Test Document

Project: LIBERTY

Task: Test the integral system

Document Version Number: 1.0
Date: 22/11/2017
Author: Edward Son
Editor: Andi-Camille Bakti

Edit History: https://github.com/Gabetn/DPM_01_Project_Documentation

McGill

TABLE OF CONTENTS

1. TESTS:

1.1 Red integration corner three test

Test 1: Full playthrough integration test

Date: 22/11/2017
Tester: Edward Son
Author: Edward Son

1) This test will validate the functionality of the integrated code so far, in terms of being a read team player, and running through the whole course.
2) This test should make the robot successfully complete bridge navigation, capturing, zipline traversal, and return to starting position. The angle, X and Y position error will be measured between each of these components. The angle should be under 4 degrees of error. The X and Y positions should be under 2 cm of error.
3) The brick should be positioned near corner 3, in any orientation. The robot receives data over wifi, the bridge coordinates (SH_LL_x, SH_LL_y), (SV_LL_x, SV_LL_y) and (SV_UR_x, SV_UR_y), as well as the search zone and zipline endpoints. Once data is acquired, the command "start" is used on the server GUI and the robot starts localization. After localization, the angle is measured, and a button press continues to navigation. It first navigates to the points (SH_LL_x, SH_LL_y) and localizes. The robot proceeds to navigate to the bridge point (SV_UR_X - 1), (SV_UR_Y - 1), where it localizes again, and the X, Y and angle error are measured. A button press gives it the go to reach the end of the bridge, and stops for another angle check. Finally, localization is run and angle error is measured at the end. The following button presses navigates it to the capture point, the zipline and the starting point. The angle, X and Y error are to be calculated before each of these button presses.
4) The brick should end up back at the starting corner after having successfully completed the whole playthrough (bridge traversal, capturing, zipline traversal).
5)

Test Run #	**Theta error after localization (in ^{0})**	**Error X (cm)**	**Error Y (cm)**

1	1.5	0.0	0.5
2	0.0	0.5	0.5
3	0.0	1.0	0.5
4	1.0	0.0	0.5
5	0.0	0.0	1.0
6	-2.0	0.5	1.0
7	0.0	0.0	1.0
8	1.0	0.5	1.0
9	0.0	0.0	1.0
10	0.0	1.0	1.0

Figure 1: Position after localization (angle precision)

Test Run #	**Starting Position (x,y)**	**SH angle reloc. (°)**	**SH_LL_x (cm)**	**SH_LL_y (cm)**	**SV angle reloc. (°)**	**SV_UR_X (cm)**	**SV_UR_Y (cm)**	**Final bridge position (cm)**
1	(1,1)	90.0	N/A	N/A	N/A	N/A	N/A	N/A
2	(1,1)	95.0	N/A	N/A	N/A	N/A	N/A	N/A
3	(1,1)	92.0	N/A	N/A	N/A	N/A	N/A	N/A
4	(1,1)	2.0	1.0	0.5	1.5	0.5	0.5	N/A
5	(1,1)	1.0	0.0	1.0	0.0	N/A	N/A	N/A
6	(1,1)	-3.0	0.5	1.0	1.5	13.5	1.0	N/A
7	(1,1)	0.0	0.0	1.0	0.0	N/A	N/A	N/A
8	(1,1)	1.0	0.0	0.5	1.0	N/A	N/A	N/A
9	(1,1)	1.0.	0.5	1.0	0.5	N/A	N/A	N/A

10	(1,1)	73.0	2.0	20.5	0.0	N/A	N/A	N/A

Figure 2: Bridge navigation test (positioning precision)

Test Run #	**Capture outcome**
1	N/A
2	N/A
3	N/A
4	N/A
5	N/A
6	N/A
7	N/A
8	N/A
9	N/A
10	N/A

Figure 3: Capture test (outcome)

Test Run #	Zipline traversal outcome
1	N/A
2	N/A
3	N/A
4	N/A
5	N/A

6	N/A
7	N/A
8	N/A
9	N/A
10	N/A

Figure 4: Zipline test (outcome)

Test Run #	Navigation to beginning outcome
1	N/A
2	N/A
3	N/A
4	N/A
5	N/A
6	N/A
7	N/A
8	N/A
9	N/A
10	N/A

Figure 5: Return test (outcome)

	X (cm)		Y (cm)		Angle (°)	
	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean
Initial localization	0.4	0.4	0.3	0.8	0.9	0.2
SH_LL	0.7	0.6	7.4	3.6	45.4	35.2
SV_UR	9.2	7.0	0.4	0.8	0.7	0.6

Figure 6: Standard Deviation and Mean of successful localizations

6) The brick successfully localizes with 100% success rate. It navigates to the beginning of the bridge with 70% success rate and localizes. The brick navigated to the next point in the bridge only 20% of the runs. Finally, it fails to navigate to the end of the bridge, and to execute capturing, zipline traversal and return to corner 3. The standard deviation and mean for initial localization is all under 1, which is considered very good. For the initial bridge, the angle standard deviation and mean are quite off, which is due to the initial 3 test runs turning in the wrong direction, causing the angle to be off by more than 90 degrees. This was due to the navigation class considering a 12x12 board instead of a 8x8 board which was actually being tested on. Finally, the SV_UR standard deviation and mean for x are off by around 8 cm, but the rest is satisfactory. In conclusion, the integration of the latter and bridge traversal is unsuccessful. This seems to be due to the brick not being able to properly finish off the bridge traversal, so it wanders off in the wrong direction when trying to execute capturing, often running into a wall.
7) There should be an investigation about bridge traversal in the robotControl class, which creates the path for the brick to run once wifi data is acquired. It may be due to wrong data being given over Wifi, or wrong data being used to drive to.

To calculate the error we used the Euclidean distance error ϵ:

$$\epsilon = \sqrt{(X - X_F)^2 + (Y - Y_F)^2}$$

X_f = *The final X position of the robot*
Y_f = *The final Y position of the robot*

The average (AM) was calculated by using the following formula:

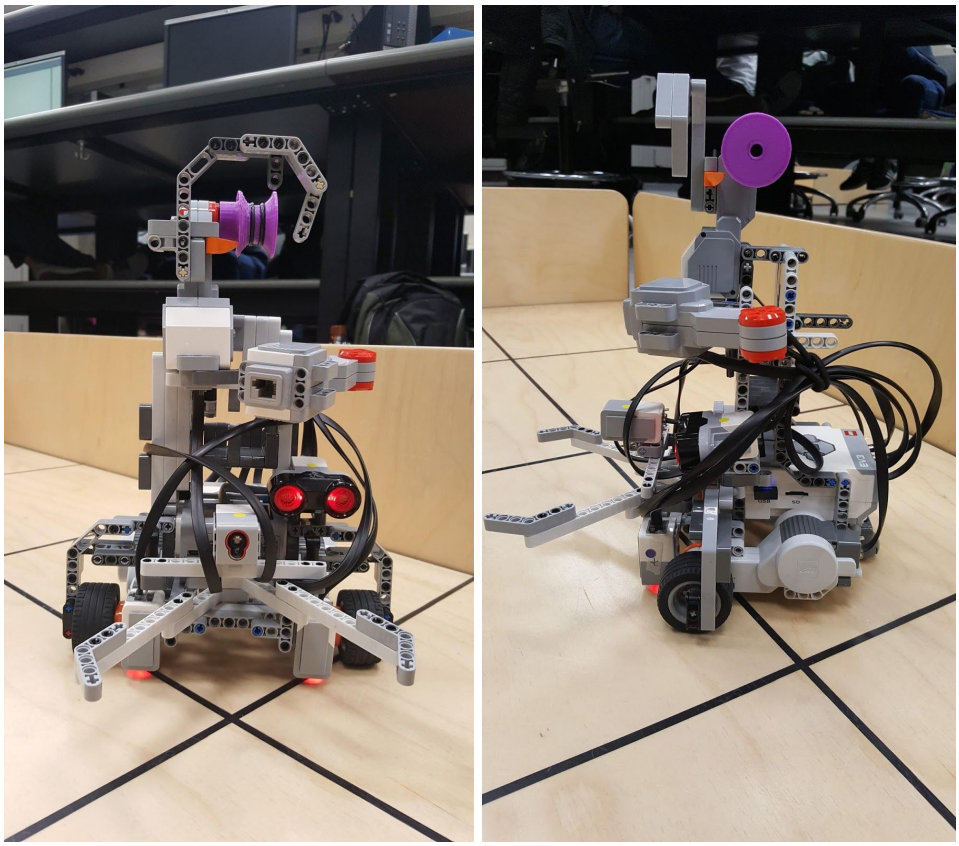
$$Average = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{1}{n}(a_1 + a_2 + \cdots + a_n)$$

We use the sample standard deviation formula (see below) to calculate the sample standard deviation.

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

$\overline{x}$ = *Mean*
N = *Sample size*
x_i = *Sample at i*

2. HARDWARE



See ***HARDWARE - 2.0.***

3. Source Code used

See github group repository at commit: d4f57913294490a9db953a5af5995e3eea75ffd5