McGILL UNIVERSITY

ECSE 211: Final Design Project

Software Design Document

Sadhvi Mehta

supervised by
Frank P. Ferrie

1 SUMMARY

This document is intended to identify and define the current software solution being implemented to solve the task at hand (refer to *Task* section for more details). It provides a brief description of the business domain, functional requirements, and nonfunctional requirements as demanded by the client. In addition, it provides an in depth view into the process of how the code was developed. This begins with a high level flow chart, followed by a domain model, which is then further explained by corresponding use cases. All knowledge pertinent to the software design is found within this document. Note, this document is subject to changes as the semester progresses.

CONTENTS

1 Summary 1

2 EDIT HISTORY 2
2.1 Document Version Number 2
2.2 Edit History

3 Project Description 2
3.1 Introduction 2
3.2 Task
3.3 Business Goals 2
3.4 Scope 2

4 Requirements 2
4.1 Functional requirements 2
4.2 Nonfunctional requirements 3

5 SYSTEM MODEL 3

6 CLASS DIAGRAM 3

7 USE CASE DIAGRAM 4

8 USE CASES 4

2. EDIT HISTORY

a. Document Version Number

i. 0.0.1: Version presented to Prof. Ferrie on the 2017/10/27

b. Edit History

i. Sadhvi Mehta - Creation of document.

3. Project Description

3.I. Introduction

The objective of this project is to implement a fully autonomous robot that plays an altered version of "Capture the Flag". This version of "Capture the Flag" consists of navigating through a grid by mounting a zipline and swimming through virtual water, in search of a specific colored flag. Once this flag is found, the robot simply needs to return to its home zone. During this journey, the robot will be competing with other robots for shortest time whilst possibly having to avoid colliding into them.

3.2. Task

Design an autonomous robot that can play a one-on-one version of the Capture the Flag while navigating through an obstacle course.

3.3. Business Goals

The successful outcome is that the robot successfully can mount and land from a zipline, navigate through the water, and capture the flag before the opposing team.

3.4. Scope

The potential of this project can be very complex and could possibly involve the implementation of very heavy algorithms. However, due to time constraints as well as the limitations of the EV3 brick (see *Constraints Document* for more information), a simple yet efficient solution and logic will be implemented.

4. Requirements

Note: For more detailed requirements, please refer to the Requirements Document.

4.1. Functional Requirements

A. The system shall be able to accept input points via wifi which indicate the pre-mount point, the mount point, and the corner the robot is to begin in.

B. Before beginning navigation, the system shall be able to localize itself with respect to its starting position using.
C. The system shall be able to navigate itself to the input points while avoiding any obstacles such as other robots, blocks, and the zipline statue.
D. The system shall be able to autonomously and successfully mount and dismount the zipline.
E. The system shall be able to correctly identify its desired block based on the block color, using the ultrasonic sensor.

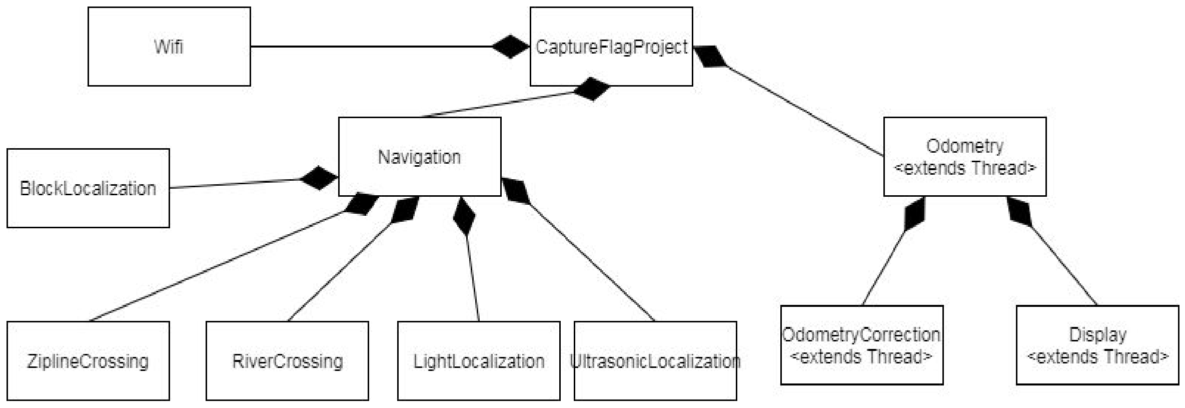
4.2. Nonfunctional Requirements

A. The system shall be able to localize itself within 30 seconds.
B. During light localization, the system shall be able to account for any discrepancies caused by ambient light.
C. The system shall be able to identify its block and return to the starting position within 5 minutes.
D. The system shall not be able to mount the zipline from either side of the statue.

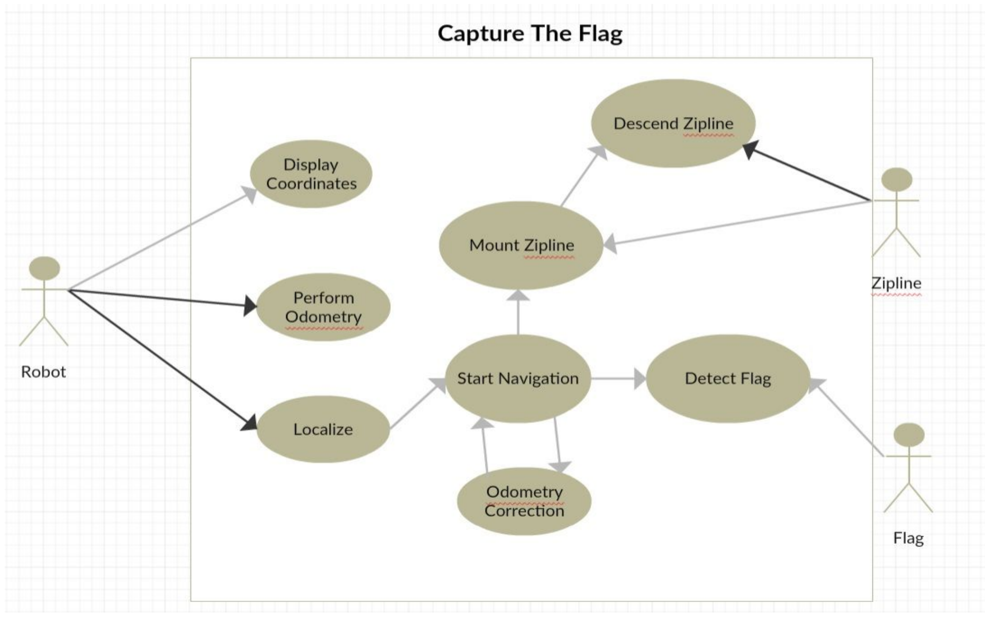
5. SYSTEM MODEL

For the flowchart that provides a high level view of the class to be implemented as well as how they communicate with one another, please refer to *section 3; System Model* in the System Requirements Document. The system model within the System Requirements Document is constantly updated and in sync with this document.

6. CLASS DIAGRAM



7. USE CASE DIAGRAM



8. USE CASE DESCRIPTIONS

A. Use Case: <u>Detect Flag</u>

Use Case Package	Capture the Flag Robot
ID	UC-CTF-DF
Use Case Goal	The robot detects the flag based on the color of the block, using its light sensor.
Actor(s)	Primary Actor: Robot Secondary Actor: Flag
Level	User-level goal
Precondition	The robot is turned on and has navigated to area where blocks are found.
Domain Entities	Robot, Arena, Flag

Main Success Scenario:

Step	Action	Notes
1	Robot localizes itself	**SeeUC-CTF-UL**
2	Robot navigates to zipline/path through water.	
3	Robot navigates to area containing blocks.	
4	Robot detects a block using ultrasonic sensor.	
5	Robot detects block's intensity based on light sensor readings and is a match with input color of desired block.	
6	Robot beeps three times.	
7	Use case ends successfully	

Alternative Flow:

Step	Action	Notes
5a.1	Robot uses light sensor to identify color of block.	
5a.2	Light sensor readings don't match with required intensity of desired block.	
5a.3	Robot reverses away from block by 10cm.	
5a.4	Return to *Main Success Scenario- Step 4.*	

B. Use Case: Ultrasonic Localization

Use Case Package	Capture the Flag Robot
ID	UC-CTF-UL
Use Case Goal	The robot is correctly oriented.
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is placed along 45 degree line of starting tile any orientation.
Domain Entities	Robot, Arena

Main Success Scenario:

Step	**Action**	**Notes**
1	Based on initial orientation, robot reorients itself to face away from wall.	
2	Robot rotates CW until wall is detected.	
3	Robot records first wall ultrasonic reading.	
4	Robot rotates CCW until wall is detected.	
5	Robot records second wall ultrasonic reading.	
6	New delta theta is calculated and added to current robot heading.	
7	Odometer is updated.	
8	Use case terminates successfully	

C. Use Case: <u>Light Localization</u>

Use Case Package	Capture the Flag Robot
ID	UC-CTF-LL
Use Case Goal	The robot is correctly oriented and positional offsets are found.
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on and placed along 45 degrees line of starting tile.
Domain Entities	Robot, Arena

Main Success Scenario:

Step	Action	Notes
1	Robot turns 45 degrees to face away from wall and towards line intersection.	
2	Robot travels towards line intersection till light sensor detects lines.	
3	Robot reverses by a tested distance in order to later be able to scan all lines.	
4	Robot begins rotating 360 degrees clockwise.	
5	Robot detects and records 4 lines based on an algorithm involving differential light intensities.	
6	New delta theta, x-offset, and y-offset are calculated.	
7	Odometer is updated.	
8	Use case terminates successfully	

Alternative Flow:

Step	Action	Notes
5a.1	Robot is unable to detect all four lines.	
5a.2	Robot is set to rotate 360 degrees clockwise again.	
5a.3	Return to *Main Success Scenario- Step 5.*	

E. Use Case: Perform Odometry

Use Case Package	Capture the Flag Robot
ID	UC-CTF-PO
Use Case Goal	The robot records its position and orientation for the display.
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on and has default initial position and orientation.
Domain Entities	Robot, Odometry

Main Success Scenario:

Step	**Action**	**Notes**
1	Robot clears current motors tacho count.	
2	Robot checks to see how much motors have rotated.	
3	Based on motor rotations, odometer calculates robot's change in theta, x, and y.	
4	Use case ends successfully	

F. Use Case: Display Coordinates

Use Case Package	Capture the Flag Robot
ID	UC-CTF-DI
Use Case Goal	The robot displays readings on LCD.
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on.
Domain Entities	Robot, LCD, Capture the Flag

Main Success Scenario:

Step	Action	Notes
1	Robot starts LCD display.	
2	Robot gets readings from odometer.	
3	Robot updates display according to odometer readings.	
4	Use case ends successfully	

G. Use Case: <u>Start Navigation</u>

Use Case Package	Capture the Flag Robot
ID	UC-CTF-SN
Use Case Goal	The robot correctly navigates to input points using minimal angle and Euclidean distance.
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on and has successfully localized itself.
Domain Entities	Robot

Main Success Scenario:

Step	Action	Notes
1	Euclidean distance to input point is calculated.	
2	Minimal angle is calculated using current theta from odometer.	
3	Robot turns to minimal angle and travels the calculated Euclidean distance.	
4	Use case ends successfully	

Alternative Flow:

Step	**Action**	**Notes**
5a.1	Flag is raised that robot is falling off desired trajectory based on odometry correction.	
5a.2	Current navigation is stopped and navigation is recalled.	
5a.3	Return to *Main Success Scenario- Step 1.*	

H. Use Case: Odometry Correction

Use Case Package	Capture the Flag Robot
ID	UC-CTF-OC
Use Case Goal	The robot correctly updates odometer every time it detects a line.
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on and is navigating.
Domain Entities	Robot, Odometer

Main Success Scenario:

Step	**Action**	**Notes**
1	Light sensor detects a line according to line detection algorithm.	
2	New x-position/y-position is calculated based on robot's previous position.	
3	Odometer is updated with new x-position/y-position.	
4	Use case ends successfully	

Alternative Flow:

Step	**Action**	**Notes**
5a.1	Flag is raised that robot is falling off desired trajectory based on odometry correction.	
5a.2	Current navigation is stopped and navigation is recalled.	
5a.3	Return to *Main Success Scenario- Step 1.*	