LUND INSTITUTE OF TECHNOLOGY
Lund University

ETSA01 – Digitala Projekt (I)
VT-13

Projektarbete
AC-58008

Victor Bodin
Albert Strömberg
Alexander Söderberg

Handledare
Bertil Lindvall

Sammanfattning

Rapporten behandlar processen kopplat till konstruktionen av ett självgående fordon vars uppgift är att undvika uppkommande hinder. Processen bestod av två delar; komponentkoppling och design, samt programmering. Konstruktionens grundläggande delar var en mikroprocessor ansluten till två elmotorer, ett servo och en analog avståndsmätare. Mikroprocessorn programmerades med ledning av initialt uppställda krav. Rapporten inleds med en kravspecifikation följt av en beskrivning av hårdvarukomponenter samt nödvändiga mjukvarubegreppsförklaringar. Detta följs av en genomförandebeskrivning och en slutsats.

Table of Contents

Kravspecifikation

Specifikationen av krav är uppdelat i primära och sekundära krav. De primära kraven är grundläggande krav som projektets konstruktion bör uppfylla för att uppnå sin huvudsakliga funktion. De sekundära kraven är krav som i mån av tid bör uppfyllas efter att de primära kraven infriats fullständigt.

Primära krav

Krav 1: Fordonet ska vara självgående.
Krav 2: Fordonet ska undvika hinder.
Krav 3: Fordonet ska vid påträffande av hinder kunna avgöra åt vilket håll som det finns mest utrymme och svänga däråt.

Sekundära krav

Krav 1: Fordonet ska efter angiven tid kunna hitta tillbaka till sin ursprungliga position.
Krav 2: Stopptid för beslut om svängriktning då fordonet påträffar hinder ska inte överstiga 3 sekunder
Krav 3: Fordonet ska kunna undvika stup.

Komponenter

Under följande del beskrivs de komponenter som ingår i konstruktionen. Framförallt behandlas grundläggande funktion och anslutningar.

Batteri 6V

För att konstruktionen ska vara mobil krävs en kraftkälla som kan ingå i konstruktionen. För detta valdes ett uppladdningsbart batteri på 6V.

AVR ATMega16

Med en matningsspänning på 5V drivs projektets processor med en klockfrekvens på 8MHz. Den har 40 pinnar och flertalet inre funktioner. De som används i detta projekt är dels 16-bitarsklockan och dels A/D-omvandlaren. Dessa aktiveras genom den kod som laddas in i processorn.

Av de 40 pinnarna som processorn har är 32 utav dessa så kallade I/O-pinnar, de kan alltså antingen ställas in på att ta in eller skicka ut signal. Då signal skickas ut antar den aktuella pinnen ett potentiellt högre eller lägre läge beroende på den kod som processorn för tillfället exekverar. Tar pinnen istället in signal “lyssnar” den på om spänningen den blir matad med är potentiellt hög eller låg. Dessa I/O-pinnar är fördelade över fyra portar benämnda A, B, C respektive D. Övriga pinnars funktioner är knutna till att bland annat driva processorn genom att ta in matningsspänning.

A/D-omvandlarens funktion är knuten till port A varför denna reserverades för detta i projektet.

16-bitarsklockan däremot är en intern resurs som genererar avbrott i den takt koden som exekveras ger. Dessa avbrott kan sedan användas till att initiera särskilda kodavsnitt som i förlängningen mynnar ut i ett särkilt utsignalmönster för processorns I/O-pinnar.

Den aktuella processorn som användes i projektet var från början förberedd för anslutning till PC via JTAG.

H-brygga L298

H-bryggan används för att reglera den ström som skickas till motorerna. Beroende på vad för signaler H-bryggan får från processorn släpper den igenom, stoppar eller byter riktning på strömmen. H-bryggan har stöd för två motorer med sina 15 pinnar. Den är kopplad till processorn där den tar upp 3 pinnar per motor, dvs 6 pinnar i det här fallet. Av dessa 3 pinnar används en för att stänga av/sätta på strömmen och de andra två används till att bestämma riktningen på den. Motorerna får sin ström via två pinnar var på H-bryggan. De resterande fem pinnarna används för matningsspänning (6V), logisk matningsspänning (5V), jord, samt sensing.

Spänningsregulator MCP1827 5-LD TO-550

Då konstruktionens kraftkälla levererade 6V och processorn samt den logiska delen av H-bryggan krävde 5V fordras en spänningsregulator som reducerar spänningen från 6V till 5V.

Spänningsregulatorns fem pinnarna har funktioner enligt tabellen nedan.

Pin	**Funktion**
SHDN	Styr huruvida spänningsregulatorn skall vara igång eller ej. Detta styrs på så sätt att hög signal är på och låg signal av.
VIN	Tar emot matningsspänningen på 6V.
GND	Jord
VOUT	Skickar ut utspänning på 5V.
PWRGD	Ger hög spänning som indikator på att regulatorns funktion är god. Måste kopplas samman med VOUT via ett motstånd på 100 kΩ. Funktionen nyttjades ej i projektet utan monterades endast för att säkerställa regulatorns funktion.

Avståndsmätare SHARP 2Y0A02

I konstruktionen används en optisk avståndsmätare av typen Sharp 2Y0A02 vilken kan mäta avstånd mellan 20 till 150 cm. Avståndsmätaren genererar en analog spänningssignal som varierar med avståndet enligt Bild 1. Inom ramen för avståndsmätarens avmätningsspann genereras en högre signal för ett kortare avståndet än för ett längre. Avståndsmätaren har tre anslutningar; spänning, jord och utsignal.

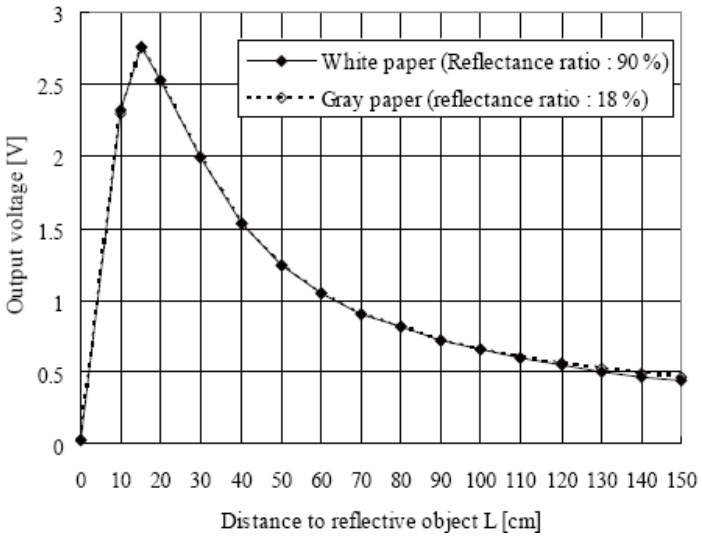

Figure 1: http://www.pololu.com/picture/view/0J1124

Toddler servo Mini E/BB

För att styra avståndsmätarens mätningsriktning används ett Toddler-servo med ett vridspann på 180 grader. Servot kontrolleras med pulssignaler av varierad längd inom ett intervall på 20 ms. Pulslängder mellan 1-2 ms ger ett utslag på -90 till +90 grader från central position med linjärt förhållande. Servot har tre anslutningar; spänning, jord och insignal.

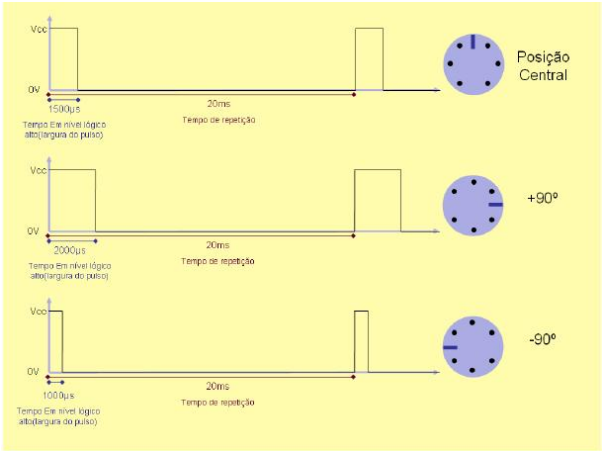

Figure 2 http://en.wikipedia.org/wiki/File:Sinais_controle_servomotor.JPG

Bandhjulsmotorer
Fordonet drivs av två separata bandhjulsmotorer med två strömanslutningar. Elmotorn har stöd för ström i olika riktning för att möjligöra drift åt valfritt håll.

LED
En röd LED används som indikator på strömtillförseln.

Mjukvara
Processorn ansluts till datorn via anslutningstypen JTag för att ha möjlighet att exekvera kod och ladda upp mjukvara. Programmering skedde i den integrerade utvecklingsmiljön AVR Studio och processorn programmerades i programmeringsspråket C.

A/D-omvandlare
Port A på processorn har stöd för A/D-omvandling vilket är nödvändigt för att omvandla avståndsmätarens analoga utsignal till en digital representation. Genom kommandon i mjukvaran kan A/D-omvandlaren konfigureras. Den digitala omvandlingen representerar den analoga signalen i registrena ADCH och ADCL som kan nås från utvecklingsmiljön.

Oändlig loop
För att upprätthålla en oändlig process krävs att processorn aldrig slutar läsa kod. Detta kan åstadkommas genom att skriva en oändlig loop som håller processen igång.

Avbrott och klockfrekvens
Processorn har stöd för avbrott med hjälp av tre olika timers. Detta innebär att processorn efter specificerad period gör ett tillfälligt avbrott i inläsning av kod och hoppar in i en metod i en annan del av koden. Processorns klockfrekvens uppmäts till 8 MHz och timerns intervall kan i mjukvaran specificeras till önskad upplösning.

Genomförande
Med tillgång till konstruktionens samtliga komponenter påbörjades arbetet med att konstruera ett kopplingschema för att skapa en uppfattning om hur komponenterna skulle anslutas till varandra. Enligt kopplingsschemat virades och löddes senare samtliga anslutningar. Alla komponenter kopplades direkt till processorn och spänning med undantag för bandhjulsmotorerna som kopplades via H-bryggan. Specifikt för LED:en var att den endast kopplades via ett motstånd till spänningskällan. Dess funktion var begränsat till att indikera

strömtillförsel. Under programmeringsprocessen genomfördes kontinuerligt förändringar av anslutningarna mellan komponenter för att öka robusthet och för att underlätta programmeringsarbetet.

Det första steget i programmeringsprocessen innefattade att testa motorernas funktion i utvecklingsmiljön. Dessa primitiva test genomfördes genom att i skicka konstanta ettor på processorn två pinnar som var kopplade till en av motorerna via H-bryggan.

När servot skulle programmeras uppstod svårigheter. Servot krävde en puls på 1.5 ms var 20 ms för att anta grundposition. Lösningen låg i att aktivera en av processorns timers. Timern fungerar som så att den räknar processorns tick, vilket sker 8 miljoner gånger per sekund, och genererar ett avbrott när den räknat till ett visst tal. Timern ställdes in på att räkna till 500, vilket ledde till ett avbrott 16 gånger i millisekunden. Detta möjliggjorde att en reguljär puls kunde skickas till servot och att pulsen kunde justeras med relativt hög upplösning.

Avståndsmätaren skickade en analog signal till processorn vilket innebar att A/D-omvandlaren fick användas. Den ställdes in så att en omvandling gjordes på kommando. Omvandlingen sparade signalen från avståndsmätaren till 10 bitar i registrena ADCH och ADCL. Vid en färdig omvandling utförs ett avbrott där de 10 bitarna sparades undan i en variabel. Ett medelvärde av de senaste två omvandlingarna beräknades som jämförelsevärde. Detta för att minska brus från avståndsmätaren. Medelvärdet står i 10 bitar vilket representerar ett decimalt tal mellan 0-1023. Ett medelvärde på 500 testades fram vilket representerade ett ungefärligt avstånd på 20 cm. Värdet kunde dock skilja sig ganska dramatiskt åt beroende på färgen på det föremål avståndsmätaren “tittade” på. Vita föremål gav ett högre värde än svarta då dessa på ett bättre sätt reflekterade det ljus som sändes ut ifrån avståndmätaren.

När alla komponenter fungerade var för sig initierades en process med att få respektive komponent att fungera ihop. Detta var det mest tidskrävande momentet i projektet då de första försöken misslyckades. Efter ett antal försök upptäcktes att AVR Studios optimeringsfunktion försvårade processen. Optimeringsfunktionen stängdes av och konstruktionens olika komponenter började fungera allt bättre tillsammans. Projektet hade nu kommit till ett stadium där endast finjustering i form av avslutande kalibrering återstod. Elmotorernas varierande motoreffekt försvårade kalibreringsprocessen något. När de primära kraven var uppfyllda genomfördes en avslutande optimering av koden för att reducera stopptiden vid avmätningsförfarande för att kunna uppfylla en av de sekundära kraven.

Resultat

Projektet mynnade ut i en fungerande konstruktion som infriade alla primära och ett utav de sekundära kraven.

De lärdomar som dragits av projektet är bland annat en fördjupad förståelse för hur kod översätts till logiska signaler som så småningom ger fysiska förändringar i elektroniska komponenter samt kontrollen av detta. Vidare har programmeringsspråket C blivit mer bekant.

Möjligheter till vidareutveckling finns dels genom att infria de två sekundära krav som återstår och dels genom diverse påhängssatser som exempelvis skulle kunna samla in damm eller klippa gräs. Det sekundära kravet om att fordonet skall kunna hitta tillbaka till dess utgångspunkt anses svårt då det skulle kräva en komponent som höll koll på fordonets nuvarande och tidigare position/-er. Att fordonet skall kunna undvika stup anses däremot lättare då detta endast skulle kräva ytterligare en avståndsmätare riktad mot underlaget framför fordonet. Svårigheterna med detta tros vara kalibreringen av det tröskelvärde som skall generera en undvikande sväng. Detta anses svårt dels då avståndsmätaren ger såpass olika resultat beroende på vilken färg underlaget har och dels att värdet måste finnas i ett intervall vilket alltså medför två gränsvärden. Att ansluta påhängssatser för ytterligare funktioner ses mer som ett mekaniskt och konstruktionsmässigt problem än programeringsmässigt. Detta då vi begränsar fordonets roll till att förflytta tillbehöret samt eventuellt leverera spänning till detta.

Bilagor

Kod

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

int adcResult;
int prevAdc;
int median;
int a;
int angle;
int looking;
int check;
int rightValue;
int leftValue;
int turning;
int isTurning;
int microSpeedL;
int speedL;
int microSpeedR;
int speedR;
int leftWheel;
int rightWheel;

int main(void){

	// Bit 7 input, rest output.
	DDRA = 0b01000000;
	PINA = 0b10000000;

	// Turn all B-ports output
	DDRB = 0b11111111;
	PINB = 0b00000000;

	//Turn off all other ports
	DDRC = 0b00000000;
	PINC = 0b00000000;

	DDRD = 0b00000000;
	PIND = 0b00000000;


	//Enable ADC for bit 7
	ADMUX = 0b00000111;
	//Set the left adjust result to 10 bit.
```

```
	ADMUX |= (1<<ADLAR);
	//Set AVCC with external capacitor at AREF pin
	ADMUX |= (1<<REFS0);
	//ADMUX |= (1<<REFS1);

	//Enable ADC.PORTA 0 27
	ADCSRA |= (1 << ADEN);

	// Prescaler
	// 8000000 / 6128PORTB 1 24
	//Set prescaler to 128 bit;
	ADCSRA |= (1 << ADPS2);
	ADCSRA |= (1 << ADPS1);
	ADCSRA |= (1 << ADPS0);
	//Enable interrupts.
	ADCSRA |= (1 << ADIE);
	//ADCSRA |= (1 << ADATE);

	//Aktiverar användningen av klockan
	TCCR1B |= (1 << WGM12);
	//Aktiverar CTC
	TIMSK |= (1 << OCIE1A);
	//Tillåter interuptions
	sei();

	//Set interruptresolution
	OCR1A = 500;

	// Starts the ADConversion
	ADCSRA |= (1 << ADSC);

	TCCR1B |= (1 << CS10);


	setup();
	start();
	return 1;

}

int setup(void)
{
	adcResult = 0;
	prevAdc = 0;
	median = 0;
	a = 0;
	angle = 342;
	looking = 0;
	check = 0;
```

```
	rightValue = 0;
	leftValue = 0;
	turning = 0;
	isTurning = 0;
	speedL = 0;
	microSpeedL = 0;
	microSpeedR = 0;
	speedR = 0;
	leftWheel = 1;
	rightWheel = 1;
	PORTB = 0b00100001;
	return 1;
}

int start(void)
{
	while(1)
	{
		if(check == 0)
		{
			ADCSRA |= (1 << ADSC);
			cli();
			if(median > 500)
			{
				if(median > 500)
				{
					check = 1;
					PORTB = 0b00100001;
				}
			}
			else
			{
				forward();
			}
			sei();
		}
		if(check == 1)
		{

			lookAround();
		}
		if(check == 2)
		{
			turn();
			forward();
		}

	}
	return 1
```

```
}

int forward(void)
{
	if(leftWheel == 1)
	{
		if(speedL > 1 && speedL < 4)
		{
			PORTB &= 0b11111101;
		}
		if(speedL > 4)
		{
			PORTB |= 0b00000010;
			speedL = 0;
		}
	}
	if(rightWheel == 1)
	{
		if(speedR > 1 && speedR <5)
		{

			PORTB |= 0b00010000;
		}
		if(speedR > 5)
		{
			PORTB &= 0b11101111;
			speedR = 0;
		}
	}
	return 1;
}
int turn(void)
{
	if(turning < 35)
	{
		if(leftValue < rightValue)
		{
			leftWheel = 0;
		}
		else
		{
			rightWheel = 0;
		}
	}
	else
	{
		rightWheel = 1;
		leftWheel = 1;
		isTurning = 0;
```

```
                turning = 0;
                check = 0;
        }
        return 1;
}


int lookAround(void)
{
        if(looking < 20)
        {
                angle = 330;
        }
        else if (looking < 22)
        {
                ADCSRA |= (1 << ADSC);
                rightValue = adcResult;
        }
        else if (looking < 42)
        {
                angle = 355;
        }
        else if (looking < 44)
        {
                ADCSRA |= (1 << ADSC);
                leftValue = adcResult;
        }
        else if(looking < 64)
        {
                angle = 342;
        }
        else
        {
                check = 2;
                looking = 0;
                isTurning = 1;
        }
}



ISR(TIMER1_COMPA_vect)
{
        a++;
        microSpeedL++;
        microSpeedR++;
        //Inträffar var 10:e millisekund.
        if(a == 160)
        {
```

```
                if(check == 1)
                {
                        looking++;
                }
                if(isTurning == 1)
                {
                        turning++;
                }
        }
        if(a > 320)
        {
                PORTA |= 0b01000000;
                if (a > angle)
                {
                        a = 0;
                        PORTA &= 0b10111111;
                }
        }
        if(microSpeedL > 40)
        {
                speedL++;
                microSpeedL = 0;
        }
        if(microSpeedR > 40)
        {
                speedR++;
                microSpeedR = 0;
        }

}

ISR(ADC_vect)
{
        prevAdc = adcResult;
        uint8_t theLow = ADCL;
        uint16_t tenBitValue = ADCH << 2 |theLow >> 6;
        adcResult = tenBitValue;
        median = adcResult + prevAdc;
        median = median >> 1;

}
```

Kopplingsschema