# Teknisk rapport

Examinationsprojekt VT 2015

Viktor Rusnak
Michael Nilsson

# Innehåll

# Inledning

Uppgiften i examinationsprojektet var att skapa ett program i C som kan reglera en pingisboll på en lutande arm. Detta skulle vidare regleras och plottas med hjälp av Matlab.

C-programmet består av två delar. Den ena delen behandlar den tidsdiskreta regleringen och avläsningen från sensorn. Den andra delen tar hand om kommunikationen mellan Matlab och Arduinokortet. Ett realtidsoperativsystem hanterar vidare de olika delarna, för att de båda delarna ska kunna köras parallellt. Högst prioritet har regleringen och avläsningen från sensorn, medan kommunikationen har lägst.

Kommunikationen mellan Arduino och Matlab sköts seriellt via USB-sladd. Kommunikationen har två syften. Det första är att skicka ett börvärde från Matlab till Arduino. Det andra syftet är att skicka fläktens styrsignal och avståndsmätarens värden från Arduino till Matlab, där de sedan ”plottas” i en graf i realtid.

Uppgiften har delats upp mellan de båda laboranterna på följande sätt. Mjukvarumässigt har Viktor arbetet självständigt med PWM och haft huvudansvaret för regulatordelen. Michael har arbetat med sensorn och kommunikationen. Då regulatordelen är en så pass stor del har båda haft att göra med denna. De olika delarna i rapporten skrevs så långt som möjligt av den som arbetat med motsvarande del i mjukvaran. För de delar där det blev svårt att dra tydliga gränser, till exempel regulatordelen, hjälptes laboranterna åt där det behövdes. Slutligen har en sammanfattning tillsammans med ett diskussionsavsnitt skrivits tillsammans av båda.

# Teori

## Den digitala signalen

Pingismodellen innehåller en sensor som har till uppgift att mäta var bollen befinner sig på armen. Informationen skickas vidare till en regulator, vilken genererar en styrsignal. Signalen skickas som en digital signal i intervallet [0…1023], eftersom en tio bitars D/A-omvandlare används. Fläkten styrs med en spänning mellan 0 - 3,3V, något som ger upplösningen:

$$\frac{3{,}3}{1024} = 0{,}0032V = 3{,}2mV$$

## Reglering

Ett stabilt system är ett system som har ett förutsägbart beteende samt behåller detta vid eventuella störningar [8]. Felaktigt val av regulator såväl som felaktig inställning kan leda till att vårt reglersystem blir instabilt. Detta är något som man vill undvika men risken att systemet blir ostabilt finns alltid hos ett återkopplat system. Ett sådant system är nästintill oanvändbart.

Snabbhet är en annan viktig egenskap hos återkopplade systemet. Hur snabbt systemet ställer in sig i förhållande till börvärdet är vad som definierar snabbheten. Man brukar prata om stigtiden vilket är ett mått som berättar hur lång tid systemet går från 10% upp till 90% av börvärdet. Då stigtiden är ett relativt mått, kommer den att variera beroende på i vilken situation samt regulatortyp man använder. Nackdelen är att en ökad snabbhet oftast leder till sämre stabilitet hos regulatorn. Ett annat mått är insvängningstiden vilket talar om hur lång tid ett system tar på sig att ställas in av börvärdet (oftast +-5%).

I ett statiskt tillstånd betraktas endast sluttillståndet. Valet av regulator påverkar således den statiska noggrannheten. Ett bra val av regulator kan leda till att det kvarstående felet nästan helt elimineras medan ett dåligt val av regulatortyp kan leda till ett större kvarstående fel vilket resulterar i ett dåligt reglersystem.

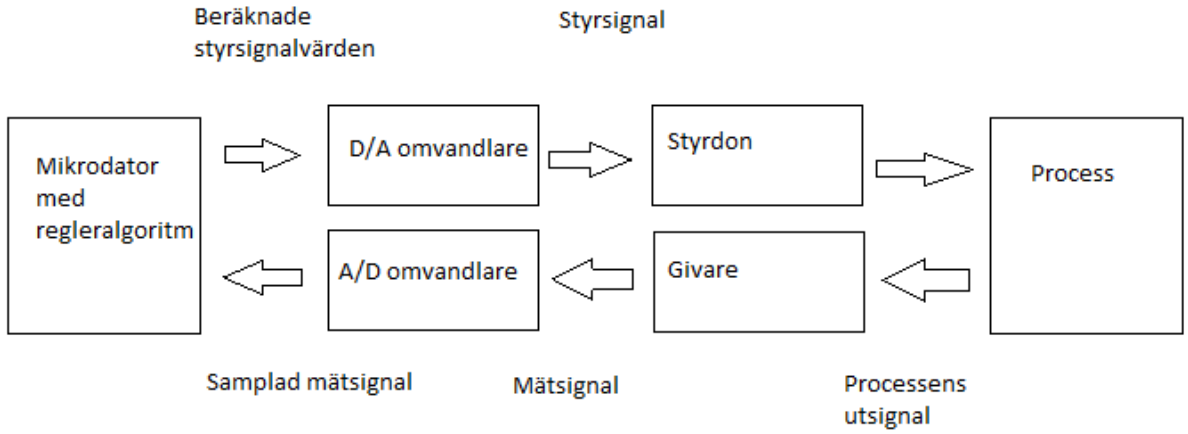
Regulatorer existerar av olika typer. Beroende på användningsområde krävs ett genomtänkt val av regulator för att få ut en så effektiv reglering som möjligt. För stabilisering av första ordningens processer är det oftast mest praktiskt att använda en P-regulator. Detta då regulatorns huvudsakliga uppgift är att minska det kvarstående felet i systemet. System som är mest lämpliga för denna typ av reglering är de som kan tåla ett konstant kvarstående fel.

En integrerande del bidrar till att regulatorn omöjligen kan förutse ett eventuellt kvarstående fel och därmed garanteras alltid en överskjutning av sitt börvärde.

En PID-regulator fås genom att tillföra PI-regulatorn ett derivat- eller ”D”-andel. PID-reglering är mycket vanligt förekommande inom industrin. Då felet ökar ger den deriverande delen (D-delen) ett bidrag till styrsignalen med samma tecken som den proportionella delen (P-delen). När felet minskar ger D-delen ett bidrag till styrsignalen med motsatt tecken jämfört med P-delen. Fördelen med detta är att det motverkar översvängningar och tendenser till instabilitet. PID-regulatorn har en snabb svarstid och bidrar till ett mer stabilt system. PID-regulatorn är populär då den även kan användes hos processer av högre ordning.

## Diskretisering av regleringen

Med tidsdiskret reglering menas att styrsignalen bara förändras vid specifika tidpunkter. Tidsdiskret reglering kallas även digital reglering, för att man använder en dator för att beräkna styrsignalen. Figuren nedan visar komponenterna i ett tidsdiskret reglersystem.




*Figur 1 Komponenter i tidsdiskret reglersystem*

I mikrodatorn finns det program som ska hantera alla insignaler och beräkningar, vilka sedan skickas vidare som utsignaler. Utsignalen omvandlas sedan av D/A omvandlaren till en analog signal. Den analoga signalen talar sedan om för styrdonet hur det ska agera och resultatet blir hur processen uppför sig. Resultatet skickas vidare till en givare och sedan en A/D omvandlare, vilken omvandlar den analoga signalen till en digital.

En tidsdiskret PID regulator kan skrivas på följande sätt:

$$u(k) = K\left[e(k) + T_D\left(\frac{e\big(k - e(k-1)\big)}{h}\right) + \frac{h}{T_I}w(k)\right]$$

$$där\ w(k) = w(k-1) + e(k)$$

I formeln ovan är betecknas styrsignalen vid en tidpunkt med u(k). Vidare är K förstärkningen, e(k) är felet vid en tidpunkt, TD är deriveringskonstanten och TI integreringskonstanten. Sampeltiden betecknas med h.

### Ziegler-Nichols metod

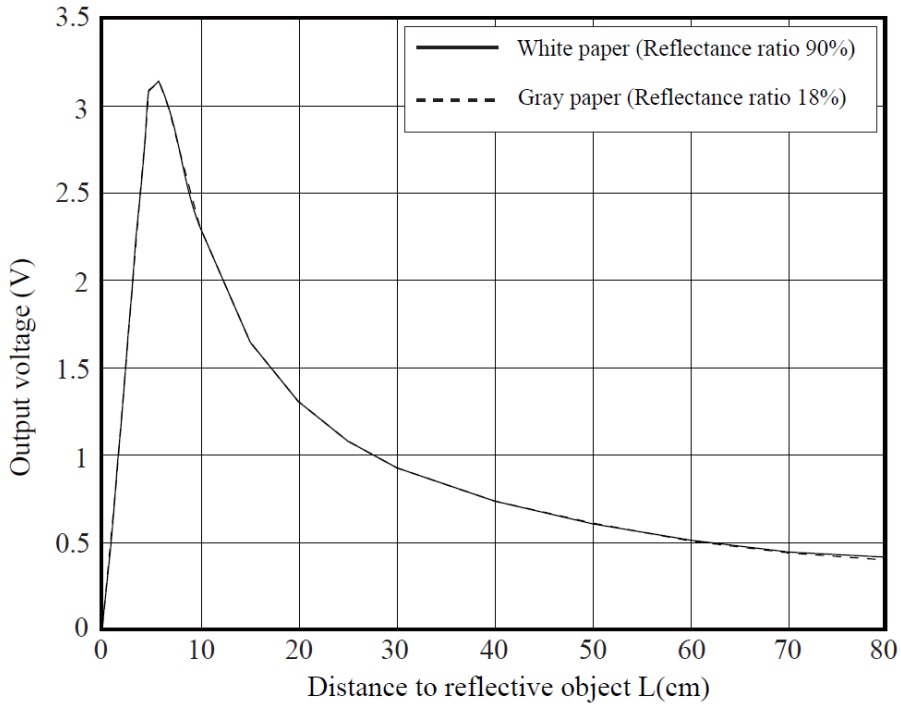
För att kunna använda Ziegler-Nichols metod behöver man ett färdigt system med en PID-regulator. Metoden handlar om att ta fram värden till Kp, Ki och Kp och sedan testköra med dessa. När man använder metoden börjar man med att anpassa om sin PID-regulator till en ren P-regulator. Det medför att Kd sätts till noll och Ki till "oändligheten". Sedan ökar man Kp till systemet precis börjar självsvänga. Kp ska då antecknas tillsammans med periodtiden för svängningarna. Därefter kan man till exempel använda tabellen i kapitlet om Ziegler-Nichols metod i vår kursbok för att beräkna de olika parametrarna.

Fördelen med Ziegler-Nichols metod är att man inte någon överföringsfunktion för att kunna använda den. Nackdelen blir dock att den är för enkel för att alltid kunna ge bra parametrar. Därför kan man behöva justera parametrarna för att få en bättre reglering.

## Metod

### Sensorn

I projektet används sensorn GP2Y0A21YK0F, tillverkad av SHARP (Givet datablad för sensorn). Figur 1 nedan visar hur förhållandet mellan avstånd till ett objekt och utspänning för sensorn ser ut. Sensorn använder infrarött ljus för att mäta avståndet. I figuren kan man se att det förekommer en topp mellan noll och tio centimeter. Ett objekt som befinner sig på ett avstånd på upp till tio centimeter från sensorn kan inte uppfattas ordentligt. Samma sak gäller för värden över 40 centimeter. Vi har därför valt att använda börvärden mellan 20 och 40 centimeter.




*Figur 2: Graf som visar förhållandet mellan avstånd till ett objekt och utspänning för sensorn.*

I och med att kurvan inte är linjär behövdes den lineariseras på något sätt först. Det beror på att en förflyttning av bollen måste motsvara samma sak, var man än befinner sig på rampen. Linearisering gick till på så sätt att ett antal intervall av sensorvärden fick motsvara ett antal verkliga avstånd i centimeter från sensorn. Om det inte utförts en lineraisering hade värden från den nedre halvan av rampen blivit lika varandra. Värdena från den övre halvan hade däremot förändrats mycket till en början med, för att sedan snabbt avta när bollen kommer några centimeter från sensorn.

Sensorn genererar värden för bollens position och skickar dessa till en av de analoga ingångarna på Arduinokortet. Metoden för detta återfinns i c-filen FuncADC.c och ser ut på följande sätt:

```
uint32_t ReadFanValue(void)
{
uint32_tcurrSensorValue;
adc_start(ADC);
currSensorValue = adc_get_channel_value(ADC, ADC_CHANNEL_10);
returncurrSensorValue;
}
```

Vidare hämtar regulatorn värden från sensorn med hjälp av metoden ovan. Hur fläkten regleras blir beroende på hur stor avvikelsen är från börvärdet. Om sensorn ger ut ett värde som är mindre respektive större än börvärdet påverkas regulatorn.

## Inställningar av reglerparametrar

Efter linearinseringen av sensorns utspänning med avseende på avståndet till ett objekt, utfördes beräkningar med Ziegler-Nichols metoden för att ta fram kp, ki och kd. Periodtiden vid självsvängning uppmättes till 2.9 sekunder, vilket gav en frekvens på 0.35 Hz. Kp bestämdes till 0,32 ki till 1,45 och kd till 0,36.

## PWM

Generering av en PWM signal sker på pinne 40 på det Arduino Due kort som användes. I metoden InitPWM görs diverse inställningar innan pinne 40 kan initieras. Nedan följer metoden där initieringen av PWM sker.

```
voidInitPWM(void){
		pmc_enable_periph_clk(ID_PWM);
		pwm_channel_disable(PWM,PWM_CHANNEL_3);

		pwm_clock_tclock_setting={
			.ul_clka=1000*1000,
			.ul_clkb=0,
			.ul_mck=sysclk_get_cpu_hz()
		};
		pwm_init(PWM,&clock_setting);

		InitPIN40();
}
```

Vidare sker en initiering av pinne 40 i InitPIN40:

```
voidInitPIN40(void)
{
		pwmPIN40.channel=PWM_CHANNEL_3;
		pwmPIN40.ul_prescaler=PWM_CMR_CPRE_CLKA;
		pwmPIN40.ul_duty=0;
		pwmPIN40.ul_period=100;
		pwm_channel_init(PWM,&pwmPIN40);
		pio_set_peripheral(PIOC,PIO_PERIPH_B,PIO_PC8B_PWML3);
		pwm_channel_enable(PWM,PWM_CHANNEL_3);
}
```

Signalen som skickas ut kan variera mellan noll och 100, där 100 är maximalt varvtal för fläkten.

## Regleringen

Samplingstiden för respektive task är vald till 100 millisekunder. Detta innebär att sampeltiden för regleralgoritmen också är satt till 100 millisekunder.

För var gång reglertasken körs filtreras signalen för att undvika extremvärden. Detta utförs med hjälp av ett glidande medelvärde. Filtret är uppbyggt på så sätt att sex sensorvärden placeras i en array. Sedan beräknas ett medelvärde, som skickas till beräkningsalgoritmen.

Varje gång reglertasken körs flyttas alla värden i arrayn ner ett steg. Den första positionen i arrayn får sedan ett nytt sensorvärde och ett nytt medelvärde beräknas.

Värdena skickas sedan in via Matlab, tillsammans med en sampeltid och börvärdet. Koden för själva regulatorn visas nedan.

```
int32_tCalcSignal(doublesampTime,doublek_p,doublek_i,doublek_d,int32_tcurrErr,int32_tprevErr,int32_tsumErr)
{
		doubleproportionalPart;
		doubleintegralPart;
		doublederivingPart;
		int32_tsignal;

		proportionalPart=k_p*(double)currErr;
		integralPart=(double)sumErr*(sampTime/k_i);
		derivingPart=((double)currErr-(double)prevErr)*(k_d/sampTime);
		signal=proportionalPart+integralPart+derivingPart;
		returnsignal;
}
```

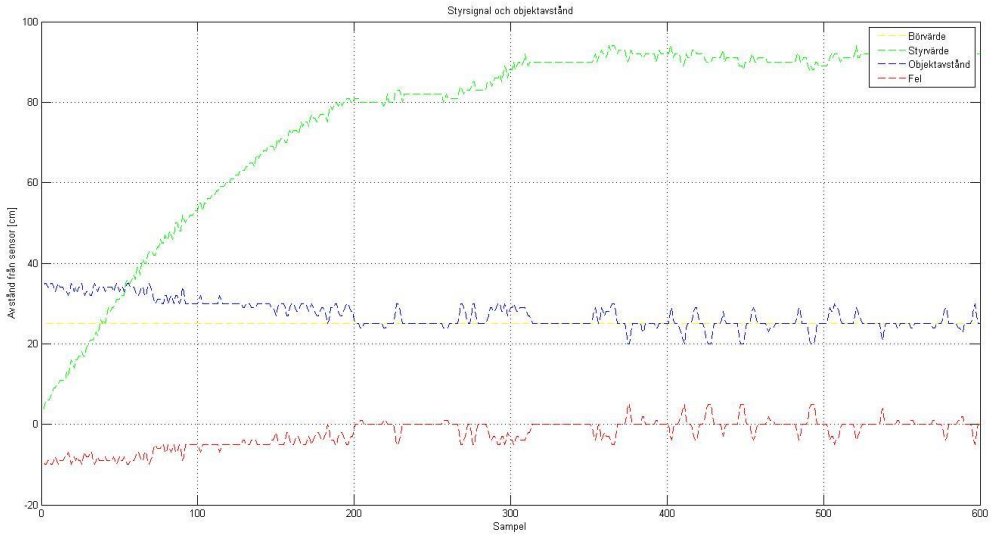
## Kommunikation mellan Arduino och Matlab

Kommunikation från matlab till arduino upprättas genom att man skapar ett serial object enligt följande rad kod: `s = serial('COM10', 'BAUDRATE', 115200)`. Därefter måste porten öppnas med kodsnutten ”`fopen(s)`”, innan man kan skicka exempelvis ett börvärde med kodsnutten ”`fwrite(s, uint32(30))`”.

Kommunikation från arduino till matlab sker genom metoden printf(). Exempel på detta i koden är `printf("%d\n",circBuffertActualValue[0]);`.

# Resultat

I figur 2 får man en bild över hur regleringen blev. Den blå kurvan visar hur objektet rör sig i förhållande till sensorn. Man kan se att objektet pendlar upp och ner med kraftiga rörelser i början, för att sedan sakta stabilisera sig runt börvärdet. Den gröna kurvan visar styrsignalen, vilken pendlar mellan noll och hundra. Rent praktiskt kan signalen dock både gå under noll och över hundra i regleralgoritmen, men den har begränsats till detta intervall. Ett värde under noll skulle betyda att fläkten suger åt sig bollen, vilket den inte kan. Ett värde över hundra skulle betyda att fläkten skulle blåsa ut med större effekt än vad den kan.




*Figur 3 Figuren visar reglerings resultatet av en pingisboll vid börvärdet 25cm. I figuren syns styrvärdet (grön), sensorvärdet (blå), börvärdet (gul) samt felet (röd).*

## Sammanfattning och diskussion

Programmet som utvecklats klarar av uppgiften att reglera en pingisboll. Regleringen blir dock aldrig helt exakt, vilket anses bero dels på utrustningens utformning men även att det var svårt att integrera så många aspekter. Programmet klarar även av att plotta en graf i realtid med värden från Arduino. Här finns dock vissa brister, då programmet inte kan hantera decimaltal vid kommunikationen.

Om sampeltiden för regleralgoritmen är större än den för tasken kommer regleringen pågå längre än vad den borde. Det betyder att när samma task startar nästa gång kanske den föregående regleringen inte är klar än, vilket kan medföra att data går förlorad.

Värdena som räknades fram med Ziegler Nichols metod ansågs vara tillräckligt bra när vi testade dem. Därför ändrades aldrig dessa. Detta motiveras med att det inte finns någon mening med att ta fram perfekta värden här, när vi inte gjort det någon annanstans i systemet.

## Källförteckning

Datasheet distance sensor for GP2Y0A21YK0F, SHARP