*ecse 211 design project*

Software Document

Version *1.02*

*03/22/2018*

*ECSE 211 TEAM 11*

VERSION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Title** | Software Document | | | |
| **Description** | Keeps track of all software related implementation | | | |
| **Created By** | Patrick Ghazal, Software Team Leader | | | |
| **Date Created** | 2st March 2018 | | | |
| **Version Number** | **Modified By** | **Modifications Made** | **Date Modified** | **Status** |
| 1.00 | Patrick Ghazal | Created the Document. | 2nd March | Formatting required |
| 1.01 | Luka Jurisic | Peer reviewed the document. Formatted the Document | 3rd March | Preliminary Week 2 submission Content complete |
| 1.02 | Patrick Ghazal Volen Mihaylov | Added sections 3 and 4; Flowchart and class diagrams respectively.  Added Software Progress Report | 22nd March | All other sections remain |

TABLE OF CONTENTS

# 1 Table of Contents

# 2 API

# 3 FlowChart

# 4 Class Diagrams

# 5 Odometer

# 6 Localization

# 7 Navigation

# 8 Color detection

# 9 Display

# 10 Robot class

# 11 sEARCH

# 12 cAPTURE THE FLAG

# software progress report

At the time of writing this report, the progress made on the software for this lab is the following. Milestones 00 through 02 were completed (in order: Shell complete, Gear system functional, Navigation functional). The robot’s Navigation has been deemed acceptable after testing. One of our main goals was to implement the Robot class for standing constants and centralization of data, this was also completed. The entirety of movement-related methods (forward(), convertDistance(), etc.) were moved to the Navigation class. The Display class was removed to allow the Robot to use more resources on the other threads. The two tables below indicate the progress made.

|  |
| --- |
|  |
| Excerpt from README.MD |

**Most Recent Git Commits:**

|  |  |
| --- | --- |
| 02.00.01 - All classes- Added JavaDoc documentation | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 16:48:28 2018 -0400 |
| 02.00.00- Navigation, Controller- Finished Navigation pretty flawless but a bit slow | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 16:18:10 2018 -0400 |
| 01.01.00 -Navigation, Controller- Navigation turning and going forward is fl awless to a mm/degree | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 16:01:22 2018 -0400 |
| 01.00.08-Controller + all classes-Changed Controller for testing again and a dded more variables to Robot and added moveBy function to navigation | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 15:56:25 2018 -0400 |
| 01.00.08-Controller + all classes-Changed Controller for testing again and a dded more variables to Rrobot and added moveBy function to navigation | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 15:22:24 2018 -0400 |
| 01.00.07-Controller-Changed Controller for testing again | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 14:51:17 2018 -0400 |
| 01.00.06-Controller-Changed Controller for testing | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 14:28:10 2018 -0400 |
| 01.00.05-All files except Rob,Nav-Fixed motor calls | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 14:09:01 2018 -0400 |
| 01.00.04-Robot,Nav-Moved all motor calls from Robot to Nav | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Thu Mar 22 14:04:56 2018 -0400 |
| 01.00.03-Several Files-Removed unused imports, updated some javadoc, etc | Author: PatrickGhazal <patrick.ghazal@mail.mcgill.ca>  Date: Thu Mar 22 13:28:17 2018 -0400 |
| 01.00.02-Cont, LLoc, Nav, Odo, Robot, USLoc-Moved all wheel behaviour and va riables to Robot class  Deleted motor instances in all classes, moved them to Robot, updated other c lasses with method calls that matched the new Robot class | Merge: 2cc3ad2 d041d29  Author: PatrickGhazal <patrick.ghazal@mail.mcgill.ca>  Date: Thu Mar 22 10:50:12 2018 -0400 |
| 01.00.01-USLoc, LightLoc, Robot-Moved some constants to Robot class | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Wed Mar 21 19:28:28 2018 -0400 |
| 01.00.00-Nav,Cont,Robot-Added gear system functions in Nav, changed robot co nstants | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Wed Mar 21 19:21:53 2018 -0400 |
| 00.00.00--Made initial shell | Author: Volen Mihaylov <volen.mihaylov@yahoo.com>  Date: Wed Mar 21 12:28:21 2018 -0400 |
| Initial commit | Author: bryanjay28 <35203044+bryanjay28@users.noreply.github.com>  Date: Mon Mar 12 11:39:02 2018 -0400 |

# 2 API

The Application Programmer Interface has been done through JavaDoc on Eclipse and generated through it as well. The needed HTML files are included with the weekly reports as demanded. The API is updated as the code changes.

# 3 Flowchart

# A close up of text on a white background Description generated with very high confidence

# A close up of text on a white background Description generated with high confidence4 Class Diagrams

# 5 ODOMETER

The Odometer class (and its associated classes OdometerData and OdometerExceptions) has been used since Lab 2. Therefore, the group is quite familiar with its functionality and usage, and we do not plan to bring any changes to its implementation. The only possible change here is an adaptation of code to ensure that communication between methods of different classes remains as smooth as possible.

# 6 localization

For Lab 5, we still had both an ultrasonic- and light-based Localization class. Given that localization will still need to be performed for the final project, it is likely that both classes will be found in our final code, albeit perhaps modified to better fit the needs of the project.

Regarding the ultrasonic localization, we have found it best to use a rising or falling edge routine depending on the initial distance read of the sensor and not to enforce one over the other.

As to the light localization, the procedure could definitely use some improvement, as it is time-costly now. We will make sure to cut the required time for the procedure as much as we can.

# 7 NAVIGATION

The travelTo and turnTo methods are the centerpiece of the Navigation class and have proven very useful in past labs. We have had to adapt the behavior of the travelTo method for Lab 5 due to necessary obstacle avoidance and block searching, however these changes will be repackaged and re-implemented differently in the code for the project, due to the change in expected behavior.

# 8 cOLOR detection

In Lab 5, colour identification was done using the mean and standard deviation of the RGB intensities, with mitigated success (the detection was successful but only at small sensor-block distances). For this lab, we plan to experiment with the existing functionalities of the sensor, namely the colorID attribute. If it shows better success in recognizing blocks, we shall implement it.

# 9 display

Display is another class inherited from past labs that has little changed since we first started using it. It is practical to assess whether the odometer is accurate or not. To reduce the number of threads running at any one time in the code, it was discussed that we could perhaps implement it within the Odometer class.

# 10 robot class

The Robot class is a new addition to our code. Its purpose is to contain and manage all the constants and variables related to the robot: motors, sensors, constants such as the track and wheel radius, etc. It will allow for the centralization of this data, which will allow any class to easily find and manipulate it. Previously, some objects were instantiated and/or passed along several times, causing confusion and possibly interfering with the robot’s functionality.

# 11 Search

The search procedure that was implemented in Lab 5 was to circle around the search area, find blocks using a sensor on the side of the robot, visit a block once detected, and act according to the color of the block. If the block was of the target colour, the search would terminate, and the robot would immediately head to the upper right point of the search area. If the block was not of the target colour, the robot would head back to the path it was following and proceed. Here, the target and procedure being slightly different, it is likely that we will adapt our search procedure to the situation. We may use an algorithm such as the wavefront algorithm if it proves useful and relevant to our situation, where we do not know how many blocks there are or even how many.

# 12 capture the flag

CaptureTheFlag is the initial entry point into the code for the entire project. Its main method will initialize the variables of the Robot class as well as the necessary threads, then run the demonstration by coordinating the other classes.