

Felsökning i NXC

Få en LEGO Mindstorm-robot att följa en linje

Christoffer Jedbäck

2014-09-04

Jedback@kth.se

Introduktionskurs i datateknik II1310

Sammanfattning

Denna rapport innehåller fakta som behandlar ett scenario där ett bristande program utdelades till ICT-studenter. Studenternas uppgift var att arbeta i par och ändra programmet så att en LEGO Mindstorm-robot skulle bete sig på ett specifikt sätt. Totalt tio ändringar behövdes i koden för att lösa uppgiften. För mer specifika resultat se [3. Resultat](#). Det som låg bakom att roboten från början inte utförde uppgiften var att: en metod satt på fel ställe, den gick för snabbt, det fanns en array av fel typ samt att den skrev ut på fel ställe, en av sensorerna var felaktigt angiven och motorerna som styrde hjulen gick i fel takt. Dessa problem löstes med hjälp av bifogade dokument samt logiskt tänkande efter en analys av programmet.

Innehållsförteckning

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

1. Inledning

Att förstå sig på grundläggande programmering är en god egenskap hos en blivande ingenjör. Denna laboration är ett bra tillvägagångssätt för att greppa det teoretiska och praktiska inom området eftersom svårighetsgraden är någorlunda anpassad till nybörjarnivå. Laborationen gick ut på att ändra koden i ett tilldelat NXC-program för att få en LEGO Mindstorm-robot att utföra en specifik uppgift.

1.1 Bakgrund

För att man överhuvudtaget ska kunna felsöka kod i ett program för vilket programmeringsspråk som helst måste man naturligtvis förstå hur de mest fundamentala programsasterna fungerar, samt känna till syntaxen för det specifika språket. Detta leder alltså till att man måste ha grundläggande kunskaper inom programmeringsspråket NXC för att kunna utföra denna laboration. Upplägget till laborationen var att två personer skulle parprogrammera och eventuella ändringar i koden skulle utförligt antecknas.

Med tanke på de moment man kommer stöta på under laborationen kan man minst sagt påstå att den är nyttig för blivande ingenjörer. Detta beror på att en ICT-student med största sannolikhet kommer att komma i kontakt med programmering, dokumentation och gruppbaseade arbeten, både under studierna såväl som senare i arbetslivet.

1.2 Syfte och målsättning

Syftet med uppgiften var dels att visa hur ett problem samt problemlösning kan se ut för en ingenjör, men också att introducera programmering för de som aldrig programmerat förut.

Målet med hela laborationen var att få roboten att följa en förbestämmd bana med hjälp av en optisk sensor. När roboten slutfört banan och ett hinder aktiverat trycksensorerna skulle roboten skriva ut gruppmedlemmarnas namn på sin display.

2. Genomförande

Till att börja med laddades alla medföljande program, drivrutiner och filer ner och installerades på datorn. Därefter startades BrixCC och "Port" ändrades till usb. LEGO-roboten kopplades in i datorn med hjälp av en USB-sladd. Efter detta analyserades programmet linefollower.nxc av samtliga laboranter i gruppen. Då en uppfattning om programmet hade skapats gjordes ändringar i programmet för att sedan kompilera det. Programmet med den nya koden exporterades till roboten och den testkördes. De ändringar som skedde i koden antecknades. Proceduren som innebar stegen från att ändra i koden till att testköra roboten utfördes på nytt tills önskat resultat uppfyllts.

3. Resultat

Radnummer	Ny kod	Kommentar
2	<code>#define Speedslow 40</code>	Ändrade värde från 80
3	<code>#define Speedfast 60</code>	Ändrade värde från 100
34	<code>string groupMembers[]</code>	Ändrade arrayens typ från <code>int</code>
35	<code>"Robin"</code>	Angav namn istället för heltal
36	<code>"Christoffer"</code>	Samma som raden ovan
46	<code>(LCD_LINE - 8*i)</code>	Tog bort <code>- 16</code>
76	<code>SensorRaw(IN_3);</code>	Ändrade sensor från <code>IN_1</code>
92	<code>OnFwd(OUT_A, SpeedFast);</code>	Ändrade från <code>SpeedSlow</code>
100	<code>OnFwd(OUT_b, SpeedSlow);</code>	Ändrade från <code>SpeedFast</code>
115	<code>// dance();</code>	Kommenterade bort

(Tabellen visar de ändringar i koden som gjordes samt på vilken rad de var och vad som förändrades)

4. Analys

Det som kunde konstateras direkt utan vidare åtanke var att metoden `dance();` skulle kommenteras bort eftersom den var placerad i själva `main`-metoden. Med tanke hur metoden var namngiven kändes det ganska självklart vad som skulle hända då man körde programmet på roboten. Anledningen till att raden kommenterades bort och inte togs bort helt var av några få skäl. För det första kan det hända att man vill ha kvar raden av experimentella skäl som t.ex. att testa metoden så fort man kör programmet. För det andra kan det vara så att man inte är riktigt säker på om raden ska stå kvar eller inte, detta är dock mer förekommande då man programmerar ett Cascading Style Sheet eftersom många saker kan slängas om med väldigt lite kod. För det tredje vill man möjligen ha kvar koden för att komma ihåg att den var där när man började koda. Orsakerna varierar beroende på situation men alla har personligen upplevts i tidigare Java-programmering.

efter att `dance()` -metoden kommenterats bort ville roboten fortfarande inte följa linjen utan den åkte istället snett till vänster. Detta löstes genom ändringarna på rad 92 och 100 eftersom en av motorerna var tvungen att gå fortare än den andra, och vise versa, om roboten ska kunna svänga. Efter dessa ändringar gjorts ville roboten fortfarande inte följa banan och därför gjordes ändringen på rad 76 eftersom `IN_1` upptäcktes vara en av trycksensorerna. Detta kunde förstås av att läsa i `main`-metoden och `readTouchSensors()` -metoden.

Ändringen som skedde på rad 34-36 gjorde det möjligt för roboten att skriva ut namnen eftersom dessa var tvungna att vara av typen `string`. Men det räckte inte bara med att ändra datatypen eftersom namnen skrevs ut för högt upp på displayen, därav ändringen på rad 46 vilket gjorde att namnen skrevs ut under texten "Gruppmedlemmar:" på displayen.

Ändringarna på rad 2 och 3 var egentligen inte essentiella för att roboten skulle klara av banan, men det upplevdes att precisionen blev bättre av att sänka dessa värden. Däremot gjorde dessa ändringar att roboten utförde uppgiften långsammare.

5. Diskussion

Syftet med laborationen framkom väldigt tydligt då den var avslutad med tanke på hur laborationen var utformad och vad som förväntades av laboranterna. Målet var från början relativt självklart och kändes realistiskt i förhållande till situationen.

Då jag och min medlaborant stötte på ytterst få problem var det endast två saker som fick oss att tänka till. Det ena var hur vi skulle få namnen att skrivas ut korrekt då vårt första resultat av detta ledde till att displayen visade "Christoffermar:". Vi förstod då att namnet hade skrivit över texten "Gruppmedlemmar:" så vi letade igenom koden för att hitta lösningar. Det var först då vi läst igenom rad 9-17 som vi förstod att namnen skrevs ut för högt upp. Det andra var hur vi skulle få roboten att följa linjen efter att ha ändrat på rad 92 samt 100. Vi var säkra på att dessa ändringar skulle få roboten att göra det den skulle, men detta var innan vi hade gjort ändringen på rad 76. Det var först när vi läst igenom `task readTouchSensors()` -metoden som vi förstod att `IN_1` och `IN_4` var de båda trycksensorerna kombinerat med i `main`-metoden förstod vi att `IN_3` var den optiska sensorn.

Det jag kommer ta med mig från denna laboration är kunskap inom ett helt nytt programmeringsspråk och begreppet parprogrammering samt dess innebörd. Jag tror nog dessvärre att jag inte kommer ha speciellt mycket nytta av detta i mitt framtida yrke eftersom laborationen var väldigt grundläggande. Däremot kommer jag att dra nytta av detta i mina framtida studier då detta moment var en bra kickstart för att förbereda för de kommande kurserna. I sin tur kommer de nya kurserna att hjälpa mig i arbetslivet så på ett sätt bidrog denna laboration också.

Om man tittar tillbaka på hur vi arbetade och reflekterar över vårt arbetssätt så skulle jag anse att det var så nära optimalt det går att komma. Hade vi kunnat göra något annorlunda skulle vi ha stannat kvar lite längre och testat fler värden på `#define Speedslow` och `#define Speedfast` föra att undersöka om det fanns några optimala värden som skulle ha resulterat i att roboten utförde sin uppgift så snabbt som möjligt, men med så lite avvikelse från linjen som möjligt. Då detta egentligen inte var målet med laborationen var det ovesäntligt.

Referenser

Kursmaterial från Bilda:

[Programming LEGO NXT Robots using NXC.pdf](#) - instruktioner för robotens olika funktioner

Bilagor



(Bilden visar ett dagboksinlägg som skrevs direkt efter laborationen)