

**ÇANKAYA UNIVERSITY**

**COMPUTER ENGINEERING DEPARTMENT**

**CENG 408**

**SOFTWARE REQUIREMENT SPECIFICATION**

**Members**

Turgut Utku ALTINKAYA

Burak ATEŞ

Yunus Emre DİNÇEL

Bayram Alper KILIÇ

İlteriş SAMUR

## Table of Contents

1.	Introduction.....	1
1.1.	Purpose of This Document.....	1
1.2.	Scope of The Project.....	1
1.3.	Glossary .....	3
2.	Overall Description.....	5
2.1.	User Characteristics .....	5
2.1.1.	Operators.....	5
2.1.2.	Developers and Engineers.....	5
2.2.	Product Perspective.....	5
2.3.	Product Functions .....	7
3.	Requirement Specification.....	8
3.1.	External Interface Requirements.....	8
3.1.1.	User Interface.....	8
3.1.2.	Hardware Interface.....	8
3.1.3.	Software Interface .....	12
3.1.4.	Communication Interface.....	13
3.2.	Functional Requirements .....	13
3.3.	Non-Functional Requirements .....	23
3.3.1	Performance Requirements .....	23
3.3.2	Safety Requirements .....	23
3.3.3	Security Requirements .....	24
3.3.4	Software Quality Attributes .....	24
4.	Use Cases.....	25
4.1.	Movement .....	25
4.2.	Data Stream.....	25
4.3.	Manual Mapping.....	26
4.4.	Autonomous Mapping .....	26
4.5.	Object Detection .....	27
	References.....	28

# 1. Introduction

## 1.1. Purpose of This Document

The purpose of this document is to provide a detailed understanding of the scope, capabilities, functions and hardware of Sentinel. It plays an important role as it is a reference for the development team, stakeholders and other organizations involved in the development and improvement of Sentinel. This document covers the entire development lifecycle of Sentinel from concept to development. It provides a detailed description of the objectives, user interactions, vehicle motion, environment mapping and system architecture for the successful physical implementation and development of this vehicle.

## 1.2. Scope of The Project

The Sentinel idea stemmed from the development of unmanned vehicle technologies in modern warfare. In today's world, many countries use technology in the defense industry for both self-defense and operational purposes. When these vehicles are not used, they cause many soldiers to lose their lives. For this reason, there is a great need for unmanned vehicles in war zones. For this reason, many countries are investing in these vehicles. This situation caught our interest and we decided to make an autonomous environment mapping vehicle. However, our vehicle is a proof of concept and isn't suitable for the war zone.

The main purpose of Sentinel is to create 2D and 3D maps of its surroundings by moving autonomously. The vehicle can be sent to dangerous places to autonomously map the environment, thus preventing personnel loss. It can be used to create a map of the environment in many areas such as buildings that are not safe to enter, war zones and places contaminated with harmful gases. The importance of this is that it gives an idea about the place without human interaction and provides detailed 3D environment mapping that shows dangers in advance. It can also be used to detect life in rescue operations when a thermal camera is used.

The vehicle is mainly equipped with Raspberry Pi5, which runs on Ubuntu, YDLidar X2, and Picamera 3. The data collected from the lidar sensor and camera are combined and used to generate the 3D map utilizing the rtabmap\_ros [1], a package used to generate a 3D point of clouds of the environment and/or to create a 2D occupancy grid map for navigation. It will mainly use the Robot Operating System (ROS) to control both autonomous and manual movement, and environment mapping.

The collected data is planned to be processed at a remote computer that is also equipped with ROS, and only the movement command from the remote computer will be returned to Sentinel. In this way, we also plan to provide a manual control mechanism which allows our users to interact with the vehicle any given time. Moreover, the generated map and the camera footage will be available to the user at the remote computer in real-time.

After the map has been generated, we also plan to detect the objects using YOLO, [2] a library for object detection, and then store these informations using rosbag [3].

In summary, users can control the movement of the vehicle, view images from the vehicle, and view 2D and 3D maps in real time. All of this is done on a remote computer. On the other hand, Sentinel is responsible for sending sensor and camera data to the remote computer and obeying the command received from the remote computer. In the following sections of this SRS report, these functionalities and the hardware used for the vehicle will be explained in detail.

### 1.3. Glossary

Term	Description
Bags	A bag is a file format in ROS for storing ROS message data.
C++	Programming Language
DC Motor	A DC motor is an electrical motor that uses direct current (DC) to produce mechanical force. The most common types rely on magnetic forces produced by currents in the coils.
DSI	The Display Serial Interface (DSI) is a specification by the Mobile Industry Processor Interface (MIPI) Alliance aimed at reducing the cost of display controllers in a mobile device.
FPC	FPC connectors have been established in response to challenges in this emerging industry which calls for smaller centerline or timer distances, smaller capacity heights, and lightweight interconnection solutions as the industry trends towards miniaturization.
HDR	High dynamic range (HDR), also known as wide dynamic range, extended dynamic range, or expanded dynamic range, is a signal with a higher dynamic range than usual.
L298N	Motor driver that helps you control the motors.
L298N Motor Driver	The L298N motor driver is a versatile module used to control both the speed and direction of DC motors.
Li-ion Battery	Li-ion battery is a type of rechargeable battery that uses the reversible intercalation of Li <sup>+</sup> ions into electronically conducting solids to store energy.
LiDAR	LiDAR is short for Light Detection and Ranging. In LiDAR, laser light is sent from a source (emitter) and reflected off objects in the scene.
NVMe	NVM Express (NVMe) is an open, logical-device interface specification for accessing a computer's non-volatile storage media usually attached via the PCI Express bus.
Pi Camera Module 3	Compact camera from Raspberry Pi
Power Bank	A power bank or battery bank is a portable device that stores energy in its battery. Power banks are made in various sizes and typically based on lithium-ion batteries
Publisher/Subscriber	Publish/subscribe is a messaging pattern where publishers categorize messages into classes that are received by subscribers.
Python	Programming Language

RAM	Random-access memory (RAM) is a form of electronic computer memory that can be read and changed in any order, typically used to store working data and machine code.
Raspberry Pi 5	Small single-board computer
ROS	Robot Operating System (ROS or ros) is an open-source robotics middleware suite that helps you build robot applications.
ROS Topic	Buses over which nodes exchange messages
Rosbag	A set of tools for recording and playing back to ROS topics.
RPM	RPM stands for revolutions per minute and is a measure of how fast the engine is spinning.
RTAB-Map	ROS framework for 3D mapping.
SD card	Secure Digital, officially abbreviated as SD, is a proprietary, non-volatile, flash memory card format the SD Association (SDA) developed for use in portable devices.
SLAM	Simultaneous Localization and Mapping. Algorithm for mapping. Build a map and localize your vehicle in that map at the same time.
Slam_toolbox	Slam Toolbox is a set of tools and capabilities for 2D SLAM.
SSD	A solid-state drive (SSD) is a type of solid-state storage device that uses integrated circuits to store data persistently.
The Sentinel	The sentinel is a Discovery Vehicle. The name of our project, The Sentinel, is an autonomous vehicle with features such as 3D mapping and object detection.
Ubuntu	Linux based operating system
YDLidar X2	360-degree two-dimensional rangefinder sensor.
YOLO	You Look Only Once. Open source library that helps you object detection.

## 2. Overall Description

### 2.1. User Characteristics

#### 2.1.1. Operators

The Operator is responsible for monitoring and controlling The Sentinel's operations. Their tasks include both manual and autonomous movement. Operators can use tools such as a keyboard or joystick for manual control or manage its autonomous functions through a user interface. This type of user must have a basic understanding of The Sentinel's functionalities, as well as the minimum technical expertise required for routine tasks.

#### 2.1.2. Developers and Engineers

The developers and engineers are responsible for designing, implementing, and maintaining both the hardware and software systems of The Sentinel. This also includes working with advanced technologies like ROS (Robot Operating System), mapping and path planning algorithms, and data processing frameworks. The developers and engineers should be capable of performing troubleshooting, optimizing the system, and making sure it works reliably in a variety of conditions. Moreover, they might be required in deploying updates, enhancing performance, or resolving various kinds of technical challenges that may occur while the system is operational.

### 2.2. Product Perspective

The Sentinel includes different hardware and software systems. The Sentinel Project aimed to produce a vehicle prototype with the specified features. So we used hardware tools for the prototype. On the software part Sentinel includes different computer science concepts: mapping algorithms (SLAM), path planning algorithms, object detection algorithms and also communication protocols. These algorithms will be implemented by different programming languages: C++, Python. Additionally, we use ROS2 and its frameworks like RTAB-Map for implementing these features. On the hardware part since our project is a vehicle, we used 4 wheels and 4 motors. The 2 motors on the right and the 2 motors on the left are connected in series with each other. Motors are powered by Li-On Batteries with the voltage of 12V. The L298N motor driver is used to controlling motors for forward, backward left and right movement. The brain for the vehicle is Raspberry Pi 5. Raspberry Pi 5 is responsible for gathering visual data from Pi Camera Module 3 and distance data from YDLidar X2 which both are mounted on Raspberry Pi. When this data gathering Raspberry Pi real time streams both visual and distance data to a remote computer from ROS2 publisher/subscriber protocol. Remote

computers are responsible for computations like mapping, object detection and path planning. For autonomous driving, a remote computer sends movement commands from the output of computations to Raspberry Pi and at Raspberry Pi this command is executed according to the input received. The detailed explanation of the hardware interface will be given at 3.1.2. Hardware Interfaces section and software interface will be given at 3.1.1.

In the below table, we stated what are the features of our vehicle, summarized explanation, and which tools we will use as hardware.

<b>Task</b>	<b>Definition</b>	<b>Hardware Tools</b>
Manuel Driving	The Sentinel shall operate by a user from a remote computer with a joystick or keyboard.	<ul style="list-style-type: none"> <li>● Raspberry Pi 5</li> <li>● Remote Computer</li> <li>● Joystick or Keyboard</li> <li>● 4 x motor</li> <li>● Vehicle Chassis &amp; tires</li> <li>● L298N Motor Driver</li> </ul>
Autonomous Driving	The Sentinel shall operate by itself while considering the ROS2 outputs.	<ul style="list-style-type: none"> <li>● Raspberry Pi 5</li> <li>● Remote Computer</li> <li>● YDLidar X2</li> <li>● 4 x motor</li> <li>● Vehicle Chassis &amp; tires</li> <li>● L298N Motor Driver</li> </ul>
3D Mapping	The Sentinel shall autonomously map unknown environments using ROS2's framework RTAB-Map.	<ul style="list-style-type: none"> <li>● Raspberry Pi 5</li> <li>● Remote Computer</li> <li>● YDLidar X2</li> <li>● Pi Camera Module 3</li> </ul>
Object Detection	The Sentinel shall classify the objects at unknown environments.	<ul style="list-style-type: none"> <li>● Raspberry Pi 5</li> <li>● Remote Computer</li> <li>● Pi Camera Module 3</li> </ul>
Real Time Data Stream	The Sentinel shall stream gathered camera and lidar data to remote computer and web applications.	<ul style="list-style-type: none"> <li>● Raspberry Pi 5</li> <li>● YDLidar X2</li> <li>● Pi Camera Module 3</li> </ul>



## 2.3. Product Functions

**Manuel Movement:** The system is designed to performs directional movements based on user inputs (left, right, up, down) either from joystick or a keyboard from joystick or keyboard.

**Autonomous Movement:** The system is designed to perform movement according to the system output from ROS2. The ROS2 output includes direction and speed. The system can handle the speed with the motor rpm rated from 0-100, adjusting the motor rpm according to the incoming speed value and providing the speed correctly.

**Data Collection:** The system is equipped to collect visual data in the form of images and videos using the Pi Camera Module 3, which is integrated to Raspberry Pi 5 hardware. Additionally it can gather comprehensive 360-degree distance measurement using the YDLidar X2 sensor, which operates with ROS2.

**Real Time Data Transfer:** The system enables real-time transfer of camera and LiDAR data using the ROS2 publisher/subscriber protocol. The data collected on the Raspberry Pi is transmitted to the remote computer efficiently, utilizing ROS2's message framework. Both the Raspberry Pi and the remote computer run ROS2 to ensure communication.

**2D Mapping:** The system can create a real-time 2D grid map of an unknown environment using distance and angle data from a LiDAR sensor. It is generated with the slam\_toolbox package which is integrated within ROS2 and RViz, to generate the 2D grid. slam\_toolbox incorporates information from laser scanners in the form of a LaserScan message and TF transforms from odom->base link, and creates a map 2D map of a space.

**3D Mapping:** The system can create a 3D map of the environment using RTAB-Map, similar to how 2D mapping is performed. However, 3D mapping includes additional considerations. During the mapping process, image data from a camera integrated with a LiDAR sensor is placed on the map based on distance and angle. Unlike 2D mapping, which primarily considers distances along the x-axis, 3D mapping also incorporates distances along the y-axis, resulting in a three-dimensional representation of the environment.

**Object Detection:** The system can perform real-time object detection in an unknown environment using a Pi Camera Module 3 mounted on the vehicle. It utilizes a deep learning-based object detection framework YOLO (You Only Look Once), integrated into ROS2 to

identify and classify objects in the surroundings. The system processes the camera's video feed to detect objects considering their position and size in the environment.

## 3. Requirement Specification




### 3.1. External Interface Requirements

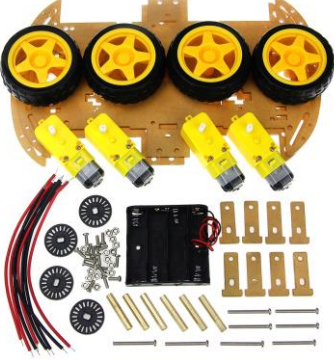
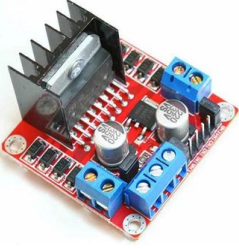


#### 3.1.1. User Interface



The Sentinel application is used for the web. The Sentinel's user interface should be clear and understandable in English. In this context, as the Discovery Vehicle advances within the area it is exploring in The Sentinel User Interface, the images obtained from the camera and the 2D and 3D maps created with various algorithms will be presented to the user. With its simple, sustainable, easy-to-use interface, Sentinel will be able to offer various functions to the user. It offers many features to the users such as visualize the vehicle model on the Admin Dashboard panel, and according to the moving the car the model also will move that vehicle direction. The interface section is constructed 4 separated boxes which is included Live Camera, Laser Scan, 2D and 3D Map, 3D Vehicle Model, and Joystick or Keyboard options to move manually and their animations. Users can also experience the mapping features of the Sentinel Discovery Vehicle by choosing between manual mapping and autonomous mapping options. In addition to the live camera image during mapping, distance and proximity data from Lidar are also presented to users in the web interface. It can also be observed how close the discovery vehicle gets to an object.

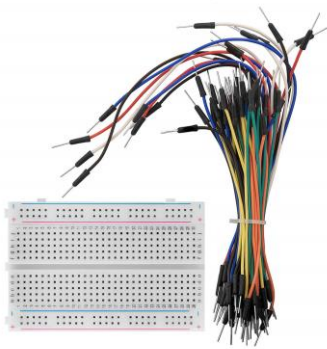

#### 3.1.2. Hardware Interface

The Sentinel requires many separate components in the hardware section. Many hardware requirements for the use of features such as autonomous movement, object detection, and 2D - 3D Mapping are as in the table below. With these products, The Sentinel Autonomous Discovery Vehicle emerges. The intended uses, product images, and descriptions are also included.

Figure	Product Name	Purpose of Use
	Raspberry Pi 5	<p>Raspberry Pi is a cheap, small computer that has 8 GB RAM that runs Linux (various distributions), but also provides a series of GPIO (general purpose input/output) pins that allow you to control electronic components for physical computing. The Sentinel uses the Raspberry Pi for remote manual control, communication with the remote computer via ROS (Robot Operating System), and sending the image captured via the PiCamera and Lidar data to the remote computer for processing.</p>
	LiDAR	<p>LiDAR is short for Light Detection and Ranging. In LiDAR, laser light is sent from a source (emitter) and reflected off objects in the scene. LiDAR is an optical technology that is often cited as a key method for range sensing for autonomous vehicles. The Sentinel uses the LiDAR sensor to provide users with features such as autonomous movement, obstacle detection, and 3D Mapping.</p>
	Pi Camera Module 3	<p>Pi Camera Module 3, is the most popular and latest in the mainstream Raspberry Pi camera class. It has 12 MP and provides good low-light photography and output thanks to its improved low-light intensity. This module also supports High Dynamic Range (HDR), which makes it easier to get a sharp image output. The Sentinel processes the images obtained by the Pi Camera for object detection and displays the detected objects on 3D mapping.</p>

	<p><b>Vehicle Chassis Kit</b></p>	<p>The vehicle chassis kit forms the foundation and body of The Sentinel. There are 4 separate 6V parallel-connected motors for 4 wheels. It is 2-story with an extra chassis piece, lidar, and camera on the top. Powerbank, Motor driver, Raspberry Pi 5, breadboard, and cables are also located on the chassis. There are also 2 battery slots for the batteries needed to power the wheels.</p>
	<p><b>L298N Motor Driver</b></p>	<p>The L298N motor driver is a versatile module used to control both the speed and direction of DC motors. Used this driver to provide precise motion and control for The Sentinel, powered by the Raspberry Pi 5. The L298N can drive up to four DC motors, making it an ideal choice for our system's requirements.</p>
	<p><b>Power Bank</b></p>	<p>The Sentinel requires the use of a power bank to power the Raspberry Pi. The power bank, which provides 5V 5A output, provides enough power to power the Raspberry Pi.</p>
	<p><b>NVMe 2.0 500 GB SSD</b></p>	<p>To provide data flow and communication via Robot Operating System (ROS), ROS must also be installed on Raspberry Pi 5. Since SD cards are insufficient in speed and storage, the SSD with NVMe technology, which can transmit data at high speed with 500 GB storage, is used in The Sentinel. SSD is also of great importance for using the GUI of Raspberry Pi.</p>

	<p>12V Lithium-ion Battery</p>	<p>For the system to work properly, the motors must be given sufficient power. In this way, The Sentinel can move with the energy given to the wheels. For this purpose, sufficient power can be provided with a 12V Lithium-ion Battery. Since it is rechargeable, it contributes to recycling.</p>
	<p>Raspberry Pi Display Cable</p>	<p>With the Raspberry Pi Display Cable, you can physically connect the Pi Camera Module 3 and Raspberry Pi 5 and establish a connection between them. Technically Shielded cable to connect a DSI display to the 22-way FPC connector on Raspberry Pi 5.</p>

	<p>External wires, Connectors, and Breadboard</p>	<p>Breadboard, external cables, and connectors are needed for many hardware operations such as connections between Raspberry Pi and motors, opening and closing motor connections, etc.</p>
	<p>Raspberry Pi M.2 HAT+</p>	<p>To use Raspberry Pi effectively and speed it up, we need an M.2 NVMe HAT to connect the NVMe SSD we use to Raspberry Pi. External drives can be associated with M.2 HAT, accelerating data transfer.</p>

### 3.1.3. Software Interface

The Sentinel uses a client-server architecture. The client runs on the Raspberry Pi 5 and uses Ubuntu as the operating system to support the ROS environment. The server runs on a remote computer. This architecture facilitates communication between the Raspberry Pi and the remote computer to control system movement based on camera and sensor data.

#### 3.1.3.1. Sensor and Camera Interface

The Raspberry Pi, as the client, is responsible for capturing the required data, such as camera and sensor information. The captured data will be transmitted to the server for processing. As an input, the client will capture data at a constant frequency and stream it to the server. The sensor data will be in a structured format compatible with ROS messages. The camera data will be sent to the server after being compressed to reduce the image size and facilitate faster communication.

#### 3.1.3.2. Movement Control Interface

The Sentinel supports both manual and autonomous movement. The movement control system is handled by ROS on the server. Manual movement includes two control systems: keyboard and joystick. In the keyboard system, when certain keys are pressed, the server sends the appropriate movement messages, such as 'Forward.' Similarly, in the joystick system, movement messages are sent, but the joystick also includes a coefficient that specifies how far the system will move. For autonomous movement, the server uses ROS to process the sensor and camera data, plan the best path for movement, and send the appropriate movement message to the client to move.

#### 3.1.3.3. ROS Node Architecture

Both server and client will run on separate ROS nodes that interact with each other through publisher/subscriber architecture.

##### **Client:**

- **Sensor Nodes:** Collects the necessary sensor and camera data and publishes it to the ROS topic.
- **Movement Nodes:** Receive movement commands (either from manual and autonomous) and send signals to the vehicle's actuators.

##### **Server:**

- **Sensor Processing Nodes:** Process the camera and sensor data to detect obstacles, identify the vehicle environment.

- **Path Planning Nodes:** Uses the path planning algorithms to optimize the movement of the vehicle.
- **Control Nodes:** Sends movement instructions to client based on the processed data and planned path.

#### 3.1.4. Communication Interface

The remote server and the vehicle are communicating with Wi-Fi technology. They must be on the same network. They will send and receive messages with the ROS publisher-subscriber model.

The publisher-subscriber model is a way of communicating in ROS (Robot Operating System) that helps share information between nodes using messages. In this model, a broadcaster allows multiple nodes to exchange data, such as sensor readings, control signals, or other information, by grouping it under a specific topic. For example, one node can send sensor data (publish), which other nodes can receive and use (subscribe).

A subscriber is like a listener that tunes in to a specific topic. It receives the messages broadcasted on that topic and processes the data to make decisions or control devices. Topics act as communication channels with names where publishers send messages and subscribers receive them.

Messages are structured pieces of data containing fields, like numbers or text, based on the application's needs. This model is important because it keeps publishers and subscribers independent. They can work without knowing about each other, which makes the system more flexible and modular. Multiple publishers and subscribers can communicate over the same topic without interfering with each other. [4]

### 3.2. Functional Requirements

<b>Use Case Numbers</b>	SRS/4.1
<b>Use Case Name</b>	Get Movement Command
<b>Related Use Cases</b>	Set Movement Command
<b>Description</b>	Sentinel gets the movement command from Remote Computer System or User at the Remote Computer
<b>Inputs</b>	Movement Command
<b>Source</b>	Remote Computer System

<b>Precondition</b>	Connection between the Remote Computer System and Sentinel must be stable
<b>Postcondition</b>	Sentinel gets the movement command from the Remote Computer System
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. The Remote Computer System or the User at the Remote Computer System sends the movement command</li> <li>2. Sentinel gets the movement command</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Connection between the Remote Computer System and Sentinel might be lost, Sentinel stops until the connection is re-established

<b>Use Case Numbers</b>	SRS/4.1
<b>Use Case Name</b>	Change Movement Mode
<b>Related Use Cases</b>	Keyboard Movement, Joystick Movement, Set Movement Command
<b>Description</b>	In order for the Sentinel to move autonomously or manually, its movement mode must be changed.
<b>Inputs</b>	Movement Mode
<b>Source</b>	Remote Computer System
<b>Precondition</b>	Connection between the Remote Computer System and Sentinel must be stable
<b>Postcondition</b>	Remote Computer System changes the Sentinel's movement mode
<b>Scenario</b>	<u>Scenario 1:</u> <ol style="list-style-type: none"> <li>1. Initially Movement Mode is set to autonomous</li> <li>2. If user at the Remote Computer wants to move the Sentinel manually, user changes the movement mode</li> <li>3. Sentinel's movement mode has been set to manual</li> </ol> <u>Scenario 2:</u> <ol style="list-style-type: none"> <li>1. Initially Movement Mode is set to manual</li> <li>2. If user at the Remote Computer wants Sentinel to move autonomously, user changes the movement mode</li> <li>3. Sentinel's movement mode has been set to autonomous</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Connection between the Remote Computer System and Sentinel might be lost, Sentinel stops until the connection is re-established

<b>Use Case Numbers</b>	SRS/4.2, SRS/4.4
<b>Use Case Name</b>	Collect Camera and Sensor Data



<b>Related Use Cases</b>	-
<b>Description</b>	System collects data from camera and sensors that are mounted on the vehicle.
<b>Inputs</b>	Camera and Sensor Data.
<b>Source</b>	Camera and Sensor hardware.
<b>Precondition</b>	Camera and sensors must be properly mounted on vehicle hardware
<b>Postcondition</b>	The vehicle system acquire camera and sensor data
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. Camera and sensors starts to work</li> <li>2. Vehicle system collects the data</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	If there is a hardware problem between the camera or sensor hardware and the Raspberry Pi, data acquisition stops. In this situation hardware problem should be fixed.

<b>Use Case Numbers</b>	SRS/4.1, SRS/4.2, SRS/4.3, SRS/4.4, SRS/4.5
<b>Use Case Name</b>	Publish Data to ROS Node
<b>Related Use Cases</b>	-
<b>Description</b>	System publishes collected camera and sensor data to a ROS node.
<b>Inputs</b>	Camera and Sensor Data.
<b>Source</b>	Vehicle System
<b>Precondition</b>	Camera and sensor data must be acquired.
<b>Postcondition</b>	Camera and sensor data published from ROS Node
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. System publishes the data to the ROS node while new data comes.</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Connection between the Remote Computer System and Sentinel might be lost. the data transfer stops until the connection is re-established between Remote Computer System and Sentinel. In this situation the connection should be re-established

<b>Use Case Numbers</b>	SRS/4.1, SRS/4.2, SRS/4.3, SRS/4.4, SRS/4.5
<b>Use Case Name</b>	Get Data From ROS Node
<b>Related Use Cases</b>	-
<b>Description</b>	The remote computer must get published data from the ROS node.
<b>Inputs</b>	-

<b>Source</b>	ROS Publisher Node
<b>Precondition</b>	The camera and sensor data must be published to the ROS node.
<b>Postcondition</b>	The remote computer acquires camera and sensor data.
<b>Scenario</b>	1. The remote computer get data from the ROS node while new data publishes
<b>Exceptional Situations &amp; Alternative Flows</b>	The connection between the Remote Computer System and Sentinel might be lost. The data transfer stops until the connection is re-established between the Remote Computer System and Sentinel. In this situation the connection should be re-established

<b>Use Case Numbers</b>	SRS/4.2
<b>Use Case Name</b>	Stream Data to Web
<b>Related Use Cases</b>	-
<b>Description</b>	Remote Computer streams data to web applications.
<b>Inputs</b>	Camera and Sensor Data.
<b>Source</b>	Remote Computer System
<b>Precondition</b>	Camera and sensors are must gathered from ROS node./
<b>Postcondition</b>	The web applications acquires the camera and sensor data
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. UDP sockets are opened between remote computers and web applications.</li> <li>2. Remote computer sends the camera and sensor data through sockets.</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Connection between the Remote Computer Web Application might be lost. The data transfer stops until the connection is re-established between Remote Computer System and Web. In this situation the connection should be re-established.

<b>Use Case Numbers</b>	SRS/4.2
<b>Use Case Name</b>	Render Page
<b>Related Use Cases</b>	-
<b>Description</b>	Web application renders the page for showing new data.

<b>Inputs</b>	Latest camera and sensor data.
<b>Source</b>	UDP Socket between remote computer and web application.
<b>Precondition</b>	Camera and sensor data are must be sent from remote computer to web application
<b>Postcondition</b>	New data can be seen from the web page.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. Web application acquires data from sockets.</li> <li>2. Renders the page for showing the last data.</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	The web application may stop working and in this case the latest data cannot be seen. In this situation web server should be restarted.

<b>Use Case Numbers</b>	SRS/4.1, SRS/4.3
<b>Use Case Name</b>	Keyboard Movement
<b>Related Use Cases</b>	Get Movement Command, Change Movement Mode
<b>Description</b>	User sends movement data to the server with a keyboard.
<b>Inputs</b>	Keystroke
<b>Source</b>	User
<b>Precondition</b>	Server must be on and connected to the working car.
<b>Postcondition</b>	Sentinel will move into the desired direction.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. User will press one of the movement keys on keyboard (w,a,s,d)</li> <li>2. Server will recognize pressed key</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Connection between the Remote Computer System and Sentinel might be lost, Sentinel stops until the connection is re-established

<b>Use Case Numbers</b>	SRS/4.1, SRS/4.3
<b>Use Case Name</b>	Joystick Movement
<b>Related Use Cases</b>	Get Movement Command, Change Movement Mode
<b>Description</b>	User sends movement data to the server with a joystick.
<b>Inputs</b>	Joystick Movements
<b>Source</b>	User

<b>Precondition</b>	Server must be on and connected to the working car.
<b>Postcondition</b>	Sentinel will move into the desired direction.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. User will use joystick to move car</li> <li>2. Server will recognize movement direction and speed</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Connection between the Remote Computer System and Sentinel might be lost, Sentinel stops until the connection is re-established

<b>Use Case Numbers</b>	SRS/4.1, SRS/4.3
<b>Use Case Name</b>	Set Movement Command
<b>Related Use Cases</b>	Get Movement Command, Change Movement Mode
<b>Description</b>	Sets current movement mode of the Sentinel
<b>Inputs</b>	Movement data
<b>Source</b>	User
<b>Precondition</b>	Server must be on and connected to the working car.
<b>Postcondition</b>	Sentinel will move into the desired direction.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. Server gets movement data from keyboard or joystick</li> <li>2. Server will send Sentinel related movement command</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Connection between the Remote Computer System and Sentinel might be lost, Sentinel stops until the connection is re-established

<b>Use Case Numbers</b>	SRS/4.3, SRS/4.4
<b>Use Case Name</b>	Create 2D Map
<b>Related Use Cases</b>	Publish Data to ROS Node, Get Data From ROS Node
<b>Description</b>	Creates 2D map using sensor data
<b>Inputs</b>	Sensor data
<b>Source</b>	Sentinel
<b>Precondition</b>	Server must be on and connected to the working car. Sensors must be

	working.
<b>Postcondition</b>	Accurate 2D map must be created.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. The remote server will process sensor data.</li> <li>2. Then, the remote server will create the map using the data with built-in library RTAB-Map</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	-

<b>Use Case Numbers</b>	SRS/4.3, SRS/4.4, SRS/4.5
<b>Use Case Name</b>	Create 3D Map
<b>Related Use Cases</b>	Create 2D Map, Publish Data to ROS Node, Get Data From ROS Node
<b>Description</b>	Creates 3D map using sensor data and camera data.
<b>Inputs</b>	Sensor data and camera data
<b>Source</b>	Sentinel
<b>Precondition</b>	Server must be on and connected to the working car. Sensors must be working.
<b>Postcondition</b>	Accurate 3D map must be created.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. Server will process sensor data and camera data</li> <li>2. Server will create the 3D map using the sensor data and camera data together using RTAB-Map</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	-

<b>Use Case Numbers</b>	SRS/4.3
<b>Use Case Name</b>	Process Data
<b>Related Use Cases</b>	Get Data From ROS Node
<b>Description</b>	Process the collected sensor and camera data, then synchronize and calibrate the camera data with the sensor data.
<b>Inputs</b>	Sensor and camera data

<b>Source</b>	Remote Control System
<b>Precondition</b>	Server, sensors and camera must be available.
<b>Postcondition</b>	The processed data must be valid for creating 2D and 3D maps, as well as for path planning.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. Get collected data from client</li> <li>2. Synchronize data flow between camera and sensors</li> <li>3. Calibrate camera data with other sensors</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	The incoming data from sensors and camera might be lost during transmission. The system attempts to retransmit the lost data.

<b>Use Case Numbers</b>	SRS/4.1, SRS/4.4
<b>Use Case Name</b>	Path Planning
<b>Related Use Cases</b>	Process Data, Get Data From ROS Node, Set Movement Command
<b>Description</b>	Plan the path and movement using path planning algorithms based on the collected and calibrated data.
<b>Inputs</b>	Calibrated sensor and camera data
<b>Source</b>	Remote Control System
<b>Precondition</b>	Server and sensors must be available. Camera and sensor data must be processed
<b>Postcondition</b>	Planned path must be reachable by The Sentinel
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. Runs path planning algorithms according to processed data</li> <li>2. Find best paths</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Generated moves may be invalid, or the planned map might be unreachable. The system recalculates the map and adjusts the moves accordingly.

<b>Use Case Numbers</b>	SRS/4.5
<b>Use Case Name</b>	Process the Data for Object Detection
<b>Related Use Cases</b>	Create 3D Map
<b>Description</b>	Remote Computer System process the data for Object Detection using YOLO library.
<b>Inputs</b>	Camera Data

<b>Source</b>	Remote Computer System
<b>Precondition</b>	Connection between the Remote Computer System and Sentinel must be stable. Also, 3D Map must be created.
<b>Postcondition</b>	Remote Computer System completed the Object detection on 3D Map objects.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. The server processes the camera data for object detection using YOLO.</li> <li>2. The Remote Computer System detects the object on the 3D Map.</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	The connection between the Remote Computer System and Sentinel might be lost, and If a 3D Map cannot created, Object detection will not happen.

<b>Use Case Numbers</b>	SRS/4.5
<b>Use Case Name</b>	Integrate with 3D Mapping
<b>Related Use Cases</b>	Process the Data for Object Detection, Show the Result
<b>Description</b>	After the completed Object Detection, detection objects must be integrated with the 3D Mapping to visualize better.
<b>Inputs</b>	3D Mapping, Detection Objects
<b>Source</b>	Remote Computer System
<b>Precondition</b>	Processing the Camera Data for Object detection must be completed.
<b>Postcondition</b>	The Remote Computer System completed the integrating of 3D Mapping and Detection of Objects.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. The server processes the 3D Mapping and detection objects together.</li> <li>2. The Remote Computer System integrated 3D Map and Detection object results.</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	Object detection might not be completed. If the server processes the camera data to detect the objects, this use case must wait.

<b>Use Case Numbers</b>	SRS/4.5
<b>Use Case Name</b>	Show the Result
<b>Related Use Cases</b>	Integrate with 3D Mapping
<b>Description</b>	After the completed the integration of 3D Mapping with object detection, the final result is shown to the user.

<b>Inputs</b>	3D Mapping with object detection
<b>Source</b>	Remote Computer System
<b>Precondition</b>	Integration of 3D Mapping with object detection must be completed.
<b>Postcondition</b>	The Remote Computer System shows the results for the user at the remote computer.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. The server completes the integration between 3D mapping and detection objects.</li> <li>2. After the completion, show the results to the user.</li> </ol>
<b>Exceptional Situations &amp; Alternative Flows</b>	The server cannot complete the integration because of the lack of detection objects or 3D mapping creating errors.

<b>Use Case Numbers</b>	SRS/4.5
<b>Use Case Name</b>	Choose the 3D Map with Object Detection mode
<b>Related Use Cases</b>	-
<b>Description</b>	Users at the Remote Computer System, choose the 3D Map with object detection mode to see the Map with the detection objects in the area.
<b>Inputs</b>	User action.
<b>Source</b>	User at the Remote Computer System
<b>Precondition</b>	The user should connect to the server to see the mode options.
<b>Postcondition</b>	Users can see the detection objects.
<b>Scenario</b>	<ol style="list-style-type: none"> <li>1. The user starts the using system at the remote computer</li> <li>2. Users choose the object detection mode.</li> <li>3. Users can see the detection objects</li> </ol>
<b>Exceptional Situations &amp; Alternative Flow</b>	The object detection algorithms cannot understand the objects from images properly. Therefore, user cannot see the results.



### 3.3. Non-Functional Requirements

#### 3.3.1 Performance Requirements

Performance Requirements	Description
Response Time	While real-time video is displayed and path planning algorithms are running, response time must be lower than 1 second. The system must work nearly real-time.
Error Handling	If an unknown error occurs, the system should restart itself, or the car must be returned to the starting location.
Workload	The system must handle multiple subsystems that are running at the same time. Path planning, video streaming, 2D and 3D mapping, and object classification will be done at the same time.
Scalability	Our vehicle will be controlled by a remote computer. Only one server can work on a vehicle at the same time. System shouldn't crash for any scalability issues.
Application Requirements	The remote server should have 1 GB ram available and a 4 GB graphics card to process images. The remote server CPU must be able to perform multi-processing.

#### 3.3.2 Safety Requirements

Safety Requirements	Description
Damage Preventing	The vehicle must make quick and effective manoeuvres to avoid obstacles and harmful objects to avoid getting or doing damage.
Error Reporting	Every system report must be reported in under 2 seconds. Every crucial report must be reported in under 1 second.
Switchable Modes	The system must allow its user to switch modes between autonomous and manual driving, so that the user can take over the errors.
Obstacle Detection	The system must detect obstacles with high accuracy.

### 3.3.3 Security Requirements

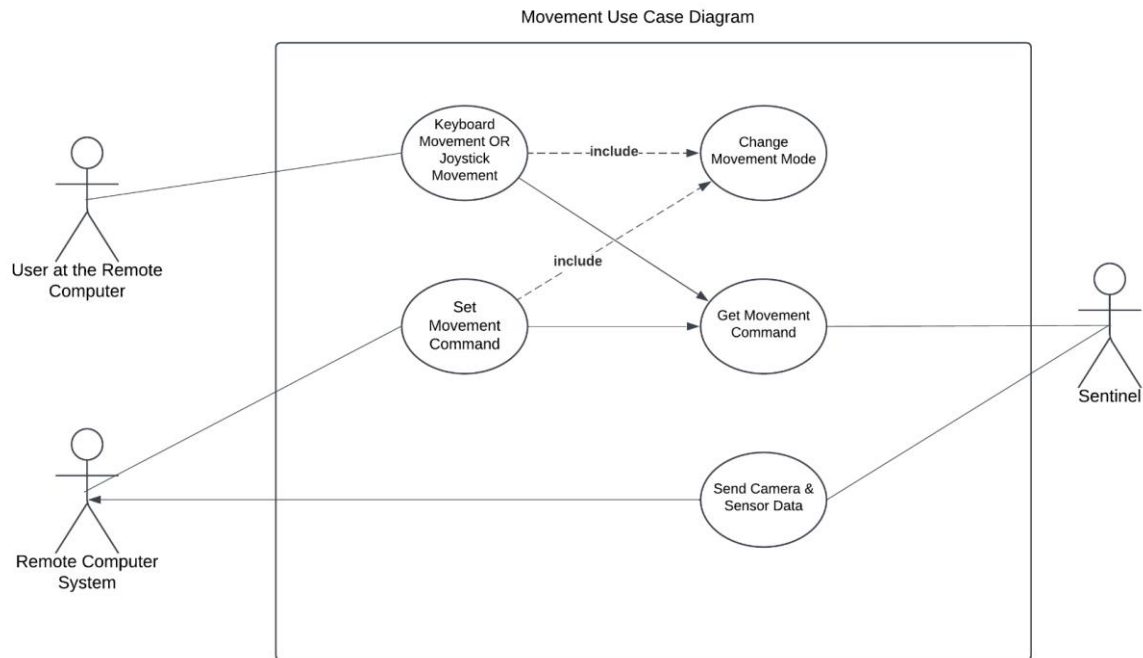
Security Requirements	Description
Wi-Fi Access	The connected Wi-Fi must be encrypted with a secure password and the must word must be hidden from the public.
Server Setting & Keys	The server settings and access keys must be protected.

### 3.3.4 Software Quality Attributes

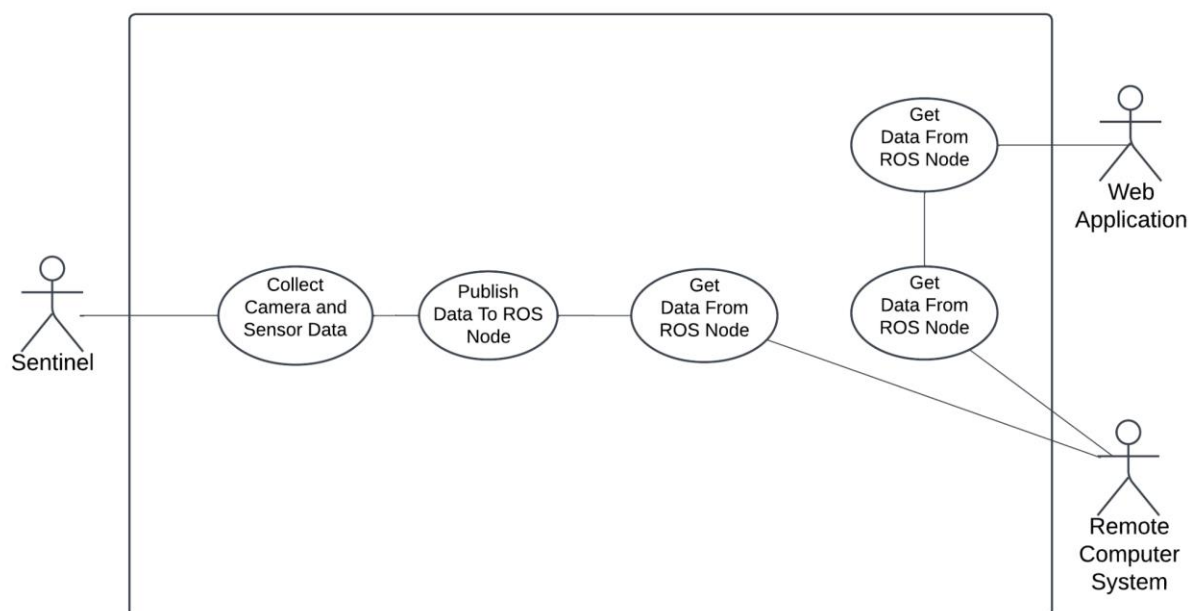
Software Quality Attributes	Description
Reliability	The system shouldn't produce errors in normal conditions and complete the task successfully.
Robustness	Under heavy environmental conditions the vehicle must be able to complete its tasks.
Portability	The system should work on Ubuntu and the server should work on Windows.
Correctness	Accuracy of predicting obstacles, classifying objects and creating maps must be over 85%.
Learnability	The usage of the system should be easy to understand for a strange user. Users must understand the concept in under 2 hours.
Maintainability	After critical error occurs, the system must restart itself without any loss of information.
Testability	The system must be tested on the different areas and different environmental conditions.
Efficiency	The system should use its resources effectively.
Usability	The vehicle should be controlled by a joystick or keyboard from a remote server.
Autonomy	The car must complete its tasks without human interruption.
Modifiability	The system may be updated, or new features will be added in the future. The system design must allow these changes.

## 4. Use Cases

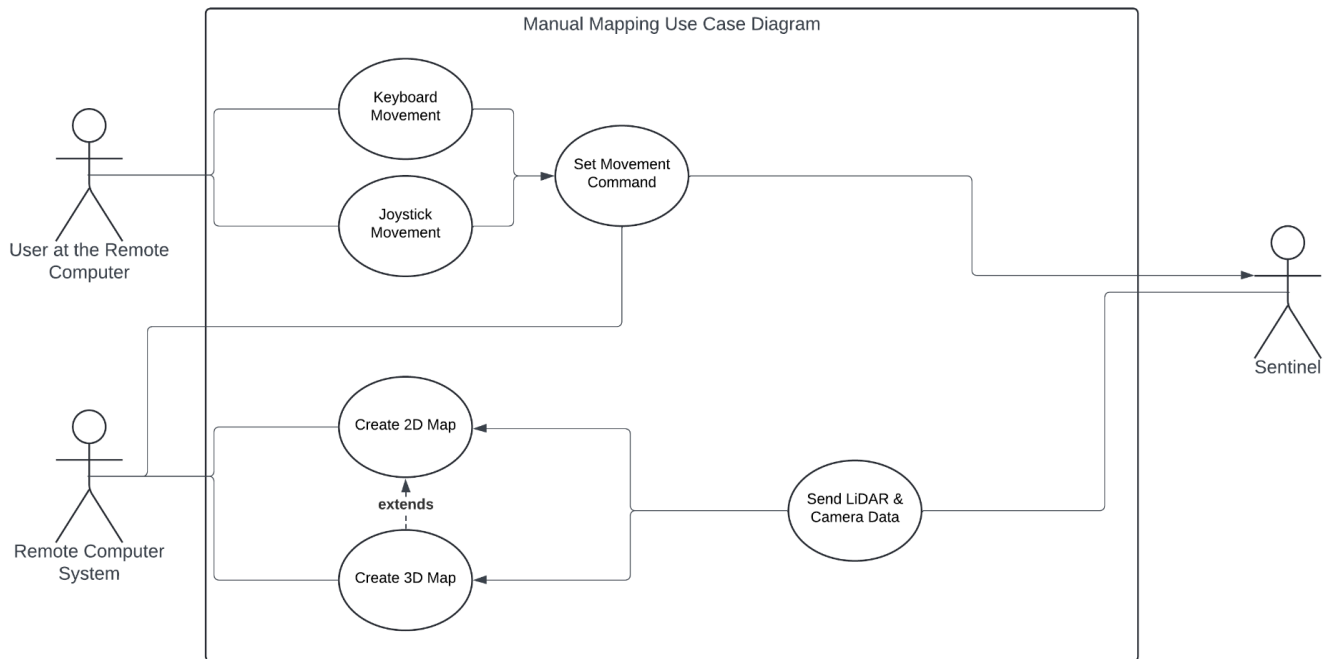
### 4.1. Movement



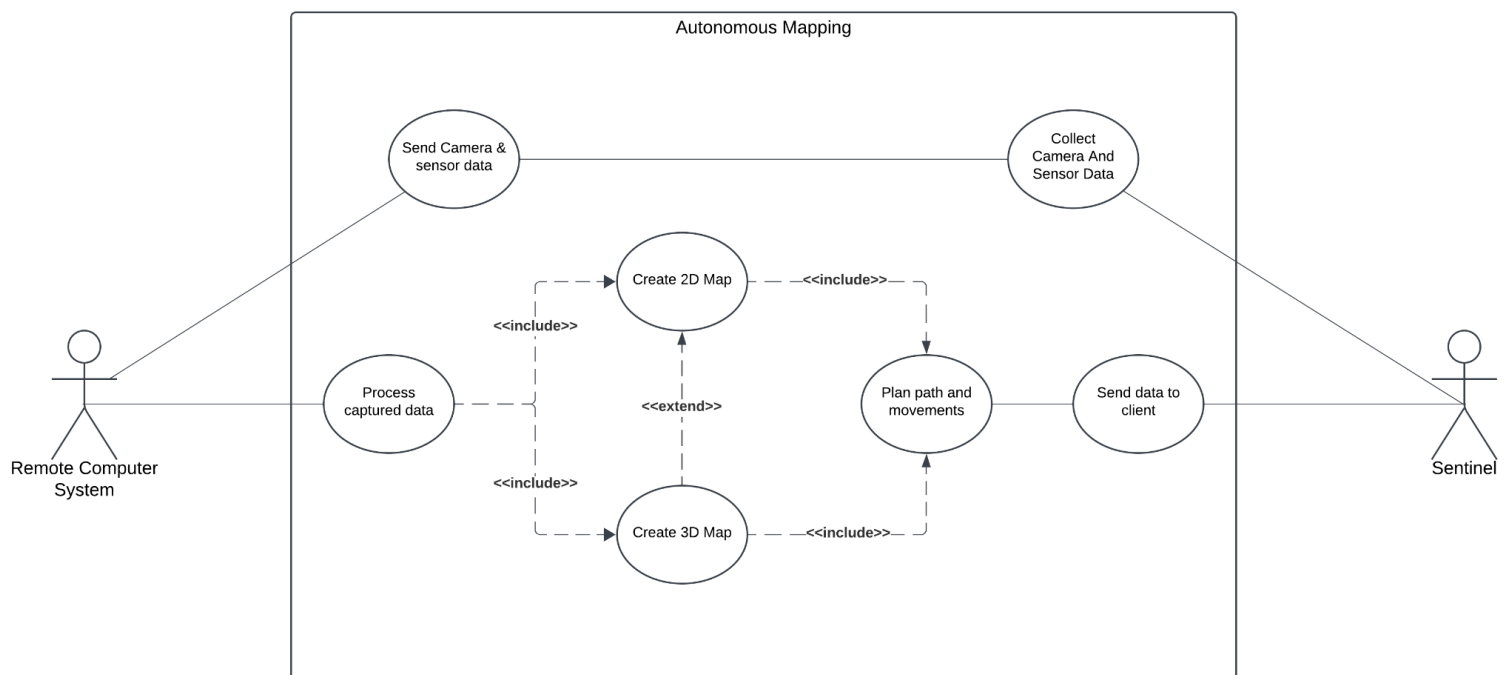
### 4.2. Data Stream



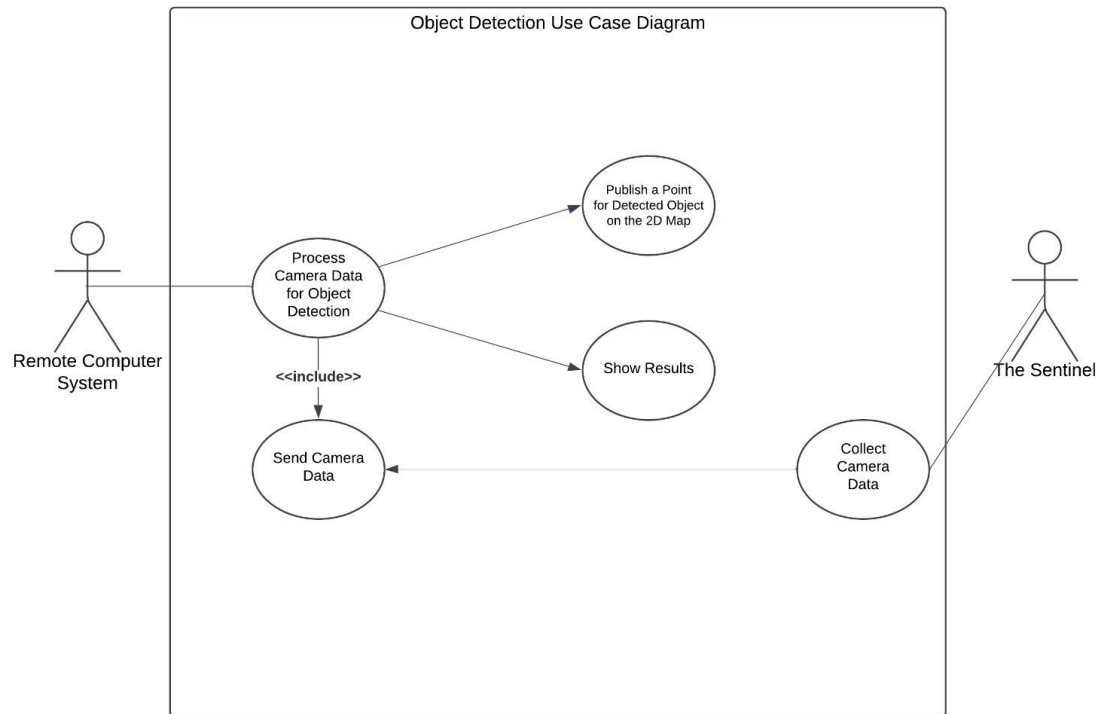
### 4.3. Manual Mapping



### 4.4. Autonomous Mapping



## 4.5. Object Detection



## References

- [1] M. Labbe, “rtabmap\_ros,” 10 December 2023. [Online]. Available: [http://wiki.ros.org/rtabmap\\_ros](http://wiki.ros.org/rtabmap_ros). [Accessed 1 December 2024].
- [2] “YOLO: Real-Time Object Detection,” [Online]. Available: <https://pjreddie.com/darknet/yolo/>. [Accessed 1 December 2024].
- [3] “rosbag,” [Online]. Available: <https://wiki.ros.org/rosbag>. [Accessed 1 December 2024].
- [4] A. Anwar, “Create Your First ROS Publisher and Subscriber Nodes,” Medium, 11 February 2021. [Online]. Available: <https://medium.com/swlh/part-3-create-your-first-ros-publisher-and-subscriber-nodes-2e833dea7598>. [Accessed 3 December 2024].