McGILL UNIVERSITY

ECSE 211: Final Design Project

System Document

Sadhvi Mehta

supervised by
Frank P. Ferrie

October 19, 2017

1 SUMMARY

This document is intended to identify and define the solution environment, i.e. the tools one can use and the basic components one must use to construct a solution. Thiss document identifies the capabilities of all parts that the team has – both software and hardware. In addition, it speaks of the compatibility of the design of the robot within our solution's environment with respect to the requirements given. It also gives a general overview of the approach being taken to solve the task at hand. Note, this document is subject to changes as the semester progresses.

CONTENTS

2. TASK

Project: Design Principles and Methods Final Project: Capture the Flag - Team 15
Task: Design an autonomous robot that can play a one-on-one version of the Capture the Flag while navigating through an obstacle course.

3. EDIT HISTORY

a. Document Version Number

i. 0.0.1: Version presented to Prof. Ferrie on the 2017/10/20

b. Edit History

i. Sadhvi Mehta - Initialize document, fill all if not most sections, combine hardware/software solutions suggested by the team.

ii. Sadhvi Mehta - Add general structure and tools section to document.

4. SYSTEM MODEL

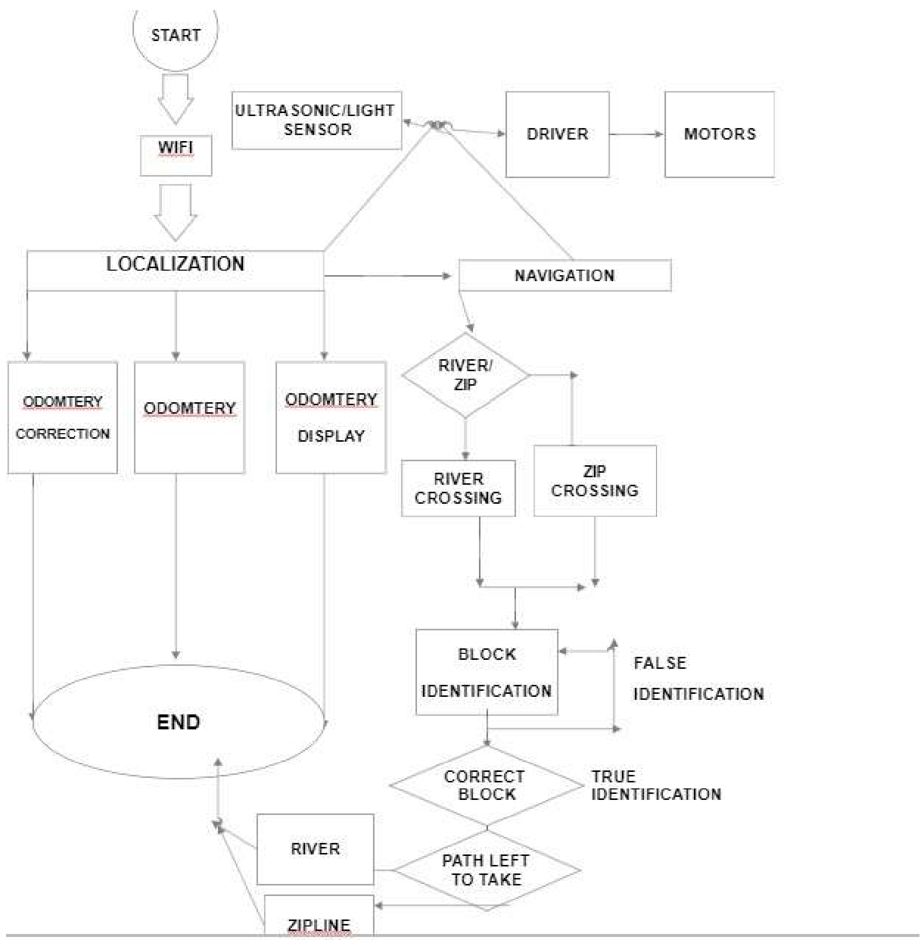

Figure 1: Tentative Block Diagram of System Based on Requirements.

5. HARDWARE AVAILABLE AND CAPABILITIES

The available hardware to build the robot is all found within a given DPM kit. Our team has access to three such kits. In brief, each kit consists of EV3 regulated motors, NXT motors, various types of Lego blocks and connectors, an ultrasonic sensor, a light sensor, a touch sensor, and a gyro. In addition, the main piece of hardware that links the software to the actual robot is also found within this kit: Lego Mindstorms EV3. Further details and constraints with respect to these parts are found below.

5.1: Lego Components:
Firstly, the Lego components have fixed compartments for connection or insertion of other Lego components. Consequently, this immediately limits the way one can connect pieces together, thus limiting one's design options and placements of sensors. More importantly, Lego components are not so robust. Eventually, with time and use, the component begin to bend and cause slight variations in the dimensions of the built design which can affect measurements used in the software.

5.2: Electromechanical Limitations:
Secondly, there are electromechanical limitations such as the power of the battery used to run the robot. This battery must be charged very often to ensure optimal performance of the robot as variations in the power can cause for variations in the robot's movements, resulting in unexpected outcomes. Also, the sensors are only able to accept integer values. Thus, time consuming scaling must be done in the software, reducing efficiency in order to gain accuracy.

5.3: Electronic/Processor Limitations:
Thirdly, although the EV3 brick has a AM1808 as its main processor, there are limitations when having to parse through readings obtained from the sensors. For example, as noticed in the Odometry Lab, the theoretical frequency to obtain the number of light sensor readings for accurate light detection would cause for too much load on the processor. Thus, other fixes must be found like reducing robot speed. Consequently, through testing, processor limitations may have workarounds.

6. SOFTWARE AVAILABLE AND CAPABILITIES

The main software tools available and that we will use include Git, Eclipse IDE, leJOS JVM, EV3 API.

- **Git -** Git is where all of our most optimized and current version of source code will be stored. Limitations wise, due to the lack of experience of two of the team members with programming, the concept of Git as a tool is providing a slight learning curve for these two members. However, once this learning curve is crossed, Git will provide an efficient way of exchanging code amongst each other.
- **Eclipse IDE -** Eclipse is the environment the entire team is using to code the software. It was chosen as everyone is familiar with this IDE and provides an efficient debugging tool.
- **leJOS JVM, EV3 API-** The leJos JVM limits the programming language to only Java. However, since Java is an OOP language, it provides enough abstraction to not have to worry about memory allocation. Also, the EV3 API greatly simplifies the movement of the robot and basic interaction of the robot. *(Please refer to constraints document, sec 5. for further info).*

7. COMPATIBILITY

Software wise, the main compatibility issue will the integration of the most efficient logic and code from the previous labs into one main program. In addition, apart from integrating previous code into our current source code, the different processes (e.g; navigation, localization) must also be compatible with one another. This will require its own separate integration period.

Hardware wise, there will be compatibility issues with the hardware design from the labs and the one needed for the final project. This is due to the new height as well as weight of the robot, which severely affects how the software conducts the robot.

8. RESUSABILITY

- **Software Reusability -** For the most part, the basic logic applied throughout the code of all previous labs will be reused. This includes the basic classes for the various functions the robot needs to perform: navigation, odometry, obstacle avoidance, etc. Please refer to section 4 of this document for a full list of this base software logic. More specific software mechanisms that will be reused include a team member's previous implementation of line detection: a difference in light intensity greater than a certain threshold will trigger line detection. This provides a more concrete solution towards the effect of ambient light.
- **Hardware Reusability-** The actual body of the robot may resemble the design of the robot used in lab 5 due to height requirements. More specific hardware reuse will also include

sensors connected to motors to allow for better visibility during navigation and obstacle avoidance.

9. STRUCTURES

- **Basic Hardware Structure:**
  1. Four motors will be used. Two of them will be used for robot movement on ground, One of them will be used to move robot across zipline, and another for the ultrasonic sensor to gain a full view of surroundings.
  2. Two wheel will be used as this serves as the bare minimum to balance the robot whilst it moves. An additional round ball will be added to the back for levelling.
  3. Ultrasonic sensor will be aligned with center of robot to simplify calculations and maintain center of balance. Light sensors will be placed at the bottom of robot at 90 degrees angle from each other.

- **Basic Software Structure:**
  Please refer to figure 1, section 4 of this document of full overview of software model.

10. METHODOLOGIES

- Use Navigation and Obstacle Avoidance algorithm implemented by Guillaume and Alia because it worked well in lab.
- Use median filter for any ultrasonic sensor readings to ensure no arbitrary values.
- Refer to section 8 of this document for light detection algorithm.
- Attachment of screws to outer sides of wheels to prevent fluctuations in wheel track.

11. TOOLS

Please refer to sections 5 and 6 of this document.

12. GLOSSARY OF TERMS

IDE- Integrated Development Environment
JVM- Java Virtual Machine