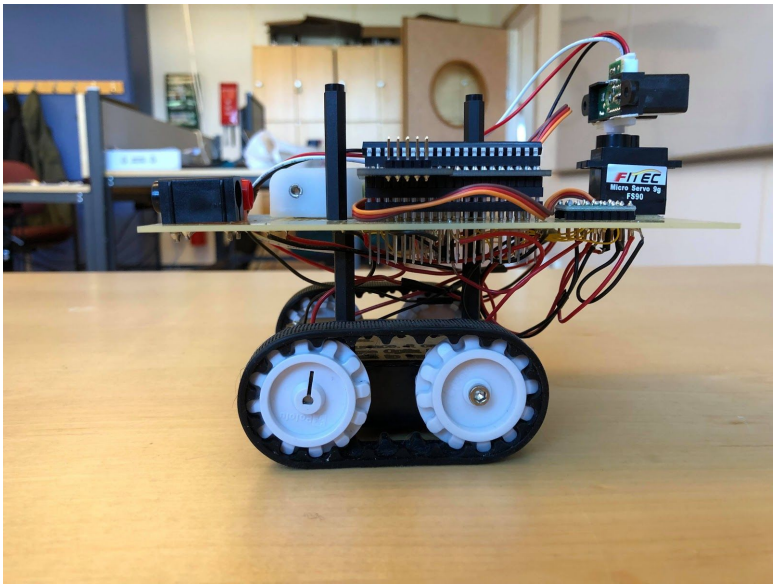
LUNDS TEKNISKA HÖGSKOLA
Lunds universitet



JOFEN-Prototypes

Målsökande pansarvagn

Projektarbete i Kursen EITA15

Jonatan Claesson, Olle Jonasson, Felix Rödén,
Edvin Rossi & Nils Olén

Handledare: Bertil Lindvall & Lars-Göran Larsson

Sammanfattning

Denna rapport behandlar produktionen av en prototyp skapad av studenter vid Lunds Tekniska Högskola. Syftet med denna prototyp är att påvisa studenternas kunskap inom datorteknik. Det gör studenterna genom att programmera en mikroprocessor, ATmega1284, och addera externa komponenter. På så vis skapas en fungerande prototyp. Prototypen som behandlas i denna rapport är en målsökande bil som kan följa ett föremål i rörelse.

Den målsökande bilen tillverkas genom att kombinera två motorer som styr varsin bandstats med en servomotor som roterar en avståndsmätare för att avgöra föremålets position. PWM, pulsbreddsmodulering, används tillsammans med ett motordrivarkort bestående av en dubbel H-brygga för att driva motorerna. Med hjälp av motordrivarkortet och PWM kan bilen köra med olika hastighet och svänga.

Mikroprocessorn har endast två timer som kan användas för att skapa en PWM-signaler. För att skapa en tredje PWM-signal som servomotorn kräver har en metod som kallas bitbanging använts.

Avståndsmätaren är ansluten till mikroprocessorn genom en pinne som har en inbyggd analog till digital-omvandlare. Avståndsmätaren har en analog utsignal i volt beroende på avståndet till objektet.

Avståndsmätaren returnerar ett värde till mikroprocessorn när servomotorn är riktad mot ett objekt och på så vis vet mikroprocessorn var föremålet befinner sig samt avståndet till föremålet. Genom att kombinera denna funktion med PWM-signalerna till motorerna kan bilen utföra olika manöver beroende på var objektet befinner sig.

Nyckelord: Pulsbreddsmodulering (PWM), bitbanging, mikroprocessor, avbrott

Abstract

This report handles the creation of a prototype created by students at Lunds Tekniska Högskola. The purpose of this prototype is to demonstrate the students' knowledge of computer science. This is done by programming a microcontroller, ATmega1284, and adding external units to create a complete prototype. This report will handle the creation of a homing car which will follow a moving object.

The homing car is created by two motors which operate one continuous track combined with a servo motor rotating a distance sensor to determine the location of the followed object. PWM, Pulse-width modulation, have been used together with a dual H-bridge motor driver to provide the car the abilities of driving with different speed and turning. The microcontroller is only provided with two timers that can be used to create PWM signals which both are used for the motors. To create a third PWM signal which is required by the servo motor a method called bitbanging have been used.

The distance sensor is connected to the microcontroller by a pin which have a built-in analogue to digital converter. The distance sensor has an analogue output voltage depending on the distance to the object it reaches.

The distance sensor is used together with the servo motor by returning a value to the microcontroller when the servo motor is pointing towards an objective. By combining this with the PWM signals sent from the microcontroller to the servo motor the microcontroller knows whether the object is out of sight, to the left, right or directly in front of the car. Depending on the location of the object the car will execute different manoeuvres.

Keywords: Puls-width Modulation (PWM), bitbanging, microcontroller, interrupt

Innehållsförteckning

1 Inledning **4**

1.1 Bakgrund 4
1.2 Syfte 4
1.3 Målformulering 5
1.3.1 Kopplingsschema 5
1.3.2 Styrning 5
1.3.3 Servomotor 5
1.3.4 Avståndsmätare 5
1.4 Problemformulering 5
1.5 Avgränsningar 6

2 Teknisk bakgrund **6**
2.1 Pulsbreddsmodulering (PWM) 6
2.2 Mikroprocessor 6
2.3 Motorkort 7
2.4 Servomotor 8
2.5 Avståndsmätare 9

3 Metod **10**
3.1 Planering 10
3.2 Kopplingsschema 11
3.3 Programmering av processor 13
3.4 Slutfasen 13

4 Analys **14**
4.1 Kravspecifikation 14
4.2 Problemlösningar 14

5 Resultat **15**

6 Diskussion **17**

7 Källförteckning **18**

8 Källkod **18**

1 Inledning

1.1 Bakgrund

Arbetet började som ett projekt i kursen Digitala System med Bertil Lindvall som handledare. Projektet består av att välja vad för prototyp man vill bygga. Det finns ett stort urval av olika saker som kan byggas eftersom alla grupper har tillgång till många olika delar som motorer och displayer, samt en mikroprocessor. Grupper valdes av studenter själva och sedan när en grupp visade ett kopplingsschema för handledaren, delade han ut delarna som behövs för att göra en prototyp av vad gruppen har bestämt att göra.

Komponenter valdes i grupperna i behov av vad som behövdes för att göra en fungerande prototyp. Alla grupper har tillgång till komponenter som motorer och displayer bland annat som de kursansvariga antingen beställde in till studenterna eller som de har sedan tidigare. När prototypen är klar ska varje grupp redovisa sitt arbete samt göra en HTML-sida med information om gruppen och arbetet.

1.2 Syfte

Avsikten med arbetet var att att påvisa kunskaper kring hur en mikroprocessor fungerar, hur den kan användas och att gruppens medlemmar kan tillämpa kunskapen praktiskt. För att visa förståelse för hur en mikroprocessor fungerar krävs det att få hårdvaran att fungera med hjälp av mjukvara. Det vill säga att hårdvaran, komponenterna, kopplas samman i form av en sluten krets som sedan kan samverka med hjälp av att programmera mikroprocessorn. Ett annat syfte med arbetet var att förbereda studenterna inför arbetslivet genom att lära studenterna hur det är att arbeta med ett projekt. Anledningen till att studenterna ska lära sig att arbeta i ett ett sådant sammanhang är att det är en vanligt förekommande arbetsmetod i arbetslivet.

1.3 Målformulering

Huvudmålet med projektet var att få den målsökande bilen att kunna följa ett rörande objekt för att stanna framför objektet. För att uppnå huvudmålet behöver fem enskilda mål som i sin tur är uppdelade i diverse delmål uppnås enligt listan nedan.

1. Kopplingsschema
    1.1. Konstruerat för att alla komponenter ska fungera optimalt
2. Styrning
    2.1. Bilen kan svänga vänster
    2.2. Bilen kan svänga höger
    2.3. Bilen kan köra framåt
    2.4. Bilen kan rotera 360 grader
    2.5. Bilen kan köra i olika hastigheter
3. Servomotor
    3.1. Rotera i 180 grader
    3.2. Avgöra om den är riktad rakt fram
    3.3. Avgöra om den är riktad åt vänster
    3.4. Avgöra om den är riktad åt höger
4. Avståndsmätare
    4.1. Upptäcka när det är ett föremål inom synfältet
    4.2. Avgöra avståndet till ett föremål

Lista 1.3.1. Mål.

1.4 Problemformulering

1. Hur ska bilen kunna svänga?
2. Hur ska de två motorerna som driver bandsatserna kunna drivas med olika hastighet och olika riktning?
3. Hur ska servomotorn fungera när processorn endast har två stycken klockor kopplade till PWM?

4. Hur ska bilen veta hur mycket den ska svänga?

Lista 1.4.1. Problem

1.5 Avgränsningar

- Prototypen kan inte minnas vilket föremål den följer utan kommer alltid att följa det närmaste objektet, oavsett om det är ett helt nytt objekt som kommer in.
- Bilen kan inte avgöra om det är objektet som den ska följa efter eller en vägg och kommer därför att svänga in till väggen och stanna där som om den uppnått sitt mål om väggen befinner sig närmare avståndsmätaren än vad objektet gör.
- Bilen kan endast se objekt inom 80 cm avstånd.

2 Teknisk bakgrund

2.1 Pulsbreddsmodulering

Pulsbreddsmodulering är en metod som används för att skapa en varierbar effektmatning. Det görs genom att utsignalen varier mellan en logisk 0 och en logisk 1 under särskilda perioder. För att genomföra detta använder man en klocka och sätter en bestämd periodtid. Beroende på hur hög effekt man vill ha sätter man sedan en puls som också är beroende av klockan. Under pulsen är utsignalen en logisk 1 medan resterande tid av perioden är utsignalen en logisk 0. På så vis kan man variera effekten av utsignalen.

2.2 Mikroprocessor

Mikroprocessorn som används är en ATmega1284 [1] som programmeras i språket C i programmet Atmel Studio 7.0 med hjälp av en JTAG. JTAG används för att felsöka mjukvaran som finns i mikroprocessorn och underlättar jobbet för programmerarna. Mikroprocessorn är hjärnan bakom det hela, den kommunicerar med alla delar av prototypen och ser till att alla komponenter arbetar tillsammans.

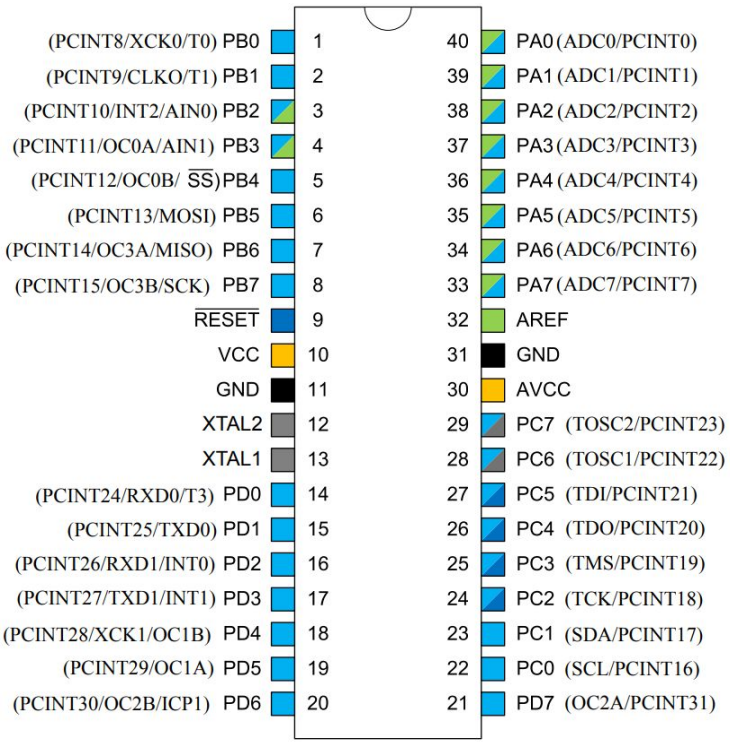


Fig 2.1.1. Beskrivning av ATmega1284 och dess pins.

2.3 Motorkort

Motorkortet, eller H-bryggan, som används är av modellen DRV8833 [2] från Texas Instruments. I prototypen används den för att driva två motorer som i sin tur driver ett band på vardera sida av bilen. Motorkortet tar emot signaler via fyra insignaler AIN1, AIN2, BIN1 och BIN2. Dessa styr i sin tur respektive utsignal AOUT1, AOUT2, BOUT1 och BOUT2. Detta sker enligt tabell 2.2.1.

xIN1	**xIN2**	**xOUT1**	**xOUT2**	**FUNCTION**
0	0	Z	Z	Coast/fast decay
0	1	L	H	Reverse
1	0	H	L	Forward

1	1	L	L	Brake/slow decay

Tabell 2.2.1. Motorkortets funktioner.

För att styra motorernas hastighet används PWM. Då ersätts de enkla ettorna med en PWM-signal vid funktionerna reverse och forward enligt tabell 2.2.2.

xIN1	**xIN2**	**FUNCTION**
0	PWM	Reverse PWM
PWM	0	Forward PWM

Tabell 2.2.2. Motorkortets styrning med PWM.

2.4 Servomotor

En servomotor av modellen FS90 från FITEC [3] används till prototypen. Servomotorn styrs med hjälp av en PWM-signal som bestämmer dess riktning. För att den ska fungera ställs periodtiden på PWM-signalen in till 20 ms. Pulstiden kan då styra servomotorn enligt tabell 2.3.1.

Pulstid	**Riktning**
0,544 ms	0°
1,500 ms	90°
2,400 ms	180°

Tabell 2.3.1. Pulstidens motsvarande riktningar.

I prototypen används en metod kallad bitbanging för att skapa en PWM-signal vid en utgång i

processorn som normalt sett inte stödjer utskickning av en PWM-signal. Metoden, i detta fall, är att en räknare skapas som baserat på klockfrekvensen bildar en fördröjning på 0,1 ms. Varje gång denna räknare når sitt toppvärde inträffar ett avbrott där periodtid och pulstid räknas upp. Periodtiden och pulstiden jämförs med varandra och beroende på vad pulstiden är inställd till skickas en etta eller nolla till servomotorn. När periodtiden når 200, alltså 20 ms, påbörjar den en ny räkning. Se fig. 2.3.1 för hur avbrottsrutinen kan se ut.

```
6  // Avbrottsrutin för servomotorn.
7
8  ISR(TIMER0_OVF_vect) {
9      period++;
10     if (period == 200) {                // Periodtiden börjar om ifall den
11         period = 0;                     // når 20 ms.
12     }
13
14     if (period >= abs(puls - 200)) {    // Ifall periodtiden fortfarande
15         PORTB = (1<<4);                 // är större än pulstiden skickas
16     } else {                            // en etta, annars skickas en nolla.
17         PORTB = (0<<4);
18     }
19
20     TCNT0 = 155;                        // Räknaren börjar om på 155 för
21 }                                        // att få rätt fördröjning (0,1 ms).
```

Fig 2.3.1. Avbrottsrutin för servomotorn.

2.5 Avståndsmätare

Avståndsmätaren som används är en Sharp GP2Y0A02 [4]. Den kan se objekt som är mellan 20-150cm ifrån. Avståndsmätaren skickar en utsignal som är beroende av hur nära den är ett objekt. Desto närmare den är ett objekt, skickar den mer spänning än vad den gör om avståndsmätaren ser ett objekt som är långt ifrån. I diagram 2.4.1, kan man se vilken spänning som avståndsmätaren skickar ut beroende på hur nära den är ett objekt.

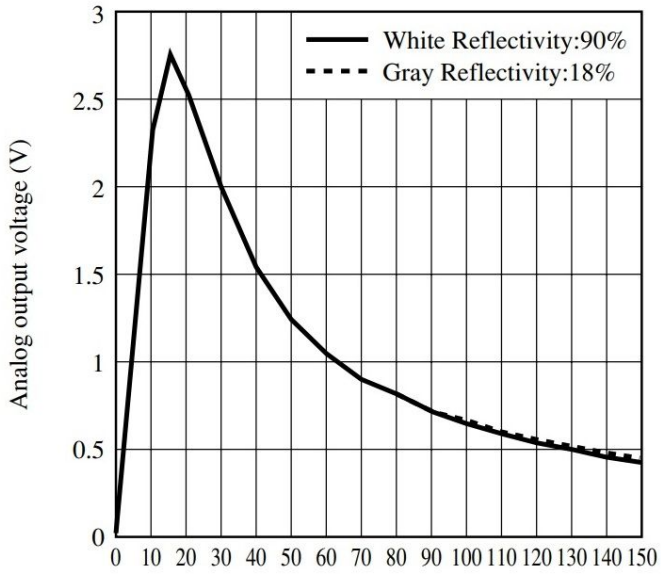


Diagram 2.4.1. Diagram över avståndsmätarens utsignal

3 Metod

Metoden innefattar de olika faser av arbetet som genomförts.

3.1 Planering

Gruppen började med att ha ett möte där det diskuterades om vilket område som ansågs vara intressant. Med hjälp av inspiration från föregående år gångars prototyper och med vetskap om vilka komponenter som fanns tillgängliga beslutades det att en prototyp av någon slags elektrisk bil skulle konstrueras. Valet föll till slut på en självkörande bil.

Under det andra mötet valdes de komponenter som behövdes för prototypen. Detta efter en grundlig informationssökning. En projektplan i form av veckoplanering gjordes där överskådliga mål och uppgifter bestämdes. Parallellt med utvecklandet av prototypen gjordes en veckorapport, dels för att enklare följa processen och dels för underlätta skrivandet av projektrapporten.

De komponenter som krävdes för en fungerande prototyp var:

1. Mikroprocessor: Detta är hjärnan, vilken vi programmerar, dvs den styr alla signaler

till resterande komponenter.

2. Motorkortet: Även kallad H-brygga, styr de två motorerna som driver bilen.
3. Motorerna: Två st motorer som styr var sitt band. De kan köra medurs och moturs.
4. Servomotorn: Används för att kunna styra hur sensorn vrids. Detta sker med hjälp av PWM.
5. Sensorn: Med annat namn avståndsmätare, är den komponent som tar in information om vilken distans ett objekt befinner sig från sensorn.
6. Batteri: Spänning på 4,8 V. Det försörjer bilens energibehov.
7. JTAG: Extern komponent. Den fungerar som en bro mellan Atmel Studios och mikroprocessorn, den kopplas in för att kunna programmera mikroprocessorn och för att felsöka prototypen.

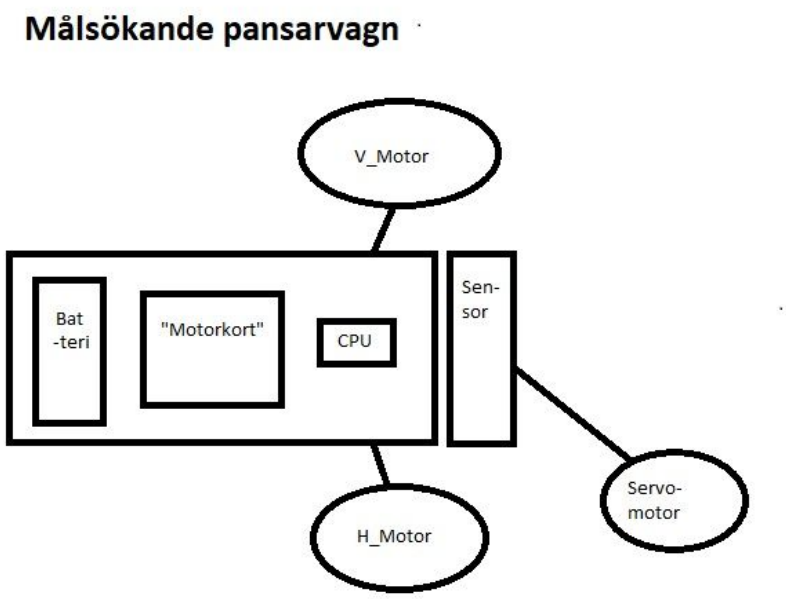


Fig 3.1.1 Första bilden som gjordes. Den beskriver de komponenter som ansågs vara nödvändiga för prototypen.

3.2 Kopplingsschema

Efter att gruppen valt ut komponenter, var det dags att börja med kopplingsschemat som även handledaren behövde godkänna innan komponenterna delades ut. Programmet Eagle

rekommenderades för att göra ett kopplingsschema där man väljer ritningar av olika slags komponenter som passar in till prototypen. Programmet var lite krångligt vid vissa tillfällen då det inte alltid blev som tänkt samt att det inte var alltför noggrant alltid.

När kopplingsschemat sedan var klart och handledaren hade godkänt det så delades komponenterna ut. Efter det var det dags för att koppla samman all hårdvara. Komponenterna placerades så att de motsvarade kopplingsschemat för att på så sätt göra det enkelt att löda och sammankoppla hårdvaran.

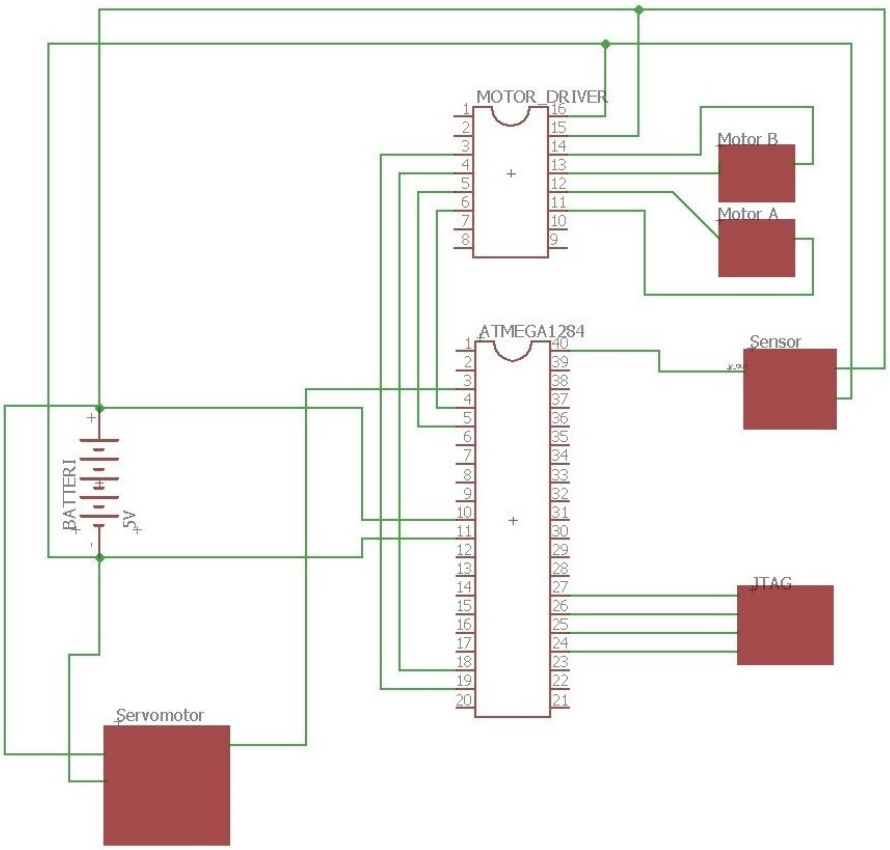


Fig 3.2.1 Kopplingsschemat för prototypen.

3.3 Programmering av mikroprocessor

Efter att all hårdvara var sammankopplad började gruppen med att programmera mikroprocessorn. Mikroprocessorn är hjärnan bakom allt då den skickar signaler till motorkortet, servomotorn och även sensorn. Kod till motorkortet kändes lämpligt att få fram först för att den kommunicerar med motorerna så att prototypen kan åka framåt eller bakåt. Kod för detta hittades genom att kolla igenom gamla laborationer i kursen och testa koder från olika sidor om avr-programmering.

Sedan var det dags att börja programmera servomotorn. Sensorn sitter på servomotorn som ska kunna rotera i 180 graders vinkel så att sensorn kan känna av objekt i närheten och inte bara de som är rakt framför den. Servomotorn behöver få PWM signaler från mikroprocessorn för att den ska svänga.

Efter det ska avståndsmätaren programmeras och anslutas så att mikroprocessorn tar emot korrekta värden så att de kan användas för att avgöra hur långt avståndet till objektet som bilen följer är. Sensorn ska känna av motstånd som är mellan 20-80 cm och med hjälp av servomotorn kan den se objekt som är runt om den. Om sensorn känner av att ett objekt är i närheten så skickar den signaler till mikroprocessorn som i sin tur skickar PWM-signaler till motorkortet som gör att prototypen svänger så att den står riktad rakt mot objektet och sedan kör emot den. Sensorn behöver programmeras så att den kommunicerar med mikroprocessorn och läser av om den hittar något objekt. Mikroprocessorn skickar sedan signaler till motorkortet som gör så att prototypen kör emot objektet. När den har kommit tillräckligt nära ett objekt så skickar sensorn en signal till mikroprocessorn som berättar att den är nära ett objekt och då ska bilen stanna.

3.4 Slutfasen för det praktiska arbetet

Här testas prototypen praktiskt för att se om några finjusteringar krävs, dvs små ändringar i den redan fungerande koden, för att få den att fungera optimalt. Mer konkret kontrolleras prototypens körning så den klarar av att stanna vid en viss distans, att den kan svänga på ett smidigt sätt. När prototypen fungerar enligt de mål som är uppsatta påbörjas slutförandet av

html-sidan och rapporten.

4 Analys

Analysen täcker områdena kravspecifikation och problemlösning.

4.1 Kravspecifikation

- Sensorn ska kunna mäta ett avstånd ca mellan 10-80 cm.
- Prototypen ska inte köra på sitt mål
- Prototypen ska stanna om den är ca 10cm från sitt mål.
- Prototypen ska kunna följa sitt mål.
- Om prototypen inte hittar ett objekt i sin vy ska den svänga runt åt ett håll tills den hittar ett objekt.
- Ifall prototypen tappar bort sitt objekt ska den stanna och påbörja en ny sökning.
- Prototypen ska inte köra för hackigt.

4.2 Problemlösningar

Det behövdes flera PWM-signaler än vad som hade planerats från början. Det ledde till att kopplingsschemat behövde ändras och andra pins behövde användas istället. Det största problemet som dök upp var när servomotorn började bli varm när den hade varit igång under en längre stund. Den blev varm för att det var problem med PWM-signalerna och dess periodtid. Periodtiden kunde inte ändras eftersom att det i timer 0 inte fanns någon motsvarighet till ICR1-registret från timer 1 och ICR3-registret från timer 3, som båda används för att ställa in periodtid i sina respektive timers. Dessa andra register, där periodtiden kunde ställas in, var redan upptagna av H-bryggan.

Lösningen till problemet var att använda bitbanging för att få fram PWM-signaler där periodtiden kan ändras. PWM-signalerna krävde en periodtid på 20 millisekunder och tack vare att använda bitbanging var detta nu möjligt. Med ett oscilloskop var det möjligt att mäta

signalens periodtid. I ATMEL var det inställt att klockans frekvens i mikroprocessorn var ⅛ av dess ursprungliga frekvens som var 8MHz så den blev 1MHz istället. Det var ett stort problem för då fick periodtiden ett annat värde än vad som behövdes. När det var åtgärdat och frekvensen var 8MHz blev periodtiden 20 millisekunder som planerat. Servomotorn fungerade inte som tänkt fast att frekvensen och periodtiden var rätt. Eftersom att den hade fått utstå många tester så blev den förmodligen bränd och behövde därför bytas ut. Med rätt frekvens och periodtid samt en ny servomotor så fungerade prototypen som planerat.

5 Resultat

Arbetet som gjorts har resulterat i en fungerande prototyp som uppfyller krav och förväntningar. I tabell 5.1 finns alla mål från lista 1.3 och om de är uppnådda eller inte.

Mål (1.3)	Uppnått	Ej uppnått
1.1	X	
2.1	X	
2.2	X	
2.3	X	
2.4	X	
2.5	X	
3.1	X	
3.2	X	
3.3	X	
3.4	X	
4.1	X	
4.2	X	

Tabell 5.1.1. Resultat av mål från lista 1.3.1.

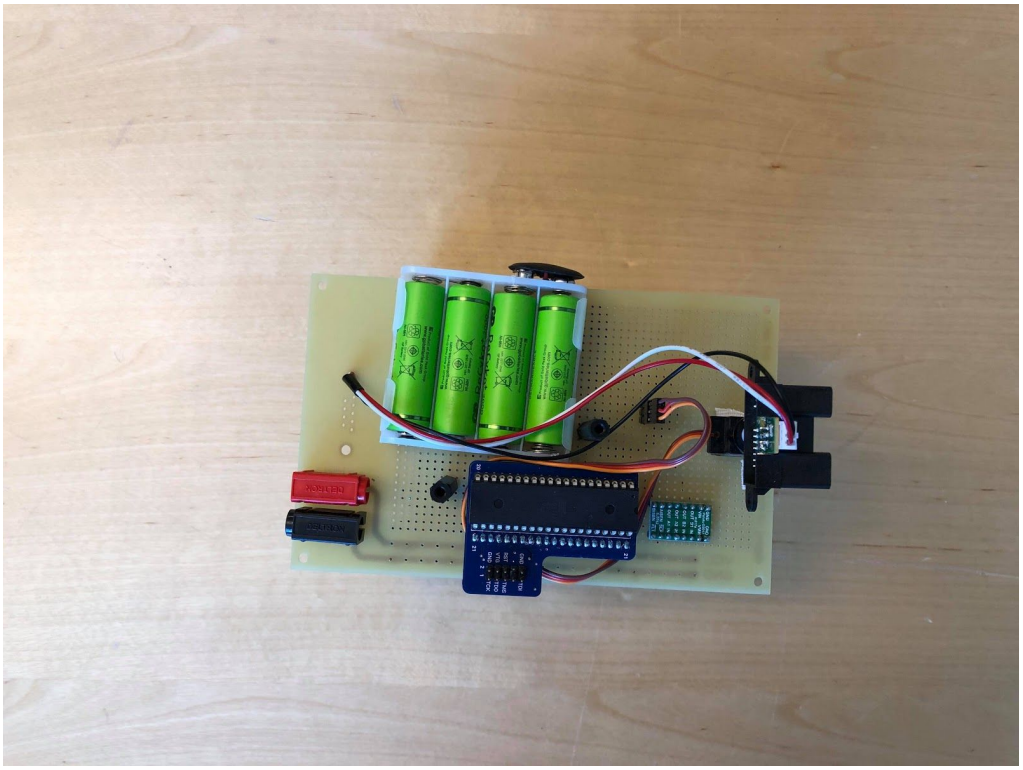


Bild 5.1.2. Den färdiga prototypen sedd från ovan. En tydlig bild på de olika komponenteras positionering i verkligheten.

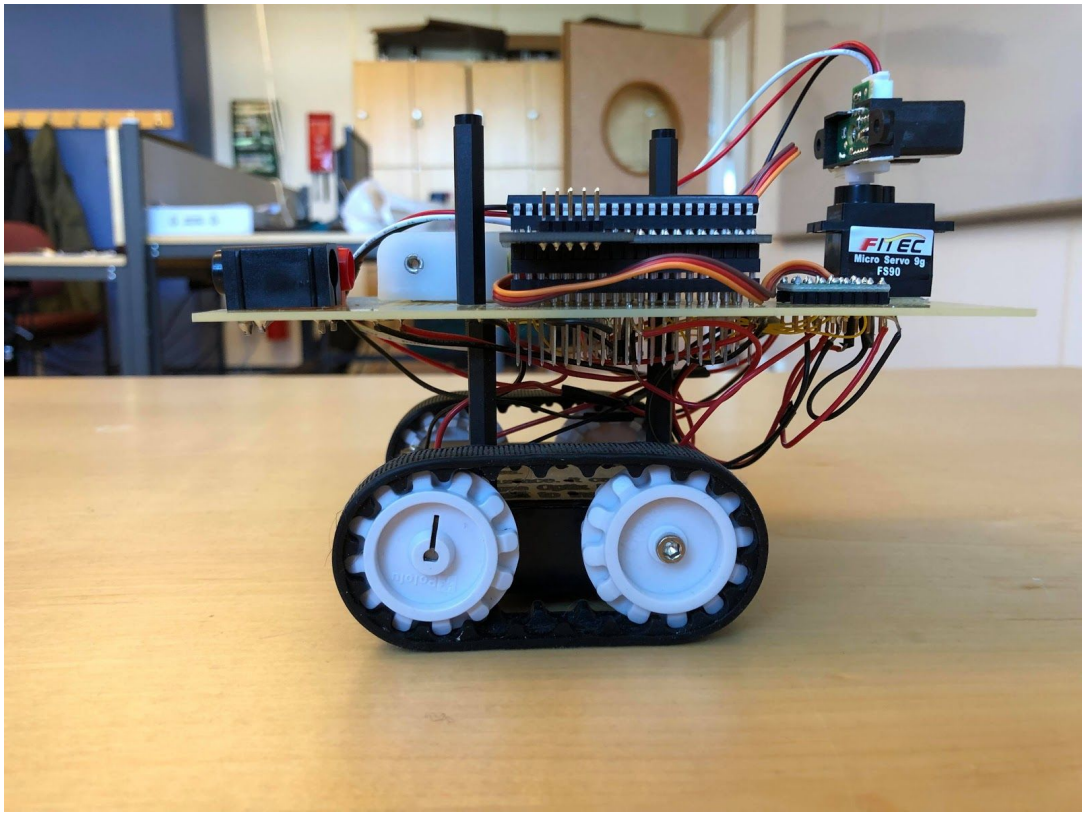


Bild 5.1.3. Kontraster mellan prototypens till synes "röriga" underrede och den övre stilrena komponentsidan. Här syns avståndsmätarens fastmontering på servomotorns roteringsblad.

6 Diskussion

Prototypen fungerar bra och alla målen är helt eller nästan helt uppnådda. Den kan svänga både åt höger och vänster och den kör rakt fram. Att prototypen ska köra bakåt är inte helt relevant eftersom att den följer efter sitt mål och när den letar efter ett nytt mål så roterar sensorn och försöker hitta ett föremål. Det leder till att den inte behöver köra bakåt. Servomotorn är den komponenten som har strulat mest. Enligt databladet ska servomotorn klara av att rotera 180 grader, vilket den klarar av på ett ungefär. Problemet med servomotorn har varit PWM-signalerna som den tar emot från mikroprocessorn. Det har varit krångligt med att få rätt periodtid för PWM-signalerna och frekvensen var felsatt så till slut behövdes servomotorn bytas

ut mot en ny. Avståndsmätaren kan känna av objekt som är i närheten då den stannar när den har kommit tillräckligt nära ett föremål. I helhet fungerar prototypen bra och den uppfyller målen och kraven som har satts.

7 Källförteckning

[1] Atmel, “8-bit AVR Microcontroller”, ATmega1284 datablad, Okt. 2016.

[2] Texas Instruments, “Dual H-Bridge Motor Driver”, DRV8833 datablad, Jan. 2011 [Reviderad Aug. 2011]

[3] FITEC, “Micro 9g servo for Air plane”, FS90 datablad.

[4] Sharp, “Long Distance Measuring Sensor”, GP2Y0A02YK datablad.

8 Appendix

8.1 Källkod

```
/*
 * MotorfunktionerPVM.c
  * Created: 2019-04-04 15:36:58
 * Author : ed5316ro-s
 */
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
volatile uint16_t period = 0;
volatile uint16_t puls = 0;
int shortest;
void dist_init() { // Initierar avståndsmätaren
	ADCSRA |= ((1<<ADEN)|(1<<ADPS2)|(1<<ADPS0));
	DDRA &= (1<<0);
	ADMUX |= ((1<<REFS0) | (1<<REFS1)); // Referens-volt 2,56 V
}
```

```
uint16_t adc_read() { // Läser avstånd och returner avståndet
	ADCSRA |= (1<<ADSC);
		while((ADCSRA & 0b01000000) != 0) {
	}
	return ADC;
}
void servo_init() { // Initierar servomotor
	DDRB |= 0b00010000;
	TIMSK0 |= (1<<TOIE0); // Overflow interrupt enable
	TCCR0B |= (1<<CS01); // Prescaler till 8
	TCNT0 = 155;
	sei();
}


void turn_right() { // Svänger bilen åt höger till det att den ser objekt vid avstånd shortest
	set_pulsea(7); // Höger hjul baklänges
	set_pulseb(0); // Höger hjul framåt
	set_pulsec(0); // Vänster hjul baklänges
	set_pulsed(7); // Vänster hjul framåt
	int medeldist = 249;
	while (medeldist < (shortest - 50)) {
		int dist = 0;
for (int i = 0; i < 8; i++) {
			dist = dist + adc_read();
		}
		medeldist = dist / 8;    //Medelvärdet av 8 avståndsmätningar
	}
	set_pulsea(0);
set_pulseb(0);
	set_pulsec(0);
	set_pulsed(0);
}
void turn_left() { // Svänger bilen åt vänster till det att den ser objekt vid avstånd shortest
	set_pulsea(0);
	set_pulseb(7);
	set_pulsec(7);
	set_pulsed(0);
	int medeldist = 249;
```

```
	while (medeldist < (shortest - 50)) {
		int dist = 0;
		for (int i = 0; i < 8; i++) {
			dist = dist + adc_read();
		}
		medeldist = dist / 8;	//Medelvärdet av 8 avståndsmätningar
	}

	set_pulsea(0);
	set_pulseb(0);
	set_pulsec(0);
	set_pulsed(0);
}


	int servo_dir() {
	sei();
	//14 är 90 grader
	int dire;
	shortest = 0;
	int dirshort = 4;
	for (int r = 0; r < 3; r++) {
		if (dirshort == 4) {
			for (int i = 7; i <= 21; i++) { //Vrider servomotorn från höger till vänster
				puls = i;
				if (i == 7) {
					dire = 2; // Höger
				}
				if (i == 14) {
					dire = 1; // Rakt fram
				}
				if (i == 15) {
					dire = 0; // Vänster
				}

				int dist = 0;
				for (int p = 0; p < 8; p++) {
					dist = dist + adc_read();
				}
```

```
                                dist = dist / 8;  //Medelvärdet av 8 avståndsmätningar
                                if (dist > 300 && dist > shortest) {  // Avgör det kortaste avstånd
och riktning till objekt
                                        shortest = dist;
                                        dirshort = dire;
                                }
_delay_ms(50);
                        }
                                for (int k = 20; k >= 8; k--) { // Vrider servomotorn från vänster till
höger
                                puls = k;
                                if (k == 14) {
                                dire = 1; // Rakt fram
                                }
                                if (k == 13) {
                                dire = 2; // Höger
                                }
                                int dist = 0;
                                for (int a = 0; a < 8; a++) {
                                        dist = dist + adc_read();
                                }
                                dist = dist / 8;   //Medelvärdet av 8 avståndsmätningar
if (dist > 300 && dist > shortest) { // Avgör det kortaste avstånd och riktning till objekt
                                        shortest = dist;
                                        dirshort = dire;
                                }
_delay_ms(50);
                        }
                }
        }
        if (shortest > 850) {
        shortest = 450;
        }
        if (dirshort < 4) {
                return dirshort; // Används sedan för att avgöra åt vilket håll den ev. ska svänga
        }
        puls = 14;
        set_pulsea(7);                          // Höger hjul baklänges
        set_pulseb(0);                          // Höger hjul framåt
```

```
	set_pulsec(0);				// Vänster hjul baklänges
	set_pulsed(7);				// Vänster hjul framåt
int medeldist = 299;
		while (medeldist < 300) {
			int dist = 0;
			for (int i = 0; i < 8; i++) {
				dist = dist + adc_read();
			}
			medeldist = dist / 8;		//Medelvärdet av 8 avståndsmätningar
		}
	stop();
	dirshort = 1;			// Har inte sett något
	return dirshort;
}

void timer1_init(){
	TCCR1B |= ((1<<CS12)|(1<<CS10)); // CLK(i/o)/1024 from prescaler
	TCCR1B |= ((1<<WGM13)|(1<<WGM12)); // Sätter ICR1 som top
	TCCR1A |= ((1<<COM1A1)|(1<<COM1B1)|(1<<WGM11)); // Clear OC0A/OC0B on
compare match, set output to low level
	DDRD |= ((1<<4)|(1<<5)); // Sätter PD4, PD5 till ut
}
void timer3_init(){					// Den här funktionen sätter
inställningarna för PVM samt sätter PB3, PB4, PB6, PB7 till ut och ger pinnarna en utsignal
	TCCR3B |= ((1<<CS32)|(1<<CS30)); // CLK(i/o)/1024 from prescaler
	TCCR3B |= ((1<<WGM33)|(1<<WGM32)); // Sätter ICR3 som top-register
	TCCR3A |= ((1<<COM3A1)|(1<<COM3B1)|(1<<WGM31)); // Clear OC3A/OC3B on
compare match, set output to low level
	DDRB |= ((1<<6)|(1<<7)); // Sätter PB6, PB7 till ut

}

void set_period(uint16_t value1){ // Sätter perioden för PVM som används för PB3, PB4, PB6,
PB7
	ICR3 = value1;// Sätter perioden för PB6, PB7
	ICR1 = value1; // Sätter perioden för PD4, PD5
}
void set_pulsea(uint16_t valuea){ // Bestämmer pulsen till PB6 och därmed hastigheten
	OCR3A = valuea;
```

```
}
void set_pulseb(uint16_t valueb){ // Bestämmer pulsen till PB7 och därmed hastigheten
	OCR3B = valueb;
}
void set_pulsec(uint16_t valuec){ // Bestämmer pulsen till PB3 och därmed hastigheten
	OCR1A = valuec;
}
void set_pulsed(uint16_t valued){ // Bestämmer pulsen till PB4 och därmed hastigheten
	OCR1B = valued;
}
void forward(int speed) { // Kör bilen frammåt med hastigheten speed, 0 lägst 20 högst
	set_pulsea(0);
	set_pulseb(speed);
	set_pulsec(0);
	set_pulsed(speed);
}
void stop() { // Stannar bilen
		set_pulsea(0);
		set_pulseb(0);
		set_pulsec(0);
		set_pulsed(0);
}
void go() {
	forward(10);
	int medeldist = 1022;
	while (medeldist < 1023 && medeldist > 300) {
		int dist = 0;
		for (int i = 0; i < 8; i++) {
			dist = dist + adc_read();
		}
		medeldist = dist / 8;		// Medelvärdet  av 8 avståndsmätningar
		if (medeldist > 700) {
			forward(6);
		} else if (medeldist > 400) {
			forward(8);
		}
	}
	_delay_ms(200);
	stop();
```

```
        while (medeldist > 900) {
                int dist = 0;
                for (int i = 0; i < 8; i++) {
dist = dist + adc_read();
                }
                medeldist = dist / 8;        //Medelvärdet av 8 avståndsmätningar
        }

}
int main(void)
{
        dist_init();
        servo_init();
        timer1_init(); // Ordnar alla inställningar för PVM, PD4, PD5
        timer3_init(); // Ordnar alla inställningar för PVM, P6, PB7
        set_period(20); // Sätter perioden till 10 för båda timers
        int dir = 4;
   while (1)
   {
                while (dir == 4) {
                        dir = servo_dir();
                }
                puls = 14;
        _delay_ms(250);
cli();
                if (dir == 0) {
                        turn_left();
                } else if (dir == 2) {
                        turn_right();
                }
                _delay_ms(250);
                go();
                dir = 4;
                _delay_ms(1000);
   }
}
ISR(TIMER0_OVF_vect) //Avbrottsrutin för servomotorn
{
                period++;
```

```
        if (period == 200) {
                period = 0;
        }
                if (period >= abs(puls - 200)) {
                PORTB = (1<<4);
        } else {
                PORTB = (0<<4);
        }
        TCNT0 = 155;
}
```