SYSTEMS DOCUMENT

Version 2.0

22nd October 2018

Author: Shaodi Sun

TABLE OF CONTENTS

SYSTEM DOCUMENT

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 its in a corner and light localization when its at an intersection. We will integrate the classes like Navigation in previous labs into our final project.
TODO: Add a software flow chart in future development phase
This section should gradually grow more specific as we gain more information from the user.

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.

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.

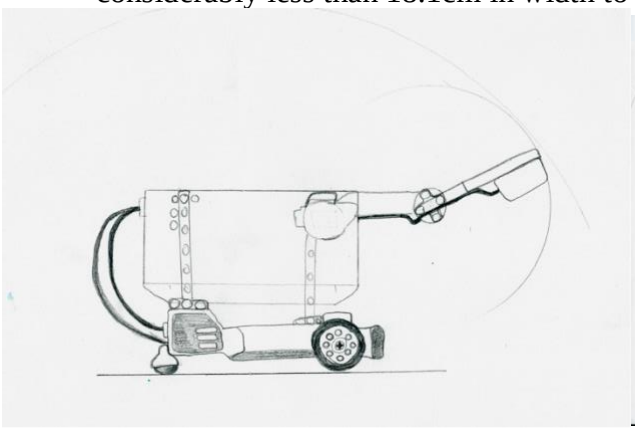
5.0 REUSABILITY

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 odometery. 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.



7.0 METHODOLOGIES

Currently, the problem is broken down into localization, odometry, bridge crossing, searching for the tree, rings scanning, color detection, and load return trip. The software classes that we may interface are:

1. Localization
2. Odometry
3. Navigation
4. Object detection

5. Color detection

More classes and methodologies may also be used in further development phase to achieve requirements.

8.0 TOOLS

1. Git and GitHub
2. Dropbox
3. Eclipse
4. JDK AND JRE
5. leJOS
6. LEGO Mindstorm

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