

LEGO Robot programmering och felsökning

Hur svårt ska det vara att följa den svarta
linjen?

Daniel Lindfors

12/9/07

dlindf@kth.se

Introduktionskurs i datateknik II1310

Sammanfattning

Denna laboration syftade till att ge laboranterna en introduktion till det breda ämnet data teknik och de arbetsmetoder som används som student på ICT-skolan för att lösa problem.

Under laborationen fick man en LEGO Mindstorm robot tilldelad med en tillhörande NXC kod som skulle få roboten att följa en svart linje på golvet. Koden gjorde dock inte som den skulle och det var laboranternas uppgift att ta reda på varför och göra de nödvändiga ändringarna i koden för att roboten skulle göra som den skulle.

Uppgiften skulle lösas genom arbetssättet par programmering där man fick praktisera att arbeta med kod skrivandet i par.

Efter många försök lyckades man få roboten att följa linjen.

Innehållsförteckning

1. Inledning.....	2
1.1 Bakgrund	2
1.2 Syfte och målsättning.....	2
2. Genomförande	3
3. Resultat	5
4. Analys	6
5. Diskussion.....	7
Referenser	7
Bilagor	7

1. Inledning

I denna laboration har man genom att felsöka programkod till en robot fått prova på att praktisera vissa av de färdigheter som krävs av ingenjörer. Syftet har varit att ge laboranterna en första inblick i de vissa av de element som kommer dyka upp regelbundet under deras utbildning.

1.1 Bakgrund

Då studenter på ICT skolan och ingenjörer förväntas kunna ta till sig nya kunskaper och med dessa lösa tekniska problem tillsammans med andra, samt kunna redovisa lösningarna och problemen så att andra kan förstå och ta del av dessa, så bör studenterna börja tidigt med dessa färdigheter.

1.2 Syfte och målsättning

Syftet med laborationen är att laboranterna, genom att praktiskt arbeta med element som genomsyrar deras utbildning, ska få en grund att stå på inför kommande arbeten under deras utbildning.

Målsättningen är att laboranterna ska få insikt och erfarenhet om följande:

- Parprogrammering i praktiken
- Hur ett program i ett programmeringsspråk kan se ut och hur det tillämpas på något konkret
- Felsökning i kod
- Skolans it system
- Laboration
- Rapport skrivning
- Lösa problem i grupp
- Vilka roliga saker man får göra om man läser på KTH

- Problem som uppstår vid ovan nämnda punkter

Vårt egna mål med laborationen var att få roboten att spela låten från Super Mario Bros

2. Genomförande

Till och börja med så saknades datorer till alla laborant grupper. Laborationsassistenten hade då sådan tillit till oss att vi fick låna dennes dator. Laborationsassistenten hjälpte oss att öppna upp den IDE och programkod som vi skulle jobba med och visade oss snabbt hur man kompilerade koden till roboten.

Roboten vi använde oss av var en LEGO robot som drevs av två motorer kopplade till var sitt hjul på robotens undersida. Roboten hade också en tryck sensor monterad framtill som kunde känna av när roboten åkte in i något, samt en ljus sensor monterad precis undertill på fronten som kunde känna av ljus intensitet från underlaget. Ljus sensorn användes för att detektera den svarta tejpens som vi skulle få att följa. Själveste kroppen på roboten bestod av en "dator" med en massa portar och en display. Portarna var med kablar kopplade till robotens motorer och sensorer.

Vi började direkt med att läsa igenom koden till programmet linefollower för att skapa oss en överblick över programmets konstruktion. Efter att vi fått en överblick över programmet bestämde vi oss för att provköra programmet. Vi satte roboten på golvet med ljus sensorn rakt över den svarta tejpens och startade programmet. Till vår förtjusning började roboten att dansa (snurra runt) och brydde sig inte alls om den svarta linjen på golvet, vart efter den började åka framåt i en stor sväng tills den körde in i väggen, och programmet avbröts med att visa texten "Gruppmedlemmar: " överst på displayen.

Vi kollade igenom koden ett antal gånger, gjorde ändringar och provkörde programmet utan några vidare resultat.

Det första problemet vi stötte på var att få programmet att visa våra namn på robotens display.

Vi la till två strängar med våra namn i listan `groupMembers` i programmet (rad 35, 36) som verkade hålla de strängar som skulle skrivas ut på displayen efter texten "Gruppmedlemmar: ". Detta resulterade i att strängen "Gruppmedlemmar: " som visades överst på displayen blev något förvrängd och fick vissa av sina bokstäver utbytta mot andra.

Efter ett tag så såg vi att problemet låg i loopen som användes för att skriva strängarna i listan `groupMembers`.

Loopen började skriva ut elementen i listan på raden ovanför den översta raden på displayen vilket resulterade i att det första namnet hamnade utanför displayen och det andra namnet på första raden där "Gruppmedlemmar: " redan hade skrivits ut. Genom att justera loopens (rad 45) så skrev programmet ut det den skulle.

Det stora problemet vi arbetade med var att få roboten följa linjen och inte dansa. Efter ett par redigeringar upptäckte vi funktionen `dance()` som kördes innan `followLine()` och `readTouchSensors()`.

Denna kommenterade vi bort från programmet (rad 114). Nu hade vi en robot som bara kunde svänga åt ett håll.

Här näst fortsatte vi laborera med koden och roboten utan resultat ett bra tag. Hur vi än gjorde så verkade roboten reagera på det ljus sensorn läste av alls. Laborationsassistenten kom och hjälpte oss. Vi hade tydligen inte läst igenom laborationsinstruktionen där det tydligen stod att vi skulle utföra en visitation av roboten och dess i/o portar. Vi såg nu att ljus detektorn var kopplad till port NI_3. Programmet använde istället port IN_1, vilket var den porten som tryck sensorn var kopplad till. Detta korrigerade vi i programmet (rad 68).

Efter ett par test så bekräftade vi att roboten äntligen reagerade på vad ljus detektorn läste av. Roboten följde dock fortfarande inte linje.

Vi kunde nu börja jobba på algoritmen som skulle se till så att roboten höll sig kvar på linjen. Vi förstod att ändringarna som skulle behövas görs i programmet för att detta skulle ske inte var många, men vi ville ändå utforska exakt hur en sådan algoritm skulle kunna implementeras. Detta tas upp i analysen.

Självaste algoritmen implementerade vi genom att ändra på de två drivande motorernas hastigheter då sensorn detekterade hög respektive låg ljusintensitet (rad 84, 90). Algoritmen bestod i att roboten skulle hålla sig precis vid den svarta linjens vänsterkan genom att alltid svänga vänster om den var på linjen, och alltid svänga höger om den var utanför (på vänster sida om linjen). Men efter en provkörning av programmet så märkte vi att roboten reagerade för trögt och lyckades komma utanför den svarta linjen på fel sida (höger sida). Detta antog vi till stor del orsakades av mekanisk tröghet.

Genom att experimentera med de förinställda hastighetskonstanterna SpeedSlow och SpeedFast kom vi så småningom fram till att problemet kunde lösas om man sänkte SpeedSlow en aning (rad 70). Detta påverkade två viktiga faktorer. Dels sänktes robotens medelhastighet, vilket gjorde så att roboten fick längre tid på sig att reagera, och dels så ökade detta robotens svängradie då skillnaden mellan hög och låg fart blev större.

Med denna sista förändring lyckades roboten ta sig igenom sin bana vid varje nytt försök vi gjorde och vi klassade laborationen som slutförd.

Vi riktade nu vår uppmärksamhet på den irriterande melodin som spelades i slutet av varje körning av programmet. Vi började experimentera med hur man kunde ändra på melodin genom att ändra på noterna och längden de spelades. Vårt mål var att få roboten att spela det absolut första man hör när man startar ett nytt spel på Super Mario Bros.

Vi la inte in denna ändring i resultat listan då läsaren, om denne skulle försöka implementera denna kod, starkt skulle börja tvivla på vår musikaliska begåvning.

3. Resultat

Vilka resultat gav laborationen? Redovisa med en tabell som innehåller radnummer från originalkoden, ändringar av koden/ny kod och kommentarer.

Denna tabell visar de raderna i program koden linefollower.nxc (original koden) där ändringar har gjorts.

Radnummer	Ny kod	Kommentar
2	#define SpeedSlow 70	Sänker speed slow från 80 till 70
35, 36	"Daniel", "Kim"	Lägger till namn i sträng listan "groupMembers"
45	(8*i)	(8*i-16) gjorde så att namnen började printas ut 1 rad ovanför skärmen
68	SensorRaw(IN_3)	Ljus sensorn var kopplad till input 3 och inte input 1 (IN_1) som original koden antydde
82	>	Vi ansåg att något skulle hända om intensiteten var över den övre tröskeln istället för under. Detta för att skapa ett odefinierat område mellan den övre och undre tröskeln.
84	SpeedFast	Ser till så att något faktist händer om if satsen på rad 82 är true
88	<	Återigen tyckte vi något. Denna gång att något skulle hända så intensiteten var under den undre tröskeln utav samma skär som för rad 82
92	SpeedSlow	Ser till att något händer om if statsen på rad 88 är false
114	//dance()	Kommenterar bort funktionen dance() som bara sabbar vårt program

De ändringar som gjordes för att få roboten att visa rätt saker på displayen är förändringarna på rad 35, 36 och 45.

Förändringarna som gjordes på rad 114 och 68 var vitala då dessa rader saboterade programmet helt.

På rad 82, 84, 88 och 92 gjordes förändringar som hör till självaste algoritmen som skulle få roboten att följa linjen.

På rad 2 gjordes den sista förändringen som gjorde att roboten, inte bara i teorin, utan också i praktiken lyckades följa linje.

4. Analys

Att det var något märkligt med loopen som skrev ut text på displayen såg man ganska snabbt. Men då displayens rader är definierade omvänt, d.v.s. att den sista radens värde är 0 och den första raden förmodligen har värdet 60, kom som en överraskning.

När vi skulle börja jobba på att få roboten att faktiskt följa linjen genom att ändra på linefollower algoritmen så valde vi att först utforska teorin bakom en sådan tänkbar algoritm.

De idéer som först dök upp i våra huvuden innehöll många if-satser och konstanter som höll reda på det ena och det andra som roboten själv saknade sensorer för att kunna upptäcka. Problemet låg i att robotens ljus sensor bara kunde känna av ljus intensiteten från en punkt på golvet. Detta innebar att roboten omöjligt kunde veta om den hade avvikit från den svarta linjen på höger sida eller vänster sida. Det enda roboten kunde känna av var om den var på eller utanför den svarta linjen. Hade roboten haft stereo ljus sensorer (2 ljus sensorer) så hade situationen varit annorlunda då skillnaden i sensorerna hade kunnat användas för att veta om roboten var på väg ut på höger eller vänster sida. Vi tänkte oss att en väldigt enkel algoritm för att få roboten att hålla sig intill den svarta linjen skulle vara att roboten alltid svänger åt det ena hållet, t.ex. höger, då den är utanför den svarta linjen. Om den istället är på den svarta linjen så svänger roboten istället åt andra hållet, t.ex. vänster. Om man nu innan programmet startar, placerar roboten precis till vänster utanför den svarta linjen så kommer den att börja svänga in mot den svarta linjen. När den till slut är på den svarta linjen så vänder den och svänger tillbaka ut igen. Sen börjar det om igen. Med denna algoritm tänkte vi oss att roboten skulle sicksacka precis på kanten av den svarta linjen hela vägen mot slutet. Det fanns dock flera faktorer som gjorde denna lösning svår. Eftersom den svarta linjen bara var ett par centimeter bred så ställdes höga krav på robotens reaktionsförmåga. Roboten skulle behöva hålla en lägre hastighet samt kunna utföra skarpa svängar om nu en kraftig kurva skulle dyka upp.

Mycket riktigt så visade sig att teorin var bra, men praktiken en helt annan sak. Roboten lyckades i början ta sig en liten bit med denna metod, men åkte efter bara ett par svängar ut på fel sida om linjen. Detta berodde enligt oss mer på mekanisk tröghet än tröga sensorer.

Denna tröghet antar vi dels bestod i rotationsmomentet roboten byggt upp under föregående sväng som den vid nästa sväng ska övervinna för att kunna svänga åt andra hållet, samt rörelse mängd som hela tiden skulle byta riktning. Genom att sänka den undre hastighetskonstanten så blev skillnaden i hastighet mellan de två drivande hjulen större, vilket resulterade i en kraftigare sväng radie. Detta är det svårt att säga om det var till en fördel eller inte så detta ökar rotationsmomentet. Men en annan klar fördel som dök upp var att detta också sänkte medelhastigheten på roboten vilket gav upphov till fördelarna minskad rörelsemängd och längre tid för roboten att reagera.

Att sänka den undre hastigheten från 80 till 70 gjorde inte någon större skillnad på medelhastigheten än $(80+100)/2 - (70 + 100)/2 = 5$. Men denna skillnad tycktes vara tillräcklig för att göra programmet felfritt för just denna bana.

5. Diskussion

Laborationen tog en aning tid längre än vad vi hade väntat. Detta p.g.a. att vi inte genomförde en visitation av roboten innan vi började laborera. Att programmet använde fel port till ljus sensorn var ett väldigt jobbigt fel som hade tagit oss timmar att hitta om vi inte hade fått hjälp.

När det gäller linefollower algoritmen så hade vi ganska snabbt kunnat prova oss till rätt kod med lite intuition. Men det kändes mer motiverande och lärorikt att först försöka klura ut hur vi själva skulle ha skrivit denna algoritm med bara två if-else satser som avgränsning. Att lösa problem med begränsade resurser på detta sätt är utmanande och lockar fram kreativa lösningar.

Arbetet bestod till mesta del av "trial and error". Detta kändes ganska lustigt då både jag och min labb kollega är vana c kodare och trodde att vi skulle kunna klura ut koden bara genom att kolla på den. Men att koppla koden till något som ska hända i verkligheten gjorde den lilla kodsnutten så mycket mer komplicerad än vad den från början verkade.

Referenser

Programming LEGO NXT Robots using NXC
Labb-PM

Bilagor

