Test Document

Project: LIBERTY

Task: Test the integral system

Document Version Number: **2.0**
Date: 29/11/2016
Author: Edward Son
Editor: Andi-Camille Bakti
Edit History: https://github.com/Gabetn/DPM_01_Project_Documentation

McGill

TABLE OF CONTENTS

1.0 TESTS:

1.1 Green integration corner one test

Test 1: Integration (Localization + Navigation + Dismount)

Date: 19/11/2017
Tester: Edward Son
Author: Edward Son

1) This test will validate the functionality of the integrated code so far, in terms of getting to the other end of the zipline.
2) This test should make the robot drive to the coordinates over wifi, more specifically to the red zipline other point (ZO_R_x, ZO_R_y). At that point, it should localize and face true zero. The angle should be under 4 degrees of error. The x and y error mean should be 2 cm or less.
3) The brick should be positioned near corner 1, in any orientation. The robot receives data over wifi, the zipline coordinates (ZO_R_x, ZO_R_y and (ZO_G_x, ZO_G_y). 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 where it turns to true zero, and the angle is measured again. The robot proceeds to navigate to the zipline point, where it localizes again, and the X, Y and angle error are measured. A button press gives it the go start the zipline traversal, and stops when dismounted for another angle check. Finally, localization is run and angle error is measured at the end.
4) The brick should end up at the point (ZO_R_x, ZO_R_y), after successfully completing each individual component
5)

Test Run #	**Theta final error (true zero) (in ^{0})**	**Theta error after localization (in ^{0})**
1	3.0	0.0
2	10.0	5.0

3	4.0	1.0
4	6.0	2.0
5	2.0	0.0
6	4.0	0.0
7	4.0	1.0
8	8.0	3.0
9	4.0	0.0
10	5.0	1.0

Figure 1: Position after localization (angle precision)

Test Run #	**Starting Position (x,y)**	**Xf (cm)**	**Yf (cm)**	**X(cm)**	**Y(cm)**
1	(1,1)	1.0	3.0	0.5	0.0
2	(1,1)	11.0	5.8	0.0	0.0
3	(1,1)	1.5	2.0	0.5	0.5
4	(1,1)	12.8	5.5	0.0	0.0
5	(1,1)	0.0	1.8	0.0	0.0
6	(1,1)	1.0	0.0	0.0	0.5
7	(1,1)	1.5	2.0	0.5	0.0
8	(1,1)	2.0	1.9	0.0	0.0
9	(1,1)	0.5	1.7	0.0	0.5
10	(1,1)	0.8	1.3	0.0	0.0

Figure 2: Navigation test (positioning precision)

Test Run #	**Theta error final (in ^{0})**
1	3.8
2	14.0
3	4.5
4	21.0
5	2.0
6	0.0
7	-1.0
8	0.0
9	1.0
10	1.0

Figure 3: Navigation test (angle precision)

Test Run #	Theta final (after localization)	Theta after dismount
1	0.0	3.0
2	N/A	N/A
3	0.0	0.0
4	N/A	N/A
5	0.0	1.0
6	N/A	N/A
7	0.0	3.0
8	N/A	N/A

9	0.5	4.0
10	1.0	3.0

Figure 4: Dismounting localization (angle precision)

	X (cm)		**Y (cm)**		**Angle (°)**	
	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean
Initial localization	0.2	0.2	0.2	0.2	1.6	1.3
zipline localization	4.6	3.2	1.8	2.5	7.2	4.6

Figure 5: Standard Deviation and Mean of position before zipline

	Angle (°)	
	Standard Deviation	Mean
Theta dismount	1.5	2.3
Theta final	0.4	0.3

Figure 6: Standard Deviation and Mean of position after zipline

6) The brick successfully reaches the end point 60% of the test runs. Most of the times it failed is due to the navigation turns being off, which causes the robot to not properly localize in front of the zipline. Thus, the robot misses the zipline. This may be due to the wheelbase or wheel radius values being off. As the standard deviation for the zipline localization shows, the x error is over 2 cm and y error is around 2 cm. This is enough to miss the zipline. On top of that, the angle error deviation is 7.2, and the mean 4.6, which indeed confirms that even with localization, the turning is not precise.
7) There should be an investigation about navigation turning to figure out why the turns are not accurate. It may be due to the wheel base not being the correct value, since the turn to

method relies on this value to turn precisely. Then, include these fixing in the next integration test, and verify that the zipline mounting success rate goes up.

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

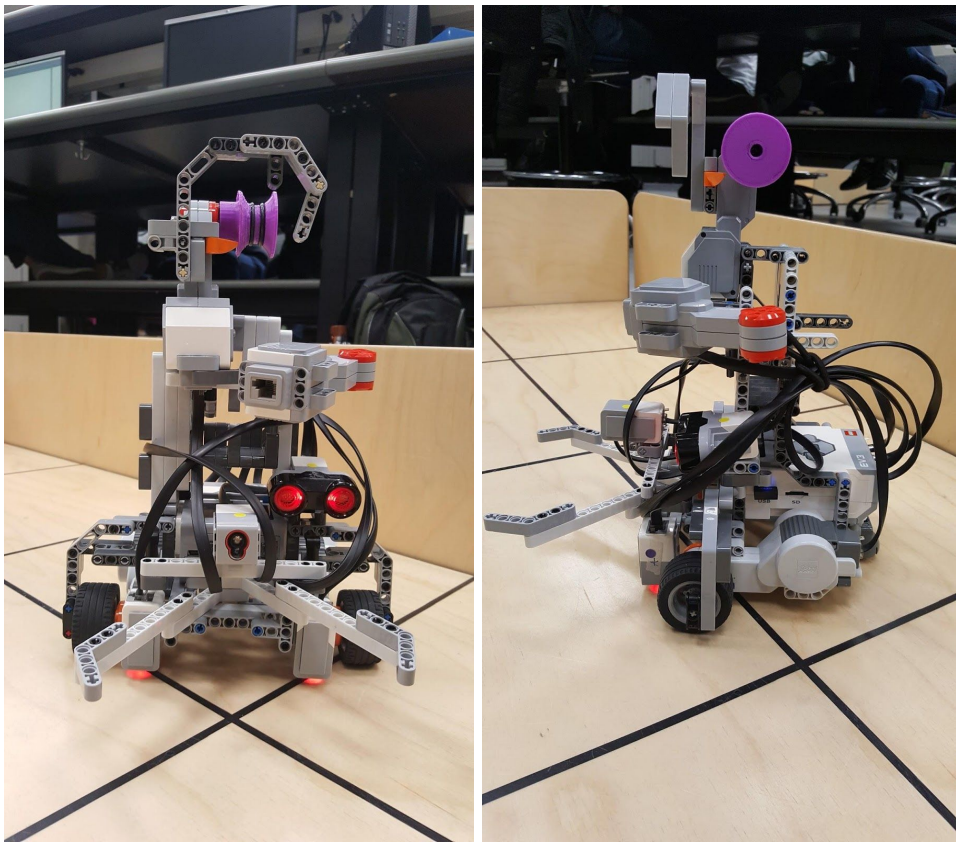
$$Average = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{1}{n}(a_1 + a_2 + \cdots + a_n)$$

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: 941f1c2c2ba3f6c1ed7a9073f08fd72fd2747e2e