Autonomous car

Apostolopoulos Theodoros, Dialektakis George

18/10/2017

School of Electrical and Computer Engineering

Embedded Microprocessors Systems (ΗΡΥ411)

## Introduction

In the project for the “embedded systems” course we have decided to construct an autonomous car, using the Lego Mindstorms technology, that is able to move freely through space, successfully avoiding any occurring obstacles and that is able to automatically park in any custom parking slot. During the first two weeks, we emphasised on familiarizing ourselves with the materials, the technology and the programming environment through the following steps.

## Materials

First and foremost, we had to construct our vehicle. We had been given an already made vehicle from the projects available in the lab. However, we decided that we should create one on our own with the specifications that we wanted, as we already had many parts from an older version of NXT. First, we created the main body using examples and instructions we found on the official site of NXT and on YouTube and later we made the appropriate corrections due to lack of accuracy on the turning of the front wheels, a problem we are still partially facing but we are due to fixing it soon enough.

## Material list

Our robot consists of the following parts:

Sensors:

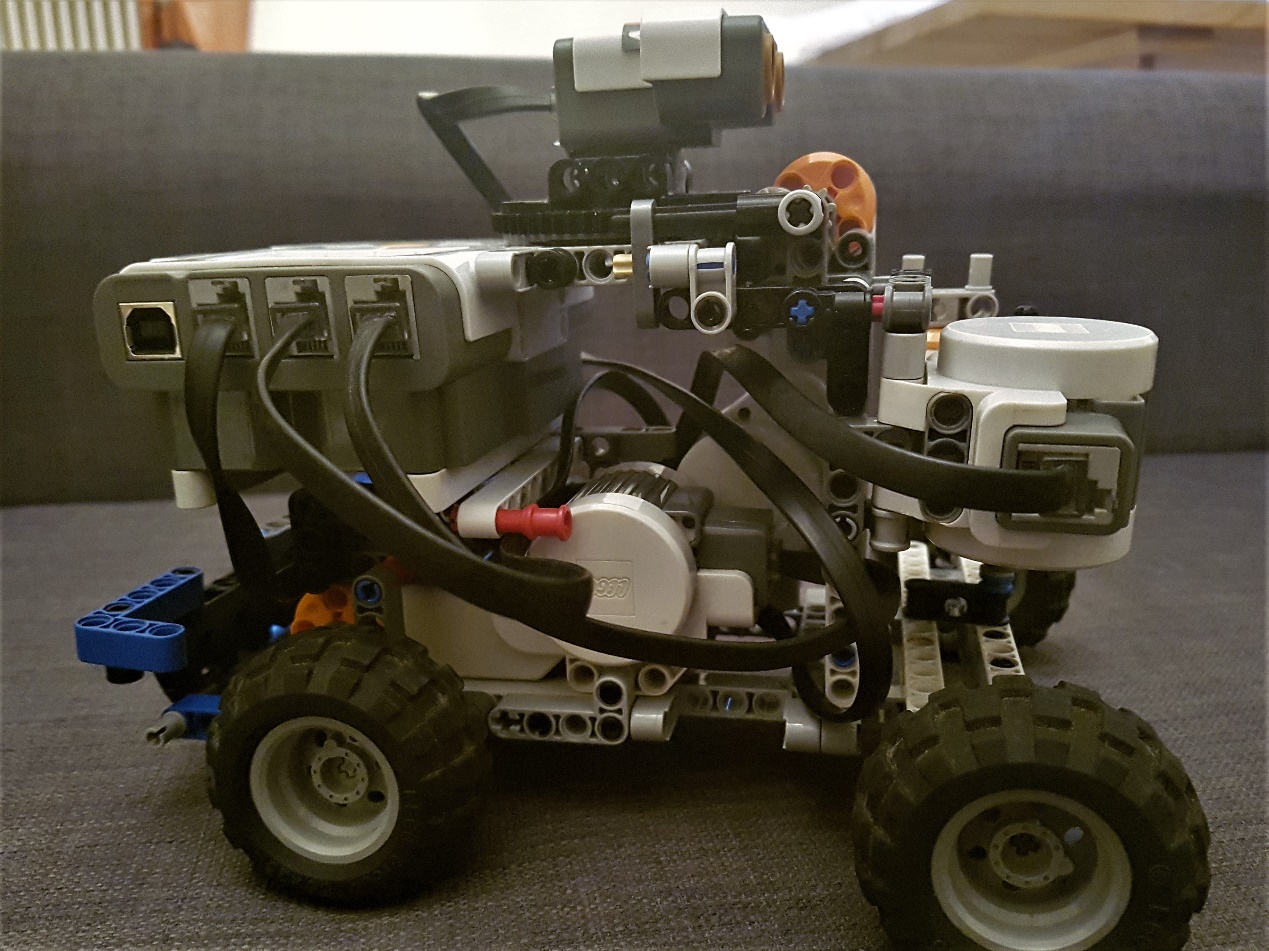
* ULTRASONIC PORT 1
* TOUCH PORT 3

Motors:

* MOVEMENT (PORT A)
* SENSOR ROTATION (PORT B)
* WHEELS ROTATION (PORT C)

Moreover, we have the following measured data:

* ROBOT LENGTH: 22 cm
* ROBOT WIDTH: 15.8 cm (back wheels)/ 17 cm (front wheels)
* WHEELS ROTATION MOTOR 130o => MAXIMUM ROTATION OF FRONT WHEELS
* SENSOR ROTATION MOTOR 56 ROTATIONS (20160o) => 1 ROTATION (360o) OF SONAR GEAR



## Technology & Programming Environment

As far as the programming part of our project is concerned, we are using the ROBOTC platform in order to implement the code of the NXT brick instead of using the original platform that is provided by Lego on it’s website. More specifically, ROBOTC is a C-based programming language which provides the user with special instruction sets that correspond to the original brick shaped instructions. On the left side of the platform is the function library that contains all the available functions however, to unlock its full content we had to switch the menu level to “Super User”, a feature that took us a lot of time to uncover. Moreover, the platform gives us access to a vast number of Sample projects that indicate various methods of programming different parts of the NXT. For the first milestone we used mostly the ones referring to motors and ultrasonic sensors and they set a foundation for us to expand on. Finally, the “Super User” mode allows us to use a Debugger Window in which we are able to depict various data from the NXT such as results from the ultrasonic sensor and debugging assisting messages.

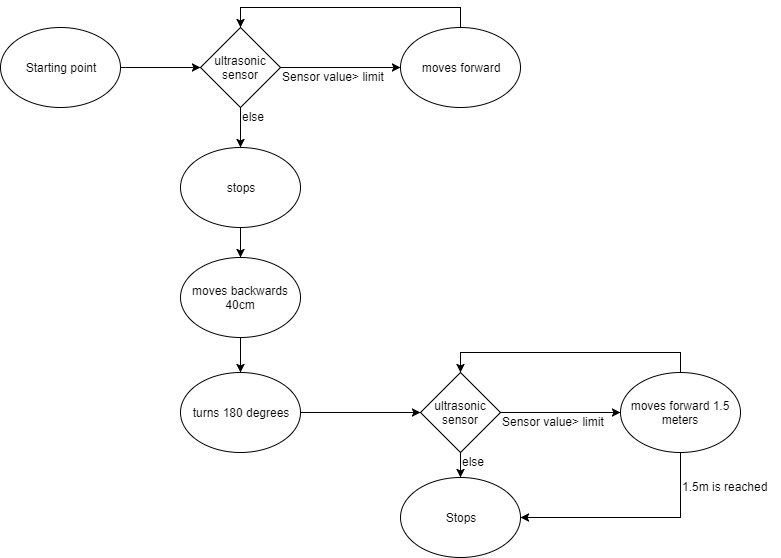
The problems we faced in this part were many and were all due our lack of experience with this specific program and its requirements. Firstly, we had to discover what the platform provided us with (Firmware, Debugger, User level, Sample Projects) and afterward we had to understand the way this C-like programming language controlled our robot. The biggest difficulty we faced was the administration of time because we haven’t constructed yet any functions that control automatically the turning of the car or the wheel and as a result we were obliged to find solutions by trial and error.

## Code

## In the first milestone we decided to keep the code simple and implement basic operations as we spent most of our time familiarising with the new information we were confronting. More specifically, we created a basic procedure in which the robot moves forward until the ultrasonic sensor detects an obstacle within a specific distance we have set and then turns 180 degrees and travels 1.5 meters unless it finds another obstacle in front of it. The robot turns “almost” 180 degrees because, as it is mentioned before, we put approximate values in our variables and therefore they have a small declination. What is more, the brick collects information from the ultrasonic sensor through polling. More specifically, the NXT due to its design is unable to work with interrupts, however it is mentioned that its processing power is strong enough to withstand polling from all of its 4 sensor ports.

## The most important instructions in our code were **“writeDebugStreamLine”** and **“nMotorEncoderTarget”**. The first one allowed us to depict the values of the Ultrasonic Sensor to the computer so that we would distinguish whether the mistakes in the route of the robot were caused by mistakes in the code or a malfunction of the sensor. The second one allowed us to set a target distance that a motor should move before system puts motor back in idle or stopped state. It was very useful due to the fact that we could specify the degrees a motor should turn and thus increasing our accuracy instead of using time as a variable for the turning of the motors.

A representation of the code of the demo for the first milestone is presented below:



## Bibliography

Basic knowledge and information about ROBOTC

<http://help.robotc.net/WebHelpMindstorms/index.htm>

Information concerning problems we faced along the way

<http://www.robotc.net/forums/viewforum.php?f=1&sid=6decc57fb332bd1c7039d8a0840df4bb>

General information

<https://www.khanacademy.org/science/electrical-engineering/lego-robotics>

Download of original firmware and general information about NXT

<https://www.lego.com/en-us/mindstorms/downloads>