle snappas upp av bilens radiomottagare för att sedan tolkas m.h.a. processorn som sedan skulle utföra ett önskat kommando.

Nedan följer den ursprungliga **kravspecifikationen:**

Ursprungliga krav
Den radiostyrda bilen skall kunna styras från en separerad enhet. D.v.s. trådlöst.
Den radiostyrda bilen skall kunna köra både framåt och bakåt.
Den radiostyrda bilen skall kunna svänga både höger och vänster.
Den radiostyrda bilen skall kunna tända och släcka strålkastare.
Den radiostyrda bilen skall ha bromsljus.
Den radiostyrda bilen skall kunna bjuda på en överraskning.

Metod

För att åstadkomma ett så lyckat resultat som möjligt valde gruppen att dela upp arbetet i fyra tydliga faser. Dessa var:

- Fas 1: Planera och bygga ihop hårdvaran.
- Fas 2: Åstadkomma kommunikation mellan datorn och kontrollen.
- Fas 3: Åstadkomma kommunikation mellan kontrollen och bilen.
- Fas 4: Programmera mjukvaran.

Vid avslutandet av varje fas lades stort fokus på att varje fas hade levererat önskat resultat. Gruppen insåg snabbt att om exempelvis inte hårdvaran fungerade perfekt varje gång så skulle arbetet med mjukvaran försvåras avsevärt.

Hårdvara

Com-port

Denna användes för att möjliggöra kommunikationen mellan datorn och kontrollen. Kort sagt är en com-port en seriell port, vilket menas med att informationen som sänds från datorn skickas en bit i taget.

Maxim multichannel RS-233

Maxim:en möjliggör kommunikationen mellan com-porten och processorn på kontrollen vilket krävs, eftersom de arbetar med olika spänningar.

Processorn – ATmega16

ATmega16 är en mikroprocessor med ett programminne på 16 kB, vilket var fullt tillräckligt för både kontrollen och bilen. Den stora fördelen med denna processor är att den stödjer SPI (Serial Peripheral Interface) vilket har använts vid sladdkommunikationen mellan kontrollen och bilen.

Motorerna

Motorerna som använts i bilen är två 6 volts likströms motorer. Dessa levererar egenskaper likt en pansarvagn, d.v.s. att de två motorerna kan snurra åt önskat håll samtidigt.

H-brygga – L298N

H-bryggans funktion på bilen är att leverera ström till motorerna vilket krävs eftersom processorn strömleveransförmåga är otillräcklig. Andra viktiga egenskaper hos H-bryggan som gruppen använt sig av är dess förmåga att driva de två motorerna oberoende av varandra samt i olika hastigheter.

Radiosändarna – Parallax 433 MHz Rf Tranciever (Används inte)

Oerhört enkel och lättanvänd radiosändare som arbetar med en frekvens på 433 MHz. Planen var att en skulle fästas på kontrollen och en på bilen och sedan låta dem sköta kommunikationen. Detta fungerade ej, varför dessa ej finns med i den färdiga enheten. Bra

egenskaper med denna variant är att den stödjer seriell kommunikation samt att den kräver väldigt lite ström, vilket hade varit fördelaktigt eftersom vår bil går på batterier.

I appendix 1.4 finns kopplingsschema som beskriver hur all hårdvara har kopplats ihop.

Mjukvaran

I appendix 1.2 finns C-koden till kontrollen. Kortfattat gäller det att koden för kontrollen är skriven så att den tar emot tecken från Hyperterminalen. Om tecknet motsvarar ett tecken som skall få bilen att agera, exempelvis ”W” som för bilen framåt, så översätts tecknet till en specifik siffra. När informationen mottagits från Hyperterminalen så skall en sändning med siffran genomföras. För att åstadkomma denna sätts kontrollen (master) till High samtidigt som bilen (slave) sätts till Low, vilket gör att en sändning nu är möjlig. Därefter skickas siffran till bilen.
I appendix 1.1 finns en manual för hur man använder kontrollen via Hyperterminalen.

I appendix 1.3 finns C-koden till bilen. Bilen avvaktar tills dess att kontrollen initieras för mottagning av information. När denna initiering sker tas siffran som skickas från kontrollen emot och översätts till ett kommando som får bilen att antingen röra sig på ett specifikt sätt eller dess lampor att lysa på ett specifikt sätt. Korta förklaringar till de flesta kommandona för att förklara koden ytterligare.

Resultat

Ursprungliga krav	Avklarat	Anledning	Lösning
Den radiostyrda bilen skall kunna styras från en separerad enhet. D.v.s. trådlöst.	Nej	Störningar i luften, icke tillräckligt avancerade radiosändare samt tidsbrist.	Kommunikation m.h.a. sladdar.
Den radiostyrda bilen skall kunna köra både framåt och bakåt.	Ja		
Den radiostyrda bilen skall kunna svänga både höger och vänster.	Ja		
Den radiostyrda bilen skall kunna tända och släcka strålkastare.	Ja		
Den radiostyrda bilen skall ha bromsljus.	Ja		

Den radiostyrda bilen skall kunna bjuda på en överraskning.	Ja		Ett ljusspel involverande samtliga lysdioder skapades.
Bonusfunktioner	**Avklarat**	**Anledning**	**Lösning**
Den radiostyrda bilen skall kunna blinka både höger och vänster.	Ja		Löses med delays
Den radiostyrda bilen skall kunna öka samt sänka hastigheten.	Ja		Löses med delays
Den radiostyrda bilen ska kunna släcka/tända samtliga lampor samtidigt med en knapp.	Ja		

Svårigheter

Den första svårigheten under detta projekt var medlemmarnas oerhört tunna kunskapsbas vid början av projektet. Detta medförde att klantiga misstag gjordes men framförallt att befintliga problem tog på tok för lång tid att finna. Ett exempel på detta var när det tog två dagar, innehållandes omlödningar och omprogrammeringar, för att inse att kontrollens processor hade gått sönder. Denna övergripande svårighet löstes dock relativt snabbt m.h.a. diverse datablad och vår handledare.

Den största svårigheten under projektet var att få radiosändarna att fungera. Detta arbete spenderade gruppen ungefär två veckor på och var tillslut tvungna att ge upp p.g.a. tidsbrist. Det huvudsakliga problemet låg i att radiosändaren på bilen, som var programmerad till att ta emot data, tog emot massor av brus utifrån. Detta brus gjorde det oerhört svårt att avgöra när det som sändes faktiskt kom från kontrollen eller när det kom utifrån.

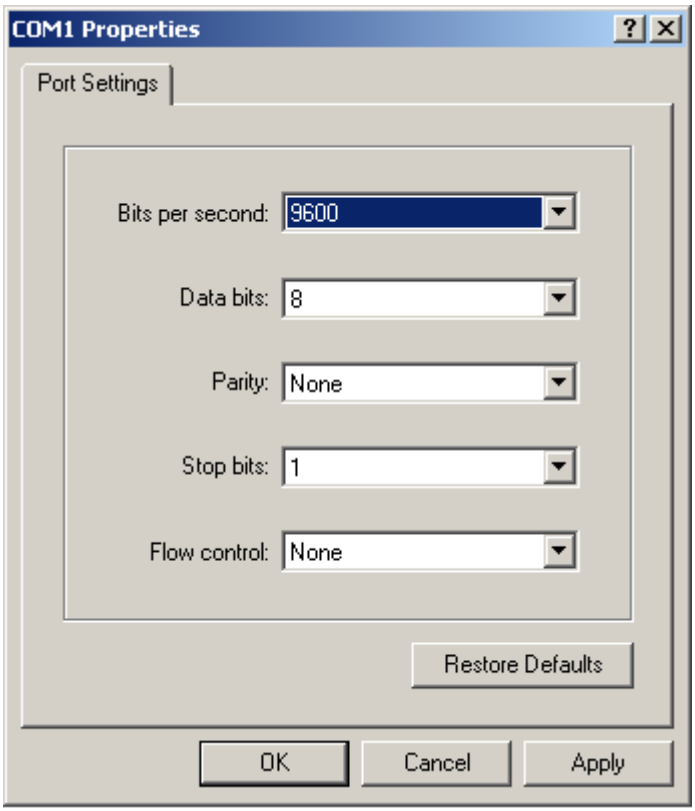
Slutsats

Trots att gruppen inte fullt ut uppnådde alla förutbestämda mål så anses resultatet lyckat. Få gånger har en kurs varit så givande. Främst i form av kunskap, där gruppen lärt sig om hur elektriska enheter är uppbyggda, hur man programmerar i C och hur man bygger en hårdvara. Utöver det så har gruppen lärt sig om hur mycket arbete som ligger bakom skapandet av en tillsynes enkel produkt. Alltifrån vikten av planering, problemlösningsförmåga, tålamod och samarbete. Vidare så har gruppen, den hårda vägen, lärt sig att hantera när saker inte blir som de är tänkta och hur mycket mer arbete som krävs än vad en bra tidsplanering någonsin kan påvisa. Slutligen skulle gruppen vilja tacka sin handledare Bertil Lindvall för dennes tålamod och engagemang och rekommendera kursen till alla som vill lära sig mer än enbart teori på LTH.

Appendix

1.1 Manual till bilen

1. Starta hyperterminalen på din dator.
2. Välj den COM-port som kontrollen är inkopplad i.
3. Välj **9600** bit per second, **8** data bits, parity **none**, **1** stop bits och **none** på flow control.



Caps	Aktivera/Avaktivera
W	Framåt
S	Bakåt

A	Vänster
D	Höger
Q	Snurra vänster
E	Snurra höger
1	Öka hastighet
2	Minska hastighet
3	Ställ in standard hastighet
F	Slå på strålkastare
Z	Ljusshow
P	Tänd Alla Lampor
(mellanslag)	Stanna bilen

1.2 C-kod, kontrollen

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

#define CSN_LOW() PORTB &= ~ (1 << PB4); //slave select låg
#define CSN_HIGH() PORTB |= (1 << PB4); //slave select hög

unsigned short int transmit;

void USARTInit(uint16_t ubrr_value)                    //initiering av USART
{

  UBRRL = ubrr_value;
  UBRRH = (ubrr_value>>8);
  UCSRC=(1<<URSEL)|(3<<UCSZ0);
  UCSRB=(1<<RXEN)|(1<<TXEN);


}

char USARTReadChar()                                                    //hämta
information
{

  while(!(UCSRA & (1<<RXC)))
  {
                                //do nothing
  }
  return UDR;
}


unsigned short int USARTWriteChar(char data)      //skriv data till USART
{
```

```
  while(!(UCSRA & (1<<UDRE)))
  {
                                //do nothing
  }

if(data == 'W')            //inmade bokstäver speglas tillbaka via com-porten. En
kommandosiffra skickas till bilen via SPI
                    {
                    UDR = 'W';
                     return 1;
                    }
                    else if(data == 'A')
                    {
                    UDR = 'A';
                    return 2;
                    }
                    else if(data == 'S')
                    {
                    UDR = 'S';
                    return 3;
                    }
                    else if(data == 'D')
                    {
                    UDR = 'D';
                     return 4;
                    }
                    else if(data == 0x20)
                    {
                    UDR = 'G';                                  //mellanslagstangenten returnerar G
                     return 5;
                    }
                    else if(data == 'E')
                    {
                    UDR = 'E';
                     return 6;
                    }
                    else if(data == 'Q')
                    {
                    UDR = 'Q';
                     return 7;
                    }
```

```
else if(data == '1')
{
UDR = '1';
 return 8;
}
else if(data == '2')
{
UDR = '2';
 return 9;
}
else if(data == '3')
{
UDR = '3';
 return 10;
}
else if(data == '4')
{
UDR = '4';
 return 11;
}
else if(data == '6')
{
UDR = '6';
 return 12;
}
else if(data == 'F')
{
UDR = 'F';
 return 13;
}
else if(data == 'P')
{
UDR = 'P';
 return 14;
}
else if(data == 'Z')
{
UDR = 'Z';
 return 15;
}
else
```

```
		{
		UDR = data;
		 return data;
		}
}

void SPI_MasterInit(void) //initiering av SPI för master
{
		DDRB = (1 << PB4) | (1 << PB5) | (1 << PB7);
		DDRB &= ~ (1 << PB6);
		SPCR = (1 << SPE) | (1 << MSTR) | (1 << SPR0) | (1 << SPR1);
}


void SPI_transmit(unsigned short int transmit)
{

		CSN_LOW();                                          //slave sätts till låg
		SPDR = transmit;            //sparar transmit på SPDR och startar
överföringen
		while (!(SPSR & (1 << SPIF))); //väntar till lyckad sändning
		CSN_HIGH();                                 //slave sätts till hög
}


void main() {

		SPI_MasterInit();                                   //initiera master
		CSN_HIGH();
		char data;
		USARTInit(51);                                                  //räknas ut
UBRR-värdet mha boadrate, bitstorlek och processorfrekvens
		while (1)
		 {
				data = USARTReadChar();
				transmit =  USARTWriteChar(data);    //läs av från USART,
skriv sedan på transmit
				if(transmit < 16 && transmit > 0)  //skicka endast vidare
styrkommandon via SPI
				{
				SPI_transmit(transmit);
```

```
                        }
                        else
                        {
                                //gör ingenting
                        }
 }
}
```

1.3 C-kod, bilen

```
#define F_CPU 8000000                    //pga klockfrekvens på 8Mhz
#include <avr/io.h>
#include <inttypes.h>
#include <util/delay.h>

unsigned int direction, speed;
unsigned short int recieve;

#define CSN_LOW() PORTB &= ~ (1 << PB4);
#define CSN_HIGH() PORTB |= (1 << PB4);


void SPI_slaveInit(void) {                                        //SPI initiate för
slaven(bilen)

                DDRB |= (1 << PB6);
                SPCR |= (1 << SPE) | (1 << SPR0);       //enable SPI

}



void Drive(int direction)                                  //sätter på motorerna
{

                while (!(SPSR & (1 << SPIF)))    //kör ända tills ny information från master
är mottagen
                {
                PORTA = direction;                                                  //ställer in
motorerna efter önskad riktining
                _delay_ms(speed);
//beroende på fart stannar motorerna olika länge
                PORTA = 0x12;
                 //stannar motorerna
                _delay_ms(speed);

                }
                if(direction == 0x12) {
```

```
		toggleLight(0b01000000);			//om direktion =
stanna så togglas bakljusen av när nytt kommando ges
		}
}

void toggleLight(int light)
{
		PORTD = PORTD ^ light;				//toggla av/på
lamporna
}

void blinkingLights(int lights)						//blinka
lampor
{
		int k;
		for(k = 0; k < 7; k++)

		{
		toggleLight(lights);

		_delay_ms(500);
		k++;
		}
}

void lightShow()
		//ljusshow med diverse blinkningar och snurr
{
		int k;

		PORTD = 0xff;
		_delay_ms(500);
		PORTD = 0x00;
		_delay_ms(50);
		PORTD = 0xff;
		_delay_ms(50);
		PORTD = 0x00;

		for(k = 0; k < 2; k++)
		blinkingLights(0xff);
```

```
		for(k = 0; k < 11; k++)
		{
		toggleLight(0b01000000);
		_delay_ms(50);
		toggleLight(0b01010000);
		_delay_ms(50);
		toggleLight(0b00110000);
		_delay_ms(50);
		toggleLight(0b00100100);
		_delay_ms(50);
		toggleLight(0b00001100);
		_delay_ms(50);
		toggleLight(0b00001000);
		}

		PORTA =	0x1B;
		_delay_ms(2000);
		PORTA =	0x12;

		toggleLight(0b01000000);
		_delay_ms(500);
		toggleLight(0b01000000);

		for(k = 0; k < 5; k++)
		{
		toggleLight(0b01000000);
		_delay_ms(50);
		toggleLight(0b01100100);
		_delay_ms(50);
		toggleLight(0b00100100);
		}

		PORTD = 0xff;
		_delay_ms(500);
		PORTD = 0x00;

}
void engineControl(void) //omtolkning av insignaler till motorsignaler
{

		if (recieve > 0 && recieve < 16) {
```

```
switch (recieve) {

case 1:
Drive(0x1E);                    //kör framåt
break;

case 2:
Drive(0x16);                    //kör vänster
break;

case 3:
Drive(0x33);                    //kör bakåt
break;

case 4:
Drive(0x1A);                    //kör vänster
break;

case 5:
PORTD = 0x40;                   //tänd bakljus
Drive(0x12);                    //stanna
break;

case 6:
Drive(0x1B);                    //snurra vänster
break;

case 7:
Drive(0x36);                    //snurra höger
break;

case 8:
if(speed < 4)                                   //kolla så
att hastigheten inte understiger 1
{
break;
}
speed = speed - 3;              //minska hastigheten
break;

case 9:
```

```
                        speed = speed + 3;                  //öka hastigheten
                        break;

                        case 10:
                        speed = 24;                                 //sätt
orginalhastighet
                        break;

                        case 11:
                        blinkingLights(0b00001000);                 //blinkers
höger
                        break;

                        case 12:
                        blinkingLights(0b00010000);                 //blinkers
vänster
                        break;

                        case 13:
                        toggleLight(0b00100100);                    //slå på
strålkastare
                        break;

                        case 14:
                        toggleLight(0b11111111);
            //toggla alla lampor
                        break;

                        case 15:
                        lightShow();                //ljusshow!
                        break;

                }
  }
}

unsigned short int SPI_recieve(void) {                  //ta emot signaler från
master

            while (!(SPSR & (1 << SPIF)));
                        return SPDR;
```

```
}

   void main() {

                SPI_slaveInit();
                                //initiera SPI
                DDRA |= (1 << PA0) | (1 << PA2) | (1 << PA3) | (1 << PA5);
                PORTA |= (1 << PA1) | (1 << PA4);                       //enable motors
                DDRD |= (1 << PD0) | (1 << PD2) | (1 << PD3) | (1 << PD4)| (1 << PD5)| (1
<< PD6);        //initiera lamporna
                direction = 0;                                  //sätter riktningen till 0
                speed = 24;                                     //anger starthastighet (delay)


                while (1)
                {
                                recieve = SPI_recieve();
                                engineControl();
                //skickar in mottagen signal i enginecontrol

 }
}
```

1.4 Kopplingsschema