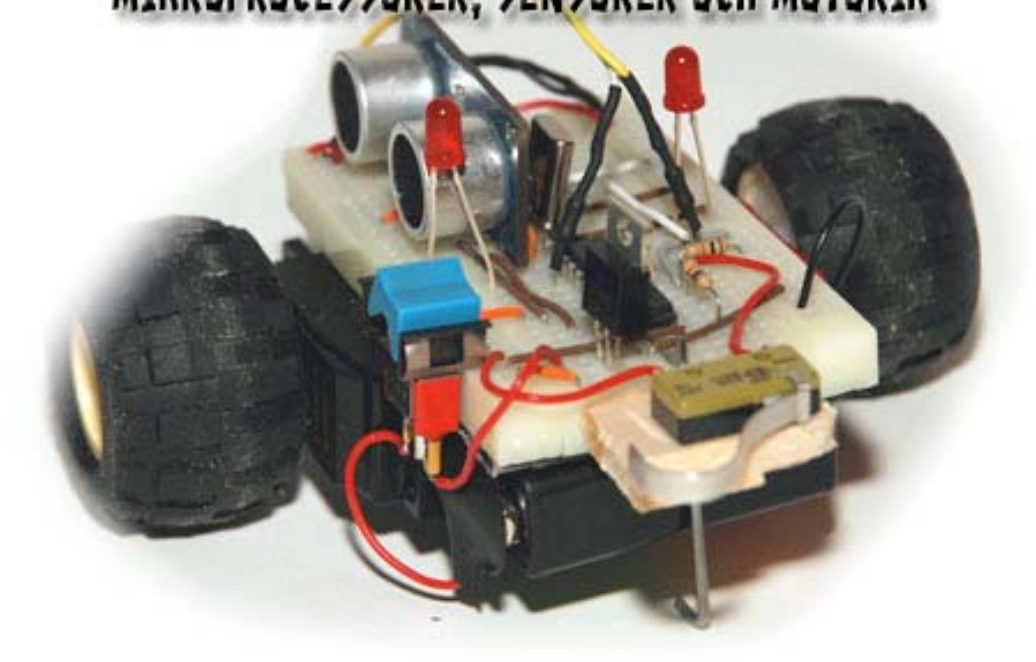
ROBOTAR

MIKROPROCESSORER, SENSORER OCH MOTORIK



AV

FREDRIK OLSEN

SEBASTIAN ÅKESSON

TE3B, POLHEMSKOLAN, LUND

INNEHÅLLSFÖRTECKNING

INLEDNING & BAKGRUND

Projektet startade med ett intresse för robotar och robotik.
Eftersom robotar finns överallt i samhället idag och processorer används till i princip allting var det ett område som man kunde göra mycket inom och som hade ett väldigt brett utbud.
Det här var dessutom ett område som vi inte hade lärt oss någonting om tidigare och vi kände att det fanns stort utrymme för utveckling och vi visste att vi var tvungna att lära oss väldigt många saker innan huvudmålet av projektet kunde uppnås.

Sedan tidigare projekt då vi gjort fjärrstyrda robotar ville vi ta steget till en autonom robot, möjligheterna ökar men också svårighetsgraden. Det är ju idag autonoma robotar som rensar avloppen, de övervakar korridorer och de gör alla andra monotona tråkiga arbeten som människorna inte vill göra själva så det var ett spännande område att forska inom.

För att vi först skulle veta vad en autonom robot var och hur vi skulle bygga den var vi tvungna att ta reda på en autonom robot var och vad den bestod av.
Halva denna rapport kommer att innefatta vad en autonom robot består av och hur den ”tänker”. Andra halvan kommer att bestå utav hur vi gick till väga för att försöka bygga vår egen robot samt de problem och framgångar vi stötte på.

MIKROPROCESSORER

I vårat projekt kommer vi att behöva använda oss utav en processor för att hantera drivning samt kunna känna avståndet till väggen och hantera signalen från mikrobrytaren.

Under första etappen av projektet då vi undersökte vad vi skulle använda för processor kollade vi på flera olika alternativ, det som främst kom upp var tre olika typer och dessa ska vi koncentrera oss på och titta lite närmre på.
Dessa tre typer är PIC, AVR och Basic Stamp.

PIC

PIC processorn är tillverkad av Mikrochip, ett företag som ligger beläget i USA med fabriker i bland annat Taiwan.
PIC processorn är byggd med RISC arkitektur, Reduced Instruction Set Computer, detta innebär att processorns instruktioner är få och enkelt utformade. Med detta kan man skapa en väldigt högpresterande kod och eftersom varje operation är enkel kan instruktionerna hämtas i väldigt snabb följd. Med att processorns instruktion är enkelt utformad menar man att koden är kort skriven och i princip på maskinnivå.
Processorn har 35 instruktioner, bland andra funktioner såsom AND, NOT, OR och NOR.
Processorns hastighet är på maximalt 20MHz, vilket betyder att den kan utföra 20 miljoner sådana instruktioner per sekund. Den processorn som vi tittade lite närmre på hade 32 I/O kanaler, 4st A/D-omvandlare och 2 PWM kanaler, vilket resulterar i att man kan ha väldigt mycket inkopplad till denna. Processorn har en inbyggd WDT, Watchdog Timer, vilket man kan säga är en klocka, baserad på en oscillator signal. Denna är väldigt användbar när man vill synka operationer för att få en viss frekvens, vilket t ex är nödvändigt när man behöver styra servon och kontrollera pulser.

Slutsats om just denna processor är att den är väldigt simpel och lättanvänd med ett enkelt sätt att koda som man snabbt kan sätta sig in i. Dock har den några nackdelar, framförallt om man skall göra lite mer avancerade program och inte vill skriva Assembler eller om det är så att man behöver något kraftigare än en 8-bitars struktur.

AVR

AVR microcontrollern är även av RISC arkitektur, dock är denna tillverkad av ett annat företag kallat Atmel som är ett norskt företag.
På samma sätt lyckas också den här processorn exekvera nästan alla instruktioner i en klockcykel, vilket resulterar precis som med PIC:en i väldigt hög effektivitet. Den största skillnaden mellan PIC:en och AVR:en är att AVR processorn är skapad för att köras med kompilerad C-kod. Den är alltså mycket mer lämpad för att programmeras med högnivå språk istället för maskinnivå språk. Den allokerar även minnet på ett annat sätt vilket gör främst att kodningen kan läggas på en mer flytande del av minnet och inte behöva ta så strikt plats som det måste i en PIC processor. En annan orsak till varför AVR processorn har blivit så populär är att den tillkommer med gratis programvara för utveckling kallat AVR Studio, det finns även programvara som stödjer Linux, vilket är ganska ovanligt.
En annan stor skillnad är att AVR processorn kan fås med en 16-bitars timer, som gör det möjligt för fler och mer avancerade operationer.
Atmel stoltserar även med att det är marknadens mest strömsnåla processor och har ett spänningsomfång på 1,8-5,5 volt för att kunna köras.

Basic Stamp

BASIC Stamp processorn skiljer sig väldigt mycket från de andra processorerna.
Då de andra är enchips processorer. BASIC Stamp är en samling chip på en krets, där hela kretsen betraktas som en processor i sig.
Detta på grund av att BASIC Stamp har en inbyggd BASIC tolk, vilket gör det väldigt enkelt att programmera den. BASIC är ett lätt språk att lära sig och detta är en av anledningarna till att BASIC Stamp har blivit väldigt populär inom vissa användningsområden.
På grund av att det är en krets och inte ett chip har den även några andra fördelar, däribland att den har fler funktioner och konfigurationsmöjligheter. Till gengäld förlorar man hastighet och i snitt har BASIC Stamp processorn inte mer än en hastighet av ca 10 000 instruktioner per sekund. Vilket i och för sig skulle räcka mer än väl för vårat projekt, men kanske inte för alla.

Tillämpningar

Mikrokontrollers eller även kallade enchipsdatorer används på väldigt många platser i dagens samhälle. Oavsett vilken elektronisk pryl du öppnar kan du hitta ett litet svart chip som är en processor, dessa används i mikrovågsugnar såväl som bilar. Alla maskiner som har input och output signaler, med undantag för en del industrimaskiner som använder PLC, använder idag mikrokontrollers för att hantera signalerna.
Just på grund av detta är det också ett väldigt spännande område att arbeta i och utvecklingen går fort och möjligheterna ökar mer och mer.

Att använda mikrokontrollers i robotar är ett krav för att den skall uppnå någon form av autonomi. Olika avancerade typer av processorer används beroende på hur mycket data den skall behandla och hur mycket en behöver ”tänka”. I våran robot behöver vi bara en väldigt enkel PIC med ett fåtal instruktioner och 6 I/O kanaler, dock för att kunna bygga till fler moduler och utveckla den vidare kan det vara praktiskt att kunna använda andra processorer med fler kanaler.

Programmering

Som vi skrivit innan så finns det lite olika sätt att programmera en mikrokontroller. Om man skall få en grundlig förståelse för hur processorn arbetar så bör man börja med att koda i assembler, ett språk där man kodar på instruktionsnivå och arbetar med att flytta bitar till olika register.

Personligen upptäckte vi dock tidigt att assembler tar lång tid att programmera, det blir effektiv kod men det tar lång tid och kräver djup kunskap för att uppnå de lite mer avancerade programmen. Därför gick vi vidare med att försöka hitta fler språk och undersöka andra möjligheter. Vi hittade ett par alternativ.

Basic är ett språk som används i t.ex. BASIC Stamp, men det finns motsvarigheter för PIC och AVR också, PICens motsvarighet kallas PicBasic och har samma struktur som Basic men inte alla funktioner på samma sätt. Alla grund funktioner som if, while, och foreach är uppbyggt på samma sätt men det finns färdiga funktioner som PULSIN och PULSOUT för att göra program med pulsning lättare, dessa färdiga funktioner motsvarar dem som finns till BASIC Stamp.

Om man går upp ytterligare en nivå i språket så kommer man till C.
Det finns en motsvarighet till C som man kan använda till miktrokontrollers som heter mikroC. Detta fungerar precis som vanlig C kod med samma syntax, på samma sätt finns det färdiga instruktioner och funktioner här men de är inte lika övergripande som PicBasic så på det sättet kommer man lite närmre Assembler med mikroC.

När man kodar i PicBasic, mikroC eller annan motsvarighet av ett högnivåspråk så kommer kompilatorn att göra om allting till Assembler och sedan HEX, detta läggs senare in på processorn. Så oavsett vilket språk man använder kommer man alltid att få ut Assembler koden så att man kan studera den i närmre detalj eller finjustera den för att göra koden mer effektiv eller förstå bättre hur den arbetar.

Många som programmerar enklare varianter av robotar och använder PIC eller AVR använder ofta just den kombinationen av mikroC eller PicBasic och i efterhand justerar Assembler koden för att få allting att fungera lite bättre.

SENSORER

Robotar är uppbyggda och beroende av sensorer.
För att en robot på något sätt skall kunna klassas som intelligent måste den på ett eller flera sett känna av sin omgivning. Den måste kunna uppta ljus, vibrationer, och tryck för att kunna veta hur omgivningen ser ut och beter sig och därefter anpassa sig efter det. Man brukar säga att ju fler sensorer en robot har desto högre grad av artificiell intelligens kan man uppnå.
Vi tänkte redogöra för några av de sensorerna som finns, och vi sorterar dem efter vilken funktion de har hos människan.

Syn

Kamera

Vanliga kameror har i princip de egenskaper som människors syn har.
De plockar upp ljusvågor i ett visst intervall och presenterar en bild i färg. Denna bild behöver sedan behandlas av mjukvara för att roboten skall kunna tolka vad som finns i den, detta görs av väldigt avancerade algoritmer för hur olika saker och ting ser ut.

Termografisk kamera

Detta är i vanlig mån vad man kallar för värmekamera eller IR-kamera.
Den plockar upp ljusvågor i det infraröda spektrat och presenterar varma saker som ljust och kalla saker som mörkt, med detta kan en robot lokalisera t.ex. eld och dylikt, men även använda det för att lokalisera däggdjur och andra varmblodiga varelser.
Termografiska kameror fungerar i ljus, mörker och varierande väder till skillnad från vanliga kameror som kan fungera olika bra beroende på klimat och omgivning. Det krävs dessutom mindre avancerade algoritmer för igenkänning av t.ex. en människa, dock får man inte samma grad av detalj och kan t.ex. inte få fram ansiktsigenkänning.

Hyperspektral kamera

Hyperspektrala kameror är en ganska ny företeelse och det är helt enkelt en kamera som använder sig av ett väldigt brett utbud av ljusspektrat för att identifiera vilket material ljuset kommer ifrån.
Satelliter använder sig av hyperspektrala kameror för att se olika mineraler på planeter som ligger under jord och om man riktar den mot en människa kan man se genom huden.

Hörsel

Mikrofon

Mikrofoner delar normalt upp i två grupper. Oriktade och riktade mikrofoner, även kallat Omnidirectional och directional från engelskan. De engelska orden förklarar grupperna bättre, en oriktad mikrofon plockar upp ljud från överallt runtomkring den, en variant av detta är den vanliga studiomikrofonen vi ser hängande ifrån taket.
En variant av en riktad mikrofon är t.ex. parabolmikrofonerna som har en parabolisk disk för att samla upp ljud från flera hundra meter bort och sedan skicka in det samlade ljudet till en mikrofon.
Mikrofoner i en robot kan användas för allting från att höra om någonting närmar sig eller bestämma i vilken riktning den står till röstigenkänning.
Ultraljudssensorn är en form av mikrofon där den skickar ut ljud och väntar på att det skall reflekteras tillbaka för att sedan kunna mäta t.ex. avstånd.

Hydrofon

En hydrofon fungerar i princip som en vanlig mikrofon fast istället för att mäta ljudvågor (vibrationer) i luft mäter den ljudvågor i vatten. Det är vanligt att dykrobotar utrustade med hydrofoner åker ner till platser under vattnet där det är olämpligt för människor att vara. Det används ofta för att lokalisera u-båtar men har också forskningssyften som att lyssna på när valar pratar på flera mils avstånd från varandra.

Seismograf (Seismisk sensor)

En seismograf är som de andra typerna av mikrofoner en sensor för att mäta vibration, dock mäter denna vibrationer i fasta material. Som t.ex. vibrationen i jordskorpan eller ett golv. De kan plocka upp en jordbävning som sker på andra sidan jorden eller kan justeras för att plocka upp fotsteg varsomhelst i ett rum och dessutom lokalisera ungefär var de kom ifrån och vilket avstånd.
Dessa kan användas för att roboten skall veta var det finns någon gående, eller t.ex. programmera en dörr att öppna sig när den känner att det kommer någon.

Känsel

Mikrobrytare

En mikrobrytare är en helt vanlig brytare som oftast bara har två lägen.
Den känner av om någonting är intryckt eller inte, ofta använder man detta på lite enklare robotar som ”stötfångare”, när den kör emot någonting så bryter den och den vet att den har kört emot och försöker då undvika det som varit i vägen.

”Nervsensorer”

Nervsensorer är en ganska högt utvecklad typ av sensor som mäter tryck. Ungefär som en våg fast oftast för mycket mindre krafter. Dessa används oftast i humanoida robotar som försöker efterlikna människor, det är väldigt små sensorer som i princip simulerar nerver och sätts t.ex. i fingertoppar och leder för att känna hur hårt den kramar någonting och ökar balansen.

Belastningsmätare

En belastningsmätare är helt enkelt en elastisk sensor som kan se ut på lite olika sätt. Den används oftast för att mäta deformationen av objekt. Man brukar sätta den i ben och armar och liknande på humanoida robotar (eller androider som de kallas) för att mäta belastningen som sätts på en led så att roboten inte tar sönder sig själv.

Miljösensorer

Miljösensorer är egentligen mer en undergrupp till känselsensorer, det är sensorer som en robot kan behöva för att känna av sin omgivning så att den inte hamnar i en omlämplig miljö. I den här gruppen ingår t.ex. fuktsensorer, termometrar och magnetfältssensorer. Det kan vara väldigt praktiskt att ha många sensorer för att känna till miljön roboten befinner sig i. Om det blir för varmt när den är på väg någonstans kan den t.ex. veta att backa tillbaka och hålla sig borta därifrån. Om den känner av att det helt plötsligt uppstår ett magnetfält som kan vara farligt för den bör den t.ex. slå på ett eget magnetfält för att motverka det, eller om den kan bege sig till någon säkrare plats.

Lukt och smak

Känslor som lukt och smak är självklart en trivial sak och är ingenting som man egentligen kan tillskriva en robot eftersom det är upp till var och en att bestämma vad som är lukt och vad som t.ex. smakar gott. Människan vet inte idag vad lukt och smak är och därför kan vi inte skapa någonting för att känna igen detta. Dock vet vi att både lukt och smak uppstår ifrån olika kemiska reaktioner och partiklar. Därför finns det två sensorer i denna grupp som är intressanta att titta på.

Kemiska sensorer

En kemisk sensor består utav ett element som reagerar med det den vill känna av.

Varje kemisk sensor är annorlunda beroende på vad man vill uppnå med den, om man vill känna av någon gas, t.ex. klor så använder man ett element i sensorn som reagerar med klor. När vissa reaktioner uppstår så sker det en spänningsutveckling och detta kan den i sin tur mäta för att se om det sker en reaktion eller inte. Dessa sensorer är relativt billiga att använda och tål mycket till skillnad från biosensorer, dock är de inte lika exakta.
Kemiska sensorer är någonting som används i dag, speciellt i det militära.
För att se om ett område är täckt i någon skadlig gas så kan de skicka in robotar för att känna av om miljön är säker för en människa att befinna sig i.

Biologiska sensorer

Dessa sensorer är känsligare och bättre än kemiska sensorer men är också dyrare och svårare att tillverka. I en biologisk sensor har man istället för ett element av något material en biologisk komponent, som till exempel enzymer eller klorofyll.
Om man har t.ex. en sensor med klorofyll kan man använda den för att varna när man får för mycket koldioxid i ett rum eller när någonting i roboten är trasigt (med avgassystem och dylikt).

Sjätte sinnet

Det som ibland kallas för sjätte sinnet hos människor är ett av våra viktigaste sinnen för att kunna överleva. Detta är kunskapen om i vilket läge alla delar av vår kropp befinner sig i samtidigt utan att behöva titta.
På grund av detta kan vi röra oss väldigt smidigt och vi kan göra koordinerade rörelser utan att behöva anstränga oss för att hålla minutiös koll på varje del av våran kropp. Detta är även någonting som är väldigt viktigt för robotar för att de skall kunna röra sig koordinerat och inte krocka med vissa delar av sig själva.
För detta finns det tre vanliga sensorer som används.

Magnetometer (eller kompass)

En magnetometer mäter helt enkelt magnetfält. Om det inte finns några störande magnetfält så kan mäter den jordmagnetfältet och det en enkel kompass.
Nackdelen med magnetometrar är just deras känslighet, de kan lätt förvirras av andra magnetfält och om man flyttar roboten långa sträckor när den är avstängd kan man behöva konfigurera om den eftersom jordmagnetfältet förändras beroende på var man är i världen. Ett sätt för att göra den lite bättre är dock att använda ett minne på den, om det skulle uppstå en stund då den känner av för många olika magnetfält av olika styrkor så kan den gå tillbaka till tidigast kända läge och försöka beräkna i vilket håll nord *bör* ligga.

Accelerometer

Människan använder sig utav innerörat för att bestämma vår position och vår nuvarande acceleration i förhållande till gravitationen. På samma sätt använder robotar en accelerometer. Den kan känna i vilket håll jordgravitationen ligger och rapporterar position i förhållande till det så den t.ex. kan känna om den är upp eller ner. Detta är robotens balansorgan.

Gyroskop

Ett gyroskop är med magnetometern en av de vanligaste sensorerna för att bestämma riktning. Ett enkelt gyroskop är ett hjul på en axel som snurrar, och när hjulet ändrar orientering så uppstår det en resistans som man kan mäta, på så sätt kan man alltid hålla koll på vilken riktning roboten har i förhållande till ursprungsriktningen. Den här kräver ju dock att man sätta ner roboten i en sådan riktning att den vet var den är

från början. Gyroskop är till dess fördel även användbara i yttre rymden där varken magnetometer eller accelerometer går att använda.

Övermänskligt

I viss mån har roboten överstigit människan i dess avkänningsförmåga, robotar har utvecklats för att hjälpa människan. Och det mest självklara att uppfinna är någonting som vi själva saknar. Här kommer en kort redogörelse för de kunskaper som robotar besitter men vi människor har väldigt svårt med eller saker som faktiskt är helt omöjligt för oss att göra.

Avståndsmätning

Även med ögonen öppna och med tillåtna referenspunkter är människan väldigt dålig på att bestämma exakt hur långt den har gått.
Fördelen roboten har är att kunna mäta exakt hur många varv dess hjul har snurrat och sedan kunna räkna ut hur långt den gått genom att veta diametern på hjulet. Men även med en humanoid robot kan den beräkna hur långt den har rört sina ben.

GPS

Många robotar idag utrustas med GPS mottagare, detta innebär att roboten kan veta exakt med ungefär 10cm precision var den befinner sig i världen. Den kan dessutom då också veta exakt hur långt kvar det är till ett mål och kan snabbt räkna ut vilken tid det skulle ta att ta sig från punkt A till punkt B, då också tagit i beräkning med att fågelvägen inte alltid är möjlig eftersom den kan använda diverse kartprogram för att synka med positionen.

Radar

Radio detection and ranging. Radar kan en robot utnyttja för att snabbt och precist veta exakt var den befinner sig, men även var andra saker befinner sig och i vilken hastighet de rör sig, samt vilken riktning.
Med detta kan den alltså direkt få in enormt mycket data och information om sin omgivning och hur den bör agera och röra sig för att undvika hinder eller undvika att vara i vägen för någonting som kommer mot den.

Ladar

Laser detection and ranging. Till skillnad från radar så är laser mer exakt och kan skapa en bättre bild av omgivningen. Hastigheten på signalen är 3*10^8 m/s istället för 300m/s, därav kan den göra en mycket större skanning på mycket mindre tid och dessutom är vågorna mycket mindre och kan alltså urskilja mindre objekt.
Med en ladar kan roboten på några få sekunder få in all information som från en radar men dessutom kunna måla upp en hel bild i 3D av hur dess omgivning ser ut och rör sig, detta utan att behöva snurra hela roboten, ungefär som att ha ögon runtom hela kroppen.

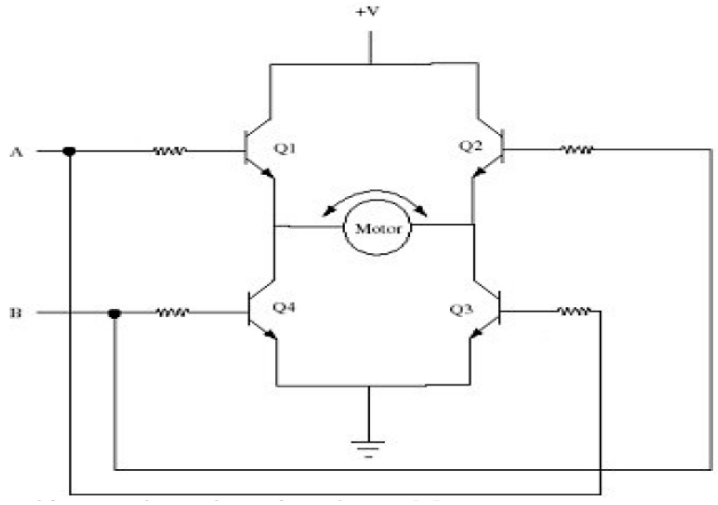
MOTORIK

Likströmsmotorer

I en likströmsmotor finns det en mekanisk del som kallas kommutator, eller omkopplade.
I mitten av motorn sitter en rotor som är en elektromagnet med nord och syd polen i en riktning 90° mot axeln, i sidorna av motorn finns en permanentmagnet.

Detta gör att rotorn vill ställa in sig med nord mot nord och syd mot syd.
Omkopplarens uppgift är att göra så att när nord står mot nord skall den skifta strömriktningen i rotorn så att det istället blir nord mot syd, detta gör att motorn kommer fortsätta vilja snurra.

Likströmsmotorer används idag inom de flesta små maskiner. Många använder dagligen likströmsmotorer utan att de knappt vet att de gör det.
Många maskiner som snurrar eller rör sig med konstant hastighet har en likströms motor. Men för att styra likströmsmotorn kan man använda sig av lite olika sätt.
Ett väldigt vanligt sätt är att använda sig av en H-brygga.
H-bryggan består av 4st olika transistorer, samt 4st resistorer. Därför man använder en H-brygga är oftast för att man vill kunna ändra riktningen på motorn. I elbilar finns ofta H-brygga för att bilen ska kunna backa, även i andra sammanhang behöver man kunna ändra rotationsriktningen på motorn. Behöver man bara ha en rotationsriktning på sin likströmsmotor kan man koppla in den direkt till spänningskällan.



Stegmotorer

En stegmotor styrs genom att man skickar pulser, pulserna skickas till ett drivsteg som skickar ut strömmen till motorns lindningar. Drivsteget ser också till att strömmen kommer i rätt ordning för att motorn ska rotera åt önskat håll.
Pulstågen som man skickar bestämmer hur fort motorn ska snurra (varv/min).
Förhållandet mellan pulstågen och steg/s bestäms av motorns storlek.
Rotationsriktningen bestäms oftast med en särskild riktningsingång till drivsteget, det gör att strömmen skickas åt andra hållet, precis som på en H-brygga.

Hur stark en stegmotor är beror på många olika saker, men generellt kan man säga att det beror på 2olika saker. Dels beror det oftast på vilken typ av spänningskälla man kopplar till motorn, men det beror också på hur många lindningar motorn har.

Stegmotorn är i jämförelse med en likströmsmotor väldigt annorlunda. Stegmotorn är väldigt exakt, tyvärr så går det inte att få upp en stegmotor i mycket mer än 1500 varv/min. En likströmsmotor kan mycket väl snurra på i 15-20000 varv/min.
Momentet är dock betydligt kraftigare i en stegmotor.
Det finns vissa regler för hur stegmotorn fungerar när det gäller momentet och hastigheten.
Hög ström gör att vridmomentet blir kraftigt vid låga varvtal.
Hög spänning gör att vridmomentet blir kraftigt på höga varvtal.

Så när man bygger något med en stegmotor får man se var det är man vill att motorn ska vara stark. Tyvärr är det så att det inte bara är att dra upp allting på max. Stegmotorn är väldigt känslig för överbelastning. Då tappar motorn all kraft och i värsta fall så stannar den tvärt och slutar fungera helt. Motorn ska inte heller utsättas för några försök att tvinga den att snurra utan att drivspänningen är inkopplad.

Servo

Ett servo kan se ut på många sätt och användas i många saker.
Oftast klassar man de i ett antal olika kategorier, modellservo, flygservo och industriservo t.ex.
I ett servo finns styrkrets, en likströms- eller stegmotor och en växellåda.
Man använder oftast servon för att det är hela paketet i ett och att de ofta ger väldigt hög precision med mycket kraft. Flygplan använder dem till att styra roder och diverse maskiner i industrin kan använda dem för reglering eller annat precisionsarbete.
Eftersom vi använder modellservo i vårat projekt tänkte vi titta lite närmre på dessa.
Här är det oftast en likströmsmotor som sitter i och snurrar runt 10 000 varv/min
Ovanpå servot sitter det sen ca 5 kugghjul för att växla ut dessa 10 000 varv till ungefär 70-80 varv/min.
En intressant sak med modellservon är att det sitter en potentiometer i dem som är kopplat till styrkretsen och potentiometerns vridning följer axelns vridning.
Styrkretsen använder då sedan motståndet i potentiometern för att bedöma vilken position axeln har, med hjälp av detta skapar man servo som har ett vridningsmoment på ca 180°.
I vårt projekt ville vi använda servot som motor dock, för att göra detta måste man modifiera servot en del, bland annat ta bort potentiometern så att den inte längre vet var den är och på så sätt bara fortsätter att snurra.
Här nedan tänkte jag göra en kort beskrivning om hur detta går till:

1. **Öppna servot**
   Det finns ett antal saker man måste tänka på här.
   Oftast öppnas servot genom att lossa 4 skruvar undertill.
   Var dock uppmärksam!
   Det finns nämligen två platser som servot öppnar sig, locket på toppen och ett lock i botten lossas, under locket i toppen sitter alla kuggar, så när du tar av detta måste du vara väldigt noggrann med att notera hur kuggarna sitter så att du kan få tillbaka dem på samma plats senare.
   Under botten ser du styrkretsen, på många servon behöver du inte göra någonting där, men vissa kan du behöva lossa styrkretsen också, vi kommer till detta i nästa steg.
   Se också till att servot är mittställt (alltså axeln är centrerad) när du öppnar det, detta kommer underlätta senare.

2. **Ta bort potentiometern**
   För att inte servot skall veta var det är får det inte lov att finnas en potentiometer, med denna kopplad till axeln kommer det nämligen inte att kunna fortsätta snurra i oändlighet.
   På olika servon finns det lite olika sätt att ”avaktivera” potentiometern, om man tar bort den helt, alltså klipper av den från styrkretsen kommer inte servot att fungera längre. Den måste ha ett motstånd, hela syftet med att göra allt detta är för att lura servot att den är i mittläget och den skall alltså vilja fortsätta snurra i maxfart.

Öppna toppen på servot, ta bort kugghjulet som är den axel man ser utifrån. Under detta kommer du se potentiometern sticka upp, om den är i plast kan du klippa av den, om den är i metall måste du öppna botten också och försöka vika undan den in i servot, där finns ofta mycket tomrum under ”växellådan” bredvid motorn.

3. **Ta bort mekaniska spärren**
   På servot finns också en mekanisk spärr, en liten plasttapp som gör att kugghjulet inte kan snurra förbi en punkt. Klipp bort denna tapp, var dock försiktig, om tappen sitter på ett kugghjul kan kugghjulet lätt spricka.

4. **Klart!**
   När du nollställt potentiometern och gjort så att den inte längre följer axeln och tagit bort den mekaniska spärren är det bara att sätta igång och använda det som motor. Skall man koppla det till en processor bör du använda pulser med 1ms topp och 1,6ms botten för att gå åt ett håll och 2ms topp med 1,6ms botten för att gå åt andra hållet, vilken längd på intervallet mellan topparna kan dock variera och det är inte så jättenoggrant, olika märken av servon vill dessutom ha olika längder här, topp-pulsen på 1-2ms brukar dock väldigt sällan avvika.

VÅR ROBOT

Grundtanke

Från början ville vi bygga en robot med en värmesensor som letade upp ett ljus, körde fram till det och släckte det.
Roboten skulle använda en TPA81 värmesensor för att plocka in värden från omgivningen och bestämma var den varmaste punkten befann sig. Därefter skulle den med hjälp av ett servo rikta in ett hjul mot den varmaste punkten. För kommunikationen mellan sensor och servo skulle vi använda en processor. Denna processor skulle också styra motorerna för drivning. Kommunikationen mellan sensor och processor skulle ske med I2C protokoll och vi hade tänkt använda Assembler som språk för att programmera processorn.
Efterhand upptäckte vi dock att våra förkunskaper var för bristfälliga för detta.

Metod

Innan projektet startade började det med att vi ville fortsätta lite grann från ett tidigare projekt i Konstruktion A där vi byggde en fjärrstyrd robot. Vi började med att undersöka vad det fanns för typer utav robotar och kom fram till att vi ville göra en autonom. Vi visste dock inte riktigt vilken typ det skulle vara så för att bestämma det tittade vi runt lite i olika butiker på nätet för att se vad man kunde hitta för sensorer man kunde använda. Vi hittade en rimligt dyr värmesensor, TPA81 som vi tyckte kunde vara väldigt kul att använda. Och idén med roboten som skulle släcka värmeljus föddes. Vi tittade igenom databladen för sensorerna och för processorn och vi hittade lite information på Internet om hur man kunde använda dem, dock inga konkreta exempel med kod. Efter detta gjorde vi upp en tidsplan på hur vi trodde att projektet skulle se ut, denna finns i bilagor. Vi bestämde oss för att beställa värmesensor, processor och en programmerare i byggsats för att kunna programmera processorn trots att vi inte hade några kunskaper om varken I2C protokoll eller mikroprocessorer överhuvudtaget.

När vi fått grejorna byggde vi ihop programmeraren, den var av den typ som också hade en liten labb-del med knappar som input signaler och LED lampor som output signaler. Vi började därefter med att programmera lite i Assembler och försöka göra små program för att lamporna skulle göra olika saker, så som blinkande och ”rinnande” ljus. När vi fått lite grepp om det där med hur den skulle programmeras så testade vi att köra den separat på en labbplatta. Här var vår första motgång, vi fick inte detta att fungera alls.
Så vi fortsatte att labba på programmeraren, vi fick bygga om den lite så att den hade sladdar ut istället för LED lamporna. På detta sätt kunde vi då koppla in avståndssensorn och få den att fungera. Innan det så var vi dock tvungna att ta beslutet att byta från Assembler till PicBasic eftersom vi hade exempelkod till sensorn som var skrivet i PicBasic och vi hade inte tillräckligt bra kunskap i Assembler för att kunna skriva om det i det.

Vi fortsatte att labba på det här sättet och försökte få värmesensorn att fungera. Detta var dock helt utan framgång, eftersom vi inte hade någon möjlighet att lagra något resultat och titta på det i efterhand och ingen möjlighet att debugga på något vettigt sätt blev det ännu svårare.

Efter många timmars testande och labbande förgäves försökte vi fråga på Elektronikforumet.com, dock fick vi inga vettiga svar där.

Vi läste runt mer på Internet för att försöka hitta svar men fanns istället att vi skulle få det svårt att använda stegmotorer.
Från början tänkte vi använda stegmotorer för att driva bilen och styra armen för att släcka ljuset men fann att det var dyrt och krävde extra drivkretsar om det inte skulle använda väldigt många I/O-portar, som vi inte hade särskilt många utav.

Efter ytterligare många timmars huvudbry och besvär kom vi fram till att vi var tvungna att ta ner hela projektet ett steg eftersom vi inte hade tillräckligt med kunskap. Vi bestämde oss för att använda avståndssensorn eftersom vi hade fått den att fungera. Vi beslutade oss för att göra en robot som tog sig ur en labyrint istället, detta skulle bara kräva att den kunde följa en högerkant och detta verkade mycket enklare. Till detta projekt skulle krävas avståndssensorn i högerled och en mikrobrytare längst fram för att säga när den körde på något.

Vi insåg dock att vi också var tvungna att leta upp någonstans där man kunde läsa om hur man skall koppla mikroprocessorn fritt från programmeraren, på en egen labbplatta alltså. Vi fann en bra guide på en privatpersons sida (se källor) som visade hur man skulle koppla den. När vi fått ihop allting så kunde börja labba lite mer seriöst, men vi hade fortfarande ingen bra kompilator. Vi använde en demo version av PicBasic Pro kompilator, nackdelen med denna var att man endast fick använda sig av 35 rader kod. Detta resulterade i att vi fick köra lite kod åt gången när vi skulle laborera.

När vi undersökte alternativ till drivning med stegmotorer hittade vi servo drivning som många robotar använder. Servon hade vi sen tidigare eftersom vi båda har varit aktiva modellflygare.
Fredrik hade tidigare plockat isär ett sådant för att få det att snurra i oändlighet och han upprepade detta på ett till servo så att vi hade två stycken vi kunde använda som drivning. Vi åkte därefter ner till SkåneHobby i flädie och inhandlade ett par billiga hjul som vi satte på servona, vi upptäckte dock att hjulen var lite skeva och eftersom de var gjorda i skumgummi gick de inte rakt. Detta gjorde att bilen inte heller gick rakt, men det fick vi bortse ifrån och försöka lösa med kodning så att den kompenserade så mycket med ett hjul att den gick rakt.

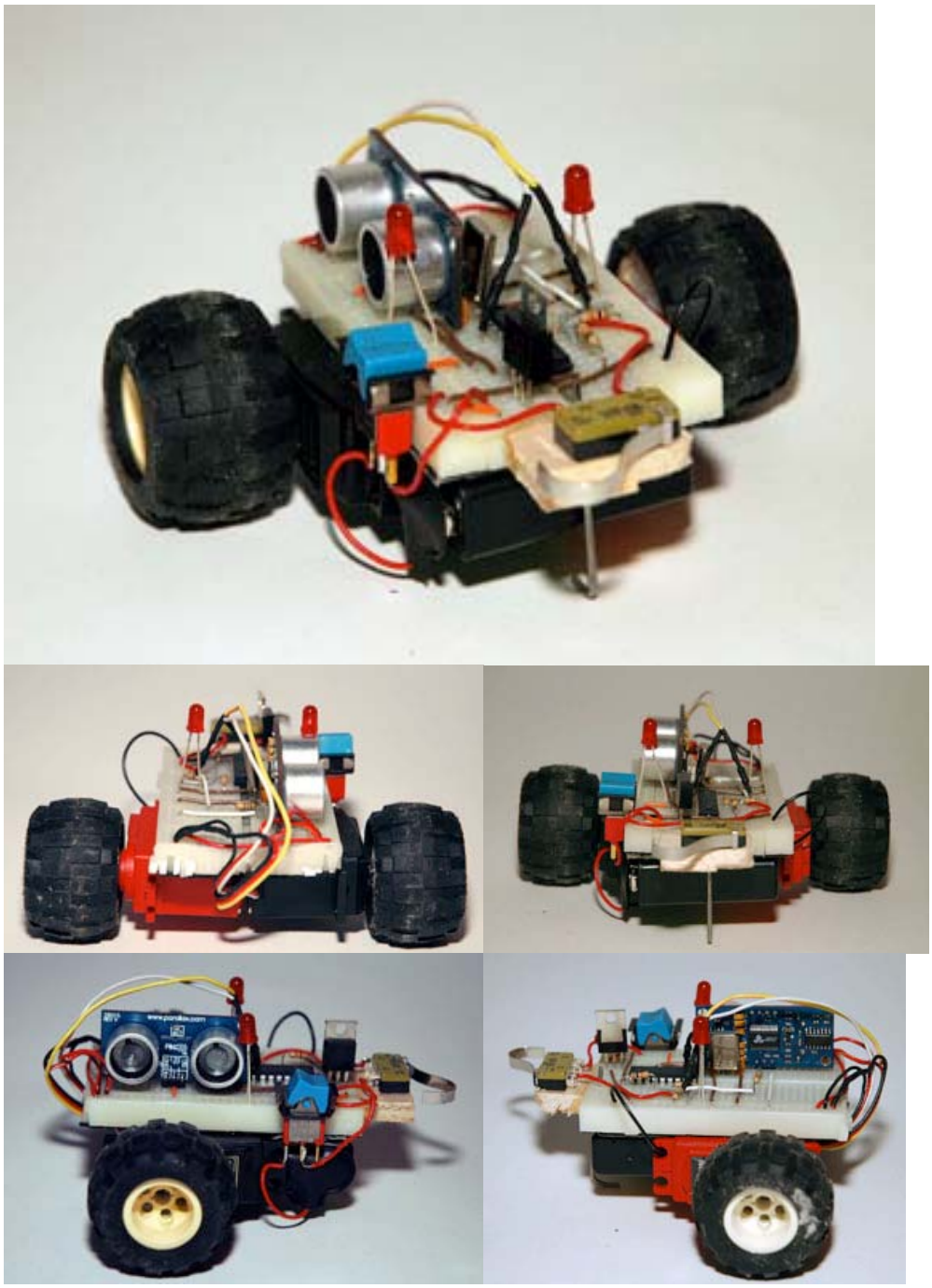
Vi var tvungna att ha en strömkälla till bilen, eftersom vi inte ville att den skulle dra en sladd efter sig. Detta löste vi genom att använda ett 9v batteri och en 5v spänningsregulator. Spänningsregulatorn ger en utspänning på 5v och accepterar inspänningar från lite högre än 5v (där finns ett spänningsfall) till 30v.

För att roboten ska känna av när den kör in i väggen så placerade vi en mikrobrytare längst fram. Vår första mikrobrytare var väldigt stor och krävde en stor kraft, det gjorde att roboten slirade med hjulen utan att trycka in brytaren. Därför var vi tvungna att köpa en lite mindre brytare, den nya brytaren krävde en väldigt liten kraft för att bryta.

Eftersom våra ursprungliga hjul var väldigt dåliga gick roboten snett, detta försvårade projektet väldigt mycket och många timmars kodning gick åt för att bara försöka få den att gå rakt och sen dessutom reagera som den skulle i samband med att den skulle följa en vägg.
Till slut fick vi ett någorlunda bra resultat, den klarade att ta sig igenom banan men fick lite kärv och stopp då och då.

Till slut hittade vi ett par andra hjul som passade, dessa var mycket bättre då det var riktig gummi och vi kunde fästa de på servot väldigt enkelt.
När vi satt på dessa gick roboten felfritt rakt fram. Detta resulterade även i att roboten gick mycket, mycket bättre i labyrinten. Nu klarar den att ta sig igenom den varv på varv utan att göra några misstag, ibland så händer det fortfarande något oväntat, men trots det så brukar den kunna ta sig ur det av sig själv.

Resultat



Som ni ser på bilderna ovan fick vi till slut fram någonting som liknar en robot. Resultat blev en autonom robot som drivs med två servon och använder sig utav en ultraljudssensor för att känna avståndet till höger kanten. Om den kommer för långt ifrån högerkanten så börjar den köra det ena hjulet mer än det andra för att närma sig väggen igen, kommer den för nära svänger den ut. När avståndet till högersidan blir väldigt stort så gör den en 90° sväng höger, då detta i ideella fall betyder att det har kommit en högersväng. När den kör emot en vägg betyder detta att den har kommit in en hörna och enda vägen ut är en vänster sväng, brytare känner av detta och då skall den alltså göra en vänstersväng. På detta sätt lyckas den (oftast) ta sig igenom en labyrint helt och hållet. Se koden för mer detaljerat rörelseprogram.

Utvärdering

Vid utgångspunkten av projektet såg det ut att kunna bli ett bra sådant och såg ut att kunna genomföras på lagom tid av ca 100 timmar per person, tagit i tanke att vi inte hade någon egentlig kunskap i området från början.

Vi gick ut ganska hårt med att undersöka komponenter och lägga upp en bra planering där vi också räknade in att vi skulle göra ritningar på roboten.

Starten på projektet gick bra och fungerade felfritt, om vi hade haft all know-how. När vi väl hade beställt alla grejor och labbat med det så insåg vi att vi hade börjat i lite fel ände av projektet, vi började med att leta upp saker vi ville använda i vårt projekt istället för att börja med att leta upp kunskapen om hur man använder de här sakerna. Vi var alltså lite förhastade i våra beslut om vad som skulle göras och byggas till en början. När vi insåg att vi inte hade kunskapen så försökte vi hämta in den i efterhand, och lyckades till stor del. Vi lyckades inhämta kunskap om hur man kopplar och använder en mikroprocessor samt hur man programmerar den och vilka språk man kan programmera i.

Under projektet har det inte alltid varit lika hög aktivitet som i början dock.
Vi begick det klassiska misstaget av att skjuta på saker trots alla varningar och dylikt, detta mycket på grund av andra faktorer som tar bort tankarna ifrån projektet.
Vi gick ut hårt och i mitten av projektet var det lite lågt och vi gjorde inte mycket mer än att försökte samla ihop lite information och träffades för att labba ungefär varannan vecka. Till vår fördel avslutade vi dock lika hårt som vi hade gått in. När vi insett att vi inte skulle få värmesensorn att fungera och bytte strategi var det inte jättemycket tid av projektet kvar och vi satte igång omedelbart att börja jobba på den nya planen. Eftersom vi inte hade en riktig kompilator gick det ganska sakta ändå att försöka koda, men det gick framåt. Fredrik kontaktade företaget som tillverkar kompilatorn för att fråga om de kunde sponsra skolan med ett exempel men fick inget svar. Efter ett tag så hittade vi istället en piratkopierad version och nu kunde vi köra hela programmet på en gång och mycket hände samtidigt.
Efter vi satt ihop allting byggde Sebastian en labyrint och vi testade i den, det fungera som tidigare nämnt halvbra efter lite justeringar i koden, men genombrottet kom när vi satte på nya hjul. Då fungerade allting som det skulle.

Skulle jag göra om projektet idag skulle jag börjat med att försöka samla in kunskap, så mycket som möjligt. Därefter skulle jag försöka bedöma om värmesensor projektet var möjligt eller inte, förmodligen skulle jag bedöma att det var möjligt och först då skulle jag beställt in grejorna. Därefter skulle jag ha börjat labba direkt med värmesensorn för att få den att fungera först.

Under projektets gång har vi varit ganska dåliga på att dela upp arbetet effektivt och många gånger har vi jobbat på samma sak trots att detta inte är att rekommendera någonstans egentligen.

I slutsats får man väl egentligen säga att vi har lyckats åstadkomma någonting. Det var under våra förväntningar i den första projektbeskrivningen, men vi kan ändå presentera en autonom robot, även om dess funktionalitet är nedsatt till ”bevakningsarbete” istället för aktiv inverkan, som ”brandsläckarroboten” skulle ha haft.

KÄLLFÖRTECKNING

Mikroprocessorer

Allmän info om mikrocontrollers http://sv.wikipedia.org/wiki/Microcontroller
Information om Pic http://sv.wikipedia.org/wiki/PIC

Information om den processor vi tänkt använda
http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Information om AVR http://en.wikipedia.org/wiki/AVR

Information om en speciell AVR, ATmega
http://www.elfa.se/elfa-bin/dyndok.pl?dok=5783.htm

Atmels olika produkter
http://www.atmel.com/products/avr/overview/AVR_ProductLineIntroduction.pdf

Vilka BASIC Stamps det finns
http://www.parallax.com/html_pages/products/basicstamps/basic_stamps.asp

Allmän info om BASIC Stamp http://en.wikipedia.org/wiki/Basic_Stamp

Guide för användning av PIC http://www.geocities.com/nozomsite/pic1.htm

Sensorer

”How to survive a robot uprising” Av Daniel H. Wilson

Diverse

Användning av I2C bus
http://www.robot-electronics.co.uk/htm/using_the_i2c_bus.htm

Datablad för värmesensor
http://www.robot-electronics.co.uk/htm/tpa81tech.htm

Kodexempel för servostyrning http://pratt.edu/~eproto/code.html

Diverse hjälp från folk på Elektronikforumet.com
(Listar alla mina trådar jag postat angående detta projekt)
http://www.elektronikforumet.com/forum/viewtopic.php?t=6859
http://www.elektronikforumet.com/forum/viewtopic.php?t=10114
http://www.elektronikforumet.com/forum/viewtopic.php?t=10264
http://www.elektronikforumet.com/forum/viewtopic.php?t=10468

BILAGOR

Tidsplan projekt

1. Planering

Genom att samla mycket information i början är det lättare att göra en realistisk tidsplan. Genom lite uppskattning och tidigare erfarenheter gjorde vi en tidsplan för hur projektet ska genomföras. För att få någon struktur började vi med att planera vilka saker vi kommer att göra och i vilken ordning de kommer att ske.

2. Undersökning

I det här stadiet kommer vi att göra en utförlig undersökning vad gäller simpel robotik (liknande den typen vi kommer syssla med) men huvudsakligen undersöka vilken typ av processor som skall användas, hur denna används inkl vilket programmeringsspråk man bör använda. Vi kommer även att undersöka vilka komponenter vi skall använda i vår robot, t ex värmekameran och avstånds kännare. Vi kommer även att här börja titta på ungefär hur vi skall bygga roboten, vilken konstruktion den kommer att ha och hur vi skall lösa problemet med att t ex släcka ljuset. Den här fasen kommer bli den som sätter bas för resten av arbetet och hur det kommer att se ut och utforma sig.

3. Inköp av byggmateriell & komponenter

Att införskaffa de komponenterna som vi behöver kommer mycket tid gå åt bara i väntan på att få våra saker. Därför kommer vi att kunna göra mer än bara precis beställa komponenter under de veckorna som vi har planerat för inköp. Bland annat börja rita och skissa på konstruktionen och så småningom börja rita den i CAD för att få en tydlig modell av hur det kommer att se ut innan vi börjar arbeta.

4. CAD-ritning

Innan vi börjar bygga den verkliga roboten är det stor hjälp om vi har en 3d-skiss på hur den kommer se ut. Med hjälp av att kunna rita i skala kommer vi få en bättre uppfattning om den färdiga roboten.

5. Byggning

Byggning kommer att ske efter den ritning som vi har fått genom CAD-modellen. Förmodligen kommer vi att använda oss av verkstaden på skolan när vi bygger roboten.

6. Programmering

Programmeringen av just denna typ är inget som vi gjort tidigare. Därför kommer vi båda att få läsa in oss på området för att kunna få roboten att agera så som vi bestämt. Målet är att roboten ska gå att koppla in och programmera via vilken dator som helst.

7. Testning

Förhoppningsvis ska detta bara vara en punkt där allt ska kalibreras in. Men vi har lämnat utrymme för eventuella justeringar eller i värsta fall små

ombyggnationer och eventuella problem som uppkommer i programmeringen, som vi sett i tidigare projekt.

8. Rapportskrivning

Efter det att alla punkterna ovan är uppfyllda och roboten fungerar kommer sammanfattningen. Allt som vi dokumenterat om varje etapp ska slutskrivas till en text där man kan läsa allt som hänt under processens gång samt slutsatser och eventuella paralleller mot användningsområden i arbetslivet.

9. Klart

När roboten är klar ska den visas upp och demonstreras för övriga i klassen. All dokumentation lämnas in. Förhoppningsvis kommer detta bli ett sevärt projekt att visa upp på kommande öppet hus eller för andra elever.

Kod

Notera att radbrytningar har blivit lite fel på vissa ställen i överförelse till word dokument.

```
Device = 16F628A            'definiera vilken processor
xtal = 4                    'klockfrekvens
Dim Ping As 2               'Ping heter ultraljudssensorn och ligger på port 2
Dim left As 3               'vänster motor port 3
Dim right As 5              'höger motor port 5
Dim rawDist As word         'avståndet som returneras vid pulsin från sensorn
Dim counter1 As word        'räknare 1 som används för att bestämma hur många
pulser som skall köras
Dim counter2 As word        'räknare 2 som bestämmer hur många kompensationer
                            'per klockcykel som får utföras
Dim counter3 As word        'räknare 3, som räknare 2 fast för andra motorn
Dim angle As word           'bestämmer om motorn har kompenserat eller inte
Dim angle2 As word          'samma som ovan fast andra motorn
counter1 = 0                'alla här nedan är bara för att de skall ha ett
                            'ursprungsvärde
counter2 = 0
counter3 = 0
angle = 0
angle2 = 0

pause 5000                  'kör en paus på 5 sekunder för att ge lite tid till att sätta
ner roboten

start:          'efter vänstersvängen kommer den hit igen (och switchen är inte intryckt
                'längre)

while PORTB.1 = 1    'om den inte sitter fast i en hörna och brytaren är intryckt är
                     'portb.1 = 1

if counter2 = 5 then 'max 1 kompensation per 5e cykel i programmet
counter2 = 0
angle = 0
endif
if counter3 = 5 then 'max 1 kompensation per 5e cykel i programmet för andra motorn
counter3 = 0
angle2 = 0
endif

pulsout Ping, 5                     'den här programsnutten hämtar avståndet till
högersidan
Pulsin Ping, 1, rawDist
rawDist = rawDist * 10      'här får vi det i uS
rawDist = rawDist / 2       'tar bort återvägen för ljudet

if rawDist > 160 AND rawDist < 210 Then  'om avståndet är mellan ungefär 50mm
                                         'och 65mm
                                         'kör rakt fram
        high 4
```

```
        high 0
    while counter1 < 50
    pulsout left, 100
    pulsout right, 100
    pauseus 1600
    counter1 = counter1 + 1
    wend
    counter1 = 0
endif
IF rawDist < 160 then                    'om avståndet är mindre än 50mm
'kompensera åt vänster
        high 0
        low 4

        if angle2 = 0 then
    while counter1 < 30 'sväng vänster lite grann
    pulsout right, 130
    pauseus 1600
    counter1 = counter1 + 1
    wend
    counter1 = 0
        angle2 = angle2 + 1
        endif

        while counter1 < 10 'kör rakt fram lite grann
    pulsout left, 100
    pulsout right, 100
    pauseus 1600
    counter1 = counter1 + 1
    wend
    counter1 = 0

endif
if rawDist > 210 AND rawDist < 700 then
'om avståndet är för långt men inte så långt att det är en högersväng
'kompensera åt höger
        high 4
        low 0

        if angle = 0 then
      while counter1 < 30 'sväng höger.. 1puls = 2,9ms
      pulsout left, 130
          pauseus 1600
      counter1 = counter1 + 1
      wend
      counter1 = 0
          angle = angle + 1
        endif

        while counter1 < 10 'kör rakt fram lite
    pulsout left, 100
    pulsout right, 100
```

```
   pauseus 1600
   counter1 = counter1 + 1
   wend
   counter1 = 0

endif
if rawDist > 700 then 'om avståndet är mer än 20cm
'kör rakt fram, sväng höger, kör rakt fram

        high 4
        low 0
   while counter1 < 160
   pulsout left, 100
   pulsout right, 100
   pauseus 1600
   counter1 = counter1 + 1
   wend
   counter1 = 0

   while counter1 < 160
   pulsout left, 100
   pulsout right, 200
   pauseus 1600
   counter1 = counter1 + 1
   wend
   counter1 = 0

        while counter1 < 300
   pulsout left, 100
   pulsout right, 100
   pauseus 1600
   counter1 = counter1 + 1
   wend
   counter1 = 0

endif
counter2 = counter2 + 1
counter3 = counter3 + 1
wend

'sväng vänster, hit kommer den om brytaren blir intryckt
high 0
low 4
while counter1 < 155
   pulsout left, 200
   pulsout right, 100
   pauseus 1600
   counter1 = counter1 + 1
wend
counter1 = 0
low 0
```

```
goto start    'gå tillbaka till början efter vänstersvängen
end
```