LUNDS UNIVERSITET

LTH

LUNDS TEKNISKA HÖGSKOLA

Autonom robot - Rapport

2022-05-23

Digitala System (EITA15)

Handledare: Bertil Lindvall

Grupp 7

Christoffer Axtegen, Karl Wedberg, Fereshta Nedjat, Alex Björklund

Sammanfattning

Denna rapport handlar om konstruktionen av en prototyp som en del av kursen Digitala System (EITA15). Prototypen är skapad av studenter vid Lunds Tekniska Högskola. Syftet med att skapa en prototyp är att använda den kunskap som erhållits inom datorteknik under kursens gång.. För att uppnå detta programmerades en mikroprocessor som styr de olika komponenterna.

Prototypen i detta fall är en robot som med hjälp av en fotoelektrisk sensor kan detektera väggar och på så sätt lösa en labyrint. Sensorn skickar en signal som säger hur långt ifrån närmaste objekt befinner sig. Den fotoelektriska sensorn skickar denna signal till mikroprocessorn som läser av signalen av med hjälp av A/D-omvandling.
Roboten drivs via en h-brygga kopplad till två motorer som i sin tur styr ett par larvfötter. För att välja riktning på sensorn används en servo som styrs med PWM.
Resultatet av projektet blev en robot som kan lösa en labyrint med hjälp av information om väggar till höger och rakt framför.

Nyckelord

Digitala system, Prototyp, Labyrint, Robot

Abstract

This report concerns the construction of a prototype as a part of the course Digitala Systems (Digital Systems)(EITA15). The prototype is created by students at Lunds Tekniska Högskola. The purpose of the construction of a prototype is to use the knowledge acquired within computer science throughout the course. To achieve this, the prototype was programmed using a microprocessor and constructed using programmable components.

The prototype described in this report is a robot that, with the help of a photoelectric sensor, can detect walls, which it uses to solve a labyrinth. The photoelectric sensor sends a signal to the microprocessor that is converted using an AD converter. The movement of the robot is controlled with an H-bridge that controls two motors, which in turn controls the wheels. To turn the servo that controls the direction of the sensor, PWM is used.
The result of the project was a robot capable of solving a labyrinth using information about surrounding walls.

Key words

Digital systems, Prototype, Labyrinth, Robot

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Detta projekt var den slutliga delen av kursen Digitala System (EITA15). Syftet var att inom en grupp bygga en fungerande prototyp genom att använda det som lärts ut under kursens gång. Vilken typ av prototyp valdes ut efter överläggning inom gruppen samt godkännande av handledaren. Till detta projekt beslutades det för konstruktionen av en robot kapabel att lösa en labyrint.

1.2 Syfte

Genom att skapa en prototyp på egen hand så är syftet att man ska få en utökad förståelse för programmering i C, samt få en inblick i hur man arbetar på ett ingenjörsmässigt sätt. Genom konstruktionen av en robot, så hoppas gruppen att så många aspekter som möjligt från kursen ska användas och på så sätt testa gruppens kunskaper.

1.3 Problemformulering

Under processen att skapa en prototyp så måste gruppen kunna samarbeta och dela upp arbetet så projektet blir klart till deadline och med att alla gruppmedlemmar bidrar till arbetet på ett bra sätt. Att gruppen kommer överens om en prototyp som samtliga gruppmedlemmar vill arbeta på är också viktigt att komma fram till. Ett annat problem som behövde en lösning var att lista ut hur konstruktionen av prototypen ska gå till. Vilka komponenter behövs, och hur fungerar de? Och hur ska programmeringen gå till för att få dessa komponenter att samarbeta i prototypen för att den ska fungera som tänkt? Detta är problem som gruppen måste lösa för att konstruktionen av en prototyp ska vara möjlig.

1.4 Avgränsningar

Då projektet är tänkt att utföras under läsperiod 4 av kursen så var det rimligt att kompromissa med vad man vill utföra och hur mycket tid som fanns utföra projektet på. Sedan så var val av komponenter begränsat, då komponenter som fanns till hands var prioriterade. Det var med detta i åtanke som prototypen skapades.

1.5 Rapportens disposition

Denna rapport består av sju delar. Del två handlar om prototypens beståndsdelar, dess komponenter samt mjukvaran bakom den. Del tre rör planering och konstruktion av prototypen. Del fyra handlar om resultatet av projektet. Rapporten avslutas med en

diskussionsdel där eventuella slutsatser tas upp. I de två sista delarna finner man referenser samt appendix till rapporten.

2. Teknisk bakgrund

2.1 Produktbeskrivning

Detta projekt är en robot som med hjälp av en sensor kan avgöra om, och var den har väggar runt om sig. Roboten kan manövrera med hjälp av larvfötter som i sin tur drivs av två motorer och en h-brygga. Med hjälp av detta samt den så kallade *right-hand rule* metoden kan roboten lösa en labyrint.

Roboten avgör om den har eller inte har en vägg framför sig respektive till höger om sig, och med hjälp av denna information avgör den vilken riktning den ska röra sig i. De finns fyra olika fall:

- En vägg till höger och ingen framför:
  Roboten kör rakt framåt.

- Vägg till höger och framför:
  Roboten svänger till vänster

- En vägg framför men ingen till höger:
  Roboten svänger till höger.

- Ingen vägg detekterad:
  Roboten svänger till höger och kör sedan rakt framåt.

Med denna logik kan roboten (med tillräckligt mycket tid till sitt förfogande) lösa en sammanhängande labyrint.

2.2 Hårdvara

Hårdvara som använts till projektet är följande:

- **ATMega1284** - 8-bitars AVR microcontroller. Programmerades för att styra beteendet hos de andra komponenterna.
- **DRV88** - H-brygga. Används för styra motorerna som driver hjulen.
- **Sharp 2Y0A02** - Fotoelektrisk sensor. Mäter avstånd till närmsta objekt med hjälp av infrarött ljus
- **Servo** - Styr sensorns riktning.
- **JTAG** - Debugger. Användes vid programmeringen
- Hjul med inbyggd DC motorkomponent

För mer detaljerad information, se *kopplingsschema* (7.1).

2.3 Mjukvara

För programmering av roboten så har Atmel Studio 7 används. Kodningen har skett i C. Design av kopplingsschema skedde i KiCad.

Vilket håll servon riktar sensorn åt bestäms genom att skicka en fyrkantsvåg vars pulsbredd motsvara en position hos servon, så kallad Pulse-Width Modulation (PWM). Den fotoelektriska sensorn används för att avgöra om en vägg är framför den. Sensorn skickar sedan en analog signal som omvandlas med en A/D-omvandlare till ett värde som kan hanteras av mikroprocessorn. Med hjälp av detta kan roboten avgöra hur långt bort ett objekt befinner sig.

Styrning av roboten sker med en h-brygga som sänder vidare signalen till motorerna så att en sida antingen drivs framåt eller bakåt. På detta sätt kan roboten röra sig framåt och bakåt samt svänga höger eller vänster beroende på riktningen på vardera sida av larvfötterna.

För en mer omfattande överblick över koden se *källkod* (7.2).

3. Genomförande

3.1 Planering

Den första perioden av arbetetet bestod av en planeringsfas, då vilket typ av prototyp som skulle konstrueras bestämdes. Tidigt i denna process kom gruppen fram till att skapandet av en robot som kan undvika hinder var intressant, och efter övervägande bestämdes det att en robot som kan lösa labyrinter var det som skulle skapas.

Innan konstruktionen av roboten kunde börja behövdes först processen planeras ut. Det första som gjordes var skapandet av en kravspecifikation (3.1.1), där de olika krav gruppen ställde på prototypen noterades.

Komponenter behövdes sedan väljas ut för att kunna konstruera roboten. Att motorer och sensorer behövdes var självklart, men vilken typ var något som behövdes undersökas. Till en början var tanken att använda en ultraljudssensor för att avgöra om det fanns väggar intill roboten. Meningen var att sensorn skulle skicka ut en ljudvåg som den sedan fick tillbaka

efter en viss tid. Tanken var då att tiden som det tog att få tillbaka vågen kunde mätas och med hjälp av det så kunde man beräkna hur långt borta väggen låg. Efter diskussion med handledare insågs det att det fanns enklare sätt att detektera väggar och det beslutades istället att använda en fotoelektrisk sensor. Den fotoelektriska sensor som tilldelades var redan ansluten till en servo. På detta sätt kunde användningen av två statiska sensorer undvikas för att istället använda en sensor som kan detektera väggar framför samt till höger om roboten.
För manövrering av roboten valdes ett hjulsystem med motorer redan sammankopplade. Detta hjulsystem består av två motorer, som driver vardera sida av hjulen. För att enklare kunna styra hjulen så används en h-brygga som motorprocessor. När komponenterna var bestämda kunde ett kopplingsschema konstrueras (7.1).

3.1.1 Kravspecifikation

Under planeringsfasen av roboten skapades en kravspecifikation bestående av 11 krav. Huruvida kraven blev uppfyllda diskuteras vidare i del 5 *slutsats*.

1. Roboten ska autonomt kunna följa en vägg belägen till höger relativt dess färdriktning.
2. ~~Roboten ska kunna färdas i en hastighet på (hastighet).~~
3. ~~Alla fyra hjulen ska vara drivande.~~
4. Alla hjul ska ha en statisk riktning.
5. ~~För att orientera sig ska två sensorer insamla data: en sensor riktad framåt och en sensor riktad åt höger.~~
6. Då en vägg detekteras framför roboten ska den svänga på stället på sådant sätt att väggen i slutet på rörelsen befinner sig till höger relativt robotens riktning.
7. Efter att roboten har startat ska den vänta i fem sekunder innan den börjar avläsa sina sensorer och navigera.
8. ~~Då inga väggar detekteras ska roboten svänga till den upptäcker en vägg eller tills en godtycklig tidsperiod har förflutit.~~
9. Roboten ska kunna navigera oavsett yttre ljusförhållanden.
10. ~~Med hjälp av en halleffektsensor ska roboten kunna stanna där den detekterar en magnet.~~
11. ~~Lampor ska visa ifall roboten rör sig framåt eller bakåt.~~

3.2 Konstruktion

Då komponenterna var samlade, påbörjades processen att testa dem separat för att säkerställa att de alla fungerade. Till en början kopplades komponenterna in enskilt till ATmegan med hjälp av en kopplingsplatta innan något permanent gjordes. På så sätt så fanns det mer flexibilitet att ändra saker. När alla komponenter var kontrollerade och kod för att avläsa/driva dem hade skrivits så påbörjades processen att sätta ihop roboten. För detta användes lödning samt virning med en virpistol. Kopplingsplattan med komponenter fästes

vid hjulsystemet och roboten testades när allt var på plats. Testning av roboten skedde när den var ansluten till en spänningskub.

3.3 Programmering

Diskussion om hur programmeringen av roboten skulle gå tillväga gick mestadels till på distans, och en klar bild över hur logiken bakom roboten skulle se ut kom ganska snart på plats. För programmeringen användes kunskapen om kodning av hårdvara från kursens laborationer samt tidigare kunskap inom C. Som nämnt tidigare skedde programmeringen av roboten i Atmel Studio 7, men en stor del av koden skrevs separat i andra kompilatorer/ordbehandlingsprogram och flyttades sedan över till huvudprogrammet i Atmel Studio.

4. Resultat

Resultatet av projektet blev en robot som med hjälp av en sensor kan avgöra om en vägg befinner sig framför eller till höger om den. Genom att använda denna information kan roboten lösa en labyrint genom att använda *right-hand rule*. Prototypen styrs genom att driva de två motorerna antingen framåt eller bakåt för att kunna köra rakt fram, eller att svänga vänster och höger. När roboten slutar upptäcka en vägg till sin högersida, så beslutar den sig för svänga åt höger, där den sedan följer nästkommande vägg. Skulle roboten upptäcka en vägg framför sig svänger den till vänster och följer den väggen. På så sätt ser roboten till att det alltid finns en vägg till höger, vilket är instrumentalt för lösning av en labyrint med *right-hand rule*.

5. Slutsats och diskussion

Ett antal krav på prototypen blev inte uppfyllda eller behövdes ändras under processens gång. Till en början var tanken att ställa krav på hur fort roboten skulle röra sig, men under konstruktionen av prototypen bestämdes det att det inte var nödvändigt att ha en specifik hastighet. Den kör med en konstant hastighet framåt, men att kontrollera den hastigheten var inte något som var nödvändigt för att roboten skulle fungera.

Det var också en tanke att en halleffektsensor skulle användas för att markera var roboten skulle stanna efter den tagit sig fram i labyrinten samt att lampor skulle användas för att visa om roboten rör sig framåt eller bakåt. Detta skulle göras i mån av tid, men beslutades att inte vara väsentligt.

De krav som ändrades var krav 3, 5 och 8. Istället för användningen av två statiska sensorer så användes istället en vridbar sensor. Detta blev mer effektivt än vad som hade planerats till en början. Så i detta fall blev det bättre än det krav som ställts. Tanken under planeringsfasen var också att fyra separata hjul, som alla var drivna skulle användas. Komponenten som användes istället bestod av två motorer som drev ett par larvfötter. Istället för roboten svänger tills den hittar en vägg när den inte detekterar någon runt om sig, bestämdes det att roboten istället skulle köra rakt fram tills den hittar en.

Gällande de problem togs upp i problemformuleringen, så löstes det mesta på ett bra sätt under processens gång. Val av vilken prototyp som skulle konstrueras framkom tidigt i processen och samtliga gruppmedlemmar var överens om att det var intressant att skapa en robot.

Valet av komponenter blev även det avklarat tidigt i processen, och gruppen bestämde fort vilken typ av komponenter som skulle användas tillsammans med handledaren. Det uppstod en del problem med komponenterna under slutfasen av projektet. Den fotoelektriska sensorn var olämplig för robotens behov då den har ett minimumavstånd på 20 cm och ett maximum på 150 cm. Det hade varit bättre med en sensor med en kortare min- och maxavstånd för prototypens ändamål. Sensorn som användes krävde otympliga avstånd för att vara pålitlig, då utsignalen för 2cm är det samma som utsignalen for 20cm.

Under processens gång skedde allt testning med en spänningskub, och all koden fungerade som den skulle när roboten var ansluten till denna. När roboten drivs med batterier hade det en negativ påverkan på hela kretsen. När detta används så fungerar inte servon som tidigare, sensorn ger annorlunda signaler, och styrningen av larvfötterna blev opålitlig.

Att roboten inte fungerade som tänkt med batterier beror antagligen på att strömförsörjningen inte höll sig på den nivå den behöver för att alla komponenter ska fungera. Detta hade kunnat lösas med spänningsreglering eller en separat strömförsörjning för processorn respektive de anslutna komponenterna, men detta fanns inte tillgängligt då problemen upptäcktes sent i projektet och kunde inte lösas innan rapportens deadline. Innan presentation av prototypen löstes detta genom att se till att potentiella glapp med kablar och batterier fixades så att spänningen kunde hållas vid 5V som roboten kräver.

Kodningen av prototypen var något som behövdes prövas fram. Datablad användes för att förstå hur komponenterna fungerade om det var tillgängligt. Mycket av förståelsen av hur kodningen skulle gå till kom ifrån laborationerna från kursen, speciellt när det gällde hur

signaler lästes in och skickades ut från ATmegans portar. Även kunskapen om hur A/D-omvandling och hur PWM fungerade kom från laborationerna.

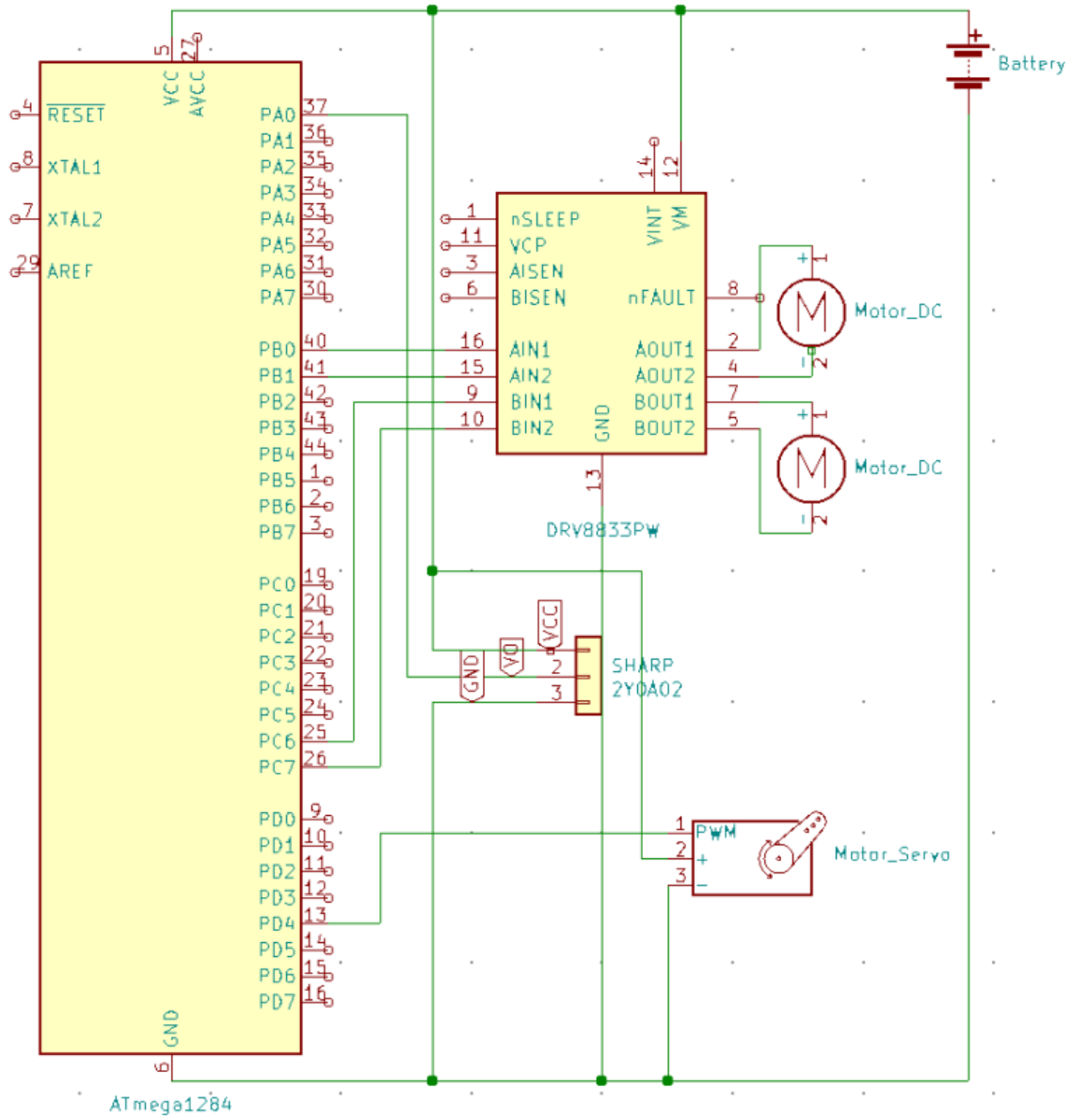
Då syftet med projektet var att använda den kunskap som erhållits inom datorteknik under kursens gång går det direkt att säga att erfarenheter från tidigare kursmoment har kommit till stor användning för att lösa de problem som uppstod.

6. Referenser

[1] Atmel, *8 bit AVR microcontrollers*, ATmega1284, Datasheet Complete
https://www.eit.lth.se/fileadmin/eit/courses/datablad/Processors/ATmega1284.pdf
[2] Sharp, *Long Distance Measuring Sensor*, GP2Y0A02YK, datablad, Okt 2016
https://www.ti.com/lit/ds/symlink/drv8833.pdf
[3] Texas Instruments, *Dual H-Bridge Motor Driver*, DRV8833, datablad, juli 2015
https://www.ti.com/lit/ds/symlink/drv8833.pdf?ts=1652890667731&ref_url=https%253A%252F%252Fwww.google.com%252F

7. Appendix

7.1 Kopplingsschema



7.2 Källkod

```
/*
 * Test1.c
 *
 * Created: 2022-05-02 13:37:07
 * Author : Grupp 7
 */

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

```
#include <stdint.h>
#include <util/delay.h>
#define F_CPU 16000000UL

void turn_sensor_forward(){

	for (int i = 0; i < 20; i++){
		PORTD = 0xFF;
		_delay_ms(2.23);
		PORTD = 0x00;
		_delay_ms(20);
	}
}

void turn_sensor_right(){

	for (int i = 0; i < 20; i++){
		PORTD = 0xFF;
		_delay_ms(1.35);
		PORTD = 0x00;
		_delay_ms(20);
	}
}

void drive_forward(){

	for (int i = 0; i < 35; i++){
		PORTB = 0b00000010;
		PORTC = 0b10000000;
		_delay_ms(5);
		PORTB = 0x00;
		PORTC = 0x00;
		_delay_ms(35);
	}
}

void drive_stop(){

	PORTB = 0x00;
	PORTC = 0x00;
}

void drive_right(){

   for (int i = 0; i < 30; i++){
	PORTB |= 0b00000010;
	PORTC |= 0b01000000;
	_delay_ms(5);
```

```
        PORTB = 0x00;
        PORTC = 0x00;
        _delay_ms(25);
        }
}

void drive_left(){

   for (int i = 0; i < 30; i++){
        PORTB |= 0b00000001;
        PORTC |= 0b10000000;
        _delay_ms(5);
        PORTB = 0x00;
        PORTC = 0x00;
        _delay_ms(25);
   }
}

void setup_adc0(){

        ADCSRA |= (1<<ADEN);
        ADCSRA |= (1<<ADPS0) | (1<<ADPS1) | (1<<ADPS2);
        ADMUX |= (1<<REFS0) | (1<<REFS1);
        DDRA &=0b00000001;

}

uint16_t adc_read(){

        PRR0 = 0b00000000;
        ADCSRA |= (1<<ADSC);
        while(ADCSRA&(1<<ADSC)){}
        return ADC;

}

uint16_t sensor_read(){

        if (adc_read() >= 0x0027){
                return 0xFFFF;
        } return 0x0000;

}

void setup_portDirections(){

        DDRD = 0b01100000;
        DDRA = 0b00000001;
```

```
	DDRB = 0b00000011;
	DDRC = 0b11000000;
	PORTB = 0b00000000;
	PORTC = 0b00000000;
}

int stat = 0b00;

int main(void){

	setup_portDirections();
	setup_adc0();

	_delay_ms(5000);

	turn_sensor_forward();



	while(1){


		stat = 0b00;


		if(sensor_read() == 0xFFFF){
			stat = stat | 0b10;
		}

		turn_sensor_right();
		if(sensor_read() == 0xFFFF){
			stat = stat | 0b01;
		}

		turn_sensor_forward();

		switch(stat){

			case 0b00:
			drive_right();
			drive_forward();
			break;

			case 0b01:
			drive_forward();
			break;

			case 0b10:
```

```
                        drive_right();
                        drive_forward();
                        break;

                        case 0b11:
                        drive_left();
                        break;

                }


        }
}
```