NTNU

Norwegian University of
Science and Technology

Controlling an nRF52 Robot from Matlab

Maria Gilje

03-08-2021

Supervisor: Tor Onshus

Abstract

This project will present how to control a BLE-robot using a high-level language, i.e. Matlab. The system consists of an estimator and a controller and was first put together by a team of three students as their project during the autumn of 2020. The focus of this report is to get BLE functionality on the Matlab-controlled robot. After implementing the BLE functionality, it should be possible to control the robot from the Javaserver used in the SLAM project. The complexity of combining Matlab with BLE resulted in unfinished code and hypothetical results.

Contents

Abbreviations

SLAM	Simultaneous Localization and Mapping
nRFx	Nordic Semiconductor's radio microcontrollers
DK	Development Kit
SDK	Software Development Kit
RTT	Real-Time Transfer
GUI	Graphical User Interface
BLE	Bluetooth Low Energy
SoC	System-on-Chip

Chapter 1

Introduction

1.1 Motivation

One motivation for this project is the familiarity many students has with MATLAB from their studies at NTNU. It is an often used software for designing controllers for different systems across the study program. Adding the possibility of writing for a microcontroller, will expand the possibilities and uses for the study program. As of 2021, C-code is not a part of the required subjects on Cybernetics. Therefore it can be challenging to verge into microcontroller programming without doing some study of the language on ones own time. This report can therefore aid in getting a foot inside the world of microcontrollers by using a familiar language on an unfamiliar platform.

1.2 Report structure

Chapter 1	Motivation for the project and the structure of the report
Chapter 2	Background on the SLAM, hardware and software
Chapter 3	Setup of required tools and programs
Chapter 4	Method followed in the project
Chapter 5	Results expected
Chapter 6	Proposed future work

Chapter 2

Background

2.1 The SLAM Project

Simultaneous Localization and Mapping - SLAM - is a project conducted on NTNU by students and professors, and consists of different robots, as well as different modules within each robot type. In this project report, the module to be worked on is the route between a high level language and physical hardware needing a low level language. More specifically, this thesis should further bridge the gap between MATLAB and C-code for nRF52840.

2.2 Hardware

The robot is made by Eivind Jølsgård[1] in his project report. It runs on an nRF52840-DK with a shield designed by Jølsgård to interface with the actuators and sensors.

An nRF51-dongle with a precompiled hex file was used to interface with the Java server. The source code for this is unchanged.

2.2.1 Known issues

There is a known bug with the robot, which manifests if there is no voltage supplied to the micro-usb port of the nRF52840-DK mounted on the robot when the power switch of the robot is turned on. The bug is that the DK will not run the program which is programmed onto it. If the DK is plugged in to an external power source upon turning on the power switch on the robot, the program runs as expected. After performing the work-around, the DK will run the program as expected. The work-around will need to be repeated every time the power switch on the robot is turned on.

2.3 Software

2.3.1 Robot

Nordic's SDK for nRF52 and nRF51 devices; nRF5 SDK, was used to edit the firmware for the robot. The SDK contains drivers and libraries for the nRF5x chips. Logging output was read with JLink RTT Viewer. JLink RTT Viewer is a part of nRF Command Line Tools, which will be described a little bit further in Section 3.1.3. The logs were used as a debugging tool and as a mean to get an understanding of the pre-existing system. The nRF51-dongle was programmed with nRF Connect for desktop[2] and the in-app programmer was used. nRF Connect for desktop is delivered by Nordic Semiconductor, and has different applications inside which are used with Nordic's microcontrollers. The programmer app is a simple GUI for programming compiled .hex-files to the flash memory of the nRF-chip.

2.3.2 Server

The server is written in Java, and is set up with NetBeans IDE 12.1[3]. The setup followed for this project was found in the delivery zip file from Jølsgård[1]. The file contained installers for Java, Java Developing Kit, and NetBeans IDE as well as the project file for the server itself. To see the robot from the server, a programmed nRF51-dongle was used.

2.3.3 Firmware

Most of the source code for the robot has been written by previous people who has worked on the nRF52 robots in the SLAM project. This code is written with version 15.0.0 of nRF5 SDK. The part of the firmware which is the controller and estimator, was generated from MATLAB and is from the MATLAB-group consisting of Gulbrandsen[4], Sjøvold[5], and Bliksvær[6], and the part which is the Bluetooth communication is from Berglund's code[7].

2.3.4 Interfacing the robot

With the Java server you can drive the robot manually. To do this you enter an (x, y) coordinate and the robot should drive to that point. During testing, the path for the robot was written into the source code.

Chapter 3

Setup

This chapter will describe the set up process followed in this project. The operating system used is Windows 10.

3.1 Chocolatey

Chocolatey[8] is a package manager for Windows. It can be installed by running Windows PowerShell as administrator and then following Chocolatey's instructions from their website.

3.1.1 CMake for Windows

CMake is needed to build the project and, by doing so, creating firmware which can be flashed onto the robot. Install CMake for Windows with

```
choco install cmake
```

3.1.2 GNU Arm Embedded Toolchain

GNU Arm Embedded Toolchain is required for development on nRF52840 (and all other nRF SoCs). It is installed with

```
choco install gcc-arm-embedded --version=9.2.1
```

The version is specified because the latest version of GNU Arm Embedded Toolchain is not guaranteed to be compatible with nRF5 SDK 15.

3.1.3 nRF Command Line Tools

nRF Command Line Tools is a collection of software and executables used to program and debug nRF SoCs. It is installed with

```
choco install nrfjprog
```

As stated in Section 2.3.1, JLink RTT Viewer originates from this collection.

3.2 Software development kit and SoftDevice

The software development kit is called nRF5 SDK[9] and is downloadable from Nordic Semiconductor's website. The SoftDevice used in this project is SoftDevice S140, which is also downloadable from the same website. The downloaded zip should be unzipped, and the SDK folder moved to a location close to a drive root, f. ex. C:\. In \%Path to nRF5 SDK base%\components\toolchain\gcc, Makefile.windows is located. This file has to be modified to reflect the actual version of and path to GNU Arm Embedded Toolchain.

3.3 Robot code

The path in the Makefile in the project specifies the path to the project. To satisfy this, the project folder should be placed in examples/SlamApp_nrf52840 inside the SDK. The Makefile also specifies file paths to the files which are included in the project. These files are both project files and SDK-files.

3.4 Server

As stated in Section 2.3.2, the whole setup of the server is in a sub-folder in Jølsgård's delivery folder[1]. Inside the manual folder in the java server there is a pdf with instructions of how to use the server application.

3.5 MATLAB

MATLAB can be downloaded from MathWorks[10] [licence required for full program] or NTNU Software [requires NTNU login] and installed. Also needed to generate C-code from the MATLAB functions is MATLAB Coder, which can be installed together with the MATLAB installer or from the Apps tab in MATLAB.

The files which were modified are in the zip file from the MATLAB-group in delivery/Matlab_files.

Chapter 4

Method

4.1 Start phase

Upon receiving the existing code, much time was spent towards understanding it. This consisted of changing a few lines of code to see what changed with the robot. Some help from the authors of the existing code was also needed to get a good enough understanding of the system as a whole. It was also necessary to get familiar with the SDK, as well as with CMake for compiling and programming.

4.2 Merging code

The next step was to get BLE functionality on the robot. Berglund's code [7] was chosen as a basis for the merge, as that code had implemented BLE, and was able to connect to and communicate with the Java server. Then, the files generated from MATLAB was included and the Makefile was edited accordingly. Merging the functions generated from MATLAB into the BLE-code proved complicated. For every function which was included, the BLE functionality was hindered. Some inclusions make the robot unable to connect to the dongle, while others allowed connections, but made the robot not show up in the Java server. The complications with the merging has resulted in an incomplete code for this delivery.

4.3 Tweaking and testing code

This section will be a hypothetical description of the final steps required for a functional BLE robot controlled from MATLAB.
After a successful merge of the MATLAB-functions into the BLE-code, the controller would be tuned in MATLAB to improve the behaviour of the robot. The existing MATLAB-code also had the route of the robot in the code, which needed to be replaced with coordinates sent from the Java server over BLE.

With a tuned controller and a BLE connection to the server, the robot would be properly tested by sending coordinates for it to follow. In the test, improvement areas would be noted down and either tweaked or proposed for further work.

Chapter 5

Results

As the code merging was not completed, this chapter will describe hypothetical results.

It would be expected that the robot would not perfectly follow a straight line when given a coordinate to move to. Though it would reach the given coordinates. A figure showing the intended and actual path would be included to illustrate these results.

Chapter 6

Future work

6.1 Upgrading hardware

As the nRF51 dongle has been discontinued by Nordic Semiconductor, it may be of benefit to upgrade the dongle to an nRF52840 dongle. This would include modifying the firmware for the new dongle, but would also ensure that there are new dongles available if the nRF51 dongles were to become lost or corrupted.

The bug described in Section 2.2.1 could be debugged and fixed for future students.

6.2 Upgrading SDK

Porting the project to nRF Connect SDK will make it simpler to continually update the project. nRF Connect SDK is a git fork of the RTOS; Zephyr, which also makes it simpler to get started on projects needing an RTOS. NCS also has some support for FreeRTOS if it is better to continue with the same RTOS as in this project.

6.3 Availability of the SLAM project code

Having the SLAM project as a git repository will also make it easy to find the code to get started with further work. Students can then fork the repository or create their own branches.

Bibliography

[1] Eivind H. Jølsgård. Embedded nrf52 robot. Master's thesis, Norwegian University of Science and Technology, December 2020.

[2] `https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-desktop/download`. Last Accessed: 2021-07-04.

[3] `https://netbeans.apache.org/download/nb121/nb121.html`. Last Accessed: 2021-07-04.

[4] Eystein Gulbrandsen. Controlling the nrf52-robot using matlab generated code. Master's thesis, Norwegian University of Science and Technology, December 2020.

[5] Eivind Sjøvold. Position estimation on an nrf52-robot using matlab. Master's thesis, Norwegian University of Science and Technology, December 2020.

[6] Viljar G. Bliksvær. Simulation of the matlab-controlled nrf52-robot. Master's thesis, Norwegian University of Science and Technology, December 2020.

[7] Gabriel Berglund. Embedded nrf52840 dk robot. Master's thesis, Norwegian University of Science and Technology, December 2020.

[8] `https://chocolatey.org/install`. Last accessed: 2021-07-04.

[9] `https://www.nordicsemi.com/Products/Development-software/nRF5-SDK/Download#infotabs`. Last accessed: 2021-07-04.

[10] `https://se.mathworks.com/products/get-matlab.html?s_tid=gn_getml`. Last Accessed: 2021-08-03.