Cover Letter

Mohammed Omar Khan

Student Number: 100983417

2nd year – Computer Systems Engineering

11 – August – 2015

Babak Esfandiari

Co-op Faculty Advisor

Department of Systems and Computer Engineering

Carleton University

Subject: Co-op Work term Report #1 for SYSC3999A

Respected Sir,

My first work term has come to an end and this would be my first work term report. During my time at the Advanced Real-Time Simulation Laboratory (ARS), Carleton University, my work was Coordinated by:

Dr. Gabriel A. Wainer, Faculty Member, Department of Systems and Computer Engineering, Carleton University

and Supervised by:

Mrs. Daniella Niyonkuru, Graduate Research Assistant, Department of Systems and Computer Engineering, Carleton University.

I was hired at the ARS lab for my work term by:

Mrs. Darlene Hebert, Department Administrator, Department of Systems and Computer Engineering, Carleton University.

I held the position of “Driver Programmer and Developer” and this is not a part of a continuing work placement with the same employer. I acknowledge that the report adheres to the guidelines set for work term reports and I also certify that this report is my own work based on my work in the lab and that any assitance whatsoever has be referenced.

Sincerely

Mohammed Omar Khan

Cover Page

Title: An Introduction to Applications of Embedded Systems

Work Term Report #1 - SYSC3999A

Written by: Mohammed Omar Khan – 100983417 omarrkhan3@cmail.carleton.ca

Employer: Dr. Gabriel A. Wainer, Faculty Member, Department of Systems and Computer Engineering, Carleton University

Supervised by: Daniella Niyonkuru, Graduate Research Assistant, Department of Systems and Computer Engineering, Carleton University

Executive Summary

In this modern age of Technology, robots are becoming increasingly popular and smarter. Hence, at the ARS Lab, a project to make an Autonoumus Smart Self-driving car was intiated. The car would be modeled by Discrete Event System Specification (DEVS) model and would use several different sensors and techniques to navigate around its path. This report will look at those several sensors and techniques used to navigate and describe them in detail.

The project mainly consisted of a group of 3 students from the ARS lab including myself. I worked on writing all the drivers and the testing part of the project and the DEVS model was prepared and tested by the other 2 members. By the end of the term, the final model was implemented on the hardware present in the lab and we had a successfully working model. Additional features were also added to implement the Internet of Things (IoT) applications.

Additionally, another project involving a Quadcopter drone was also initiated with an objective to attain a totally autonomous flight navigation without human input. These projects and their speifics are described in detail in the report below.

Acknowledgements

I consider myself as a very lucky individual as I had the great oppourtunity to work and learn from the wonderful people at the ARS Lab.

The special thanks goes to Mrs. Daniella Niyonkuru who in spite of being extraordinaly busy with her duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my duties for the project.

I express my deepest thanks to Dr. Gabriel A. Wainer, Faculty Member and Project Coordinator for giving necessary advices and guidance and taking part in useful decision.

Projects performed during my work term were highly interesting and educative. I am also highly thankful to the Department of Systems and Computer Engineering and specifically Mrs. Darlene Hebert under whoose careful and precious guidance, I was successfully able to complete my Co-op work term.

This oppoutunity was extremely valuable and I perceive it as a big milestone for my career. Last but not the least, I would like to thank all colleagues with whom I worked together in the projects during my work.

Table of Contents:

1. Introduction [1](#_Toc325634774)

1.1. Organizational Context [1](#_Toc325634776)

[2. Work Experience 2](#_Toc325634777)

[2.1. Nature of the work 2](#_Toc325634778)

[2.2. Experience 3](#_Toc325634779)

[2.2.1 The ParallaxRobotShield Library 4](#_Toc325634779)

[2.2.2 Testing 5](#_Toc325634780)

[2.2.3 The Quadcopter 5](#_Toc325634779)

[2.3. Challenges and solutions 5](#_Toc325634780)

3. Reflections on Work Experience  [6](#_Toc325634774)

3.1. Contributions  [6](#_Toc325634776)

[3.2. Relation to academic studies 6](#_Toc325634777)

[3.3. Career Development 7](#_Toc325634778)

[4. Summary 7](#_Toc325634779)

[5. References 7](#_Toc325634780)

[6. List of Abbreviations (if applicable) 8](#_Toc325634778)

1. Introduction

This report is a short description of my 12-week co-op work term at the ARS Lab. The work was carried out in the lab located in the Visualization and Simulation Building (VSIM) at Carleton University from May to August 2015. I, being in a Computer Systems Engineering major was interested in programming code for the drivers of an ongoing project at the ARS Lab. The goal was to build a library of drivers for several modules and sensors of a robot, which contribute to make an “Autonoumus Smart Self-driving car”.

This report contains my activities that have contributed to achieve that goal. In the following module, a description of my work environment, i.e. the ARS Lab and its’ projects is given. It is then followed by my work experience at the Lab explaing my roles, duties and the challenges faced at work in detail. After this is a reflection on my functioning and a conclusion on my experience relating to my academics.

1.1 Organizational Context

“The Advanced Real-Time Simulation Laboratory (ARS) is an advanced Modeling & Simulation research laboratory, located in the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada. The Laboratory is physically located at the Carleton University Centre for Visualization and Simulation (V-Sim).

Our laboratory is investigating means of automatic generation of executable models derived from systems specifications. The research is based on the Discrete Event System specification (DEVS) formalism, and will augment previous work with new theory, methodology, and supporting development tools. We are also interested in integrating the simulation results obtained with 3D visualisation.”[1]

The lab is supervised by Professor Gabriel A. Wainer. My work environment can be described as a small computing lab with advanced computers and ample of open space for testing robots. My direct supervisor would be Daniella Niyonkuru who is a Graduate Research Assistant directly supervised by Professor Gabriel A. Wainer. I was assigned to Professor Wainer at the beginning of my offer by Mrs. Darlene Hebert the Department Administrator of the Department of Systems and Computer Engineering at Carleton University.

2. Work Experience

During my work term, I worked on two different project. The most focused on would be the “Autonoumus Smart Self-driving car” which was acomplished successfully and the second on the “Autonomous Quadcopter navigation”. Some of my duties to mention a few were to:

* Design and implement software in C/C++ to run on the STM32F411 and STM32F405 microcontrollers
* Create an ARM mbed compatible library for the Parallax Robot Shield control and implement control drivers for its servomotors, whiskers, bump, ultrasonic and light sensors
* Design and implement an Internet of Things application that connects the Parallax Robot Shield to the Internet in order to communicate
* Work with team for application testing based on the newly designed libraries
* Analyze and test the current Crazyflie 2.0 (Fig. 2) nano quadcopter firmware

2.1 Objectives

The goal of the project was to implement a DEVS modeled complex controller system onto the Parallax boe-bot (Fig. 1) present in the Lab as a starting point to obtain a completely autonomous robot. This was overcome and a bigger objective of implementing an IoT application with it was formalized. An autonomous robot that finds it’s path by tracking a line, detects obstacles in front of it and uses WiFi to communicate with the user how many obstacles it detected. As well, a second project which would be developed for a while and be the final project for the Master’s Thesis of a team member was started with a similar goal of modeling the Quadcopter (Fig. 2) with DEVS and attaing autonomous path navigation. Both of these DEVS models are run as simulations and are tested several times before interfacing them with the drivers and hardware.

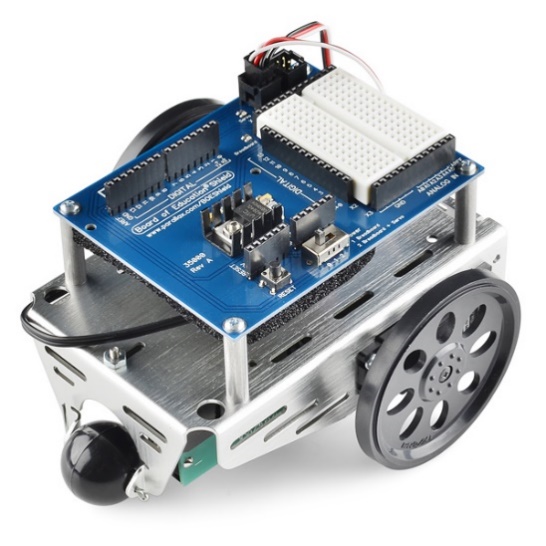


FIGURE : The Parallax Boe bot [2] FIGURE : The Crazyflie 2.0 nano-quadcopter [3]

2.2 Experience

My work at the ARS Lab mostly involves dealing with embedded systems. The “big picture” was to embed software with DEVS. As can be seen in the picture below (Fig. 3),

FIGURE : Embedded Software with DEVS

The DEVS Model or in other words the controller interfaces with the execution engine and the Execution Engine helps it to get input and output data values from the sensors. The execution engine gets this data by sending input and output commands to the Hardware API. The Hardware API is the place where there is a link between the software world and the hardware world. It is where the robot interacts with its environment and gets data values from its sensors. In our project, the whole Hardware API was written from scratch by me and the DEVS Model and the Execution Engine by the other two members.

The hardware consisted of:

* Two STM32 Nucleo boards (the microprocessors)
* The Parallax Boe bot (the robot with servomotors and a breadboard on top, Fig. 1)
* The Seeed Shield bot (another robot with servomotors and 5 light sensors preinstalled)
* Whiskers (for Tactile navigation)
* A Light sensor (for tracking a line)
* A bump sensor (for obstacle detection)
* An Ultrasonic sensor (for measuring distance)
* A Bluetooth Shield
* A GPS Shield
* A WiFi Shield
* The Crazyflie 2.0 Quadcopter (Fig. 2)

Software’s used in the projects:

* The mbed online compiler [4]
* Eclipse Luna IDE
* The Virtual Machine provided by Bitcraze (Crazyflie’s maufacturer) [5]

Using all the above mentioned hardware and software, I was responsible to make all the sensors and shields work as required by the DEVS model. Instead of writing a library specific to a particular application, I built a library for the drivers of the Parallax bot for each of the sensors.

2.2.1 The ParallaxRobotShield Library

This is a reusable and cross-platform library ready to publish for the end user. This library includes all the drivers required for each of the sensors mentioned above. To be more specific, it has functions defined for:

Movements:

* Forward: To make the robot go forward
* Backward: To make the robot go backward
* Left: Turns both servomotors in opposite direction to turn left on spot
* Right: Turns both servomotors in opposite direction to turn right on spot
* Turn Left: Turn only the left servomotor to turn left while moving back (useful in situations where there is a wall in front)
* Turn Right: Turn only the right servomotor to turn right while moving back (useful in situations where there is a wall in front)

Servomotor Control:

* Disable (one or both servomotors): Stops the servomotors in order to make the robot stop (useful when an obstacle is detected)
* Enable (one servomotor each): Resets the servomotors to their initial state for them to work again

Sensors:

* Whiskers: A function for the Left and Right Whisker individually to read if any obstacle came into contact
* Light Sensor: A function which predicts if a black line is detected
* Bump Sensor: A fuction which detects if an obstacle is detected within a range of 10cm
* Distance Sensor: Uses the simple ultrasonic technique of the ultrasonic wave being transmitted and received and the time taken to travel is used to calculate the distance from the robot

All of these drivers are documented with an example main( ) written to test the working of each driver individually. Also, all these drivers were rigourously tested, the library refined and then a process of regression testing followed.

Additional features were also added to the Seeed Shield bot’s manufacturer provided library. A driver which allows us to implement a very similar DEVS model as the Parallax bot on the Seeed Shield bot. Also, the ParallaxRobotShield Library was built with portability in mind. Hence, it was developed using the mbed online compiler where the same program can be compiled for several different mbed-compatible boards. The Library is also vendor independent and can be used on any robot having a similar hardware as the Parallax bot.

2.2.2 Testing

After testing all the drivers individually, the DEVS model was implemented for the Line tracking path navigation. On achieving success in the former, I started to make test modules for the drivers, in other words, Integration testing. The behaviour of the DEVS model was duplicated and simple code was written to see if all the sensors and drivers work properly. These wouldn’t have anything to do with the DEVS model itself but were a test to run before implementing the DEVS model on the hardware. This would help in identifing if there exists any problems in the hardware so we don’t confuse them with the DEVS model.

The first step taken was to to prepare tracks for the robot to test on. Two tracks were made after thoughtful planning of the design to test the bot in every possible scenario. These tracks were crafted out of carton boxes using black electrical tape. Four different test modules were generated:

* Obstacle free path navigation: with Whiskers
* Line tracking path navigation: with the Light sensor
* Smart Line tracking navigation with obstacle detection: with the Light sensor and the bump sensor (For both, the Parallax and Seeed bot)
* Smart(er) Line tracking navigation with obstacle detection and reporting the number of obstacles detected via WiFi (M2X – online server client): with the Light sensor, the bump sensor and the WiFi Shield (For both, the Parallax and Seeed bot)

2.2.3 The Quadcopter

The Quadcopter came with its own Virtual Machine [5] which has all the required softwares, plugins and tool-chains preinstalled. There are also example projects and a ready-to-use GUI which allows us to control the quadcopter with a joystick.

I got used to the GUI and its working and found out that the floor in our Lab isn’t perfectly horizontal. To counter this, after several test flights, I had the offset values required to fly the quadcopter normally as expected. As the goal was to flash the default firmware with our very own DEVS model, I understood the default code of the firmware and flashed my very own version after making a couple of changes to the default one. Now when we knew that we can successfully flash our own firmware, I filtered a list of controller parameters and identified them as inputs and outputs for the DEVS model.

2.3. Challenges and solutions

There were serveral challenges that I came across. As the PC I worked on had Windows 8.1 installed which is fairly new, the required driver for the serial port of the Nucleo board didn’t get installed perfectly. As a consequence, I couldn’t print anything on the screen (no printf( ), no cout, etc). As most of the printing values returned by the sensors were 0 and 1, I instead used an LED as an indicator. Also, I used physical devices such as a ruler for example to see if the distance sensor is working as expected.

Once when refining the ParallaxRobotShield Library, I replaced 8 objects of different classes by just 2. The problem that came across was that the motors were no longer in sync. To tackle this, I spent time running several tests until I finally came up with an idea to add calibration parameters to cater for each servomotor. By doing that, I calculated the calibation values for 100% and 20% speed because those were the most suitable for the final model. Also, the offline compiler had different values for calibration and I calculated separate values for it.

Somethings that I faced but couldn’t solve in the lab were that I couldn’t find any satellites for the GPS inside the lab. One solution would be to do all the tests in an open space like a garden but that wouldn’t be practical. Another issue was the power mangement of the board. When more than one shield (eg: WiFi + GPS) is plugged in onto the board, there are fluctuations in the current and the servomotors start behaving abnormally. Also that when plugging more than one shield, I ran out of “serial” ports which are required for transmitting and receiving of data from the shield to the board.

3. Reflections on Work Experience

3.1. Contributions

During my work as a co-op student, I feel that I have contributed not only to the ARS Lab, but the mbed coding community as a whole. The ParallaxRobotShield Library which has drivers for the general modules present in the Parallax bot as well as additional sensors. It can be used as a base for any future research projects all around the world. As its cross platform friendly, that brings even more possibilities.

In the ARS Lab, the library helped is running the DEVS model on the hardware to see how the model behaves in the real world. Also, the test modules developed would be of use in the future when the corresponding DEVS model is produced. The Hardware API also helped Daniella Niyonkuru in her thesis and its detailed documentation would be very helpful for the next people to work in the Lab, to directly start on the project.

My contributions to the Quadcopter project, although not as much as the Parallax bot, would also help the project members to get a head start on the project. I also believe that sharing my experience on how to use the GUI to control, to get data or to flash the firmware was helpful to get started.

3.2 Relation to academic studies

As I had a very stong background in coding from highschool and took 3 courses in programming in 3 different languages at Carleton, I had a very good knowlegde to begin my work term with. Also, I used a couple of concepts of physics several times during my course for example the working of an ultrasonic Radar.

By playing with microcontrollers and programming them for a long while now, I now have hands-on experience which I don’t expect to get from lectures in class. I have also learnt a lot from the people I worked with all thanks to the weekly seminars which highligted the ongoing work of everyone working in the ARS Lab. I now know how much it takes to build a control system and a bit of what goes on in the background.

As I didn’t take CCDP 2100 before my co-op work term, I didn’t really know much about giving presentations. Although I learnt a huge amount by doing the seminar myself, I believe that I would have been more prepared if I would have taken the course before.

3.3 Career Development

I believe the biggest lesson I have learnt from my co-op program is that it’s okay to fail. Only when you fail, you try to find a solution for the problem and in the process, you learn a lot of new things which you wouldn’t have known if you didn’t fail. I have also developed a great amount of knowledge on how to figure things out if they go wrong by research. Now, I also know what my intrests are and where to look at for my next application. Additionally, the advice I got from working at the ARS Lab is invaluable.

4. Summary

On the whole, the co-op was a useful experience. I have gained new knowledge, skills and met many new people. I was successful in developing a Library for the drivers of the ParallaxShieldBot which is now published on the web open for anyone to use. Programmers can now directly start implementing their ideas and don’t have to worry about the drivers.

One possible direction for my future successor would be to build a bot with all the shields on-board which would give it the maximum functionality and allow the Parallax bot to communicate with the Seeed bot via Bluetooth. The Parallax bot can also be the leader which navigates an obstacle free path for the Seeed bot and the Seeed bot follows even though it doesn’t have any sensors on it. For the Quadcopter, an intresting thing to do would be to make it fly like a delivery drone which picks-up and delivers a package at a given location.

The co-op work term was also good to find out what my strengths and weaknesses are. This helped me to define what skills and knowledge I have to improve in the coming time.

5. References

[1] “The Advanced Real-Time Simulation Laboratory (ARS) is an advanced Modeling &……” retrieved from (<http://vs3.sce.carleton.ca/wordpress/>)

[2] Picture retrieved from (<https://www.sparkfun.com/products/11494>)

[3] Picture retrieved from (<http://www.seeedstudio.com/depot/Crazyflie-20-p-2103.html>)

[4] Can be used on for free on login (https://developer.mbed.org/compiler/)

[5] Can be downloaded for free on (<https://wiki.bitcraze.io/projects:virtualmachine:index>)

6. List of Abbreviations

ARS lab - The Advanced Real-Time Simulation Laboratory

DEVS- Discrete Event System Specification

IoT - Internet of Things

VSIM - Visualization and Simulation Building