LTH

EITF11-Digitalaprojekt

Projektrapport

2011-05-10

Projekt utfört av Per Gifvars och Johan Linåker

http://www.eit.lth.se/fileadmin/eit/courses/edi021/Sammanfattning/2011/LP4/Grupp3/html.html

Abstract

This report covers the project of building a robot with the ability of moving forward and avoiding obstacles by turning away from them. The project has been carried out by two students of Industrial Engineering and Management and had the initial aim of carrying a flamethrower. However, the flamethrower was never realized, for several reasons. In this project we have been using some standard robot components such as a turning servo, proximity sensor, DC-motor and H-bridge.

Inledning

Projektet inleddes med målsättningen att bygga något som i första hand var roligt och i andra hand något som utnyttjade den specialkompetens som fanns inom projektgruppen. Då ett robotbygge var något som både intresserade projektgruppen och som den ansåg vara ett nåbart mål. För att addera ytterligare en dimension på projektet sattes målsättningen att den brandingenjörskompetens som gruppen besatt skulle utnyttjas genom att bygga en fjärrstyrd eldkastare att montera på roboten. Detta skulle givetvis göras i mån av tid och det målet fick dessvärre prioriteras bort vartefter projektet fortlöpte och tiden blev knapp.

Syftet med projektet var att åskådliggöra hur digitala produkter fungerar och är uppbyggda samt ge prov på hur komplexa digitala produkter är som vi i dagens samhälle tar för givet. Projektet har även utvecklat projektdeltagarnas kunskaper i såväl Digitalteknik, Ellära samt Datorteknik och programmering.

Kravspecifikation

Primära krav

- Köra rakt fram
- Svänga höger och vänster (45 grader åt vardera håll)
- Stanna var ≈30:e cm och undersöka avstånd
- Behåll aktuell färdriktning om inget hinder observeras
- Byta håll om avstånd < 0,5 m i aktuell färdriktning
- Välja riktning efter var avståndet är störst. I det fall när största avstånd kan uppmätas åt både höger och vänster – slumpa.
- Om roboten fastnar i ett hörn, sväng 180 grader
- Roboten skall ha en brytare för spänningsmatning på/av

Sekundära krav

- Radiostyrd signal för extra tillbehör
- Bygga robot med fyra individuellt drivna hjul samt pivotstyrning

Teori

Processor

Den mikroprocessor vi använt oss av är en Atmel AVR ATMega16 på 8MHz (http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf), vilken är en billig mikroprocessor mer 40 pinnar, programmerbart minne som lämpar sig bra för den här typen av enklare digitala tillämpningar. De drivs av en spänning på 5V och är mycket energisnåla. I projektet använder vi oss dels av 4 st. I/O pinnar för att styra varje elmotor. Varje motor kräver 2 I/O pinnar styck och vi använder oss av PD3-PD6. Därtill använder vi oss av PD7, som har en inbyggd funktion för PWM (Pulse Wide Modulation) för att rikta servon. Sist men inte minst använder vi pinne XXX för att ta emot de analoga signalerna från avståndssensorn. Detta för att port A är den port som har en inbyggd fnktion för analog till digital omvandling, så kallad ADC. Detta givetvis för att vi får en analog signal från givaren.

Servo

Vi monterade avståndsgivaren på en Parallax Standard Servo (#900-00005) för att kunna vrida den och på så sätt mäta avstånden inte bara rakt fram från roboten utan även snett fram. Servons riktning ges av olika långa pulser där periodtiden inte får överstiga 20 ms. Därför använde vi oss av pinne PD7, som nämnts tidigare.

H-brygga

För att kunna styra motorerna med en starkare ström än signalen från processor gav, använde vi oss av en L298 H-brygga som egentligen är en dubbel H-brygga. En för varje motor.

Avståndsgivare

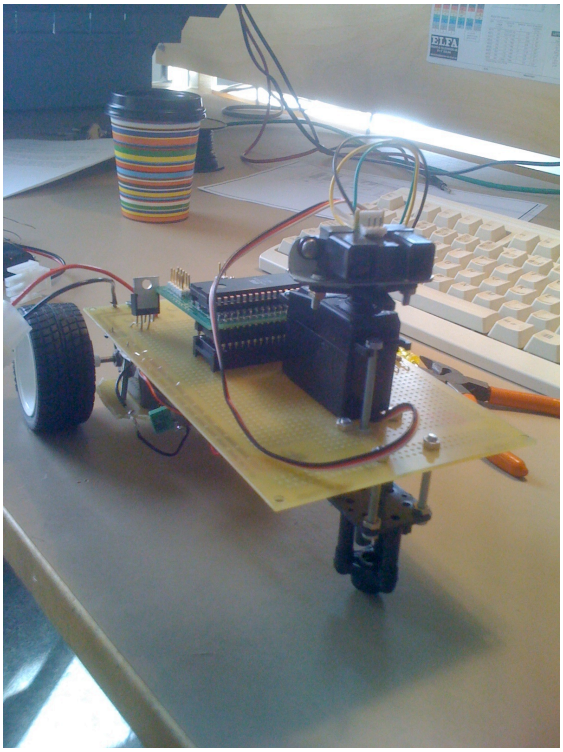
Den avståndsgivare som monterades på servon för att mäta avståndet till omgivningen var en GP2Y0A02YK från SHARP. Den har förmåga att bedöma avstånd mellan intervallet 20 – 150 cm. Avståndsgivaren är direktkopplad till en ADC port på mikroprocessorn som omvandlar den analoga signalen till digital och sedan tolkar den. Avståndsgivaren är satt i så kallat free running mode som innebär att den hela tiden gör avlsningar.

Motorer

För att driva roboten framåt användes två stycken likströmsmotorer som vardera drevs av 6V, vilka även styrdes från H-bryggan.

Spänningsregulator

Då vår spänningskälla var på 6V men samtliga digitala komponenter ej tålde mer än 5V var vi tvugna att använda oss av en spänningsregulator som omvandlade 6V till 5V. Den komponenten vi använde oss av var en



Utförande

Projektet delades inledningsvis upp I tre huvudfaser. Planeringsfasen då kravspecifikationen skrevs och kopplingsschema gjordes, Monteringsfasen då vi fysiskt monterade alla komponenter och kopplade samman dessa samt programmerings och testfasen då arbetet med att dels få alla komponenter att tala med varandra gjordes samt utveckla styrprogrammet och kalibrering gjordes.

Planeringsfasen inleddes med att skriva en kravspecifikation som täckte in de mål vi hade med roboten. Att så tidigt I projektet avgöra vad som var rimliga mål att sätta och vad som inte är det, var något vi upplevde som en svårighet. Detta på grund av projektdeltagarnas avsaknad av erfarenhet av liknande projekt. De krav vi inledningsvis ställde upp, blev utmed projektets gång reviderade på så sätt att kraven kunde skärpas och funktionaliteten på roboten ökas. Exempelvis sattes först kravet att roboten skulle stanna var 30:e cm för att mäta avståndet och sedan välja riktning. Då projektdeltagarnas kunskap om hårdvaran ökade insåg man att det inte fanns någon praktisk mening med det utan att göra kontinuerliga avståndsmätningar och val i realtid var både mer funktionellt och lättare programmeringsmässigt.

För att styra motorerna kopplades fyra I/O signaler från PD3-PD6 på processorn till H-bryggan som i sin tur reglerade spänningen på 6V till likströmsmotorerna. För att reglera hastigheten på motorerna användes en for-loop i programmet som skickade signal med jämna intervaller. Vi valde även att koppla in 2st lysdioder som lyste om roboten svänger höger eller vänster. Den praktiska funktionen av detta är tämligen begränsad, men det var projektgruppens samlade bedömning att en robot bör ha lysdioder och detta var det bästa sättet att använda dessa. Lysdioderna kopplades med signal från PB0,PB2 via varsitt 500 ohms motstånd och monterades längst fram på roboten.

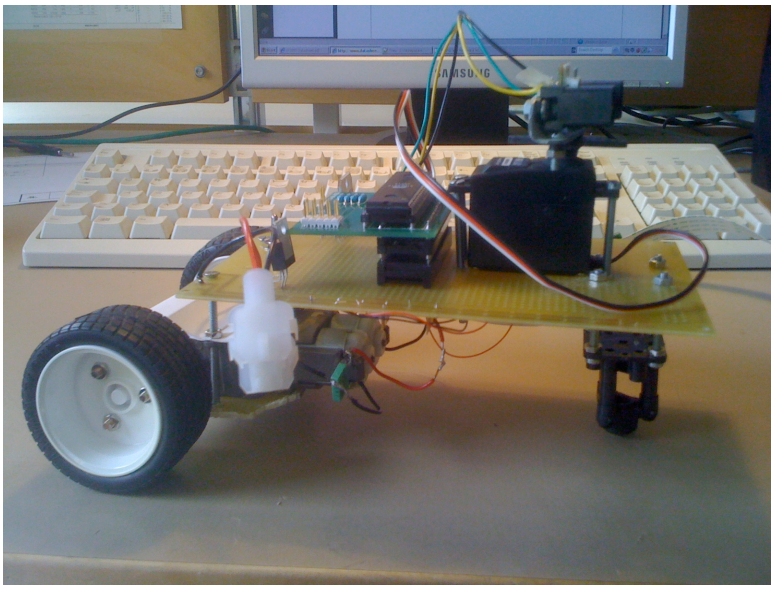
Det servo vi använde oss av var en Parallax Standard Servo (#900-00005), viken styrs av en PWM signal vilket står för Pulse Wide Modulation och innebär att riktningen på servon ges av hur lång signalpulsen är under en given periodtid. För att åstadkomma detta använde vi processorns utgång med förberedd funktion för PWM, nämligen pinne PD7. Maximalt tillåten periodtid för att servon skall behålla sitt läge är 20ms, men för att ha marginal använder vi en periodtid på 16ms.

GP2Y0A02YK från SHARP var den avståndssensor vi monterade på servot för att vara robotens ögon. Från sensorn avges en analog signal i spänningsintervallet 1,8-2,3V, beroende på avstånd. För att ta han om signalen kopplades den till pinne PA0, vilken har en inbyggd funktion för analog-till-digital omvandling(ADC – Analog to Digital Converter). I funktionen för ADC använder vi oss av en referensspänning på 2,56V vilket ger oss en bra upplösning på avståndssignalen.

Koden har sedan utvecklats succesivt med fokus på de olika momenten i robotens rörelsemönster och kalibrerats grov efter empiriska test. Det finns mycket förbättringspotential i roboten och gruppen har en plan för hur den skall kunna förbättras i framtiden, däremot är funktionaliteten tillfredsställande och förbättringarna kan därmed ses som detaljer.

Resultat

Jämfört med vad som stipulerats i den ursprungliga kravspecifikationen kan det konstateras att projektets resultat uppfyller i stort sett alla de primära kraven, dock med en del justeringar i vissa fall. Tyvärr har kravet om att roboten ska svänga 180 grader när den fastnat i ett hörn ej lyckats uppfyllas på grund av tidsbrist. De sekundära kraven har helt och hållet åsidosatts.



Roboten startas med hjälp av en switsch som är belägen i den bakre delen. Motorerna startar och börjar driva hjulen framåt. Servot roterar ir-sensorn i ett 90-gradigt intervall med start från höger. När sensorn uppfattar att avståndet i robotens aktuella riktning är mindre än 0,5 meter kommer roboten att svänga 45 grader åt höger eller vänster beroende på vilket håll som har störst avstånd uppmätt.

Källförteckning

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

AVR Libc. (u.d.). AVR Libc User Manual 1.6.7. Hämtat från http://www.eit.lth.se/fileadmin/eit/courses/edi021/avr-libc-user-manual/index.html

National Semiconductor. (u.d.). LP3852/LP3855 1.5A Fast Response Ultra Low Dropout Linear Regulators Datasheet. Hämtat från http://www.national.com/ds/LP/LP3852.pdf

PARALLAX. (u.d.). Parallax Standard Servo (#900-00005) Datasheet. Hämtat från http://www.parallax.com/Portals/0/Downloads/docs/prod/motors/900-00005StdServo-v2.1.pdf

SHARP. (u.d.). Long Distance Measuring Sensor GP2Y0A02YK Datasheet. Hämtat från http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a02_e.pdf

STMicroelectronics. (u.d.). Dual full-bridge driver - L298 Datasheet. Hämtat från http://www.st.com/stonline/books/pdf/docs/1773.pdf

Bilaga 1: Källkod

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

#define ALL_FORWARD _BV(PORTD5)|_BV(PORTD3)|_BV(PORTD2)|_BV(PORTD1)
#define A_FORWARD 0x0A
#define B_FORWARD 0x24
#define A_BACKWARD 0x12
#define B_BACKWARD 0x44
#define PRESCALE_TICK 6103 //TAR PROCESSORN 0,1 mSEK ATT RäKNA UPP TILL
DETTA VÄRDET
#define SERVO_LEFT 26
#define SERVO_RIGHT 10
#define SERVO_MID 18

char servoDirMemory;
unsigned short dist;
unsigned short currServoDir;
int turnCount = 0;

void main(void) {

	init();
	sei();

	while(1) {
		dist = ADCH;
		currServoDir = OCR2;
		forward(ALL_FORWARD);
		if(dist > 150)
		{
			if(currServoDir < 18)
			{
				turn(B_FORWARD);
			} else if(currServoDir >=18)
			{
				turn(A_FORWARD);
			}
		}
	}
	return;
}
```

```
	void init()
	{

		DDRD =0xFF;
		DDRB = 0xFF;
		TCCR0 = 0b00000101;   //clk(I/O) / 1024 (from prescaler)
		int m1 = 200;
		int m2 = 200;
		int servo = SERVO_MID;     //Ställer servon i miten
		TCNT0 = 0xFF;                    //Resettar 8-bitars timern
		//OCR1A = PRESCALE_TICK;
		TIMSK = _BV(TOIE0)|_BV(TOIE1);   //Timer0 overflow interrupt, Timer1
overflow interrupt
		ADCSRA = _BV(ADEN)|_BV(ADSC)|_BV(ADATE);         //Enable Analog-to-
Digital-omvandling, Autotriggering ADC enable
		ADMUX = 0b11100000;         //Set ref to 2.56V and set ADC0
		servoDirMemory = 'r';
		TCCR1B = _BV(CS12)|_BV(CS10);

		ASSR=0b00;
		TCCR2=0b01100110;                //Prescale 256, PWM phasecorrect, set
on upcounting & clear on downcounting
		TCNT2=0xFF;
		OCR2=SERVO_LEFT;
		//PORTD = ALL_FORWARD;


	}

	ISR(TIMER2_OVF_vect)            // Avbrott för servo-PWM
	{

		TCNT2 = 0;

	}

	ISR(TIMER0_OVF_vect) //Varje avbrott
	{

		turnCount++;
		TCNT0 = 0;                  //Nollst%ller r%knare för servot

			if (OCR2 < SERVO_LEFT && servoDirMemory == 'r') {
```

```
                OCR2++;
            }
            else if (OCR2 == SERVO_LEFT) {
                OCR2--;
                servoDirMemory = 'l';
            }
            else if (OCR2 > SERVO_RIGHT && servoDirMemory == 'l') {
                OCR2--;
            }
            else if (OCR2 == SERVO_RIGHT) {
                OCR2++;
                servoDirMemory = 'r';
            }

}

void turn(int dir){
      turnCount = 0;

      while(turnCount < 40) {
            if(dir == B_FORWARD)
            {

                        PORTB = _BV(PORTB0);
            } else if(dir == A_FORWARD){
                        PORTB = _BV(PORTB1);
            }
            forward(dir);
      }
      PORTB = 0;
}


void forward(int dir){
      for(int k = 0; k < 20; k++)
      {
            if(k < 18){
                PORTD = dir;
            } else {
                PORTD = 0;
            }
      }
}
```

Bilaga 2: Kopplingsschema