Software Document

Project: LIBERTY

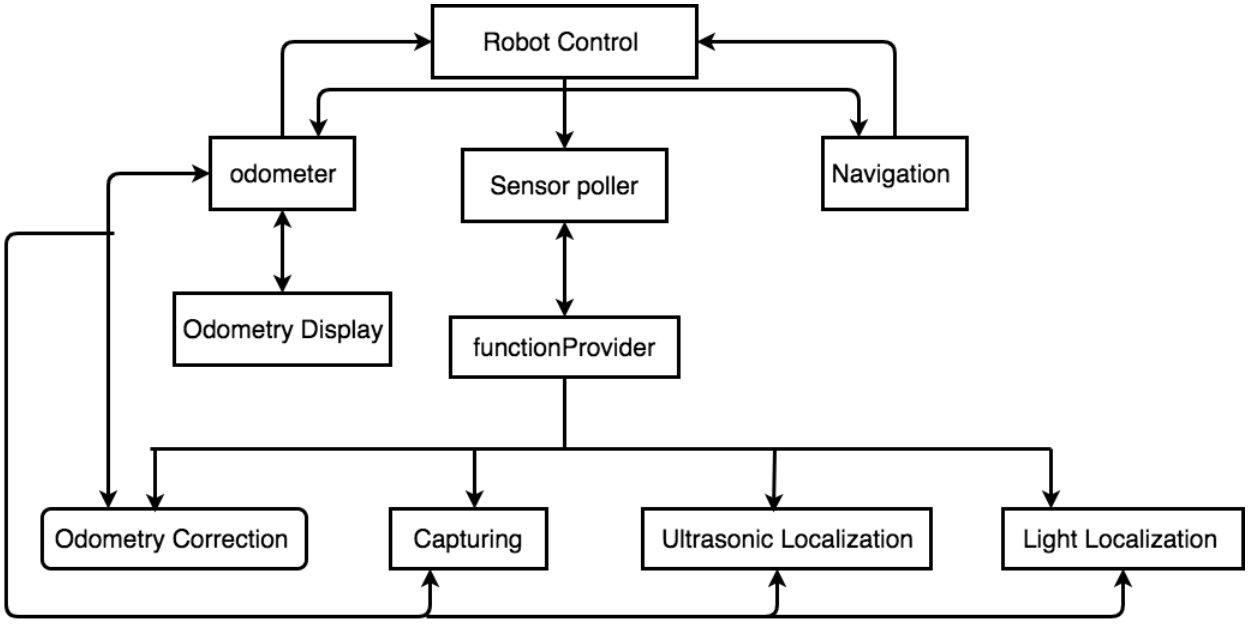
Task: Class Hierarchy

Document Version Number: 2.1
Date: 2017/10/28
Author: Bill Zhang
Edit History: https://github.com/Gabetn/DPM_01_Project_Documentation

McGill

TABLE OF CONTENTS

1. Class Hierarchy



2. Description

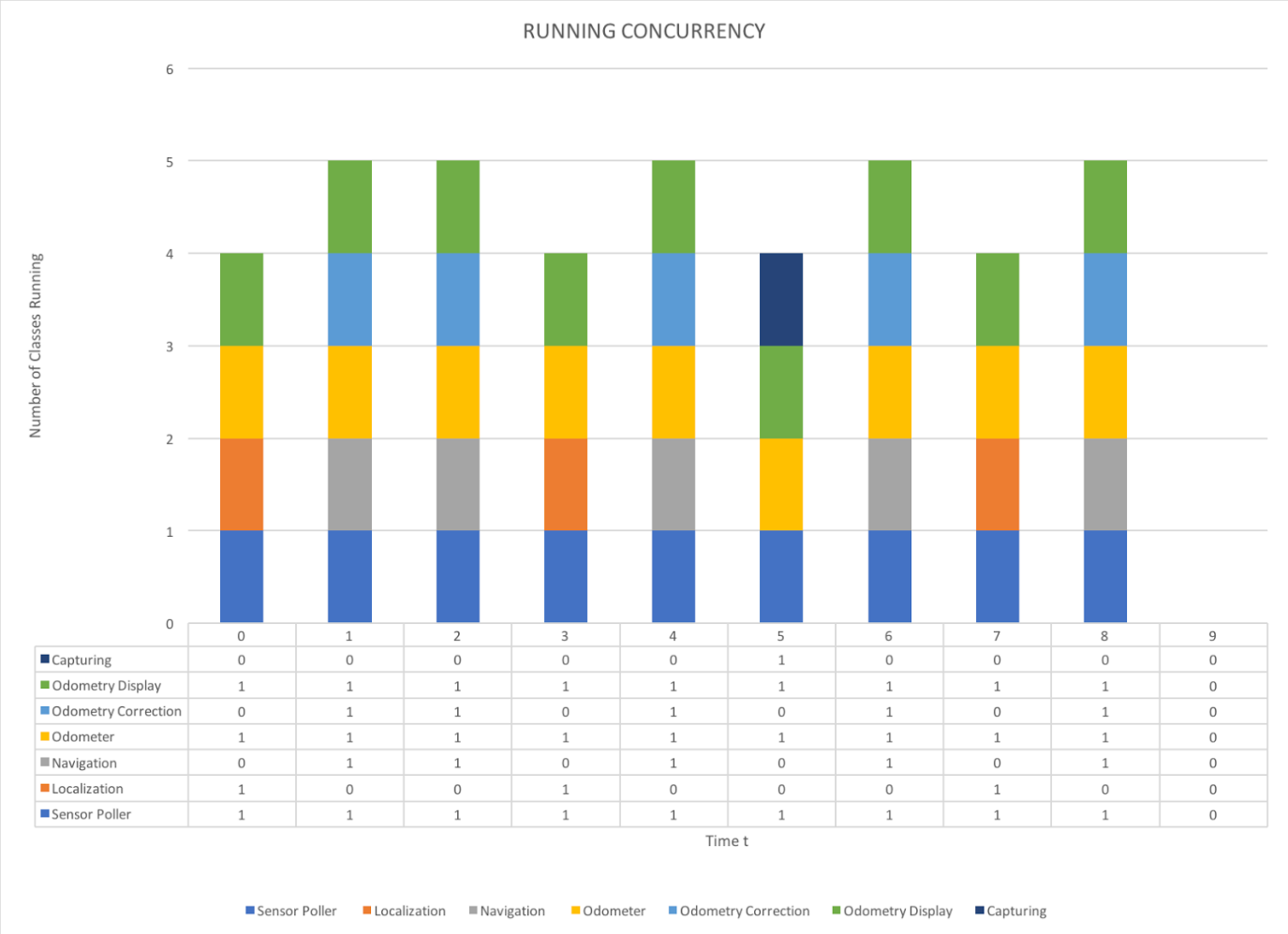
There are three basic layers in the class structure in terms of hierarchy. The top layer is in charge of controlling the robot (Robot Control) which is a managing class that is responsible for calling the other threads and other classes in the system and the overall process. It is also the core of controller in our software architecture. The second layer is composed of the odometer and the sensor poller. The odometer layer is placed here because of the multiple classes that are dependent on it. The sensor poller class is where the instances for all the sensors are created. Due to the limited numbers of port, each light sensor or ultrasonic sensor serves multiple purposes in different classes. However, if the same sensor object is defined more than once in different places then this causes an error in the system. To prevent this a sensor poller class is introduced to create instances for the sensor objects and pass the sensor data as parameters to the classes that

Figure 1: Class Hierarchy Diagram

need them. The third layer includes the functional classes. They deliver the different functionalities that are required, and these classes are managed by robot control and sensor poller. They are the core of model in the software architecture.

3. Software Concurrency

In our design, the task assigned to each class was reduced as much as possible to improve the readability and to be easier to refactor. Thus multiple classes have to run concurrently to achieve the desired functionalities. The concurrency flow is shown below.



	0	1	2	3	4	5	6	7	8	9
Capturing	0	0	0	0	0	1	0	0	0	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 2: Concurrency Flow Diagram

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 1: Corresponding time frames for **Figure 2.**