Software Document

Project: LIBERTY

Task: Software Design

Document Version Number: 2.0
Date: 2017/11/05
Author: Bill Zhang
Edited by: Andi-Camille Bakti

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

McGill

TABLE OF CONTENTS

1. SUBSYSTEM DIVISION

The requirements for the project are divided into five subtasks (see ***REQ - GEN***) and so it follows that the software component of the design will have to meet these five sub-requirements which allows for a logical division of the system into the following:

- **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 three times
- **Zipline**: traverse the zipline

Furthermore, as research during labs 3 to 5 have suggested, the navigation of the subsystem have a low tolerance for error and can easily be affected physically (slippage of the wheels and varying battery levels) which accumulates over large distances. To address this issue we add the additional task of correcting the odometer (**Odometry correction**).

2. CLASS DIAGRAM AND INTERACTIONS

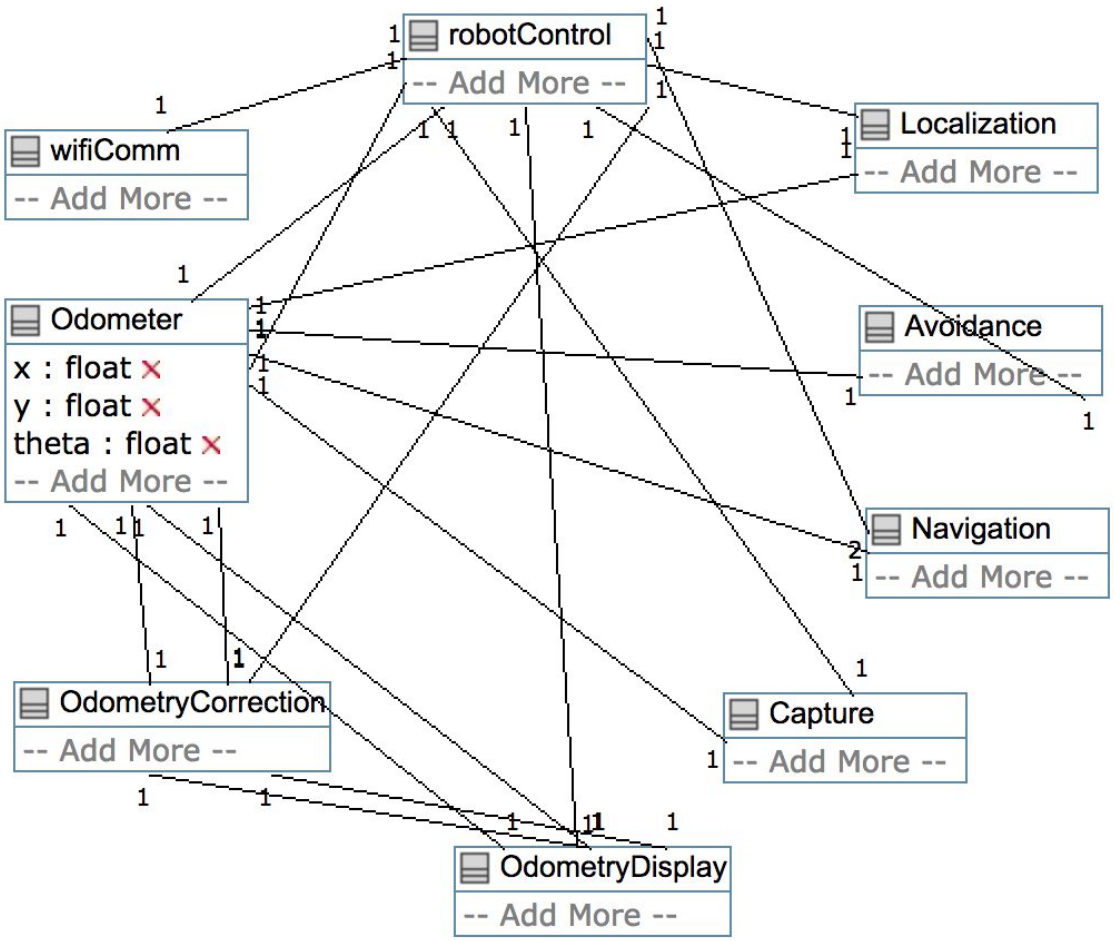


Figure 1: Class Diagram

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

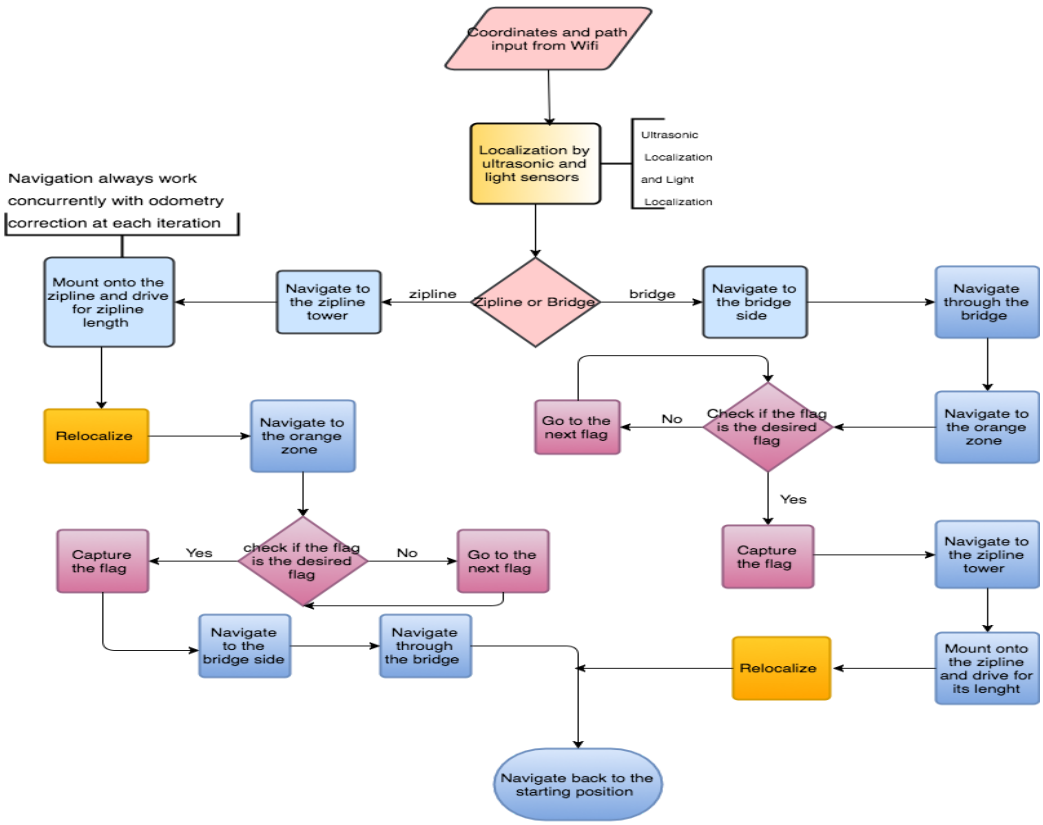
Capture: determine if the flag is the wanted one and beep 3 times to capture it

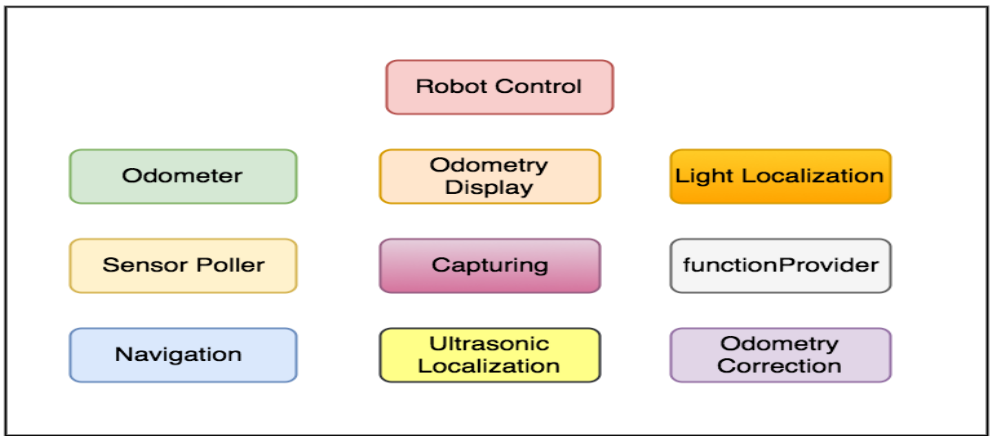
3. DEPENDENCY AMONG CLASSES

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





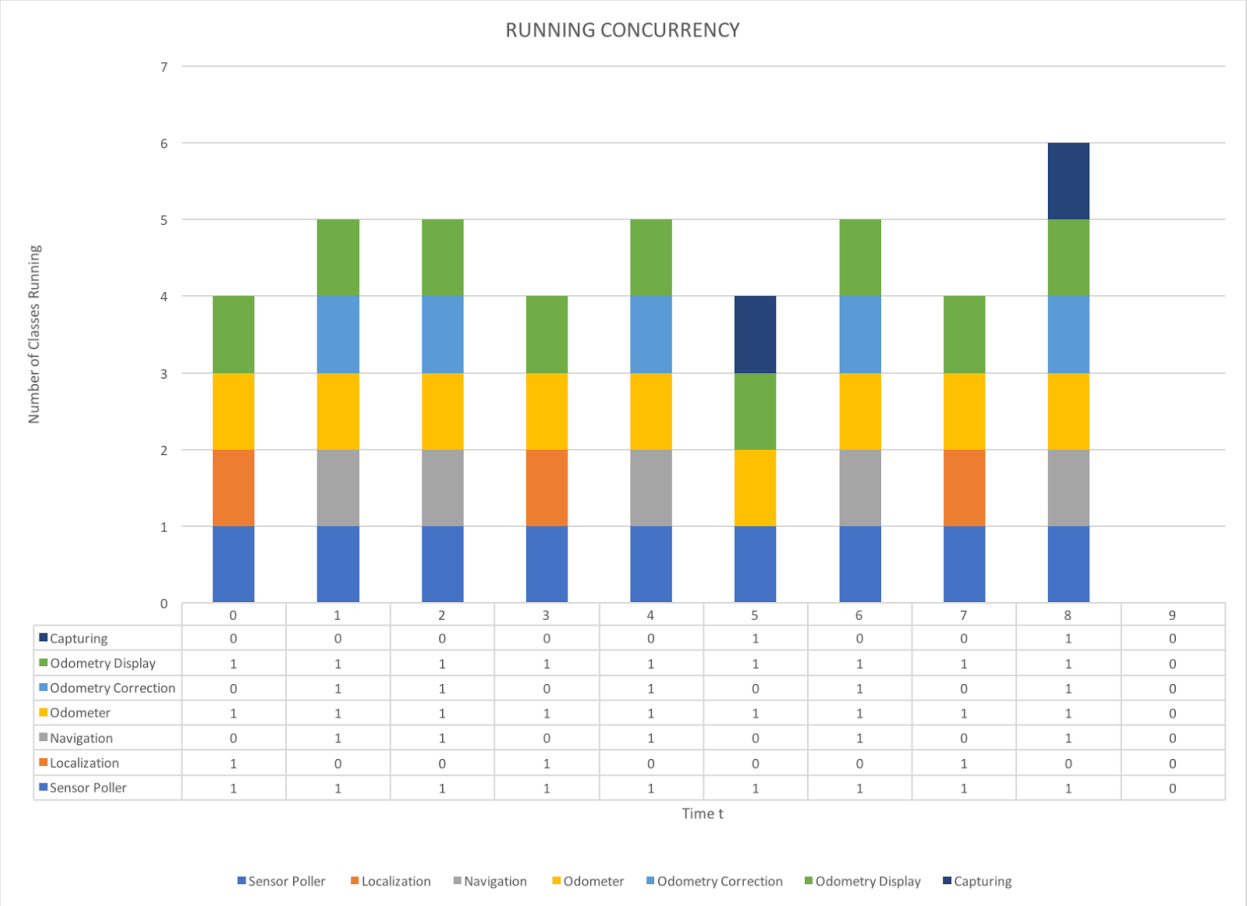
5. SOFTWARE STATUS

Class	Workload	Percentage
robotControl	170 lines	85%
Navigation	300 lines	85%
Light Localization	270 lines	90%
Ultrasonic Localization	200 lines	100%
Odometry Correction	150 - 200 lines	15%
SensorPoller	100 lines	100%
Capturing	200 lines	10%
functionProvider	40 lines	100%

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

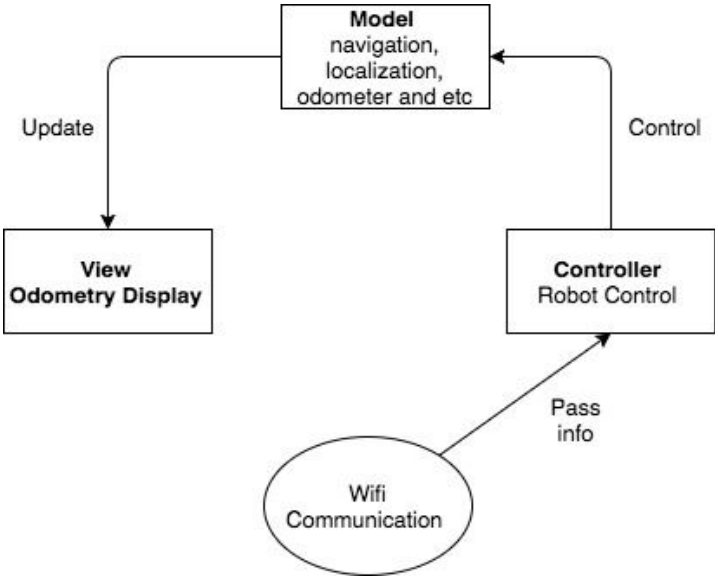


Figure 4: Architecture Design Diagram