ROYAL INSTITUTE OF TECHNOLOGY

Programming LEGO Robots with NXC – for beginners

A lab report introducing the workflow of effective problem-solving using the NXC language.

Arvin Behshad
2013-09-03
arvinb@kth.se
Introduction to Computer Studies (II1310)

Abstract

Learning a new skill is often difficult and programming is no exception. Lots of beginners are often confused by the syntax and lack of instructions, which make the process much more difficult for them. This is a problem as programming is one of the most sought-after skills on the market today. The lab detailed in this report aimed to ease these concerns by introduce programming in a fun and easy way. The task was to work in pairs in order to program LEGO robots using the NXC (Not eXactly C) language. Equipped with an LCD screen, light sensors, pressure sensors and speakers; LEGO robots would have to follow a black line in the shape of a semi-circle and collide with a perpendicular wall, then upon collision, play a tune and list the names of the coding pair. The code, along with the drivers and tutorials were provided for us. However, the code was intentionally altered to become faulty, leaving us to troubleshoot and ultimately correct it. The results show that we managed to correct it, and have the robot complete the task properly. This is a promising result at the very start of our education in Computer Engineering, and a great way to familiarize ourselves with programming techniques before entering a long and hopefully prosperous career in this field, where real-world problems are tackled.

Table of Contents

1. Introduction

The first steps taken to learn programming are often the most daunting for beginners. There are too many questions, such as how you should get started with programming, which language is the best and what does it mean to "know a language"? A more seamless way to be introduced to programming is by working in pairs to solve a common problem together, in a fun and easy way. As such, we programmed a LEGO Mindstorms robot during our lab session. It was one effective way to answer all of these questions, or to at least gain a deeper understanding of the topics surrounding the questions.

This lab report will detail how we were given a small introduction to programming during a lecture, and later used that knowledge to debug faulty code, allowing us to complete a task using the LEGO Mindstorms robot. The task involved making the robot follow a black line in a semi-circle, then collide with a wall and play a sound as it prints the names of the group members with the help of its LCD screen, light sensors, pressure sensors and speakers. Our results show that we were successful in finding and correcting the errors in the given code, allowing us to adjust the code in order to complete the task with the robot.

1.1 Background

The methods involved in this lab served very important purposes, not just for the future of our careers as engineers, but also in preparation for upcoming courses in programming. It also introduced the general workflow of debugging and testing code, which are both essential for effective work within programming. The general work phases (from start to finish) of problem-solving for engineers were introduced to us as idea conception, design, implementation and trials, and operation/maintenance.

1.2 Aim and Purpose

The aim of this lab was to work in pairs to debug and compile a faulty piece of code in NXC, which would allow the robot to perform its tasks in our desired way. The robot's task was to follow a black tape which had been drawn across the floor in the shape of a semi-circle. The robot would do this with the help of its light sensor. At the end of the semi-circle it would then collide with a wall, causing its pressure sensors to detect a collision and play a sound with its speakers. This lab report also has its own purpose in that it acts as proper documentation and reflection on the work that we did.

2. Procedure

With respect to the phases mentioned above, the idea conception, design, and much of the implementation had already been done for us. The remaining tasks for us were to troubleshoot and debug the code, compile it and make sure the robot performs the required task properly.

During a previous lecture we were introduced to essential syntax and statements in the NXC language. These included, but were not limited to:

- `for/while loops`
- `if/else statements`
- `arrays`
- Other syntax such as `int`, `==`, `||`.

In preparation for the lab session, the necessary courseware[1] was downloaded from our internal site. This included:

- A tutorial titled "Programming LEGO NXT Robots using NXC", which we were instructed to read.
- Drivers for the LEGO NXT Robot
- The code that we would be debugging
- Editor and compiling software Bricx Command Center (BricxCC)
- A template recording our results and analysis onto, as seen under Results.
- An image showing how the robot should be assembled
- A template for writing this lab report

BricxCC and the drivers were compressed in .zip format upon download, which I had to extract. I then went on to setup both BricxCC and the drivers through their respective installation wizards. In addition to this, we were instructed to familiarize ourselves with the API, and to go through a detailed explanation of the lab session on our internal webpage, Lab-PM[2].

Upon beginning the lab together with my partner, we first checked the inputs to make sure everything was connected properly, referring to the image with instructions. We then turned on the robot and connected it to our PC with the provided USB cable. After the drivers were installed, we tried compiling the code in BricxCC to see what would happen. We immediately got an error pointing to an array (rows 36-40 in the Results). We entered our names and the code compiled without error. We then downloaded the code onto the robot and on the second execution the robot began driving in circles. Realizing that we had arrived at the real problem, we decided to work in 10 minute intervals, where I had control over the PC, and my partner would comment on everything I did. Together we looked through the code step by step and clarified to ourselves what each step did. As help, we used the API and the tutorial document to search for specific syntax.
After a while we found `void dance()` (rows 57-64) which we recognized as irrelevant code, so we commented it out. We downloaded the code to the robot again and on the third execution the robot went in a straight line. We decided to test this at the black line, so we adjusted the light sensor pointing down towards the line and executed a fifth time. The robot seemed to follow the line straight, but didn't turn, causing it to go beyond the circle.

After reasoning together and referring to the tutorial, we decided that the wheel on the outer area of the semi-circle needed to move faster in order for the robot to turn properly (rows 98-104). This edit applied `SpeedFast` to the outer wheel in order to bring the robot back on track when the light intensity is lower than the top threshold. The edit also allowed both wheels to keep moving at

`SpeedSlow` in case the intensity level isn't altered. We then noticed that there was redundant code repeating the same command in different terms, which we commented out (rows 106-113).

We also noticed that our names weren't being printed properly. None of us had dealt with arrays before, so we actually didn't know where the problem lay. With the help of our supervisor, we were pointed to the definitions regarding the pixel-mapping of the lines on the LCD display. We noticed that the lines were increasing in intervals of eight, so we removed instructions to subtract by 16 in the array loop (rows 48-53). On the sixth execution the robot still kept going in a straight line. We reasoned that this probably had to do with the light sensor. We checked the inputs once more and noticed that in the code, the wrong input was assigned to the light sensor (rows 83-86).

On the seventh execution the robot completed the task and printed our names successfully. However, our supervisor noticed that the robot was going too fast, causing it to deflect slightly from the line. He advised us to redefine `SpeedFast` and `SpeedSlow` (rows 2-3).

3. Results

The final result was that the robot completed the task of following a black line in the shape of a semi-circle by using its light sensors and comparing the light intensity with a predetermined threshold. Upon coming to the end of the line, the robot would collide with the perpendicular wall, detect the collision with its pressure sensors, and play a tune on its speakers, as well as print our names on its LCD screen.

Table showing changes made to the provided code for the LEGO robot

Row	New code	Comments
2-3	#define SpeedSlow 20 #define SpeedFast 90	Lowered speed for more controlled steering

33-37	string groupMembers[] = {
	"Arvin",
	"Aziz"
	};
		Added values for groupMembers

43-47	int i;
	for(i = 0; i < ArrayLen(names); i++)
	{
	TextOut(0, (LCD_LINE2 - (8*i)), names[i]);
	}
		Removed instructions to subtract i by 16 as it wasn't in accordance with LCD_LINE2

50-55	/*
	void dance()
	{
	OnFwd(OUT_A, 87);
	OnFwd(OUT_B, 20);
	Wait(SEC_3);
	}
	*/
		Commented out dance() as it gave faulty instructions

74-77	void readLightSensor()
	{
	lightIntensity = SensorRaw(IN_3);
	}
		Changed from IN_1 as it was the wrong input

89-95	readLightSensor();
	if(lightIntensity < TopThreshold)
	{
	OnFwd(OUT_A, SpeedFast);
	} else {
	OnFwd(OUT_AB, SpeedSlow);
	}
		Changed the if-statement to SpeedFast, in order to increase the speed of the outer wheel and bring the robot back on track.
		
		Changed the else-statement to incorporate OUT_AB, as we don't want a change in reaction in this case.

96-101	/*
	if(lightIntensity > BotThreshold)
	{
	OnFwd(OUT_A, SpeedFast);
	} else {
	OnRev(OUT_AB, SpeedSlow);
	}
	*/
		Commented out redundant code, as the above is enough to complete the task

4. Analysis

The results show that our method worked as intended in order to complete the task. A few notable changes included the logic that, since there's no steering wheel for the robot, one wheel (the outer one) has to speed up relative to the other in order to turn it. This was appropriately done using an `if/else statement`. Another was that making the robot drive slower will always result in more accurate results, using `#define`, as there is less margin for error. The `#define` statement was also used to create a reference point for `TopThreshold` and `BotThreshold`, which the robot depended on to compare with the value for light intensity.

5. Discussion

The purpose of this lab was to introduce programming to us in a fun and easy way. The coding was effectively done in pairs, meaning that we could help each other. Neither my partner nor I had any previous experience with extensive programming, so we're both very pleased with our work, though he did require the supervisors help for some areas.

The lab session helped me realize a few specific things about programming. First of all there was the logic that went into the problem-solving. The fact that there's no steering wheel for the robot meant that the outer wheel needed to speed up relative to the other in order to turn it. Another point is that cleaner code is always better. It's important to use a clear structure and to document the code well and to remove the redundant code, although it was interesting to see the same commands expressed in different ways, yielding the same result.

My two biggest realizations were that large scale solutions in form of code don't just come tailored to the client's current needs, but instead have a base formula applied (e.g. rows 43-47 involving a `for-loop`) where the code only needs to be adjusted minimally in order to fulfill the client's future needs as well. Also that the programmer generally has to have a strong understanding of the API in order to be able to troubleshoot it properly.

I'm satisfied that I had the chance to work with an API for the first time, and realize its importance in creating software. Their importance was previously unknown to me, which is ironic as I now consider learning APIs to be more important than the language they're written in. Learning a bit of NXC was fun, but I don't think I'll continue with it in the foreseeable future as I'd like to focus on more popular languages like Java, Python, PHP and JavaScript. That said, I'm very grateful for the knowledge I acquired over the small time period which I used it.

References

1. All relevant courseware (https://bilda.kth.se/courseId/10164/content.do?id=21060029)
2. KTH Lab-PM (https://bilda.kth.se/courseId/10164/content.do?id=21060042)

Attachments

Journal entry from KTH-Social