LTH

Thunder's Truck projektrapport

EITF11 – Digitala projekt

2012-05-10

Anton Forssell
Stephanie Gardner
Felix Lindell

Anton Forssell
Stephanie Gardner
Felix Lindell

Abstract

The goal of this project was to build a truck operated by an IP-remote. The robot is built using standard parts such as two DC-motors, an H-bridge, a stepping motor to operate the truck function, an IR-receiver as well as an IR-decoder. The processor used in the project is an ATmega 16 micro processor programmed using the C programming language.

Anton Forssell
Stephanie Gardner
Felix Lindell

Innehållsförteckning

Anton Forssell
Stephanie Gardner
Felix Lindell

1 Inledning

Syftet med projektet var att utveckla kursdeltagarnas kunskaper inom ellära, programmering och digitalteknik genom praktisk erfarenhet. För att uppnå detta valdes som projekt en fjärrstyrd robot som svarar på IR-signaler och som har en truckfunktion.

2 Kravspecifikation

- Thunder's Truck skall styras med hjälp av en infraröd fjärrkontroll.
- Thunder's Truck skall kunna köras framåt och bakåt.
- Thunder's Truck skall kunna svänga till höger och vänster.
- Thunder's Truck skall ha en lyftanordning som skall kunna lyfta mindre objekt.

3 Teori

3.1 Processor

Processorn som används i projektet är en Atmel AVR ATmega16. Detta för att denna energisnåla processor lämpar sig väl för enklare projekt så som detta. Processorn programmerades genom anknytning till dator via en JTAG och programspråket är C. För att förenkla programmeringen användes port D till H-bryggan och port A till IR-dekodern.

3.2 H-brygga

I projektet används en dubbel H-brygga av typen L298N för att stödja de två motorerna. H-bryggan används för att möjliggöra styrning av motorerna, som drivs av 6V, med signaler från processorn.

3.3 Motorer

För att driva Thunder's Truck används två likströmsmotorer som styrs av H-bryggan och driver höger- respektive vänsterhjulet.

3.4 IR-mottagare

I projektet används en IR-fjärrkontroll för att styra Thunder's Truck. För att detta ska vara möjligt krävs en IR-mottagare som kan ta emot signalerna från fjärrkontrollen och kommunicera dessa till IR-dekodern. IR-mottagren som används i projektet är av typen TEW9062 och reagerar endast på IR-ljus som blinkar med en specifik hastighet som överensstämmer med fjärrkontrollen.

3.5 IR-dekoder

De IR-signaler som tas emot av IR-mottagaren kan inte läsas av processorn och måste därför dekodas innan processorn kan ha användning av informationen från fjärrkontrollen. För detta används en IR-dekoder av typen SAA3049A som skickar ut omvänd binärkod då en siffra trycks på fjärrkontrollen. Om en etta trycks på fjärrkontrollen skickas alltså signalen 011111 till pinnarna PA0 till PA5 på processorn (se Appendix A). I projektet används endast fjärrkontrollens siffror för att styra bilen.

Anton Forssell
Stephanie Gardner
Felix Lindell

3.6 Spänningsregulator

Då motorerna kräver en spänning på 6V används ett batteri med den spänningen för att driva Thunder's Truck. Dock kan de digitala komponenterna endast hantera en spänning på 5V. Detta problem löses med hjälp av en spänningsregulator av typen LP3852-5.0 som konverterar 6V till 5V. För att fortfarande förse motorerna med önskad spänning kopplas dessa runt regulatorn och direkt till batteriet.

3.7 Stegmotor

En stegmotor är en digital motor som kan styras direkt från processorn och därför lämpar sig bra för att driva Thunder's Trucks truckfunktion. I en stegmotor omvandlas digitala pulser, i detta projekt alltså från processorn, till en stegvis rotation av motoraxeln. I detta projekt användes stegmotor SM 30-2003 som har fyra faser.

4 Utförande

Efter en vecka av teori i form av föreläsningar inleddes projektet av skapandet av en kravspecifikation. I denna fas av projektet var förståelsen för arbetsprocessen låg och det fanns en problematik med att så tidigt formulera krav för projektet. Av denna anledning hölls kraven simpla. Efterhand som projektet fortskred visade sig detta vara till fördel då friheten vid programmeringen blev större än om t.ex. vilka knappar på fjärrkontrollen som skulle kontrollera styrningen var specificerat redan i kravspecifikation.

I nästa steg i arbetsprocessen skapades ett kopplingsschema som inkluderade alla huvudsakliga komponenter så som motorer, processor och IR-komponenter (se Appendix A). I detta skede av arbetsprocessen inkluderade kopplingsschemat även ett drivsteg till stegmotorn som senare insågs vara överflödigt och plockades bort. Ett problem kopplat till kopplingsschemat som uppstod då kopplingen påbörjades var att kopplingsschemat inte inkluderar strömkällor och behovet av en regulator och motstånd därför inte var uppenbart förrän senare i projektet. Brist på erfarenhet inom ellära och ovana av att läsa datablad ledde till att vissa nödvändiga kopplingar missades och kopplingsschemat fick kontinuerligt uppdateras under arbetets gång.

För att kunna styra Thunder's Truck med IR-signaler behövde processorn programmeras för att ta emot signaler från IR-dekodern och skicka signaler till H-bryggan och stegmotorn. För att möjliggöra kontinuerlig avläsning av A-porten, som är kopplad till IR-dekodern, och anpassning av utsignaler från D-porten, vars pinnar PD1-PD6 är kopplade till H-bryggan, och B-porten, som är kopplad till stegmotorn, genererar processorns interna 8MHz klocka interrupts vid overflow. Vid varje interrupt läses A-porten av och utsignalerna från D- och B-porten anpassas. Med hjälp av en count som räknas upp vid varje interrupt då en metod, t.ex. forward() (se Appendix B), kallas kan Enable A och Enable B på H-bryggan sättas till 0 för att stänga av motorerna och på så sätt kontrollera hastigheten.

I den sista fasen av projektet, testning, upptäcktes att trots att Thunder's Trucks styrning fungerade galant då hjulen inte var i kontakt med marken generade de inte tillräckligt mycket kraft för att driva roboten framåt. Spänningen till likströmsmotorerna blir för låg då hjulen utsätts för motstånd och H-bryggan hettar upp. För att åtgärda detta byttes H-brygga, kopplingar och motstånd men resultat förblev det samma. Trots mycket analyserande av kopplingsschemat kunde problemet inte åtgärdas och problemet ligger troligtvis i den gamla utrustningen snarare än ett misstag i projektet.

Anton Forssell
Stephanie Gardner
Felix Lindell

5 Resultat

Resultatet av projektet är en robot med truckfunktion vars hjul kan styras med en IR-signal men som inte kan drivas på mark. Truckfunktionen kan lyfta och sänka en ej påmonterad truckgaffel. Detta demonstreras genom att vifta en påmonterad flagga. Överlag bedöms projektet som framgångsrikt.

6 Källförteckning

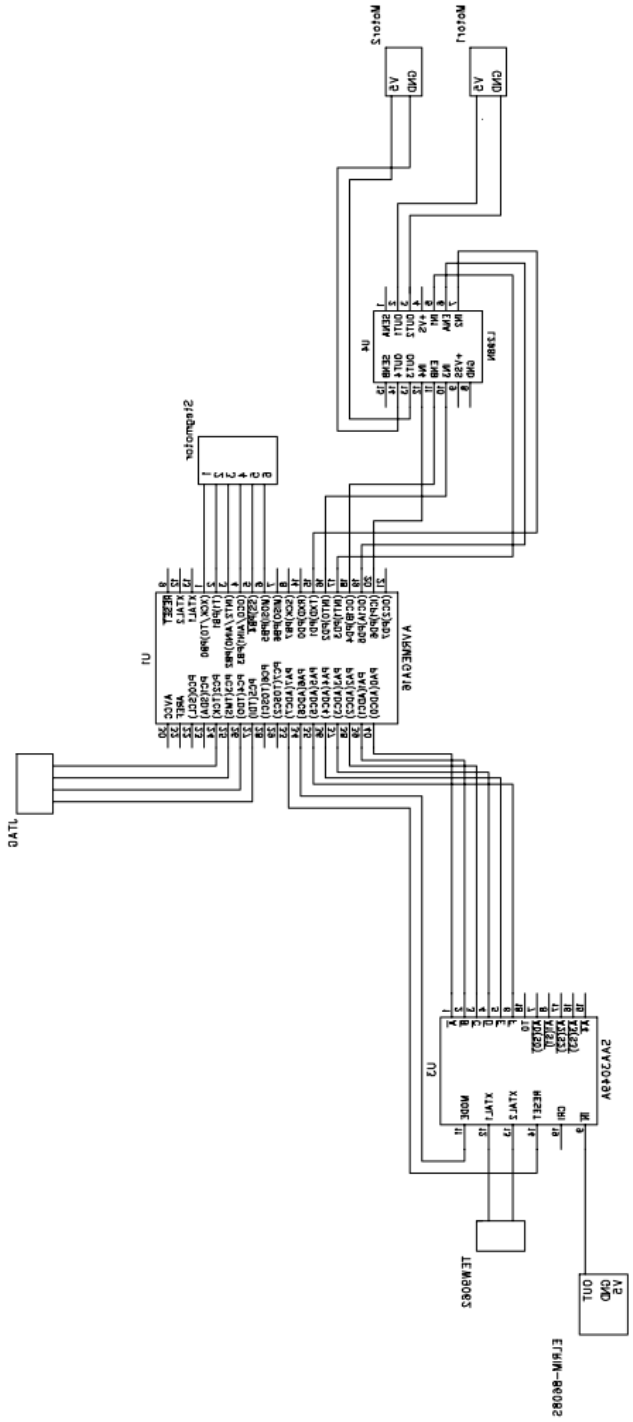
ATMEL (u.d.) ATmega16 Datasheet. Hämtat från:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf

Stegmotorstekning, All motion technology. Hämtat från:
http://www.allmotion.se/files/pdf/AMT_stegmotorteknik.pdf

H-brygga, KTH. Hämtat från:
http://www.ict.kth.se/courses/IL1390/elmotorstyr/hbrygga/

Anton Forssell
Stephanie Gardner
Felix Lindell

Appendix A – Kopplingsschema



Anton Forssell
Stephanie Gardner
Felix Lindell

Appendix B – Källkod

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

#define ENL1 _BV(PD5) // Enable left på
#define ENL0 ~_BV(PD5) // Enable left av
#define ENR1 _BV(PD4) // Enable right på
#define ENR0 ~_BV(PD4) // Enable right av
#define L_F1 _BV(PD3)  // Left motor fram
#define L_F0 ~_BV(PD1)
#define R_F1 _BV(PD6)  // Right motor fram
#define R_F0 ~_BV(PD2)
#define R_B1 _BV(PD2) // Right motor bak
#define R_B0 ~_BV(PD6)
#define L_B1 _BV(PD1) // Left motor bak
#define L_B0 ~_BV(PD3)
#define OFF 0x00 // Båda motorer av

#define ONE 0x3E //hard left
#define TWO 0x3D //forward
#define THREE 0x3C //hard right
#define FOUR 0x3B //left
#define FIVE 0x3A //backward
#define SIX 0x39 //right
#define SEVEN 0x38 //sänker gaffel
#define EIGHT 0x37 //viftar flagga
#define NINE 0x36 //höjer gaffel
#define NO_SIGNAL 0x3F //ingen signal


volatile uint8_t count;

void main(void)
{
                init();  // ställer in alla startvärden
                sei();  //tillåt interrupt

   while(1)
                {

                }
   return;
}
/* Här ställs alla startvärden in
*/

void init()
{
                PORTA |= _BV(PA7);
                PORTA &= ~_BV(PA7);
```

Anton Forssell
Stephanie Gardner
Felix Lindell

```
    DDRA = 0xC0;  //decoder, mode och reset utportar
                DDRB = 0xFF; //stegmotor, utportar
                DDRD = 0xFF;  //till h-brygga
                TCNT0 = 0x00; //Reset timer 0
                TCCR0 = 0x04; //Sätter prescaler FCPU/256 till timer 0
                TIMSK|= _BV(TOIE0);// Enable timer 0 & 1 overflow interrupt.
                count = 0;


}

/* Ett interrupt genereras vid timer overflow
*/

ISR(TIMER0_OVF_vect)
{
                count++;
                switch (PINA)
                {
                                case ONE:
                                                rightForward(); // Hard left
                                                leftBackward();
                                                break;
                                case TWO:
                                                rightForward(); // Forward
                                                leftForward();
                                                break;
                                case THREE:
                                                leftForward(); // Hard right
                                                rightBackward();
                                                break;
                                case FOUR:
                                                rightForward(); // Left
                                                leftSlowForward();
                                                break;
                                case FIVE:
                                                rightBackward(); // Backward
                                                leftBackward();
                                                break;
                                case SIX:
                                                leftForward(); // Right
                                                rightSlowForward();
                                                break;
                                case SEVEN:
                                                forkUp(); // Sänker gaffel
                                                break;
                                case EIGHT:
                                                flagWave(); // Viftar på flagga ;)
                                                break;
                                case NINE:
                                                forkDown();// Höjer gaffel
```

Anton Forssell
Stephanie Gardner
Felix Lindell

```
                                        break;
                                case NO_SIGNAL:
                                        enginesOff();
                                        break;
                                default:
                                        enginesOff();
                        }

                        if(count == 21)
                        {
                        count = 0;
                        }


}

/* Sätter bits på PORT D så att vänster hjul kör framåt.
Regelerar hastigheten genom att ändra på Enable för vänster motor
*/

void leftForward()
{
                        PORTD |= L_F1;
                        PORTD &= L_F0;
                         if(count == 21)
                         {
                                        PORTD &= ENL0;
                         }
                         else if(count == 1)
                         {
                                        PORTD |= ENL1;
                         }
                         else {
                         }
}

/* Sätter bits på PORT D så att höger hjul kör framåt.
Regelerar hastigheten genom att ändra på Enable för höger motor
*/

void rightForward()
{
                        PORTD |= R_F1;
                        PORTD &= R_F0;
                        if(count == 21)
                        {
                                        PORTD &= ENR0;
                        }
                        else if(count == 1)
                        {
                                        PORTD |= ENR1;
```

Anton Forssell
Stephanie Gardner
Felix Lindell

```
		}
		else {
		}
}

/* Sätter bits på PORT D så att vänster hjul kör bakåt.
Regelerar hastigheten genom att ändra på Enable för vänster motor
*/

void leftBackward()
{
		PORTD |= L_B1;
		PORTD &= L_B0;
		if(count == 21)
		{
			PORTD &= ENL0;
		}
		else if(count == 1)
		{
			PORTD |= ENL1;
		}
		else {
		}
}

/* Sätter bits på PORT D så att höger hjul kör bakåt.
Regelerar hastigheten genom att ändra på Enable för höger motor
*/

void rightBackward()
{
		PORTD |= R_B1;
		PORTD &= R_B0;
		if(count == 21)
		{
			PORTD &= ENR0;
		}
		else if(count == 1)
		{
			PORTD |= ENR1;
		}
		else {
		}
}

/* Sätter bits på PORT D så att vänster hjul kör framåt.
Regelerar hastigheten genom att ändra på Enable för vänster motor
*/

void leftSlowForward()
{
```

Anton Forssell
Stephanie Gardner
Felix Lindell

```
		PORTD |= L_F1;
		PORTD &= L_F0;
		if(count ==  1)
		{
			PORTD |= ENL1;
		}
		else if(count == 11)
		{
			PORTD &= ENL0;
		}
		else {
		}
}

/* Sätter bits på PORT D så att höger hjul kör framåt.
Regelerar hastigheten genom att ändra på Enable för höger motor
*/

void rightSlowForward()
{
		PORTD |= R_F1;
		PORTD &= R_F0;
		if(count ==  1)
		{
			PORTD |= ENR1;
		}
		else if(count == 11)
		{
			PORTD &= ENR0;
		}
		else {
		}
}

/* Stänger av båda motorerna
*/

void enginesOff()
{
		PORTD = OFF;
}

/* Genererar en sekvens som får stegmotorn att föra gaffeln nedåt
*/

void forkDown()
{
		switch (count)
		{
			case 1:
				PORTB = 0x1B;
```

Anton Forssell
Stephanie Gardner
Felix Lindell

```
                                        break;
                                case 6:
                                        PORTB = 0x33;
                                        break;
                                case 11:
                                        PORTB = 0x36;
                                        break;
                                case 21:
                                        PORTB = 0x1E;
                                        break;

                }
}

/* Genererar en sekvens som får stegmotorn att föra gaffeln uppåt
*/

void forkUp()
{
                switch (count)
                {
                                case 1:
                                        PORTB = 0x1E;
                                        break;
                                case 6:
                                        PORTB = 0x36;
                                        break;
                                case 11:
                                        PORTB = 0x33;
                                        break;
                                case 21:
                                        PORTB = 0x1B;
                                        break;

                }
}

/* Genererar en sekvens som får stegmotorn att föra gaffeln nedåt och sedan uppåt
*/

void flagWave()
{
                switch (count)
                {
                                case 1:
                                        PORTB = 0x1B;
                                        break;
                                case 4:
                                        PORTB = 0x33;
                                        break;
                                case 7:
```

Anton Forssell
Stephanie Gardner
Felix Lindell

```
                    PORTB = 0x36;
                    break;
                case 10:
                    PORTB = 0x1E;
                    break;
                case 12:
                    PORTB = 0x1E;
                    break;
                case 15:
                    PORTB = 0x36;
                    break;
                case 18:
                    PORTB = 0x33;
                    break;
                case 21:
                    PORTB = 0x1B;
                    break;
            }
}
```