McGill University

Capture the Flag Robot – Software Model Specification

Team 14: Amlekar R, Bouchard W, Bercovici D, Commodari A, Wright A

ECSE 211: Design Principles and Methods
Dr. Lowther and Dr. Ferrie

2015/11/02

Revision History

Date	Changes to Document	Version	Changes made by
2015/11/02	Creation of Document	1.0	Asher Wright

Table of Contents

List of Illustrations

Project Description

Introduction

The Capture the Flag Robot (CFR) is an autonomous robot that can play a modified version of capture the flag. The goal of the CFR is to, after being placed on a game-grid, navigate to and locate a colored flag, pick up the flag, and take it to a home zone. It must do this before the enemy robot completes the same task.

Context

The Capture the Flag Robot is designed with the versatility to adapt to different layouts of the grid (with blocks in different locations). It is not pre-programmed to avoid blocks, but rather does it during operation.

Business Goals

The successful outcome of this project will be a fully autonomous robot that can capture the enemy flag before its own flag is captured.

Scope

The Capture the Flag Robot has the potential to be complex but given the project timeframe and for the sake of simplicity and elegance, certain features have been omitted. These include:

Actors

The Robot is the primary actor. There are no secondary actors since the robot must be completely autonomous.

Requirements

Please see the Requirements document for detailed requirements. This document will briefly outline the functional and nonfunctional software requirements.

Non-functional requirements:

The software system shall be autonomous, only initially connecting to the Wi-Fi to receive a message containing 13 integers. These integers indicate the starting corner number, the coordinates of the Home Zone for both the robot and its opponent, the location of the destination for the captured flag, and the colour of the two flags.

The software system shall attempt to be as efficient as possible to minimize the time required to complete the task.

Functional requirements:

The software system shall localize to the grid within the time constraint of 30 seconds using the USLocalizor and LightLocalizor classes.

The system shall navigate around an obstacle course to the opponent's home zone using the Navigator, Odometer, and Wall Avoider.

The system shall us the BlockDetector to search for the flag and determine whether it is a flag by shape and colour.

Architecture

The domain model was created with the components that were used in the class labs. The five labs were:

1. Wall Follower
2. Odometer
3. Navigator
4. Localizer
5. Block Detector

Many of the classes from these five labs are integral to the success of the robot. There are certain classes that will be used in threads and will be running constantly. These are:

1. Odometer
2. Odometer Corrector
3. Driver
4. LCD Displayer
5. Ultrasonic Poller
6. Light Sensor Poller

It is necessary that these classes are always running. The following classes are classes that will be created/invoked at certain times:

1. Wall Avoider
2. Block Detector
3. Navigator

The class that is powering and creating most of these classes is the "Controller" class.

In creating this domain model, a few architectural styles were considered.

The first architectural style that was considered was Model-View-Controller (MVC). The advantages of this were that it is easy to change components individually, and also easy to add elements. However, this robot is autonomous, and does not really have a "View".

The next architectural style that was considered was n-Layered Architecture. In particular, 2-Layered Architecture. This allows for one layer to be the Robot's Logic, and for the other to be the Robot's controller. This is Controller-Logic layered architecture.

The upper layer is the Robot's logic (brain), and contains classes such as the Controller, Driver, etc.
The lower layer is the Robot's actions (muscles), and contains the Wall-avoider, navigator, odometer, localizers, etc.

This can be seen in the following domain model.

Domain Model

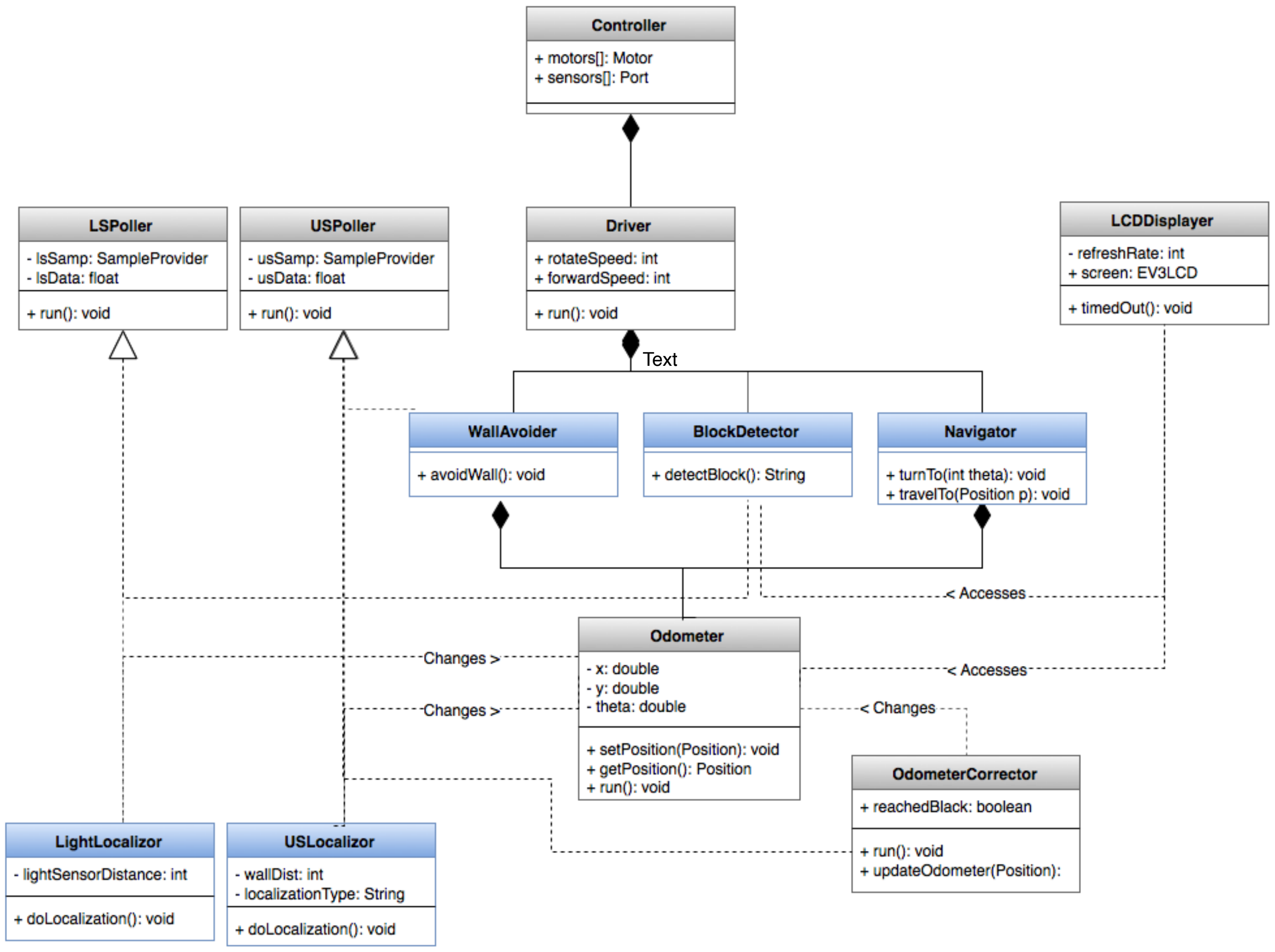

Figure 1 - Domain Model

Grey classes are classes that are always running (on separate threads or TimerListeners). Blue classes are classes that are instantiated/used when needed.

Use Cases

Overview

The following Use Case Diagram shows the various use cases of the robot. Note that, since this is an autonomous robot, it is the only actor. Additionally, the structure of the use case diagram is very exact (since the robot will localize → Start → Drive → Get Block → Return).

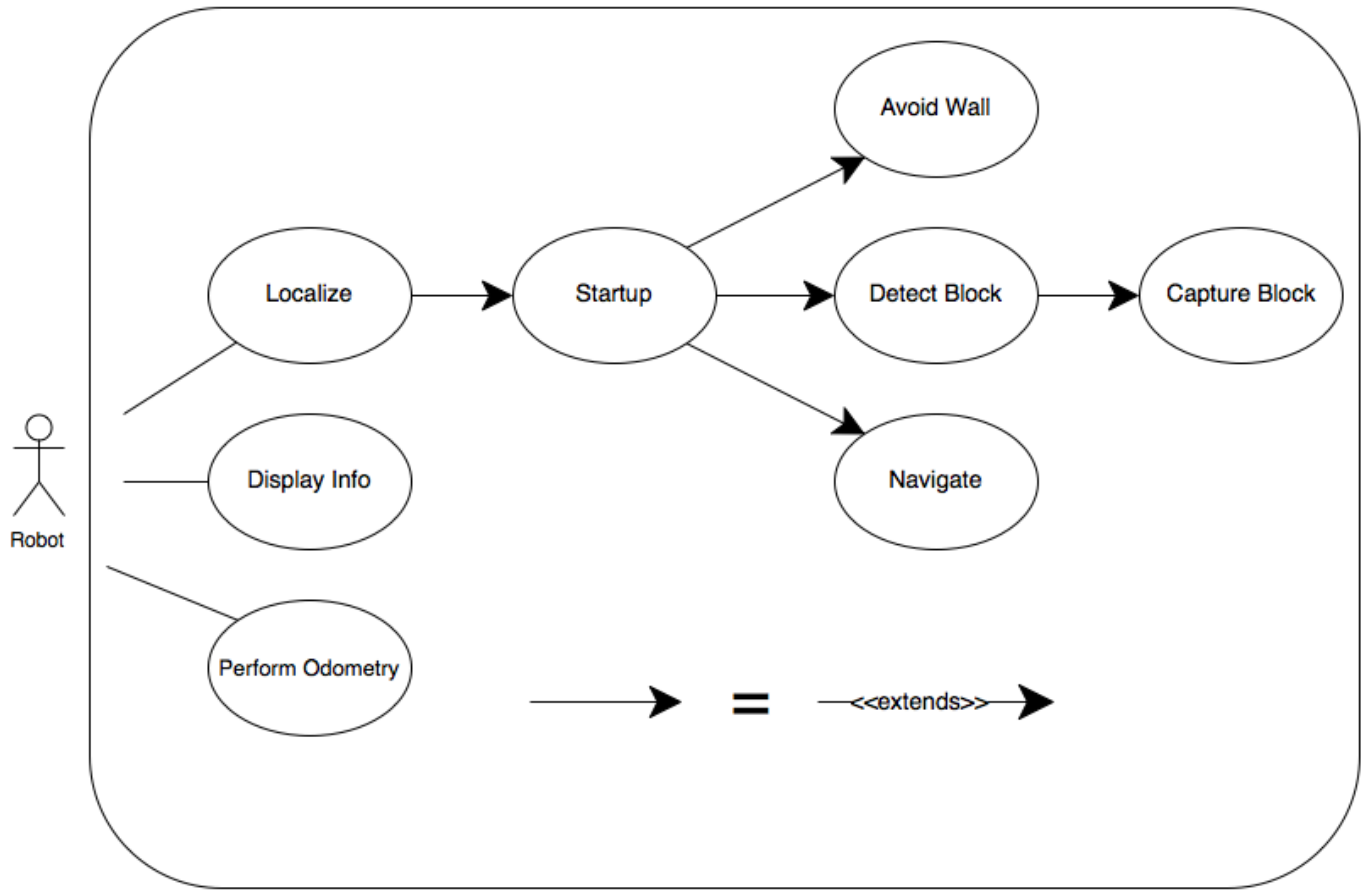

Figure 2 - Use Case Diagram

Brief Description:

Localize: The robot finds its orientation given that it starts randomly oriented in a corner.
Startup: The robot starts to find the enemy's block. It uses the navigate, detect block, and avoid wall use cases to find the block and bring it home.
Avoid Wall: The robot avoids a wall that is in the way of where it is travelling.
Detect Block: The robot searches for and detects a block as it looks for the flag.
Capture Block: The robot uses its arms to grab the block, and then travels to the end location.
Navigate: The robot moves from one point to another in the field.
Display Info: The robot displays its position and sensor readings on the screen.
Perform Odometry: The robot records its position as the motors rotate.

Significant Use Cases

Use Case: Capture Block <<extends>> Detect Block

Successful Outcomes: The robot captures the enemy's block and returns it

Use Case Package	Capture the Flag Robot
ID	UC-CTF-CB
Use Case Goal	The robot captures the enemy robot's block
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on, placed in a corner of the arena
Domain Entities	Robot, Arena

Main Success Scenario:

Step	**Action**	**Notes**
1	Robot localizes itself	**See UC-CTF-L**
2	Robot navigates to the enemy's home zone.	
3	Robot searches for the enemy's block in the home zone	
4	Robot finds enemy block	
5	Robot moves arms to secure block	
6	Robot, with block, returns to home zone	
7	Use case ends successfully	

Alternative flows:

Step	**Action**	**Notes**
2a.1	Robot finds obstacle in its path	
2a.2	Robot uses wall-avoider to travel around obstacle	**See UC-CTF-W**
2a.3	Robot goes back on track	
2a.4	Return to MSS Step # 2	

Step	**Action**	**Notes**
4a.1	Robot finds other block (not goal)	
4a.2	Robot moves back away from block	
4a.3	Robot continues to search for other blocks	
4a.4	Return to MSS Step e	

Use Case: Avoid Wall <<extends>> Start up

Successful Outcomes: The robot avoids the wall blocking its path

Use Case Package	Capture the Flag Robot
ID	UC-CTF-W
Use Case Goal	The robot moves around the wall
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on, navigating to a place blocked by a wall
Domain Entities	Robot, wall, navigation

Main Success Scenario:

Step	**Action**	**Notes**
1	Robot measures distance to wall	
2	Robot measures dimensions of wall	
3	Robot uses measurements to move to the left or right of wall	
4	Robot travels around wall	
5	Robot restarts navigation to initial endpoint	
6	Use case ends successfully	

Alternative flows:

Step	**Action**	**Notes**
4a.1	Robot finds another wall in the way	
4a.2	Robot uses Avoid Wall to avoid this wall	**See UC-CTF-W**
4a.3	Return to MSS Step # 1	

Use Case: Ultrasonic Localization

Successful Outcomes: The robot orients itself correctly in the arena.

Use Case Package	Capture the Flag Robot
ID	UC-CTF-L
Use Case Goal	The robot is correctly oriented
Actor(s)	Primary Actor: Robot
Level	User-level goal
Precondition	The robot is on, placed in a corner of the arena
Domain Entities	Robot, Arena

Main Success Scenario:

Step	Action	Notes
1	Robot starts recording ultrasonic sensor distances	
2	Robot rotates clockwise until it reads a wall	
3	Robot keeps rotating until it no longer reads a wall	
4	Robot stores this angle as X	
5	Robot rotates counter-clockwise until it hits a wall	
6	Robot rotates counter-clockwise until it no longer reads a wall	
7	Robot stores this angle as Y	
8	Robot uses X and Y to figure out how it is oriented	
9	Robot turns to 0 degrees	
10	Use case terminates successfully	

Use Case: Perform Odometry

Successful Outcomes: The robot records its position and orientation over time.

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

Main Success Scenario:

Step	**Action**	**Notes**
1	Robot gets access to left and right motors	
2	Robot checks to see how much motors have turned	
3	Robot alters position and orientation based on amount the motors have turned.	
4	Use case ends successfully	

Use Case: Display Info

Successful Outcomes: The robot displays its position and sensor readings on the LCD screen.

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 gets readings from Ultrasonic Sensor	
4	Robot gets readings from Light Sensor	
5	Robot displays readings on LCD Display	
6	Use case ends successfully	

Robot Properties (constants)

The following properties are physical properties of the robot that are used in the software powering the robot. They are necessary to ensure that the robot travels the correct amount, does not hit objects unintentionally, and has an accurate reading for how much it has travelled.

Property Name	Property Description	Property Value
Track	The distance between the centers of the two wheels	15
Left Wheel Radius	The radius of the left wheel	2.1cm
Right Wheel Radius	The radius of the right wheel	2.1cm
Ultrasonic Sensor Offset	The distance from the center of rotation to the center of the ultrasonic sensor	3cm
Light Sensor Offset	The distance from the center of rotation to the center of the light sensor	5cm

Note that these software values will change as the hardware design changes.

Glossary

Keyword	Description
Arena	The wooden area in which the robot moves
Block	Any object in on the arena that is not a robot
Flag	The block that the robot (or enemy robot) wants to capture. It is a block with a specific color/size.
Home Zone	The zone where the flag is located
Robot	The primary actor: the autonomous robot that has been created to complete the task
Track	The distance between the two wheels of the robot (see Robot Properties)

References

This format of this document was adapted from Asher Wright's ECSE 321 Software Requirements Specification (with his permission).

Some aspects of this document were also adapted from:
http://brazos.cs.tcu.edu/0910/deliverables/SRS_v1.0.pdf