System Document

Version 4.0

McGill University

ECSE 211

Team 20

SYSTEM DOCUMENT

Project: DPM Final Project
Task: Construct a robot that can navigate a closed course to search and retrieve colored rings

Document Version Number: 4.*0*
Date: 07/11/2018
Authors: Shaodi Sun, Ayden Malik
Edit History:
[18/10/2018] Shaodi Sun: Created System Document
[22/10/2018] Ayden Malik: Inception of doc
[02/11/2017] Ayden Malik: Added hardware sketch in structures
[07/11/2018] Shaodi Sun: Updated system document based on project description version 2

TABLE OF CONTENTS

1. Systems model
2. Hardware available and capabilities
3. Software available and capabilities
4. Compatibility
5. Reusability
6. Structures
7. Methodologies
8. Tools
9. Glossary of Terms
10. Edit history

1.0 SYSTEM MODEL

Our software consists of the best design ideas from previous labs of each team combined together since we deemed that to be more efficient. Therefore, that would require rigorous testing to make sure all these different approaches to solve the problem at hand, is well integrated. This is to allow parallel processing and communication between the classes. We use two ways to fix the direction of the robot i.e. Ultrasonic localization for when it's in a corner and light localization when it's at an intersection. We will integrate the classes like Navigation in previous labs into our final project.

2.0 HARDWARE AVAILABLE AND CAPABILITIES

Provided to us are the parts that come with three Lego mindstorm kits, each kit consists of an EV3 brick, 2 precise motors, 2 regular motors, 1 gyroscope sensor, 1 Ultrasonic sensor, 1 sensor motor, 1 light sensor, 7 Ethernet cables and a plethora of Lego bricks. We may also '3-D print' any parts we deem may help in achieving the perfect design for our final design competition robot. The Lego bricks provided are made of ABS plastic (**acrylonitrile butadiene styrene)**.

Each EV3 brick has 4 inputs sockets, 4 output sockets and a limited number of buttons available for different functions. Each brick has an 8V Lithium ion battery and has an ARM processor clocked at mere 300MHz and has a small memory of 64MB. The sensors listed above are of very limited precision when it comes to taking readings. It also consists of a USB port, microSDHC slot Wireless Fedility and Bluetooth connectivity.

From experience with the first five labs, we have seen that it is always better to keep the robot as simple and precise as possible. Although the Lego bricks are very resilient in nature but adding more parts while designing a very simple joint will create 'flex' within the joints that may hold an actuator out thus reducing the accuracy of the device overall and it may giggle with movement. That may be handled by adding offsets but adding too many offsets over-complicates the entire process in terms of hardware and software. Therefore, the design has to be smartly executed.

3.0 SOFTWARE AVAILABLE AND CAPABILITIES

For the project at hand we are required to use Java to program the robot using LeJos as a plugin to communicate the code to the EV3. Other languages may be used to complete the task at hand but Java is most commonly used amongst developers, hence, our choice.

The main programming that we used for this project is Java, which is one of the easiest-to-use and most compatible to various platforms object-oriented programming language. Java has a lot of features like garbage collection, memory allocation, and threading. To its advantage, developers do not need to worry about the low-level features like memory allocation when developing with Java, and using Java virtual machine for compiling allows Java to be run and compiled on any platforms where a Java Runtime Environment available with some degree of performance.

However, Java has a lot of overhead code produced by running in a virtual machine. So, the software developed by Java will be hard to optimize. This is even less preferable with EV3 system, which have limited resources. Some other languages like C which allow developers to manually allocate memory using pointers, that will manage resources better.

Also, when java compiled, it will produce class files first to be compiled at run time, which would produce suboptimal code.

4.0 COMPATIBILITY

Certain design ideas have been created during the course of the previous five labs. Most of them will be directly integrated or maybe be slightly modified when being implemented into the final design. For our design, all the concrete parts that we used to build our robot come from the 3 EV3 kits that we have. It is safe for us to deduct that only the LEGO pieces from EV3 kit are adequate to build a functional robot, but additional parts can certainly be produced for future improvement.

From hardware perspective, available hardware pieces are from cumulatively three EV3 kits. Additional materials could also be 3D printed. For all the LEGO parts, not only the amount of them are limited, the number of ways to connect them is also limited. Both of them will limit the design options that we can have in some extent.

From software perspective, all of our software is developed using Java as a programming language and Eclipse as an integrated development environment. We also adopted collaboration software like Git and GitHub during our research and development phases so all of our team members all collaborate regardless of what kind of operating system they use.

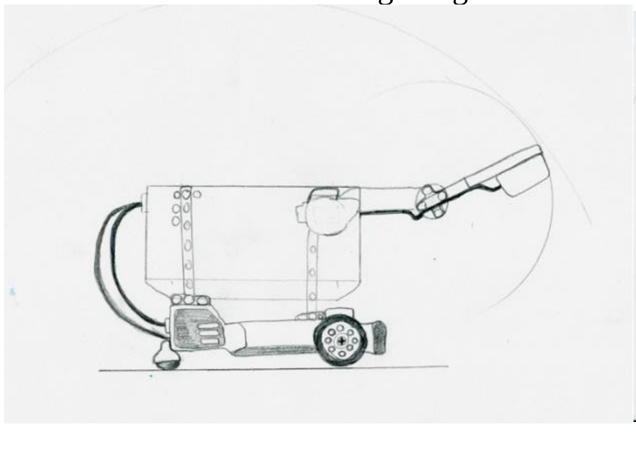
5.0 REUSABILITY

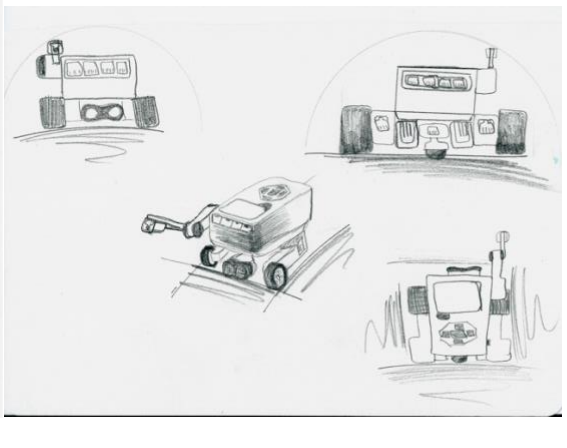
Some hardware design ideas will be used again here from the R&D phases from the labs. For example, in terms of hardware, the placement of the light sensor and ultrasonic sensor for localization and odometry. Plus, the basic software framework is also being recycled for the aforementioned tasks.

6.0 **STRUCTURES**

In terms of hardware, the robot shall follow the following guidelines:

The base of the vehicle shall consist to two wheels attached to two motors at the front a ball-in socket at the back. An Ultrasonic sensor at the front to avoid obstacles, a light sensor at the back to detect gridlines, another light sensor to detect the colors of the rings placed on the tree and a gyroscope sensor to correct the heading of the robot in real-time. The robot cannot be more than 25cm in height because of the dimensions of the tunnel. It has to be considerably less than 18.1cm in width to avoid the gratings on the sides of the tunnel.





More details of both software and hardware structures will be provided in software document and hardware document respectively.

7.0 METHODOLOGIES

All the methodologies that we adopted during our research and development phase are based on base ground rules that an efficient engineering team should adopt.

From software perspective, the development and testing happen concurrently. We adopted divide- and – conquer method as our development scheme. We separated the whole project into several subproblems which we will develop and test them piece by piece to make sure the subtasks are performing optimally. All the subtasks and progresses are also recorded into Gantt chart to make sure our resources are not wasted. For each subtask, we created specific issues for them on GitHub repository and assign them to team members. The subtasks are specific enough for granularity concerns. A software document which contains an API, class diagram, finite state machine, and explanations of flow logic, classes and methods. The software document will be built on a weekly basis to record latest changes and progresses.

From hardware perspective, the robot in general should be as simple as possible but still adequately functional. The design in general is constrained by both hardware, environmental, and requirement constraints. Some other implicit constraints may also affect the design, for example, the design need to take friction into consideration because it will be run on a wooden panel, etc. And it will also be constrained by our software capabilities. We proposed three designs in the beginning of the research and development phase, and evolve on our initial design along the way when the requirements become more specific. We also have a hardware document to record our proposals, final designs, justifications and evolutions. That document is also being updated on a weekly basis to reflect latest changes.

8.0 TOOLS

1. Git and GitHub

Git is a vision control system, it is used to manage our source code history. GitHub is a hosting service for Git repositories, where created private repositories and collaborate.

2. Dropbox

Dropbox is a file hosting service that we used to manage all of our documentation. We created a folder for our group, and also subfolders for each week.

3. Eclipse

Eclipse is the integrated development environment that we used to develop the software. We add leJOS as a plugin and use Java as programming language.

4. JDK and JRE

JDK is Java development kit; and JRE is Java runtime environment. JDK also include command line tools such as compiler and debugger which are necessary when developing Java applications. JRE is where our Java application runs in.

5. LeJOS

leJOS is a firmware replacement for Lego Mindstorms programmable bricks. It includes a Java virtual machine, which allows Lego Mindstorms robots to be programmed in the Java programming language.

6. LEGO Mindstorm

Lego Mindstorms is a hardware software platform produced by Lego for the development of programmable robots based on Lego building blocks.

9.0 GLOSSARY OF TERMS

10.0 EDIT HISTORY

Edit History:

Date	Version Number	Edit	Description of edit
22/10/2018	1	Ayden Malik	Inception of doc