Software Document

Project: LIBERTY

Task: Software Design

Document Version Number: 1.0
Date: 2017/10/22
Author: Bill Zhang
Edit History: 2017/10/30 by Bill Zhang

McGill

TABLE OF CONTENTS

1. SUBSYSTEM DIVISION

Localization: localize the robot using light and ultrasonic sensors
Navigation: navigate the robot between two points also included path generation
WiFi Communication: take the input coordinates from the server
Capture: capture the flag by beeping 3 times
Zipline: traverse the zipline

2. CLASS DIAGRAM AND INTERACTIONS

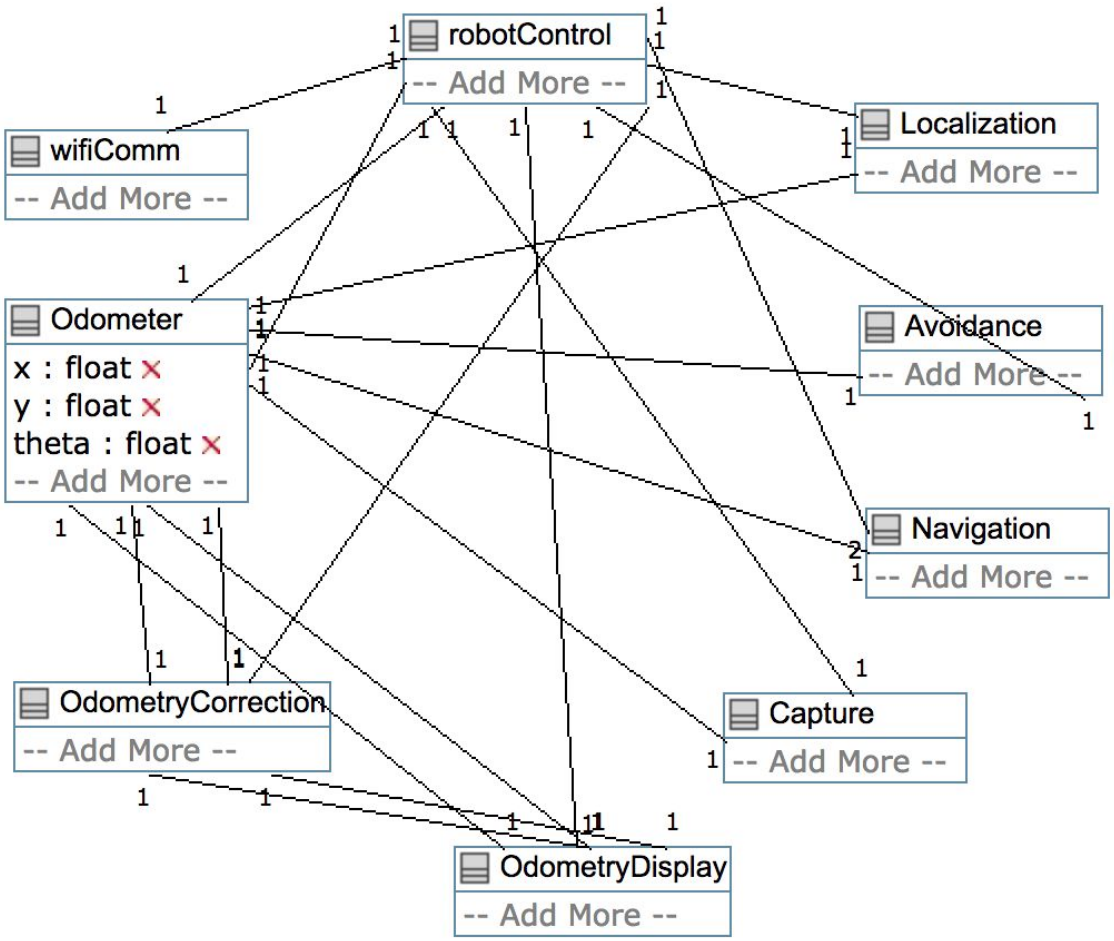


Figure 1: Class Diagram

Robot Control: the main class to call other classes when needed. The main method is written in this class

Localization: first perform the ultrasonic localization and then the light localization to ensure the accuracy of x, y and θ when starting

Navigation: drive between two points and generate a path based on the points given. Two instances should be running in the control class. One is for the driving motor, the other is for the zip-line traversal motor which only rotates.

Odometry Correction: correct the x, y and θ using light sensor when encountering a black line

Avoidance: Avoid any obstacles including useless flags

Capture: determine if the flag is the wanted one and beep 3 times to capture it

3. DEPENDENCY AMONG CLASSES

Class Name	**Dependency**
Robot Control	All other classes
Localization	Robot Control & Odometer
Navigation	Robot Control & Odometer
Odometry Correction	Robot Control & Odometer
Avoidance	Robot Control & Odometer
Capture	Robot Control
Wifi Communication	Robot Control
Odometry Display	Odometry Correction, Odometer, Robot Control
Odometer	Robot Control, used by Localization, Navigation, Odometry Correction, Odometry Display and Avoidance

Table 1: Class Dependencies

4.0 OVERALL SOFTWARE WORKFLOW

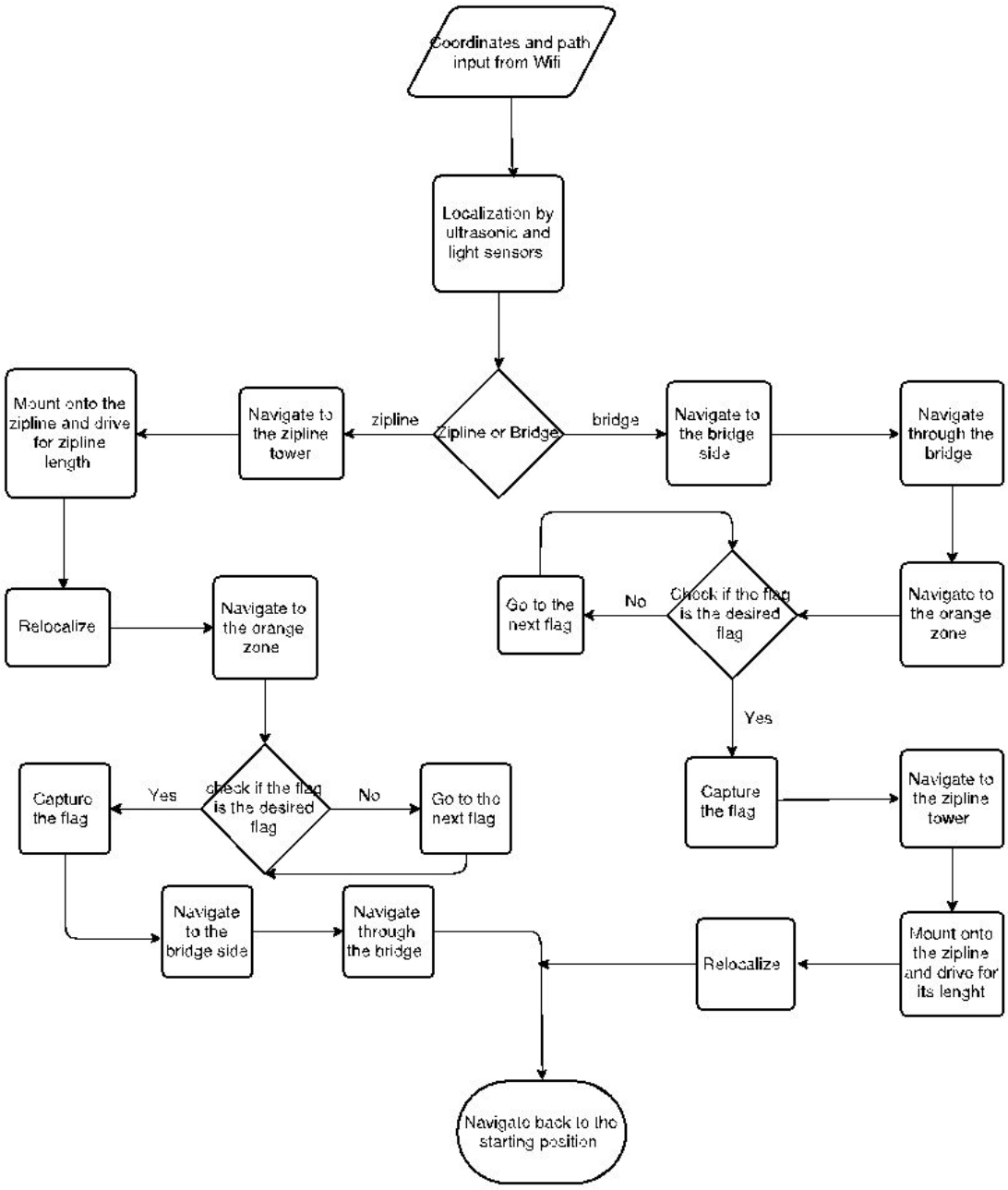


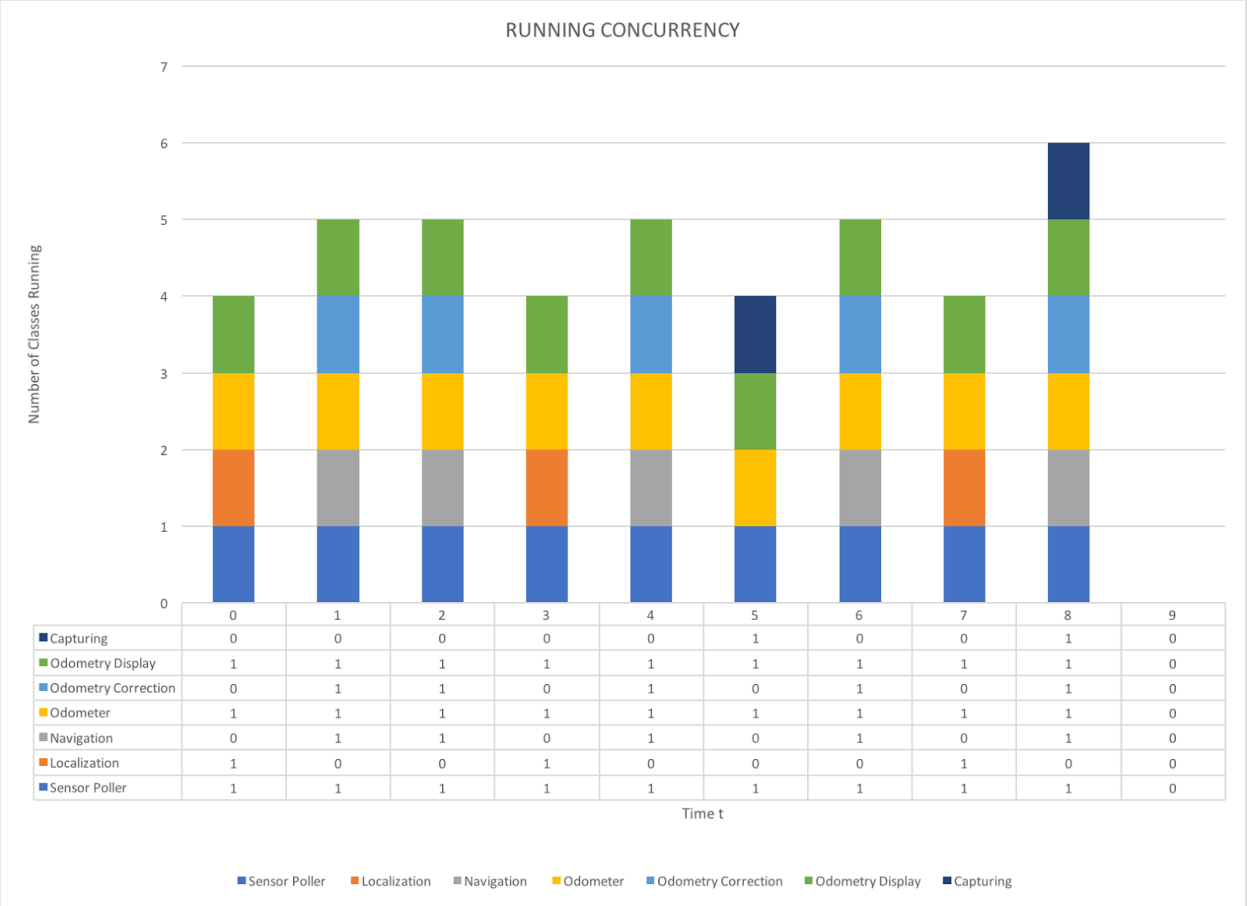
Figure 2: Overall System Workflow

5. SOFTWARE STATUS

Class	Workload	Percentage
robotControl	100 lines	20%
Navigation	300 lines	50%
Ligth Localization	270 lines	100%
Ultrasonic Location	200 lines	100%
OdometryCorrection	150 - 200 lines	15%
SensorPoller	50 lines	0%
Capturing	100 - 150 lines	0%

Table 2: Software Completion

6. SOFTWARE CONCURRENCY



	0	1	2	3	4	5	6	7	8	9
Capturing	0	0	0	0	0	1	0	0	1	0
Odometry Display	1	1	1	1	1	1	1	1	1	0
Odometry Correction	0	1	1	0	1	0	1	0	1	0
Odometer	1	1	1	1	1	1	1	1	1	0
Navigation	0	1	1	0	1	0	1	0	1	0
Localization	1	0	0	1	0	0	0	1	0	0
Sensor Poller	1	1	1	1	1	1	1	1	1	0

Figure 3: Class and Thread Concurrency

Time t	Action
0	Localize around the starting position
1	Navigate to the waypoint
2	Traversal the zipline or navigate through the bridge
3	Re-localize on the other side
4	Navigate to the orange zone
5	Capture the flag
6	Navigate back to the other side via zipline or bridge
7	Re-localize on the other side
8	Navigate back to the starting point

9	System stops

Table 3: Corresponding time frames for **Figure 3.**

7. ARCHITECTURE DESIGN

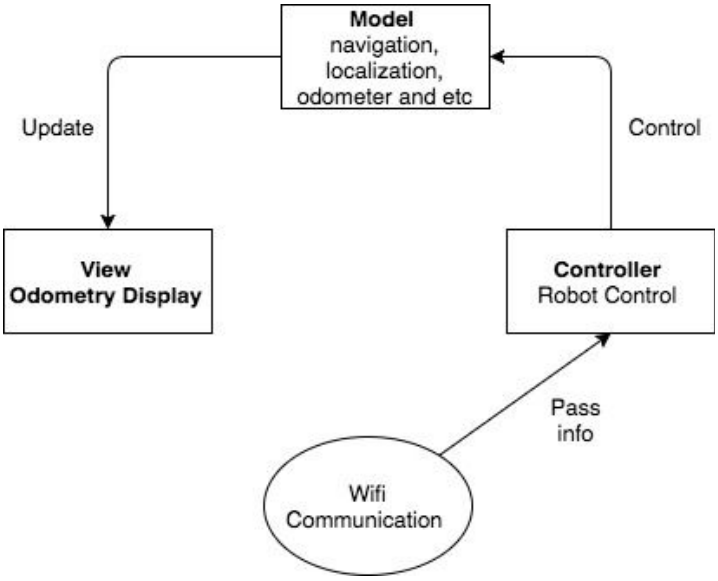


Figure 4: Architecture Design Diagram