**SOFTWARE DESIGN DOCUMENT**

**Project:** Obstacle track-racer – A competitive robot design project

**Task:** Design and implement a working model out of EV3 Lego sets that would autonomously navigate to a racetrack on an island shown in the world map and complete as many laps as possible within the 5-minute time limit, eventually returning to its starting point.

**Document Version:** 2.03

**Date:** 13/03/2021

**Author:** Lide Cui, Dominic Chan

**Edit History:**

|  |  |  |  |
| --- | --- | --- | --- |
| Version 1.01 | 25/02/2021 | Lide Cui | Work on the initial document. |
| Version 1.02 | 03/03/2021 | Dominic Chan | Formatting and Template of document |
| Version 2.01 | 05/03/2021 | Lide Cui | Detailed design of avoiding obstacles |
| Version 2.02 | 12/03/2021 | Lide Cui | Add state machine |
| Version 2.03 | 13/03/2021 | Lide Cui | finish obstacle avoidance, navigation,  review abstract |

**1.0 TABLE OF CONTENTS**

[**2.0 ABSTRACT** 1](#_heading=h.30j0zll)

[**3.0 SOFTWARE FLOWCHART AND WORKFLOW** 2](#_heading=h.1fob9te)

[**4.0 STATE MACHINE** 2](#_heading=h.3znysh7)

[**5.0 CLASS HIERARCHY** 4](#_heading=h.2et92p0)

[**6.0 DESIGN OF DIFFERENT FUNCTIONS** 4](#_heading=h.tyjcwt)

[**6.1 LOCALIZATION** 4](#_heading=h.3dy6vkm)

[**6.2 NAVIGATON** 5](#_heading=h.1t3h5sf)

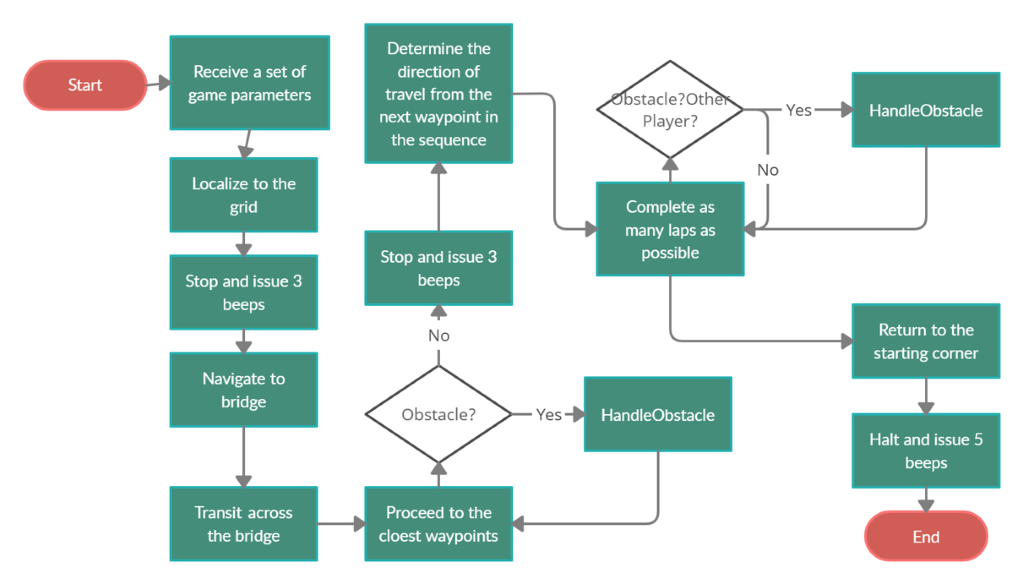
[**6.3 OBSTACLE AVOIDANCE** 5](#_heading=h.4d34og8)

# **2.0 ABSTRACT**

In this software document, the **state machine** and game play **flowchart** describe the sequential behavior of the robot. All implemented methods are described respectively with a brief description. The algorithm of all main methods is illustrated in a flowchart form. A **class diagram** is also included to showcase the structure of the software for this project. Lastly, this document contains a **weekly log** to demonstrate the evolution of the robot’s software as well as the feedback that was received by the testing team.

Also, some detailed descriptions of functions, such as navigation, localization, and obstacle avoidance, are provided.

# **3.0 SOFTWARE FLOWCHART AND WORKFLOW**



*Figure 1 Flow Chart*

# **4.0 STATE MACHINE**

Graphical user interface, diagram, application

Description automatically generated

*Figure 2 State Machine*

*Transition:*

*Event 0: Parameter Received*

*Event 1: Position Determined*

*Event 2: Find Tunnel Location*

*Event 3: Arrive at Competition Yard*

*Event 4: Find Bridge Location*

*Event 5: Arrive at the Other Side of Bridge*

*Event 6: Iterate All the Points*

*Event 7: Detect Obstacles*

*Event 8: Arrive at the Safe Point*

*Event 9: Time Approaches 5 Minutes*

# **5.0 CLASS HIERARCHY**

A picture containing timeline

Description automatically generated

Only public methods are shown in the class diagram, for they can be used by other classes. Private methods are subject to change.

# **6.0 DESIGN OF DIFFERENT FUNCTIONS**

## **6.1 LOCALIZATION**

（Refer to Lab5 report)

## **6.2 NAVIGATON**

Title: General Navigation

Actor: Robot, Obstacles

Intention: The intention for the robot is to cross the bridge, run laps and return home within 5 minutes.

Precondition: The robot has received all parameters and finished localization.

Main Scenario:

1. The robot turns to the bridge and moves to the edge of the bridge.

2. The robot moves across the bridge.

3. The robot moves the closest waypoint.

4. The robot locates itself, and if there is time left, the robot moves the next waypoint (back to step 3).

5. The robot returns to the edge of the bridge.

6. The robot across the bridge.

7. The robot moves to the starting point.

Alternative\Exception #1:

3a. If there is an obstacle, then run the obstacle avoidance process, continue at step 4.

Alternative\Exception #2:

Notes:

The edge of the bridge is the midpoint between TNR\_UR\_x and TNR\_LL\_x.

The distance of the bridge is TNR\_\_UR\_y and TNR\_LL\_y.

The estimated return time is at 4’10’’.

## **6.3 OBSTACLE AVOIDANCE**

Title: Avoid an Obstacle

Actor: Robot, Obstacles

Intention: The intention of the robot is to safely avoid an obstacle by passing by it and move to the destination point.

Precondition: The robot has accurately located a point, turned by a minimal angle towards destination point, and calculated the distance to the destination.

Main Scenario:

1. The ultrasonic sensor of the robot measures distance to the nearest ‘obstacle’.

2. The robot divides the path into 3 parts.

3. The robot travels the first part (the length of first part = distance to obstacle – react distance).

4. The robot turns 90 degrees and detects if there is enough space to turn.

5. The robot moves the second part (the length of second part = 3\*obstacle size + 2\* react distance).

6. The robot turns -90 degrees.

7. The robot moves third part to the destination point (the length of third part = distance to destination – length of first part – obstacle size – 2 \* react size).

Alternative\Exception #1:

1a. If the distance to obstacle is greater or equal to distance to destination, then the main scenario is jumped, the robot directly travels to the point.

Alternative\Exception #2:

1b. If the path of current position and destination position pass through an overpass, then the main scenario is jumped, the robot directly travels to the point.

Alternative\Exception #3:

4a. If the robot detects a wall so that it has no sufficient space to turn

4a.1. The robot turns 180 degrees.

4a.2. The robot does step 5.

4a.3. The robot turns 90 degrees and continues from step 7.

Postcondition: The robot approaches the destination point and starts the localization process.

Notes:

React distance is estimated to be 0.1 m.