CONSTRAINTS DOCUMENT

Project: BOB: Bob's Our Builder
Task: To collect blocks and build a tower.
Document Version Number: 5.0
Date: November 12, 2016
Author(s): Jenny Zhao, Juliette Regimbal, Harley Wiltzer, Jastaj Virdee, Camilo Garcia La Rotta, Mathieu Lapointe, Adham El-Khouly
History Log / Version Revision Updates:
V1.0 Created Major Versions of Sections 1.0-6.0
V2.0 Bullet Point "*Forklift, Grabbing Arm, Forklift and Arm*" added to Section 3.0 "*Hardware Constraints*"; (Adham El-Khouly, Mathieu Lapointe)
V3.0 Updated Section 2.0 "*Environmental Issue*"; (Mathieu Lapointe)
V4.0 Updated Section 2.0 "*Environmental Issue*" to include localization specifications (Mathieu Lapointe)
V5.0 Updated Section 2.0 "*Environmental Issue*" to include size of Green/Red zone specification in (Mathieu Lapointe); finalized Project Name to "*BOB: Bob's Our Builder*"

1.0 TABLE OF CONTENTS

2.0 ENVIRONMENTAL ISSUES

The robot will be placed in a 12' by 12' wood surface. The competition will take place on the 2nd floor of the Trottier building where external light from outdoors will be present. This can affect the readings from the light sensors as they try to read the color of the Styrofoam blocks, which must be captured. In addition to this, there will be an opponent's robot on the field as well, which will add noise to the readings of the ultrasonic sensor. Both of the external factors must be taken into account.

The 12x12 wood surface will be constructed by joining 9 4x4 boards. The joints of these boards will not necessarily be smooth. There might be unevenness due to different heights of the different boards. This will affect robots and light sensors which are too low, and not accounting for these bumps might cause the light sensors on the underneath of the robot to get stuck, which will throw the robot off course and introduce errors into the odometry of the robot. Also, it is evident to note that the odometry correction depends on the black grid lines; therefore, an inconsistency in height might introduce errors, which will hinder the whole process. However, given the 8x8 board that is already present in the lab, the height has not caused any problems so far. However, on the night before the demo when we are testing on the 12x12 board, checking inconveniences resulted from the new introduced height inconsistencies should be on top of our list.

According to the specifications provided, obstacle blocks might be placed as close as one block size away from each other. This adds a constraint on the size of the robot that can be built. A robot having a size that does not fit inside one block will not be able to successfully avoid two obstacle blocks if they are placed one block apart from each other.

The height of a Styrofoam block is half its width, which means an ultrasonic sensor must be placed much lower on the body of the robot so as to be able to detect the presence of the blocks. Placing the ultrasonic sensor so close to the ground adds to the noise in the readings of the sensor as reflections of the sound waves from the ground are read as noise.

Also, the robot is forbidden from stabbing or damaging the Styrofoam block. Also, after localization, it must make a sound beep to signal that localization is over.

The initial heading of the robot and its position in the starting corner will be specified by one of the competition officials.

Once the parameters have been received the robot will start it cannot be touched by a team member at this point; there cannot be further button pushes. The clock time starts from the time the parameters are transmitted by the server.

The Red and Green zones will never be smaller than 1 square by 2 squares and never larger than 2 squares by 4 squares.

Given the instability of inter-brick communication in the current version of leJOS, once a connection is opened between the master and slave brick, we can only signal motors to move once. Beyond this point, any attempt to signal motors again after a threshold of a few seconds will throw an exception. Very limited resources are available on this subject matter and thus we have to accommodate this fact in our design. Given this is the case, we only have one chance to use the second brick to catch the flag and accordingly this adds more pressure to creating a very intact object detection algorithm.

*See the requirements document 2.5 Operating Environment for more information

3.0 HARDWARE CONSTRAINTS

We are provided with four EV3 bricks. This allows the use of 16 motors (16 output ports in total) as well 16 sensors (16 input ports in total). However, even though we can use all of them in theory, it will likely be unnecessary to do so. Moreover, increasing the size of the robot would be considered a disadvantage since it would limit its mobility.

There are limitations posed by the dimensions of most the lego pieces, especially the thickness of the pieces. Some examples discussed by the team are:

- *Forklift:* the use of a forklift designed using the Lego kit was suggested initially. The idea was then deemed unreliable. The ground base of the forklift on which the block would lie would be too thick, making it very difficult to get a block onto the forklift. In addition, using the forklift during the stacking process of blocks would be very difficult since it would end up destroying the whole tower, again due to the thickness of the lego pieces. A very thin surface that is not part of the Lego kit could solve this issue.
- *Grabbing arm*: The use of grabbing arm was suggested as well, and is going to be used. However, a limitation that is posed by the Lego kit is that there are no pieces that can guarantee that an arm would be able to grab a block in any orientation. In particular, if a block is captured on the ground there would be no problem, but once it is lifted off the ground there is no guarantee that it would not fall. The use of tires on the grabbing arm would significantly decrease the risk of dropping a block by increasing friction with the picked up object.
- *Forklift and Arm:* After progressing in the building process of the robot, a design that has both, a forklift and an arm, was suggested. This combination of both ideas allows the grabbing and stacking processes to be done without any difficulty. Two motors are being used for this design, one for the lifting of the arm using a forklift, as well as another motor for the grabbing action. The front part of the arm, the claw, was made wide enough to be able to catch a block in any possible orientation.

4.0 SOFTWARE CONSTRAINTS

4.1. LeJOS EV3 API:

The project's application program interface for the Lejos project is a well-documented library that contains useful routines, protocols and tools to build advanced functionalities for our final robot project. Specifications on how to interact with the physical components of the robot, such as sensors and motors, are a solid base case to complete the services we want our EV3 block to perform. The list below gives a brief description of such cases:

- *lejos.hardware*: provides access to all major physical components of the brick (motors, sensors and the brick LCD). Includes mutators and accessors such as Listeners, Samplers and mode setters. This class is an absolute necessity for the team to automate the robot's behavior based on external input factors.

- *lejos.remote:* provides access to the bricks components status (battery, LCD, SSH keys, ports, WiFi, among others) from the PC. This class will come in handy for debugging in the development stage, as well as in the wireless controlling in the test stage.
- *lejos.utility:* provides general helper methods such as debug messages, filters, sensor capabilities and timers. Alongside with *lejos.remote*, this class falls under the category of debugging tools at our disposal.
- *lejos.robotics:* this class is not allowed to be used under the current constraints given by the Professors for the final project. It contains classes and methods native to robotics applications such as localization, mapping, navigation, object identification, path handling, among others. During the pre-labs done as preparation for the final project, we have already built, tested and understood the functionality of the most important of these capabilities. In consequence, we are in a position where we can recreate the necessary classes from this library if we need to implement them in any service offered by our robot.

4.2. Programming Language:

The leJOS project is based on a Java Virtual Machine. It is ported for the NXT and the EV3 Lego Mindstorms. Lejos being the main framework for our ECSE 211 project, we are constrained to Java as the main programming language. It is interesting to note, that there are other programming languages we could've used for the robot, such as the NXTG Retail and RoboLab 2.9, both of which provide graphical programming, yet are inherently slowed down in execution speed.
NXTG and RoboLab projects are not backed by a worldwide used language, as it's the case with Oracle's Java. Java's documentation is up to date, as it is expected from a language that has topped for the last decade as one of the most used languages in the industry. Furthermore, as pointed out in the constraints document, section 3.0 Software availability and capability, functionalities intrinsic to robotics such as Object Oriented Programming, Multi-Threading, Synchronization and Exceptions, all add to our toolkit to write modular, portable and maintainable code for this project.

In terms of constraints, we are limited more by our capability to write and maintain quality modular and portable code in Java, than by Java's API and toolkit.

5.0 AVAILABILITY OF RESOURCES

5.1 General Meetings

According to Capabilities Document: Section 5.0 Availability, the only moment of the week where a general meeting could be held is on **Tuesdays from 1h p.m. to 1h30 p.m.**. This coincides with the TA meeting and therefore, the first 15 minutes will be reserved for updating Jason Dias on our progress. Hence, it's possible that a general meeting be held during a weekend. The exact moment will be determined a week prior to the meeting due to the unpredictability of everyone's weekend schedule.

5.2 Software Team Meetings

According to Capabilities Document: Section 5.0 Availability, the software team constituted of: Camilo, Harley, Juliette, Mathieu, Jastaj and Zisheng, cannot meet all at the same time during the week. However, everyone except Juliette is available on **Mondays from 4h p.m. to 4h45 p.m.** as well as on **Wednesdays from 4h p.m. to 4h45 p.m.**. If need be, we will meet without Juliette and update her with the important points highlighted during the meeting via our communication tool, Slack.

5.3 Hardware Team Meetings

According to Capabilities Document: Section 5.0 Availability, the hardware team constituted of: Adham, Juliette, and Jenny. The team manager should also be present during the meeting and therefore the only possible moment is on **Mondays from 1h p.m. to 1h30 p.m.**. If need be, we can meet without Jenny on **Wednesday from 10h a.m. to 12h p.m.** and update her afterwards via Slack.

Location for all meetings: Trottier Building, 5th floor in a private room.

REFERENCES:

http://www.lejos.org/ev3.php
http://www.legoengineering.com/nxt-g-examples-motors/
http://www.legoengineering.com/robolab-2-9-4c-patch/

6.0 GLOSSARY OF TERMS

1. mutator: methods used to control changes to a variable. They are also widely known as setter methods. accessor: which returns the value of the private member variable. The mutator method is most often used in object-oriented programming, in keeping with the principle of encapsulation.