Software Requirements Specification

for

<Robot Radar>

Version 1.0 approved

Prepared by <Anubhav Sigdel>

<Umang Patel>

<Tyler West>

<Sven Kappeler>

<SUNY OSWEGO CSC 380 810 Team G>

<December 3rd 2021>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Umang	12/1/2021	Final update for submission	2
Tyler	9/14/2021 12/2/2021	Scenarios/UI Display and small additions to Sections 3-6 Final edits	1.3 2.1
Sven	9/14/2021	Added a new Use Case Diagram and Sequence Diagram	1.4
Anubhav	9/21/2021	Added UML Class Diagram	1.5

1. Introduction

1.1 Purpose

To develop a Robot Radar program for GoPiGo robots that will allow the user of the robot to move the robot around whereby it understands its environment and avoid collisions. The application uses ultrasonic translateralization beacons to detect objects in the path and gain an understanding of its environment. The object would then be displayed on a user interface as well as potential objects with colored dots on the radar that would be in its path This document describes its requirements and documents in its usage.

1.2 Document Conventions

Non-Italic text - Our solution for the section
N/A - Section deemed irrelevant
TBD - Not enough information yet
~~Strikethrough~~ - Candidate for deletion
Bold- Important term or section title.
IP - In progress / needs review

1.3 Intended Audience and Reading Suggestions

- Team members - read the document for implementation
- Stakeholder - read the document to understand the project as well as confirm its expectations/outcomes
- Professor - read the document to determine our deserved grade
- User- read the document to understand how the project works

1.4 Product Scope

The product is intended to serve as a guiding application for the users of GoPiGo robots on the obstacles in their path. The purpose of this project is to allow for cars, drones, and other utilities to be aware of their surroundings, allowing for them to reduce possible destruction/damage of itself and other objects and increase their efficiency in mobility. Utilizing this software designed for the robot can be taken even a step further for controlling multiple units simultaneously in any environment.

It is **not** an autonomous steering application since the GOPiGo robot will have a user interface for controlling them. We are just aiding in its ability through sensors to not crash and to track objects around it and a feature to display a radar to show past objects will be implemented.

1.5 References

- GitHub Code for this **RobotRadar**
https://github.com/asigdel29/RobotRadar
-Radar emulator with Arduino+Python
https://makersportal.com/blog/2020/3/26/arduino-raspberry-pi-radar
- Python for GoPiGo
https://www.dexterindustries.com/GoPiGo/get-started-with-the-gopigo3-raspberry-pi-robot/3-program-your-raspberry-pi-robot/python-programming-language/
- Measuring Distance with Sensors
https://gopigo3.readthedocs.io/en/master/tutorials-basic/distance_sensor.html
- GoPiGo **Circuit Board** Picture
https://www.dexterindustries.com/GoPiGo/learning/hardware-port-description/

2. Overall Description

2.1 Product Perspective

The GoPiGo robot has simple functionality to move around without sensing the surroundings and often leads to crashing onto walls and obstacles on its way. This project will be adding some functionality for the robot to stop as soon as the obstacle is scanned close to the robot and the program will have a GUI for the user to interact with the robot with moving commands and would be able to see scanned object on the radar which will be collected using Ultrasonic sensor on the robot`s servo.

2.2 Product Functions

The robot and/or interface will

- Use its ultrasonic sensors to **detect** obstacles
- The robot`s sensor will notify the robot to stop when the object is too close to the robot.
- The robot will have basic movement ability to move right , left forward and backward and its servo will have the ability to rotate on a 180 degree angle to detect around.
- Measure the time interval between sending a small invisible laser pulse and receiving the light back
    - Based on the time it takes, calculate the distance to an object
- Display obstacles on the user interface.
- After the scanned data is collected the robot update the radar according to the robot movement
- Use color coding to determine proximity to other objects in its immediate area-using the Hue values.

2.3 User Classes and Characteristics

- User of the gopigo robots
- Any student at any level could use this system to avoid crashing their Gopigo

Primary User - would require a little different extend version to this programs(different hardware)

- Cars Drivers
    - when parking
    - crash detection- immediate pedestrian/animal crossing road

Secondary User - would require high level of security and fully automatic extension of the program (different reliable hardware)

- Homeowner -
    - robot vacuum cleaner -fully automatic
- Military -
    - mine locator robot
- Astronomers
    - Outer planet rovers

2.4 Operating Environment

- GoPiGo 3 Raspberry Pi robot
- Python API`s
    - pygame,
    - gopigo
    - math
    - time
    - color
    - sys
- Raspbian OS
- WIFI
- Ultrasonic sensor
- Battery/Power source

2.5 Design and Implementation Constraints

- Slow Wifi Speed
- Radar senses objects within 15mm to 4500mm range
- Sensors only in the front of robot ; doesn't account for parallel or behind the robot
- sensor servo only move 180 degree max (without camera)
- Operating Voltage: minimum 9 V, low voltage affects speed of robot and accuracy of distance sensor
- External Interfaces:

    - I2C ports: 2 Grove ports connected to the Raspberry Pi I2C bus through a level conversion chip.
    - Serial ports: 1 Grove port connected to serial pins on the Raspberry Pi through a level conversion chip.
    - Analog digital ports: 2 Grove ports connected to the GoPiGo3 microcontroller.
- Encoders: 2 Magnetic encoders with 6 pulse counts per rotation (with 120:1 gear reduction for a total of 720 pulses per wheel rotation).
- Wheel Diameter: 66.5mm
- safe environment(Don't put the robot on a road/highway)

2.6 User Documentation

- Deployment guide
- ReadMe would be available on the Github https://github.com/asigdel29/RobotRadar/blob/main/README.md
- A comprehensive manual of how to use the robot would be made
- A glossary featuring all project-relevant terms and definitions. at the Appendix A: Glossary on page 10

2.7 Assumptions and Dependencies

- Network failure in WIFI
- Dexter - GoPiGo Sensor Failure/ damaged
- Temporary failure in name resolution (Raspbian software corruption)
- Python
- Robot functions as expected
- Wheels broken/ out the axel
- Any physical component of the robot broken

3. External Interface Requirements

3.1 User Interfaces

A user of the application should see the live Radar from the robot when they open the application connected to the robot using WiFi. The screen layout will be mostly covered with green black radar with a green line going from right to left which shows the servo movement on the robot and would print out colored dots on the radar depending on the object distance as green equals object far, yellow equal object is at safe distance, and red equal object is close. It's implemented as a traffic light. These dots on the radar move according to the robot movement and sensor data collected by its ultrasonic sensor. A list of commands is displayed on the left top corner on the UI which shows Angle at which the server is at, movement command to help the User control the robot from the keyboard inputs. Forward up key, backward down key, left turn left key, right turn right key, to stop equals 0 and exit equals 1.

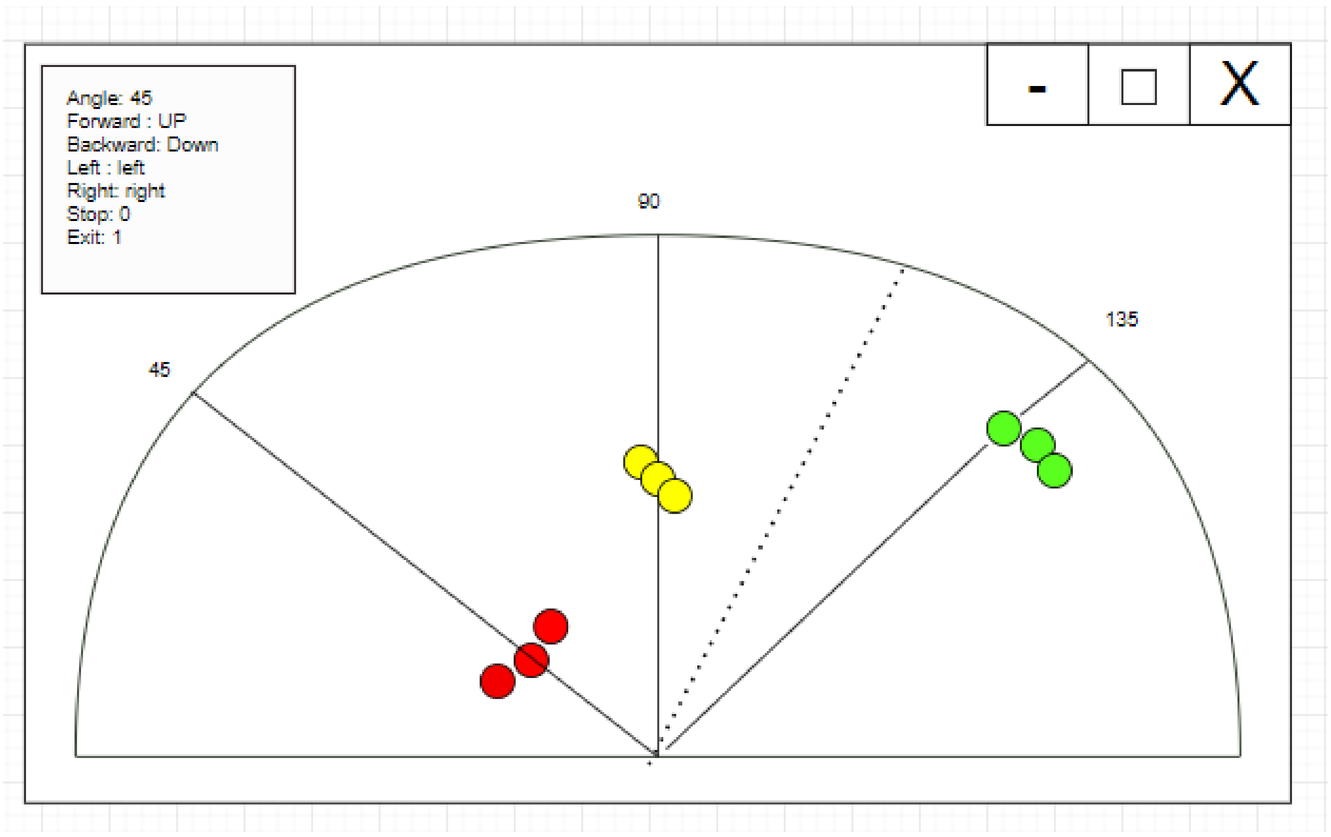


Figure: 1 GUI

3.2 Hardware Interfaces

Any device that can connect to the Colosseum Router and connect to the robot through the router is supported (via kitt.local or Snicker.local), the most common devices to interface with the robot will be Windows and macOS computers. The computer and the robot must be within range of the router at all times in order for communication to occur. The computer will connect to the web application host from the GoPiGo.

The radar sensor is plugged into the GoPiGo circuit board that interfaces with the Raspberry Pi; the circuit board connects many of the important parts for the robot, including the radar, video camera, battery and wheels (Circuit Board pictured below)

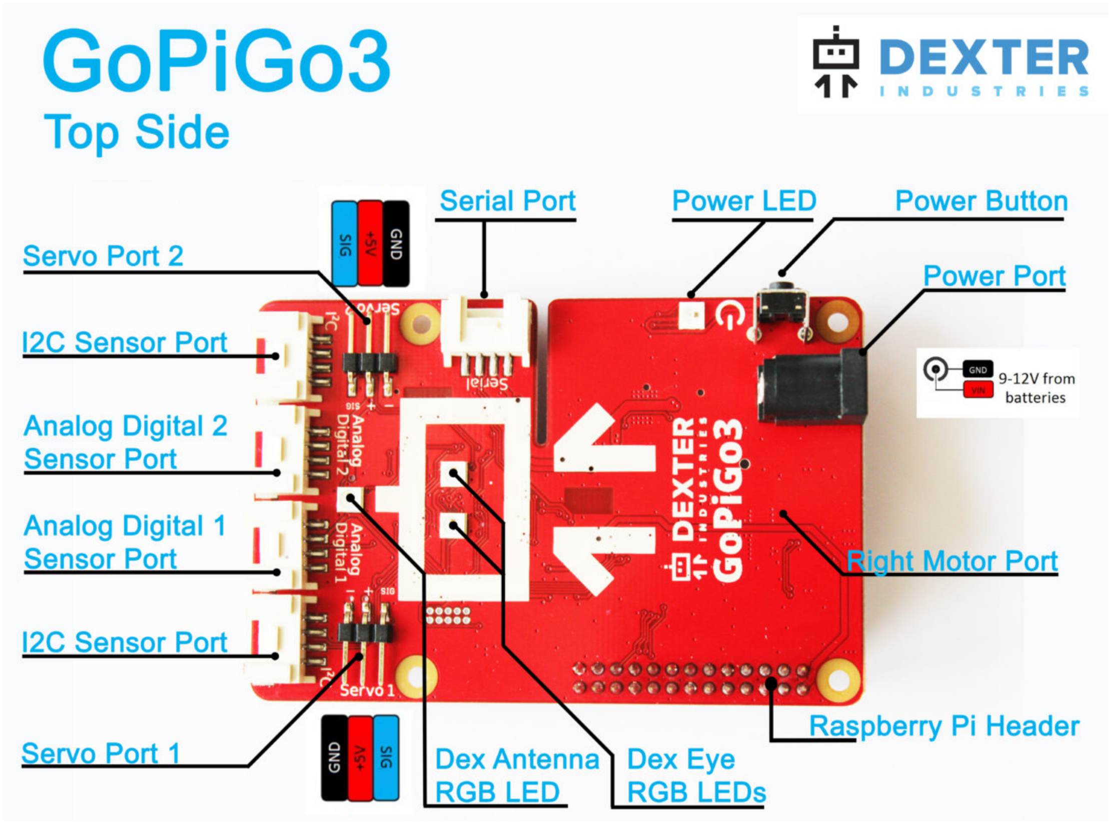


Figure: 2 GoPiGo Circuit Board Picture

The software will run **locally** on the GoPiGo and will be saved on the onboard storage, via the 16 GB SD card plugged into the Raspberry Pi. The keyboard of the device visiting the web application will be used to drive the robot and control the radar's rotation.

3.3 Software Interfaces

The system requires Gopigo and computer to connect to the gopigo wifi then go to any browser and write in the URL input the gopigo name or ip address in this project the gopigo name was Kitt which got changed to snickers later. this will give you access to the raspberry pi. On that you can run the program and see how it works but also make sure necessary external softwares, libraries are updated/downloaded on the gopigo list of these is given in section 2.4.

3.4 Communications Interfaces

N/A

4. System Features

4.1 Robot movement

4.1.1 Robot Movement from the wheels has features to move forward,backward,right,left and Robot servo has features to rotate left to right. These features are required for the robot to move around and for the sensor to detect around. The robot has a limitation to this feature to not move too close to the object. the robot will stop at about 200mm distance from the object around it. and the servo has the limitation to rotate about 180 degrees at the front of the robot.

4.1.2 Stimulus/Response Sequences

User will have an option through the keyboard to input command to move the robot

- hold the up key to move the robot forward with the function fwd()
- hold the down key to move the robot backward with the function bwd()
- hold the right key to move the robot right with the function right()
- hold the left key to move the robot left with the function left()
- Press the 0 key to move the robot servo to scan from left to right with the function target()
- Press the 1 key to exit the program.

4.1.3 Functional Requirements

When a user presses both movement keys at the same time, the robot will move in a curve or stop depending on the keys pressed by the user. the robot stops depending on the sensor when an object is too close to the robot. and the servo will rotate for a full swipe from left to right to detect objects. This feature is medium priority because the radar is supported to help the movement happen in a smooth sequence and not that the robot crashes.

4.2 Display object on radar

4.2.1 This feature is based on the robot`s Ultrasonic distance sensor in the robot`s servo. It is an important feature because the robot will not be able to sense the environment around. The sensened objects arounds anre later displayed in the radar with a color combination change when the as the object is close the dot will be red and the the object is far then the the dot will be yellow and if the object is very far the the dot will appear in green color. Tha radar updates automatically according to the user's command to move the robot around.

4.2.2 Stimulus/Response Sequences

Users will have an option to do a full swipe from left to right for about 180 degrees then the scanned data will be converted to a display radar in green and black with different color dots red, yellow, green.

- hold the 0 key to move the robot servo to scan from left to right with the function target()

4.2.3 Functional Requirements

When the user moves the robot around the sensor with default move the server at the front face straight and will not rotate and scan around but the user will be given an option to scan when the robot stops by pressing the 0 number key. then it will update the object on scan on the side of the robots environment. This feature is high priority because this is the main reason for the product to detect the object around and display it to the user using the display/screen.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Robot can not parallely run both move it wheel as users directed and servo movement to scan the objects around so when the robot scans around it is required that the users does not press anymore key until the scan is complete to move the robot.

5.2 Safety Requirements

Make sure your SD card is not corrupted to make the program successful work. Another safety requirement is to make sure the gopigo parts are perfectly connected and not damaged before running this program to avoid more potential damage to the robot for example make sure the wheels are perfectly fitted in the axle of the robot and also make sure if you want to exit the program you have to click number key 1 and not direct stop it in any other way and do not off the robot in the middle on the program running. Also make sure the robot is charged before running the program.

5.3 Security Requirements

The robot has to be accessed via the colosseum network and logging into the VNC with the information provided on the robot

5.4 Software Quality Attributes

The scope beyond the project can have utilities with other devices like cars which would make the software itself very versatile. The robot radar itself is quite operable and reliable. It could potentially be able to carry items for someone from one place to another. It is also quite intuitive in operating that just about anyone could utilize it effectively. It could be a simple toy for a child or a weapon in other hands.

5.5 Business Rules

The user of the program has access to all features. The program does not have any subsystem feature because the robot detects and shows it to the user.

Appendix A: Final Analysis Models

These are final diagrams and are also attached differently in a pdf to view them better as the diagrams get bigger and more detailed.

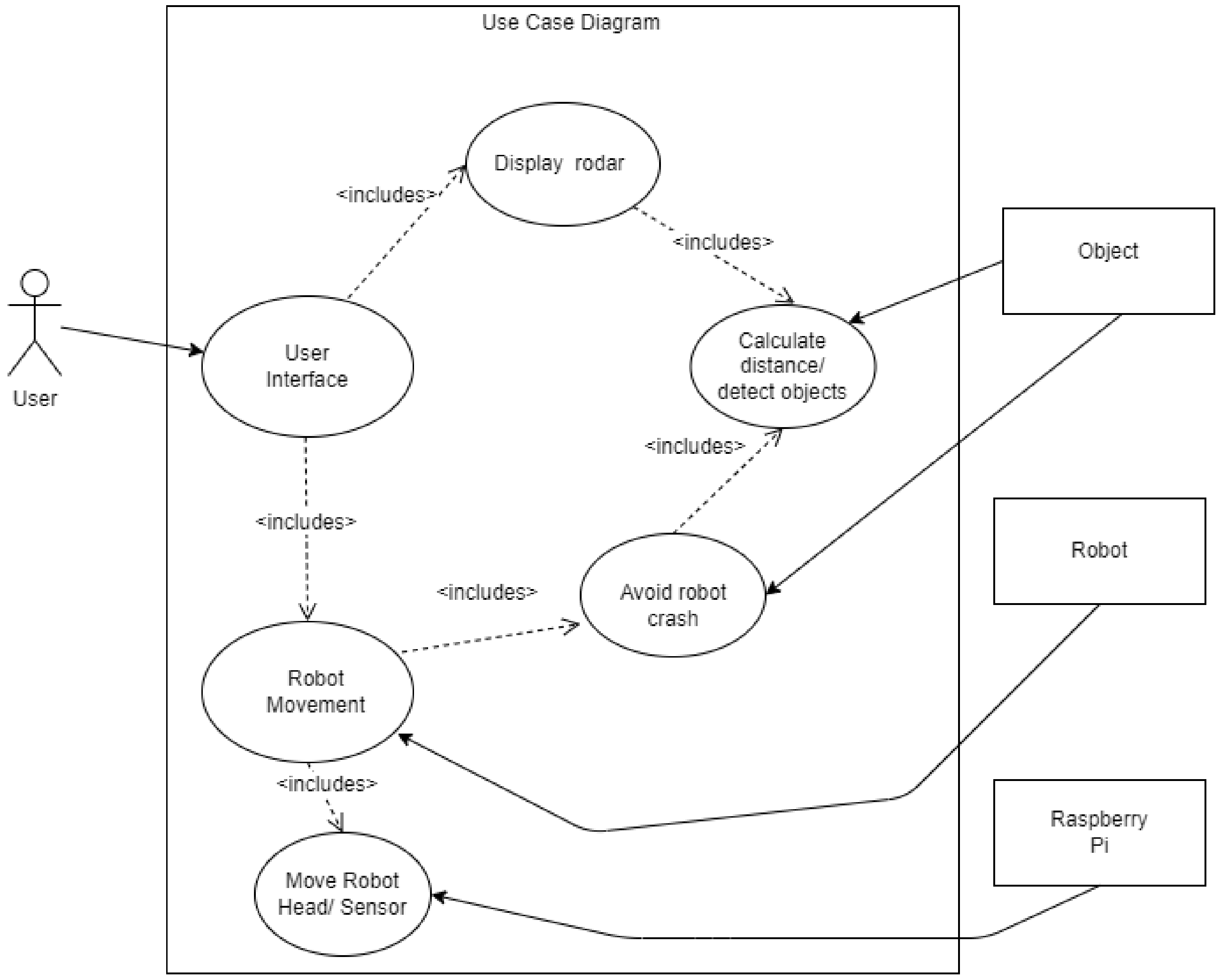


Figure 3 Use case Diagram

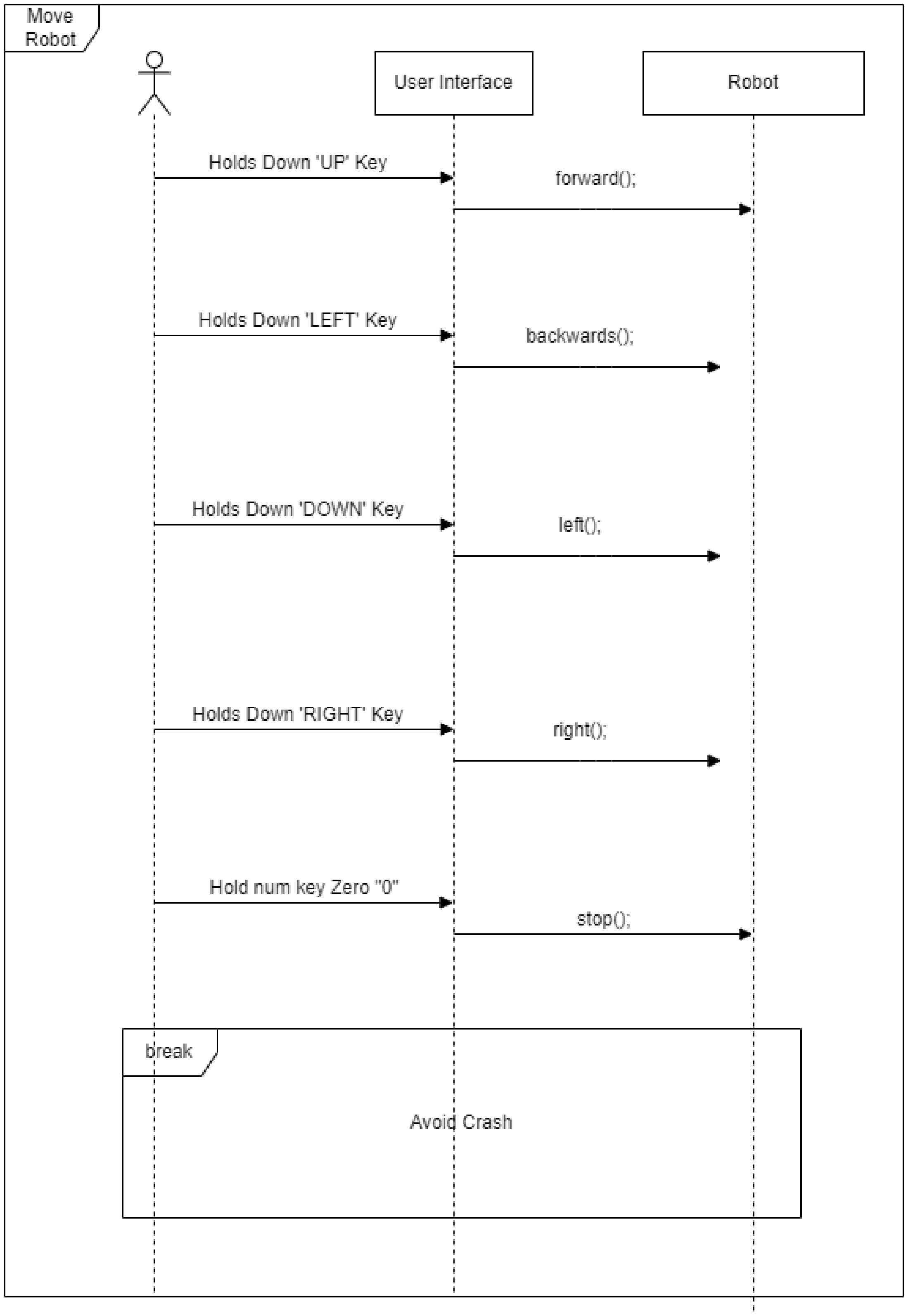


Figure 4: Sequence Diagram of Robot movement

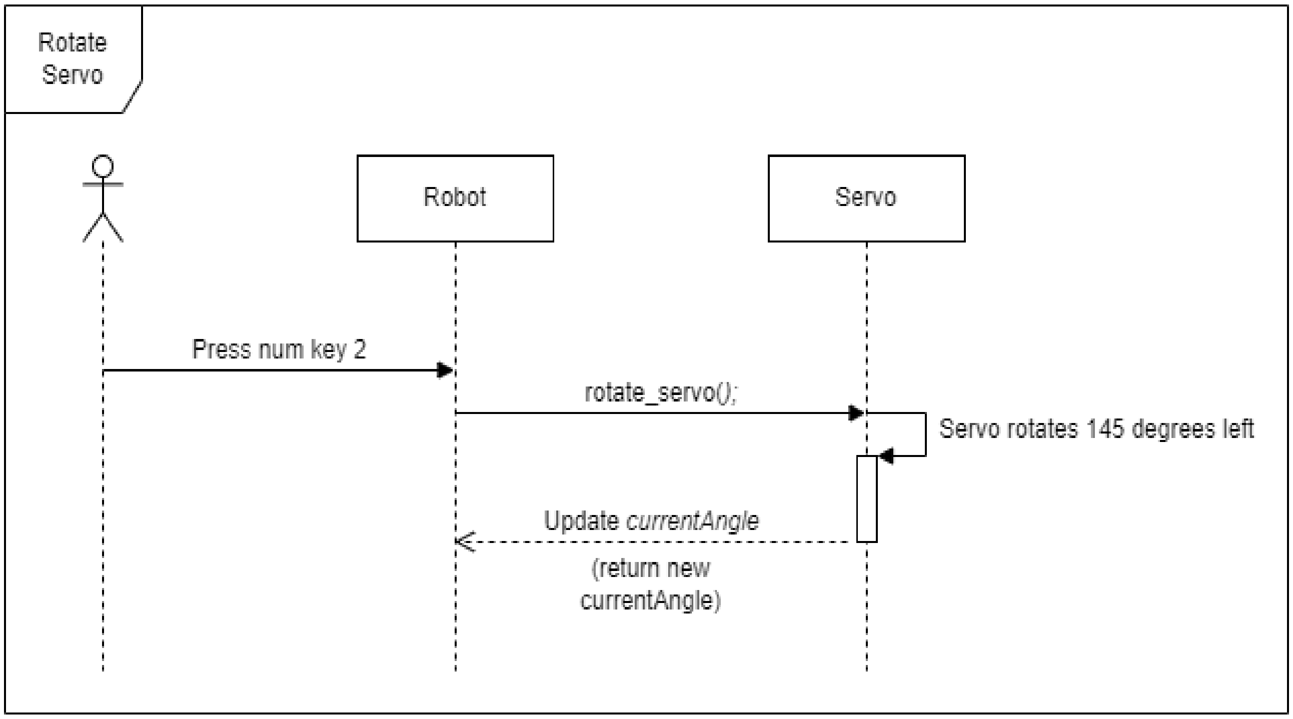


Figure 5: Sequence Diagram to rotate servo

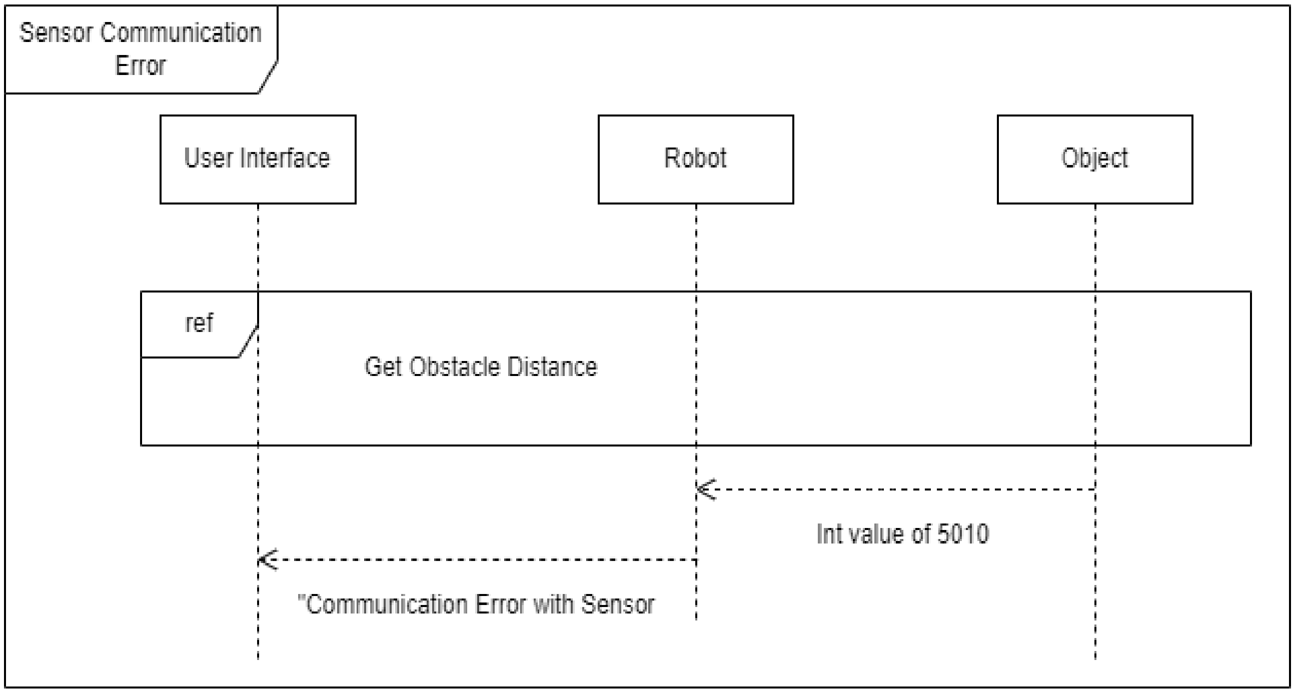


Figure 6: Sequence Diagram sensor communication error

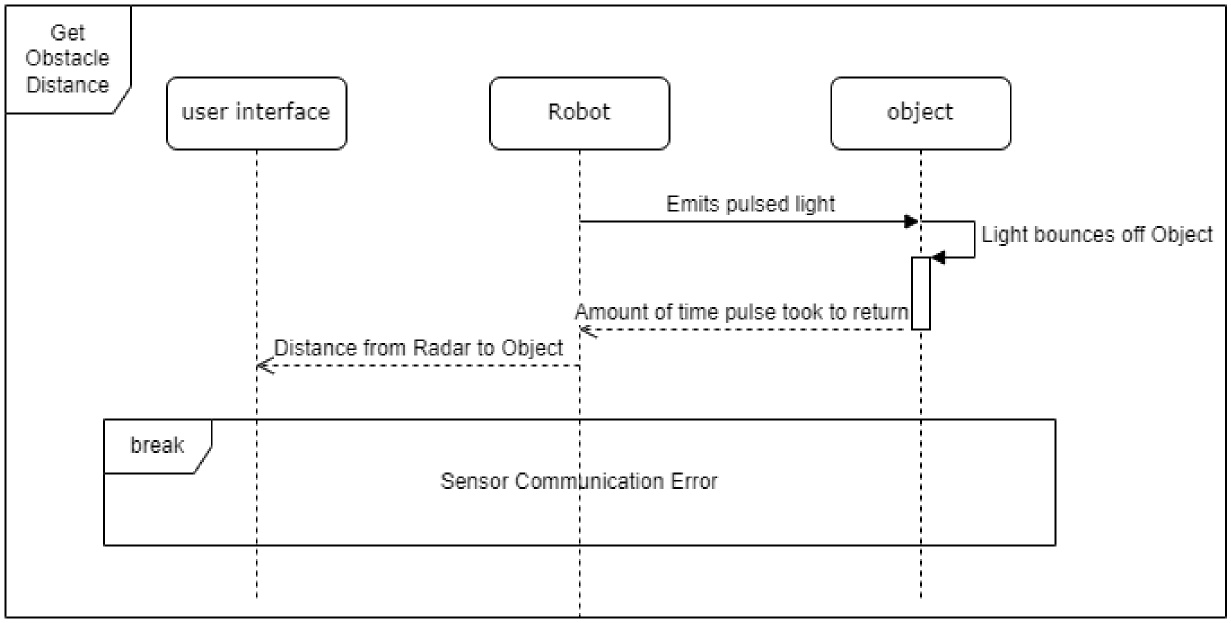


Figure 7: Sequence Diagram get obstacle distance

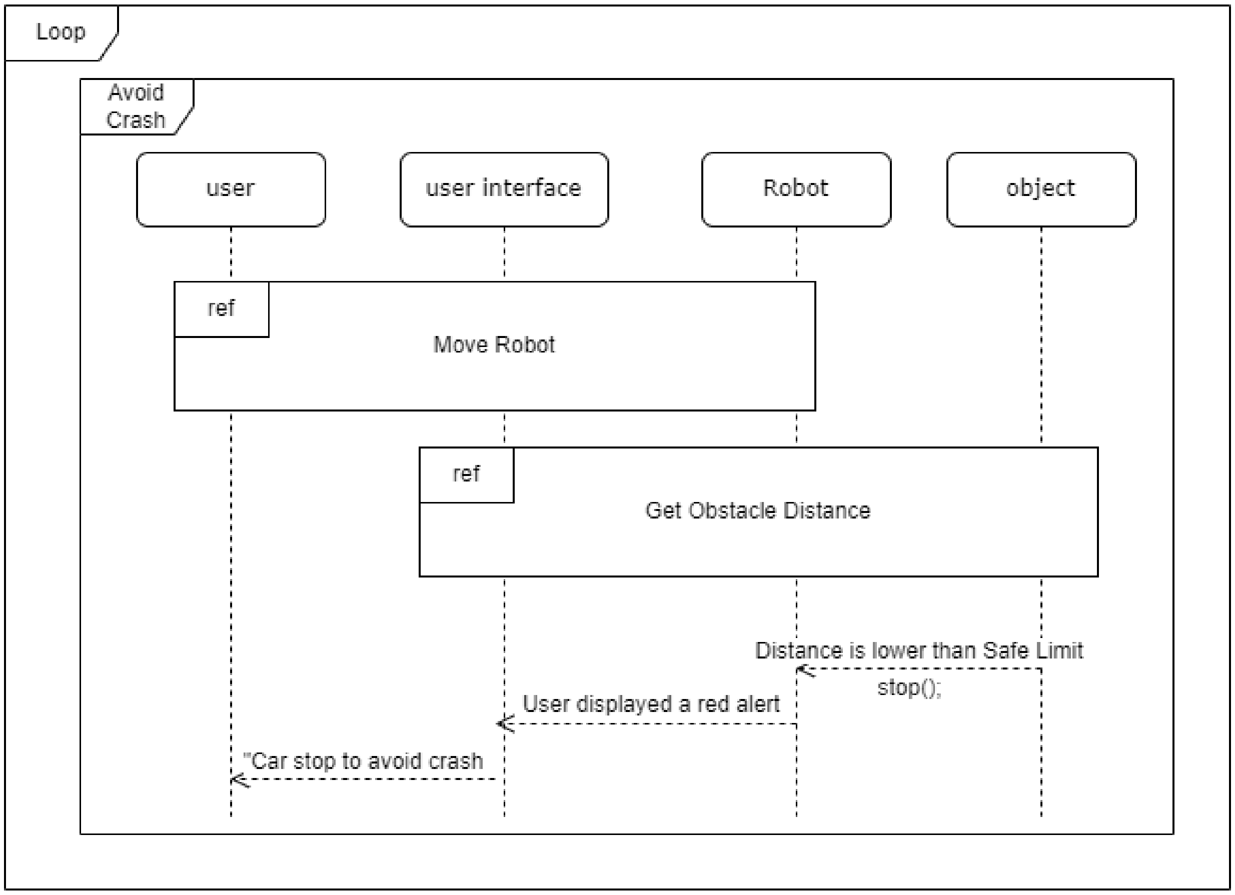


Figure 8: Sequence Diagram to avoid crash

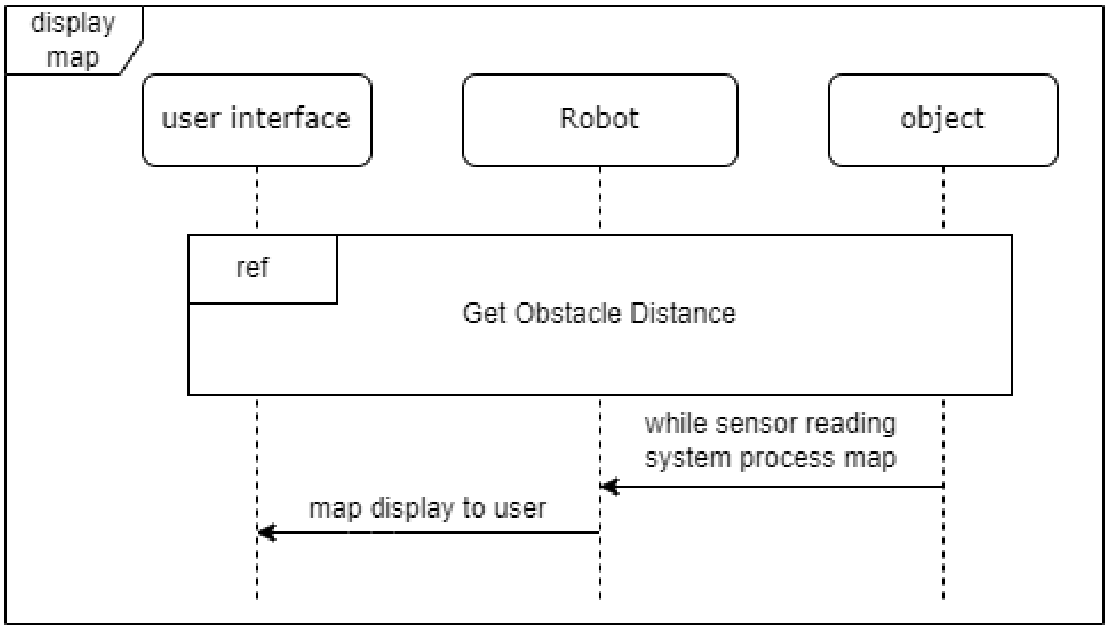


Figure 9: Sequence Diagram to display map

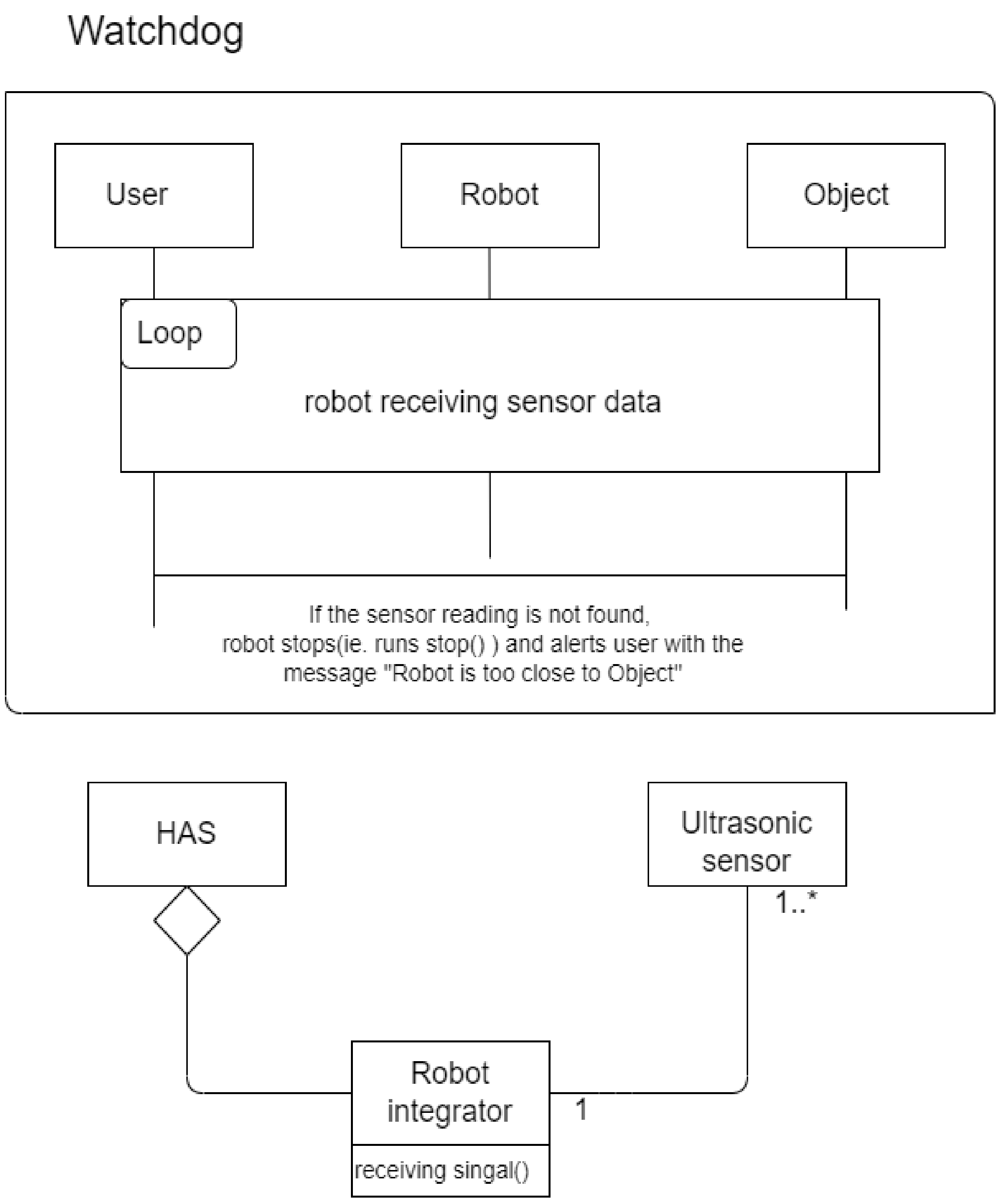

Watchdog
User
Robot
Object
Loop
robot receiving sensor data
If the sensor reading is not found,
robot stops(ie. runs stop()) and alerts user with the
message "Robot is too close to Object"
HAS
Ultrasonic sensor
1..*
Robot integrator
1
receiving singal()

Figure 9: Watchdog pattern diagram

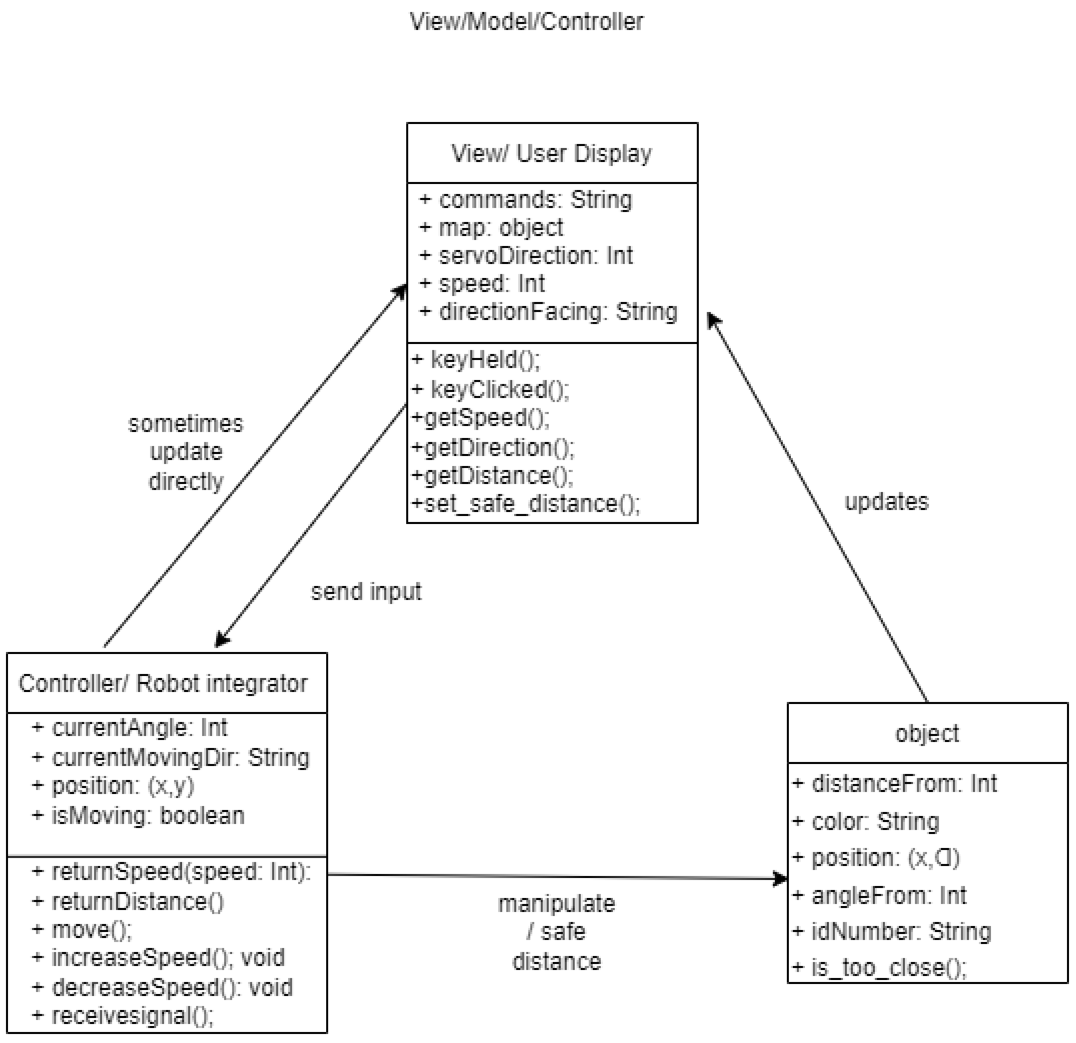


Figure 10: model/view/controller pattern diagram

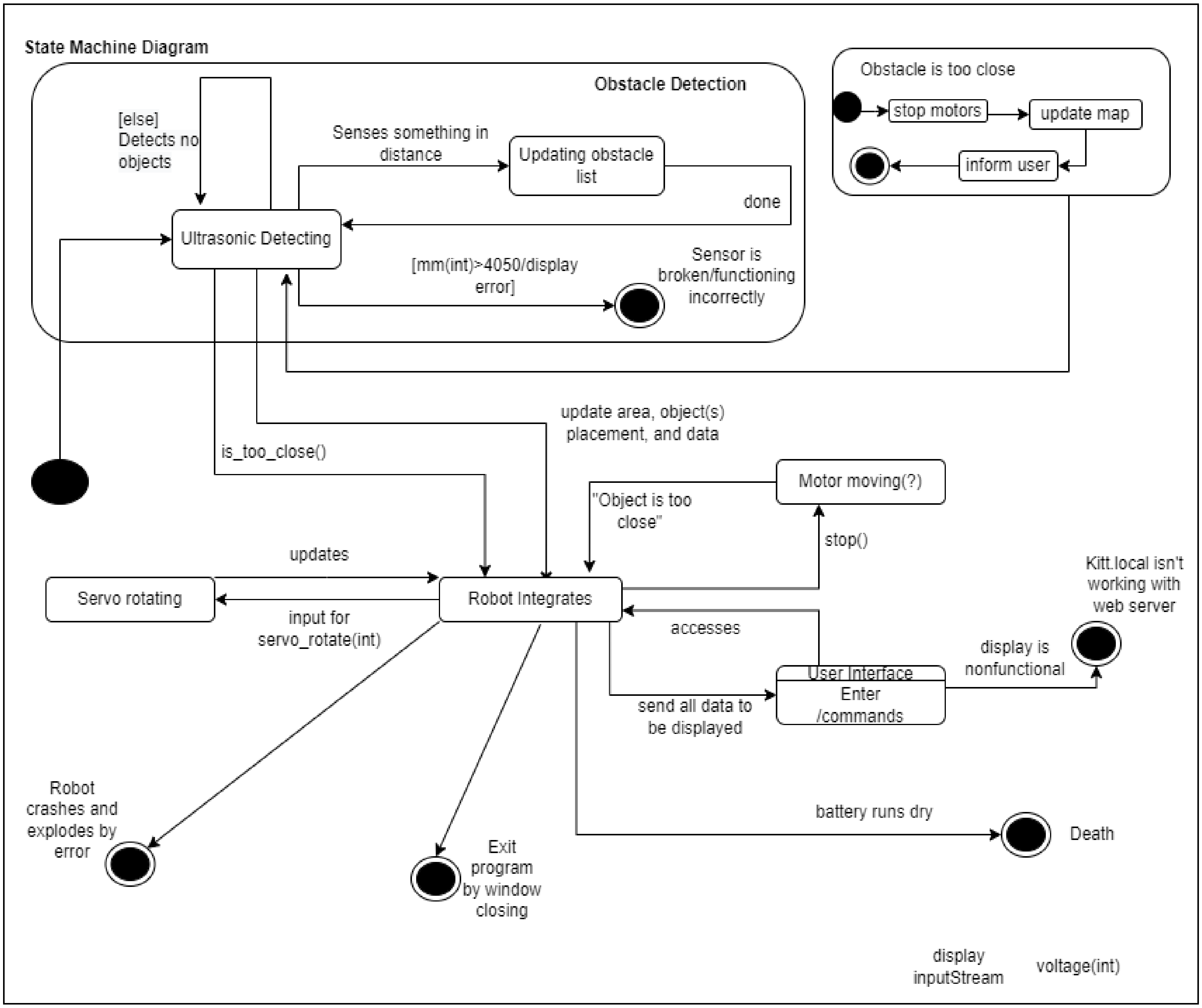


Figure 11:State machine diagram

(Diagram attached separately too to view it better)

Activity Diagrams (Swimlanes // Movement)

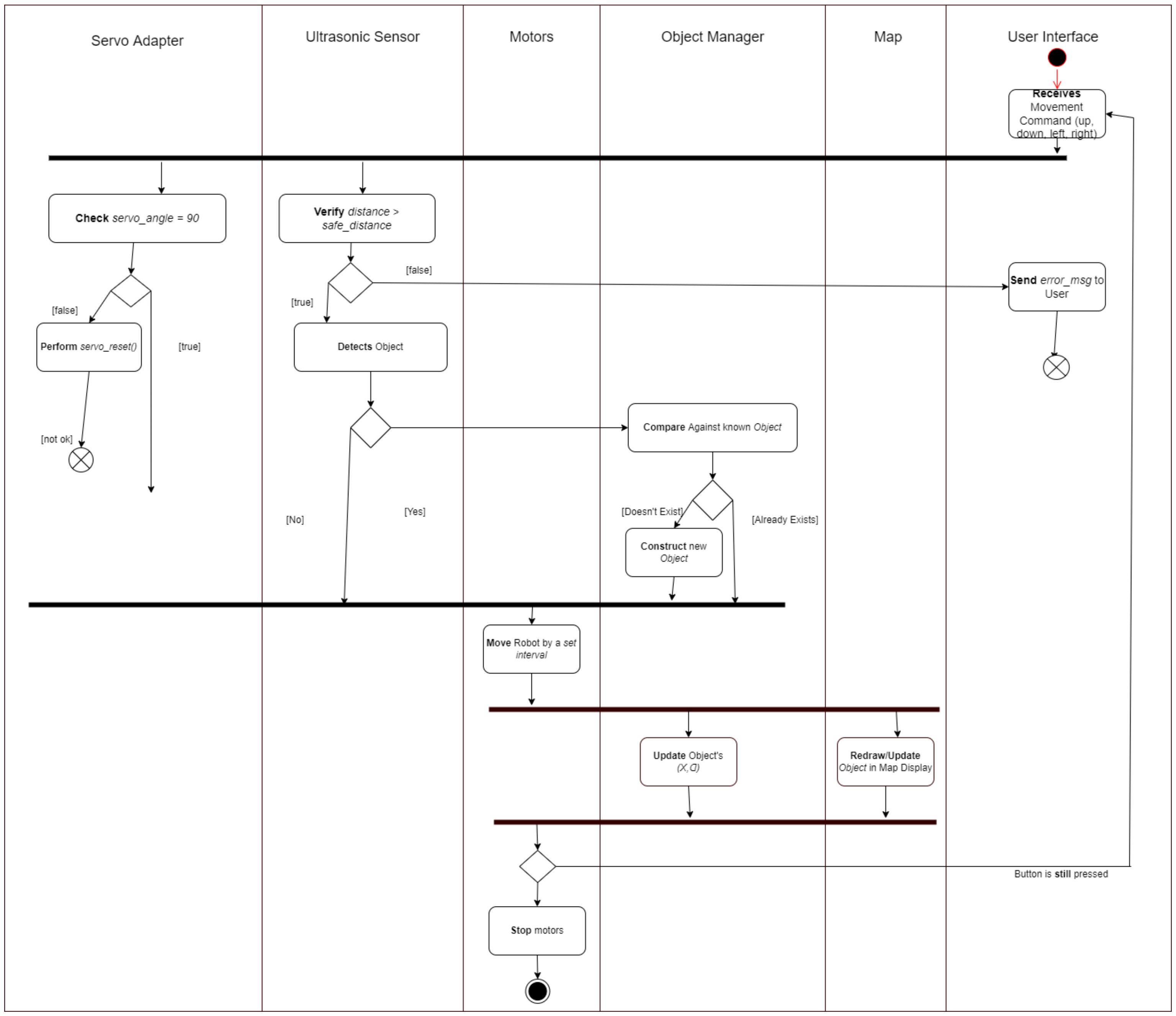


Figure 12: Activity Diagram swimlanes

(Diagram attached separately too to view it better)

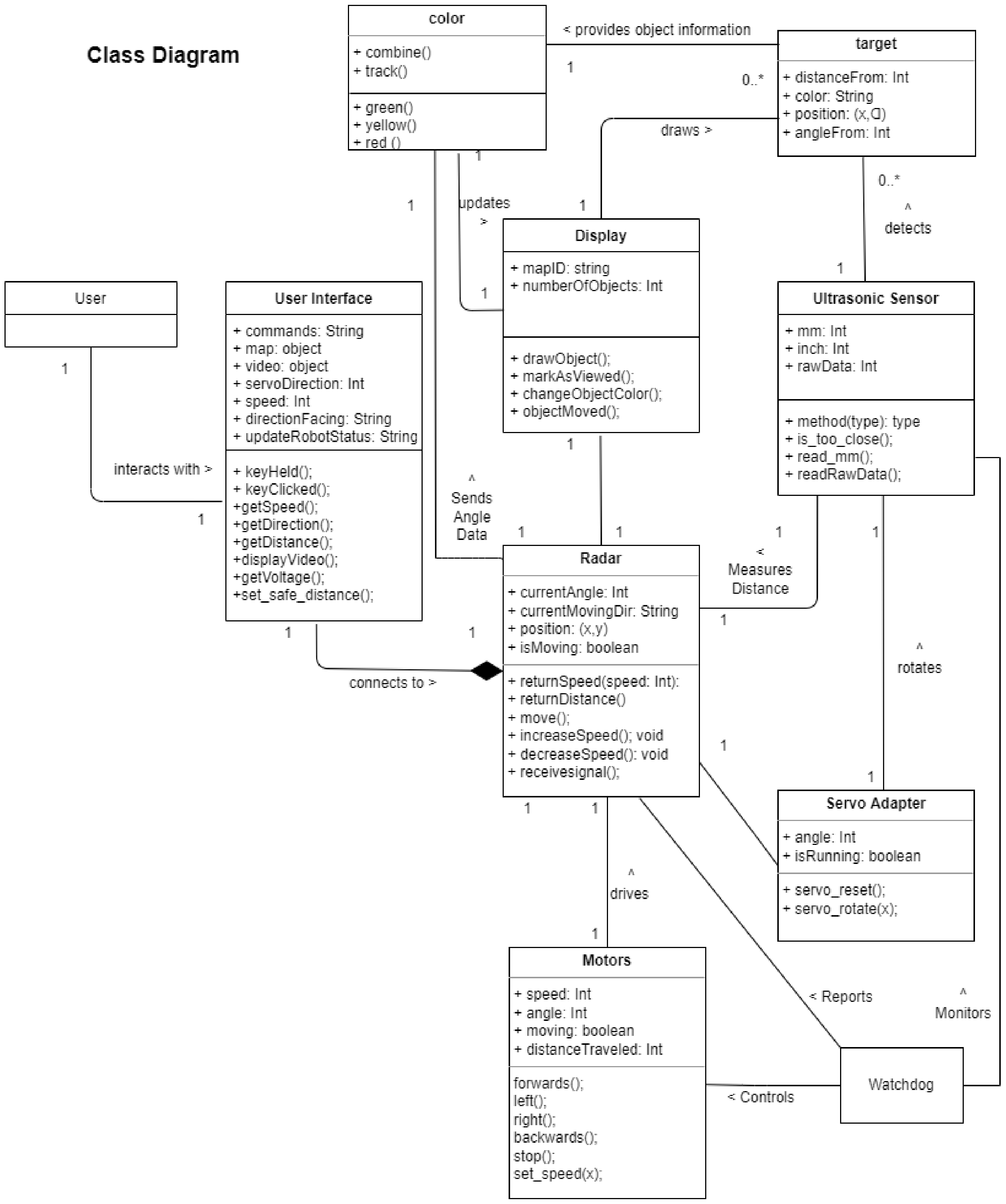
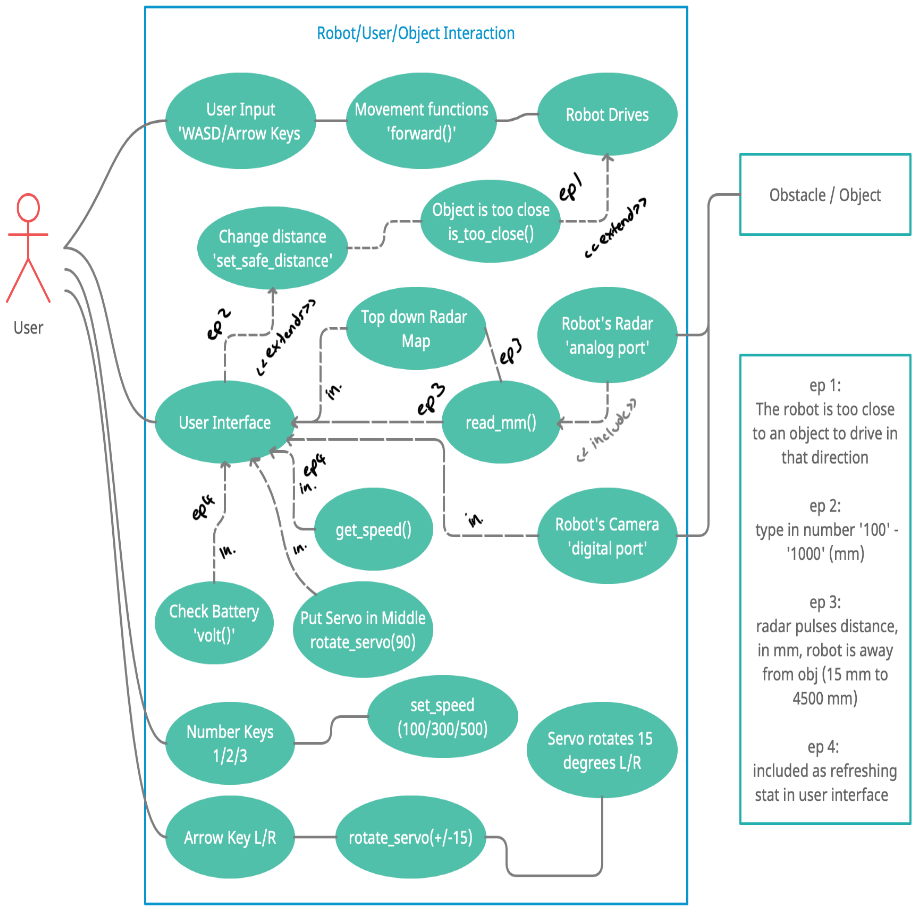
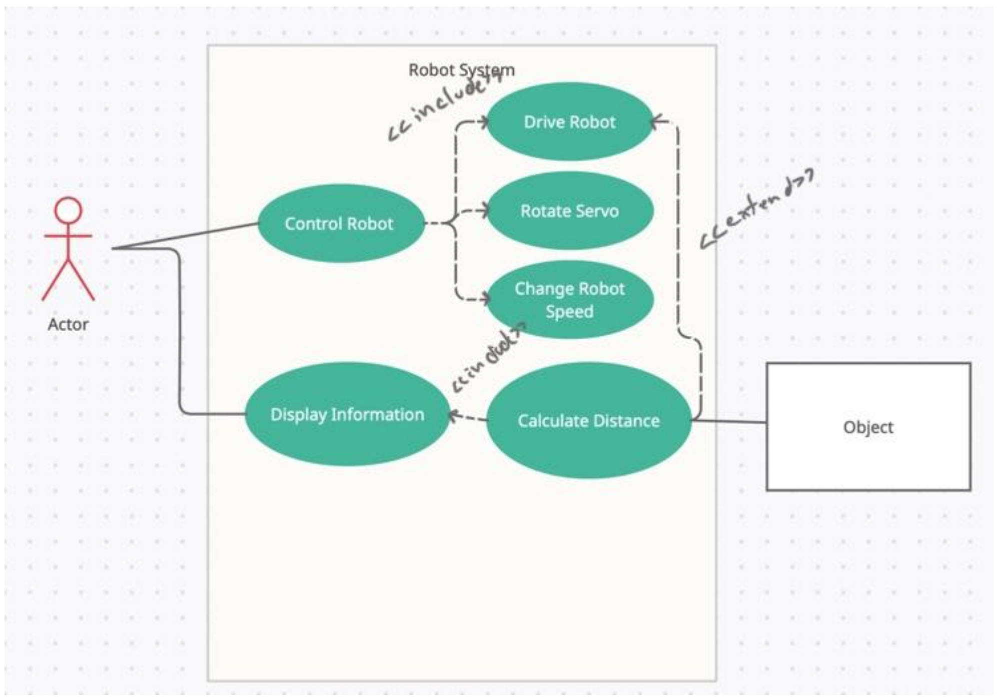


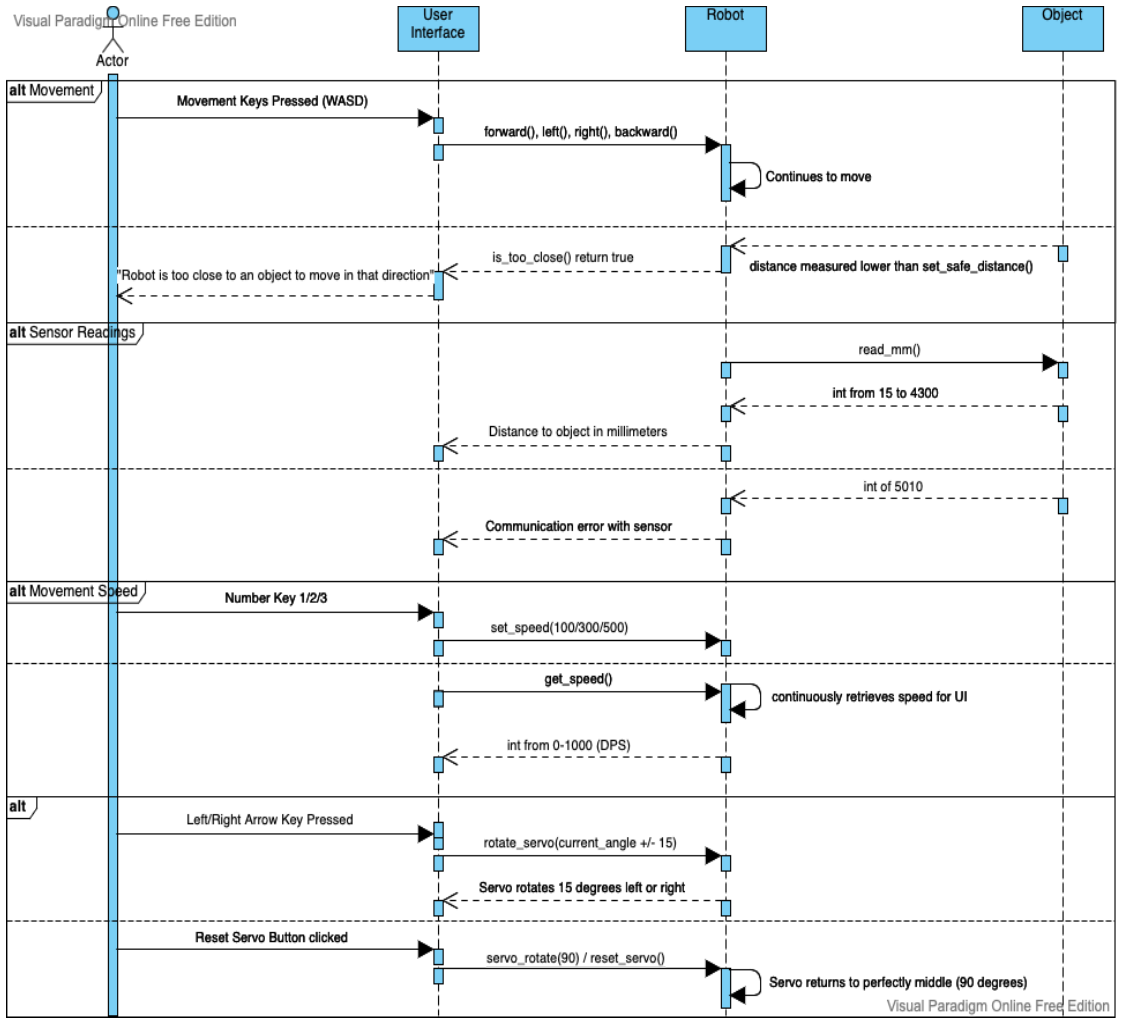
Figure 13: Class Diagram (Diagram attached separately too to view it better)

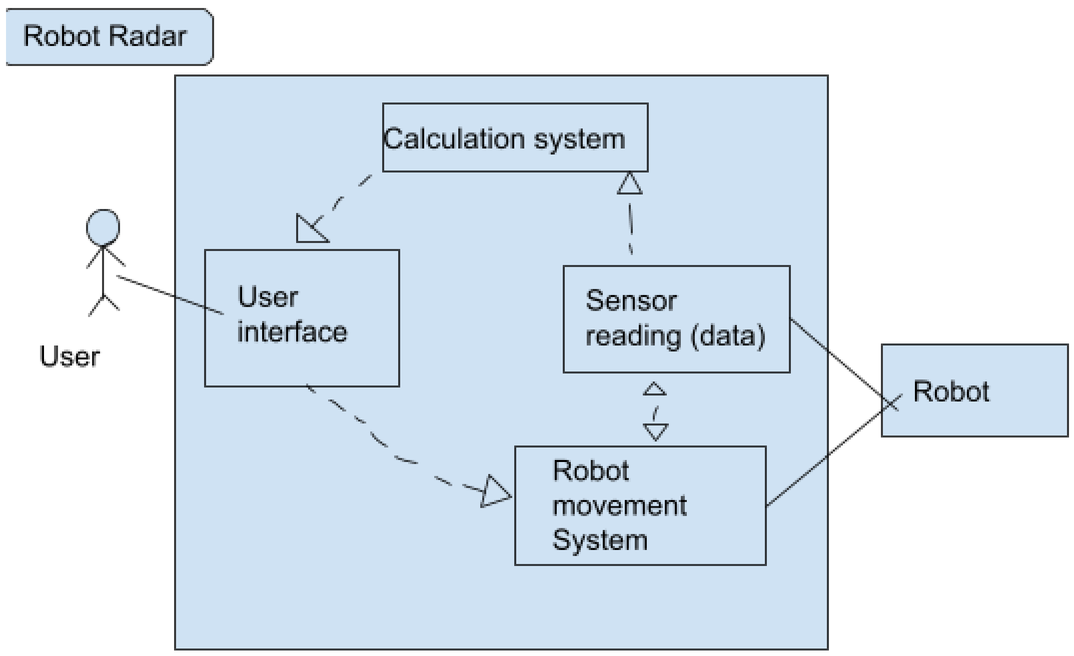
Appendix B: Initial Analysis Models

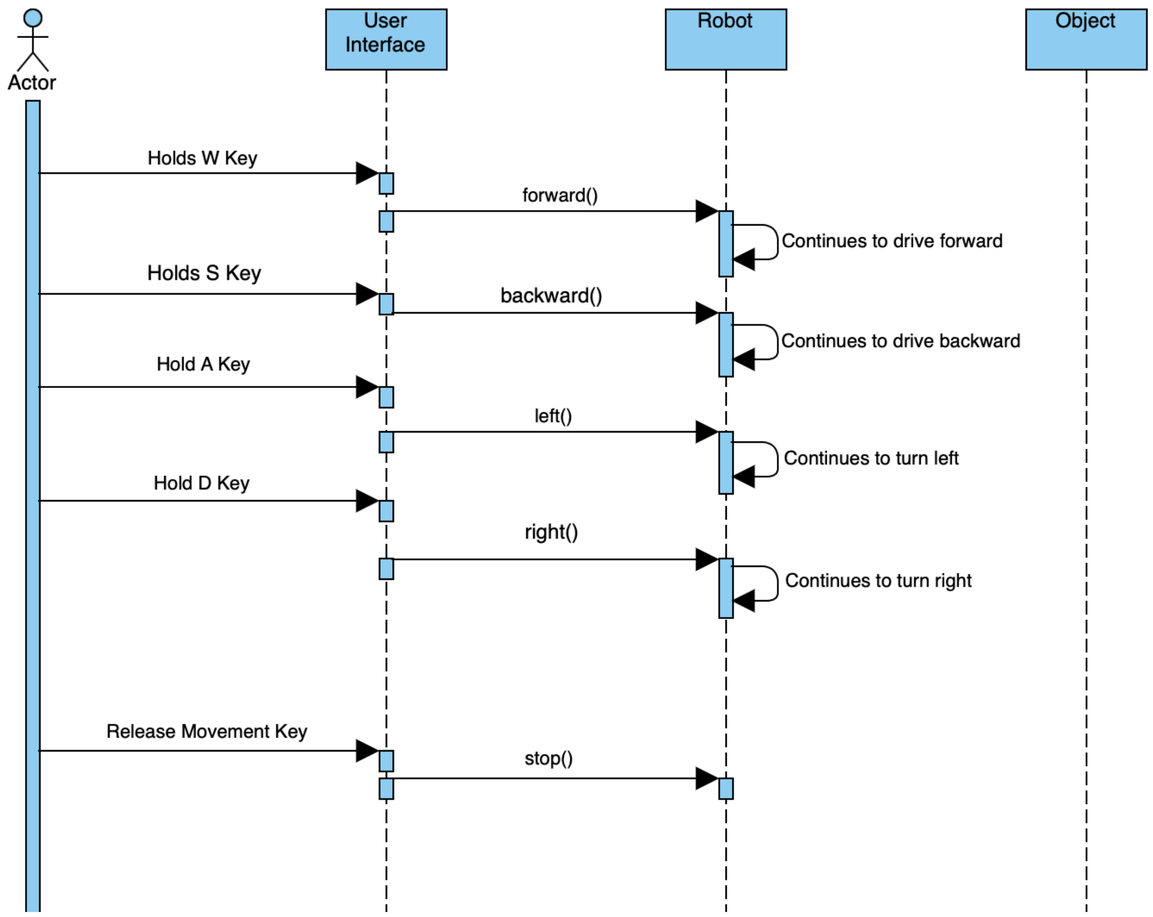
These are initial diagram to get a sketch/ come up with an idea for final diagram

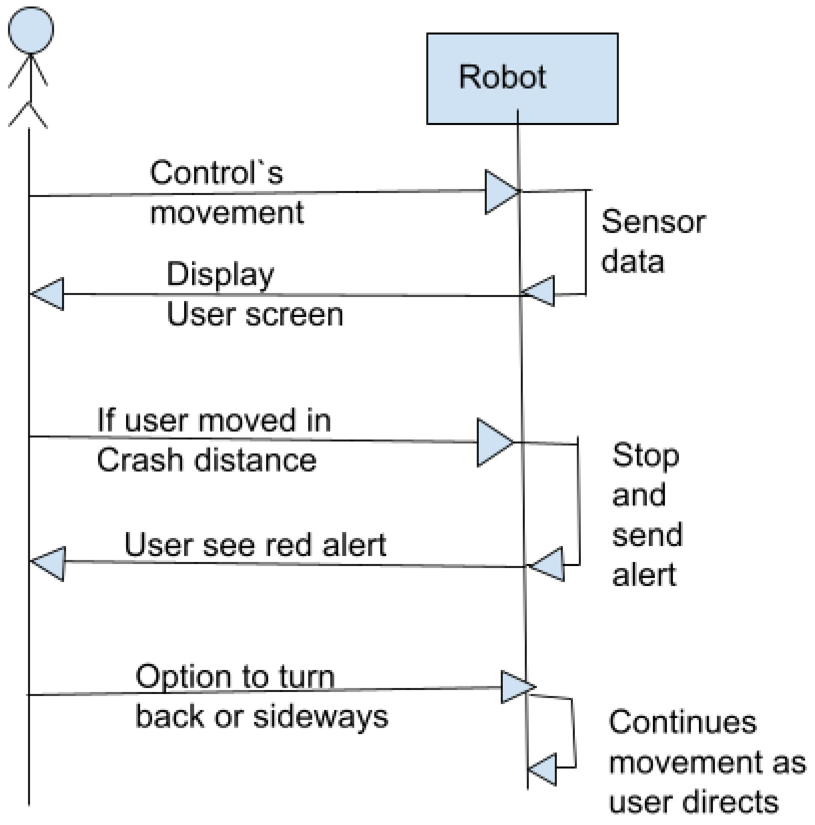

Robot/User/Object Interaction
User
User Input 'WASD/Arrow Keys
Movement functions 'forward()'
Robot Drives
Change distance 'set_safe_distance'
Object is too close is_too_close()
ep1
<<extend>>
Obstacle / Object
ep 2
<<extend>>
Top down Radar Map
Robot's Radar 'analog port'
ep3
in.
ep3
User Interface
read_mm()
<< include>>
ep4
in.
get_speed()
Robot's Camera 'digital port'
in.
ep4
in.
in.
Check Battery 'volt()'
Put Servo in Middle rotate_servo(90)
Number Keys 1/2/3
set_speed (100/300/500)
Servo rotates 15 degrees L/R
Arrow Key L/R
rotate_servo(+/-15)
ep 1: The robot is too close to an object to drive in that direction
ep 2: type in number '100' - '1000' (mm)
ep 3: radar pulses distance, in mm, robot is away from obj (15 mm to 4500 mm)
ep 4: included as refreshing stat in user interface


Robot System
Actor
Control Robot
Drive Robot
Rotate Servo
Change Robot Speed
Display Information
Calculate Distance
Object
<<include>>
<<include>>
<<extend>>

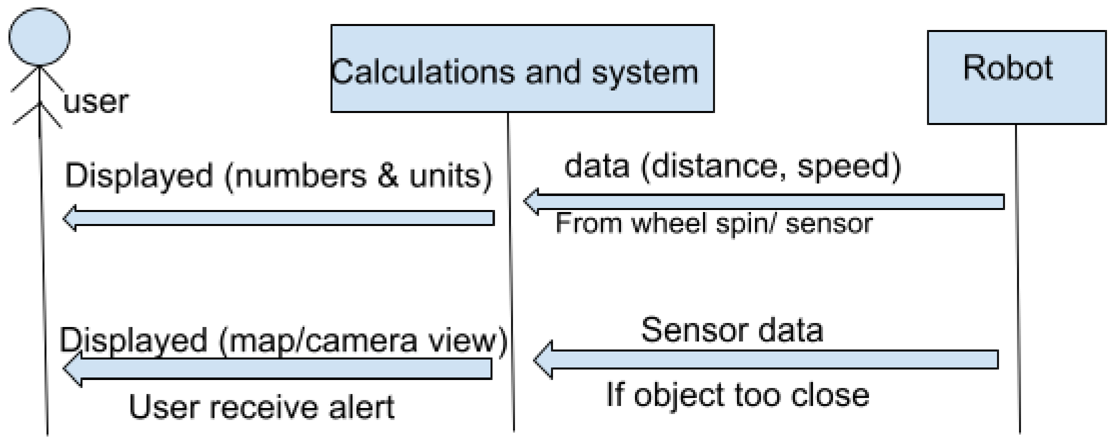

Visual Paradigm Online Free Edition
Actor
User Interface
Robot
Object
alt Movement
Movement Keys Pressed (WASD)
forward(), left(), right(), backward()
Continues to move
is_too_close() return true
distance measured lower than set_safe_distance()
"Robot is too close to an object to move in that direction"
alt Sensor Readings
read_mm()
int from 15 to 4300
Distance to object in millimeters
int of 5010
Communication error with sensor
alt Movement Speed
Number Key 1/2/3
set_speed(100/300/500)
get_speed()
continuously retrieves speed for UI
int from 0-1000 (DPS)
alt
Left/Right Arrow Key Pressed
rotate_servo(current_angle +/- 15)
Servo rotates 15 degrees left or right
Reset Servo Button clicked
servo_rotate(90) / reset_servo()
Servo returns to perfectly middle (90 degrees)
Visual Paradigm Online Free Edition

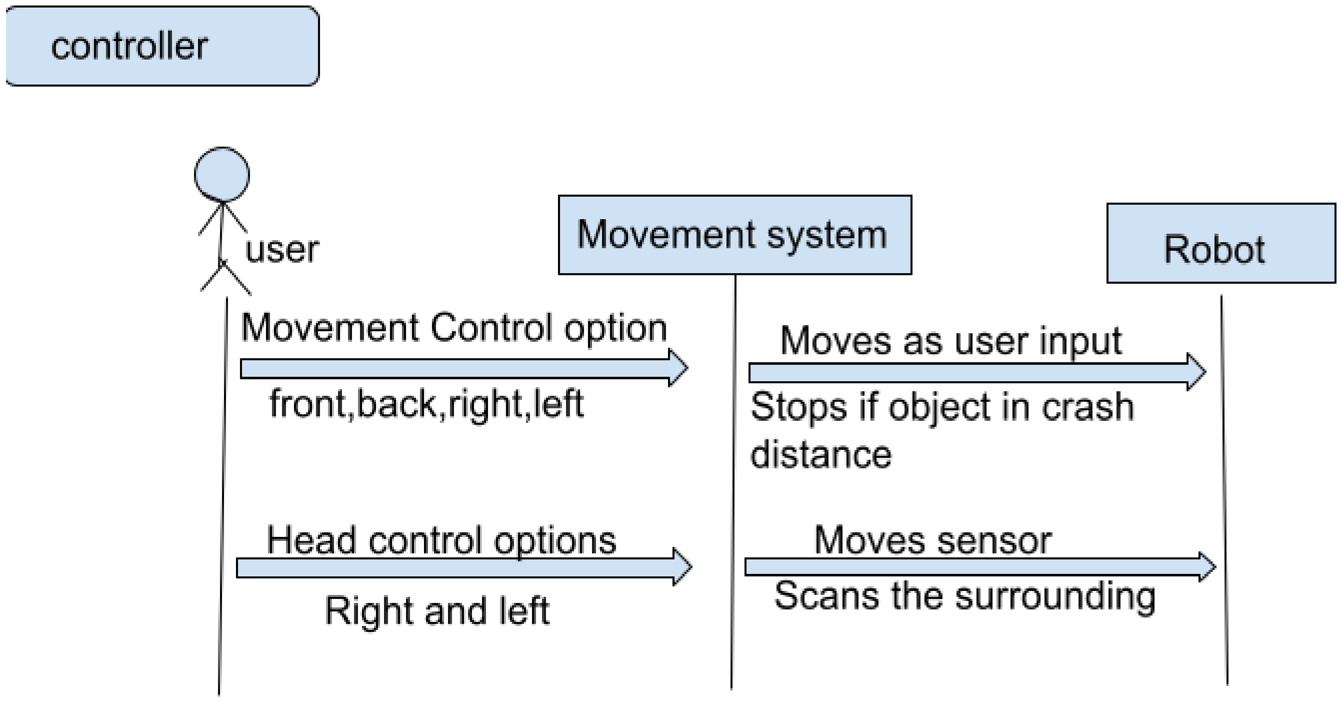

Robot Radar
Calculation system
User
User interface
Sensor reading (data)
Robot
Robot movement System

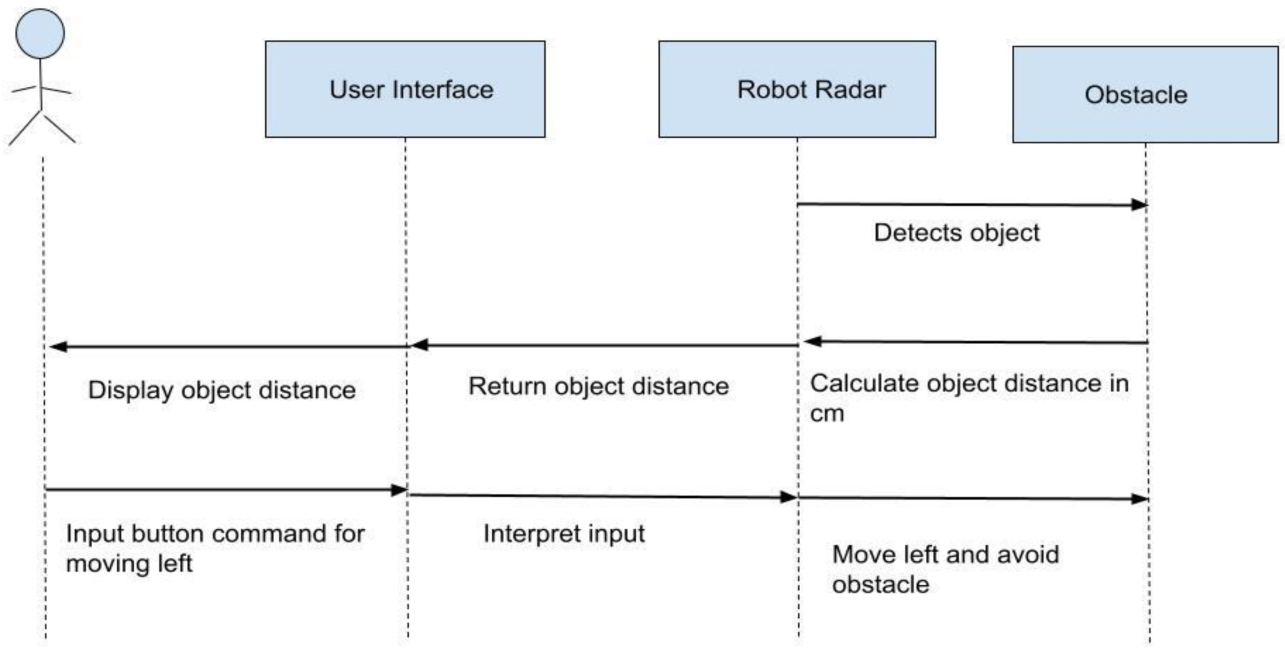

Actor
User Interface
Robot
Object
Holds W Key
forward()
Continues to drive forward
Holds S Key
backward()
Continues to drive backward
Hold A Key
left()
Continues to turn left
Hold D Key
right()
Continues to turn right
Release Movement Key
stop()


Robot
Control`s movement
Sensor data
Display User screen
If user moved in Crash distance
Stop and send alert
User see red alert
Option to turn back or sideways
Continues movement as user directs

User Display




controller
user
Movement system
Robot
Movement Control option
front,back,right,left
Moves as user input
Stops if object in crash distance
Head control options
Right and left
Moves sensor
Scans the surrounding


User Interface
Robot Radar
Obstacle
Detects object
Display object distance
Return object distance
Calculate object distance in cm
Input button command for moving left
Interpret input
Move left and avoid obstacle