Firefighter Indoor Navigation

Product Design Specification

Client:	Mostafa Daneshgar Rahbar
GTA:	Samira Taghavi
Team Members:	Thomas Anter
	Nawar Mikha
	Jordan Shimel

Version History

Version	Implemented & Approved by	Date	Purpose
1.0	Nawar Mikha Thomas Anter Jordan Shimel	10/5/19	Initial design definition draft
1.1	Thomas Anter Nawar Mikha	10/7/19	Updated version for design doc presentation
1.2	Thomas Anter	10/17/19	Updated FR and DFD to match stretch goals
1.3	Thomas Anter Nawar Mikha Jordan Shimel	10/20/19	• Completed Class Diagram, Dependency Diagram. • Finalized formatting • Modified sequence diagram
2.0	Thomas Anter	10/22/19	Added lifetimes to sequence diagrams

Table of Contents

1 Introduction

This document is intended to be a blueprint for the implementation team. It has non-ambiguous diagrams and instructions on how to build the system outlined in the associated software requirements specification document. The design document will also be a valuable resource for the entirety of the FIN system's software lifecycle.

1.1 Purpose

The Product Design Specification document will define and illustrate the architecture and methods used to build the Firefighter Indoor Navigation system. This will provide the development team with guidelines and instructions to move from the planning phase to the development phase. The target for the design is to allow the implementation of the requirements defined in the SRS document that meet the stakeholders' needs. The intended audience is the client, Mostafa Daneshgar Rahbar, the GTA, Samira Taghavi, and the development team, Thomas Anter, Nawar Mikha, and Jordan Shimel.

1.2 Scope

The goal and scope of the Firefighter Indoor Navigation application is to build a working distributed system to create a real time digital mapping of the route taken by a firefighter and display it in a GUI within the application window. The mapping will be done by creating a point cloud that will display the environment, the current location of the firefighter in that environment, the path they took to get to that location, and the orientation of the camera along that path. The system will support up to two cameras at simultaneously. There will be two to three computing devices in the system. One to two units will be a depth-finding camera connected to an UP board. This will be referred to as the remote unit. There will also be a main console, referred to as the base unit (a laptop).

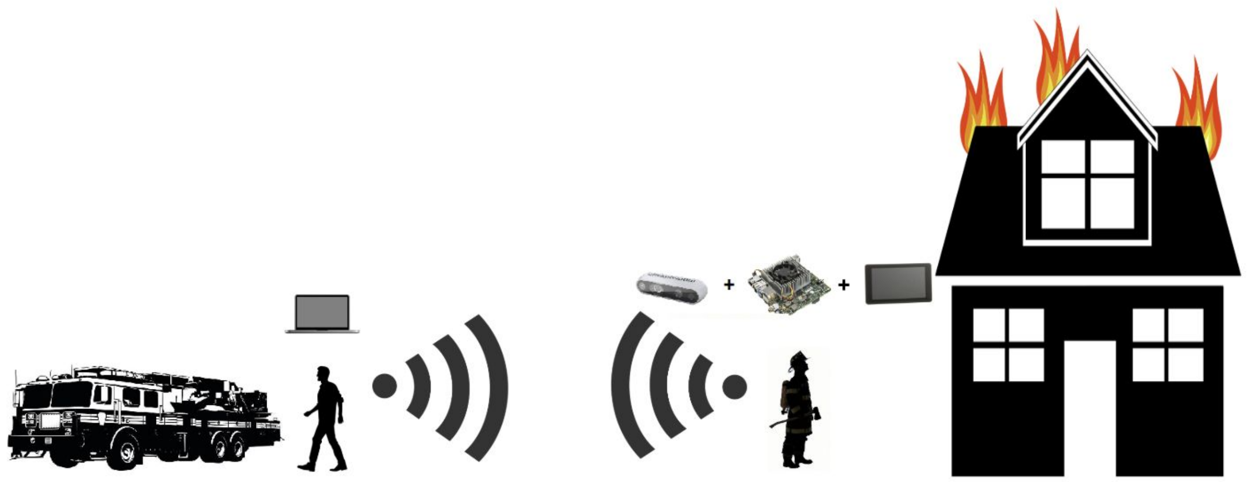
The remote unit consists of the depth finding camera and the UP board, which will be connected via USB. The camera requires a direct connection to transmit the data being received. The remote unit will have a 5 inch touch screen display for the firefighters GUI. The UP board has no built in Wi-Fi capabilities, so there will be a Wi-Fi module connected as well. The objective of the remote unit is to collect the depth information where the camera is pointing, along with a record of the camera's location and orientation, to convert that live data stream to a series of ROS messages, and to publish those ROS messages via point-to-point Wi-Fi to the base unit. The remote unit is intended to be mounted to the gear of a firefighter and is needed to track the location of that firefighter. Firefighters without a camera will not be tracked by the application.

The base unit will be a laptop. The objective of this unit is to receive a live stream of ROS messages via point-to-point Wi-Fi from the remote unit and perform SLAM on that data. The SLAM package's API's will create a point cloud from that data, and output both that point cloud, and a live video feed from that camera, on an active GUI on the laptop. The GUI will display information from one camera at a time, but will have the option of switching between camera feeds. The point cloud visualization will support panning and scanning operations. The location of the cameras, the path taken to get to that location, and the orientation of the camera on that path, will all be displayed in the point cloud.

The goal of this software is to provide firefighters with a better alternative to GPS as it pertains to tracking workers in an unfamiliar building. This will allow firefighters to refer to a digital map created in real time as they explore a building. The map will be created with the remote unit, and the operator outside, at the main unit, will be able to see the location of all firefighters in the building and be able to navigate them efficiently.

1.3 Context Diagram

Below is a diagram that illustrates how the FIN system will be used in practice. In the event of a fire, firefighters will arrive at the scene in their truck. Each firefighter entering the building will have one remote unit, that is, an UP board connected to a depth finding camera, Wi-Fi module, and a small display. These firefighters will enter the building, responding to the fire, according to their training. Remaining outside, the team leader will coordinate the firefighters inside. This is the operator, the user of the base unit, the laptop at the fire truck. This operator will see the path the firefighters are taking, view their live streams, and upon request, send a firefighter in the building the location information of one of their colleagues.



Depicted on the left, the operator is stationed outside of the building, with the fire truck, using the base unit to communicate with the remote unit via a point-to-point Wi-Fi connection. On the right, the firefighter(s) inside the building enter with the UP board, depth finding camera, and display, sending and receiving data to/from the base station via the point-to-point Wi-Fi connection.

2 General Overview

Given the particular hardware of the system and the client's other requests, a set of assumptions, constraints, and standards were explored before the actual design of the product was produced. This portion of the document will discuss the guiding principles and strategies that lead to the specific design of the FIN system.

2.1 Assumptions

Both the remote unit and base unit must have Ubuntu Linux 16.04 for their OS as well as ROS Kinetic. The ROS system is what enables the direct communication between devices.

Remote unit assumptions:

- The remote unit hardware (UP board) is capable of functioning with the required hardware and software to successfully capture the required depth and motion data
- The transmission hardware is capable of providing sufficient bandwidth and latency to enable transmission of the data from the remote unit to the base unit

Base unit assumptions:

- The transmission hardware is capable of providing sufficient bandwidth and latency to enable receipt of data from at least two remote unit instances
- The base station hardware is capable of real time processing of the depth and motion data into a 3D point cloud

Software dependencies:

- DBoW2
    - Library for transforming images into bag-of-words (Requirement of ORB-SLAM2)
- Eigen3
    - Linear algebra library (Requirement of g2o)
- FFMPEG
    - Video capture and encoding (Requirement of OpenCV)
- G2o
    - Graph optimization library (Requirement of ORB-SLAM2)
- Glew
    - Extension for OpenGL (Requirement of Pangolin)
- GTK+
    - Toolkit for creating GUI (Requirement of OpenCV)

- Intel RealSense SDK 2.0
    - API for interfacing with D435 camera
- OpenCV 4.1.1
    - Computer vision library for image processing
- OpenGL
    - Graphics API (Requirement of Pangolin)
- ORB-SLAM2
    - Real-Time SLAM for camera data
- Pangolin
    - Visualization and UI (Requirement of ORB-SLAM2)
- Image View
    - ROS package for video frame extraction

2.2 Constraints

Given the real-time nature of the system, there are several constraints that will need to be accommodated to ensure proper functionality. From software compatibility to hardware limitations, critical attention was paid to make sure catastrophic design problems don't appear after the implementation process is well under way.

2.2.1 Hardware Constraints

There are many pieces of hardware in this system, each with their own specifications and limitations. The five components are: the base unit laptop, depth camera, UPboard minicomputer, the Wi-Fi adapter for the remote unit, and the touch screen display; the latter four comprising the remote unit. The features and specifications for these devices are listed below.

Range of Intel D435i camera:

- 0.11m - 10m
- Dependant on lighting conditions and camera calibrations

Range and speed of point-to-point Wi-Fi connection with TPE-N150USB Wi-Fi module:

- Data Rate: 150Mbps @ 400 GI Max
- Outdoor Distance: with 802.11n can get about 250m (about .15 miles) in good conditions

Processing Power of the Remote Unit (UP board):

- Intel® Atom™ x5-Z8350 Processor
    - Quad core
    - 64-bit architecture
    - 2MB cache
    - Up to 1.92Ghz operating frequency
- 2GB DDR3L memory
    - 1600Mhz operating frequency

Processing power of Base Unit:

- Intel® Atom™ x5-Z8350 Processor
    - Quad core
    - 64-bit architecture
    - 2MB cache
    - Up to 1.92Ghz operating frequency
- 2GB DDR3L memory
    - 1600Mhz operating frequency
- 32GB eMMC flash memory
- HDMI 1.4a
- 1x Gigabit ethernet RJ-45
- 4x USB 2.0, 1x USB 3.0 OTG
- 5V DC-in
- Measures 85.6 mm x 56.5 mm

Display Specifications of Remote Unit:

- 5.0" TFT LCD Module
- Resolution:800x400
- Touch Screen Type: Resistive LCD driver IC: ILI9486L
- Refresh rate: 60Hz
- LCD Size: 121.11mm*77.93mm
- Micro USB: get 5V power (not needed when connected by GPIO)

- GPIO (13*2): get 5V power & touch function
- HDMI interface: for HDMI transmission

2.2.2 Network Constraints

The Wi-Fi adapter is the limiting factor for data transfer between the base and remote units. The device in use here as listed in the previous section, has a maximum transfer rate of only 150 Mbps, therefore the depth data that is being sent to the base unit will need to be less than this number. The tentative plan is to run the camera at a frame rate of 30 fps with a resolution of 640x480. At 12 bits per pixel, that comes out to roughly 110 Mbps. The frame rate will be the first thing to be reduced if issues occur in this area, since there are more possible complications from adjusting the resolution.

2.2.3 Design Constraints

The scope of this project includes the use of 2 remote units. In the future, that number is likely to be expanded. Based on the clients request, the application will have a tab option to switch between a given remote unit's data stream. An option rather than multiple displays was designed, because only so many firefighter's inputs can be displayed on one screen without degrading the size quality of the output.

Another design constraint that was considered was the quickness with which the base unit operator is able to use the software. The time critical nature of this system means that we cannot have any complex feature menus or any buttons that require navigation through a deep hierarchy of buttons. If the operator has to waste valuable seconds navigating through the application that can result in disastrous results for the fire rescue mission.

2.3 Standards

The system must be able to render the video stream and point cloud map of two camera's without delays. Due to the nature of the network and the processing time required of the system, a delay of 5 seconds is considered within the real-time goal of this product. As discussed in the network constraints section, the camera feed's resolution and frame rate may have to be adjusted and tested to ensure that the remote unit isn't transmitting more data than the network has bandwidth for. Otherwise, the incoming data will be distorted beyond reliable use for the base unit operator.

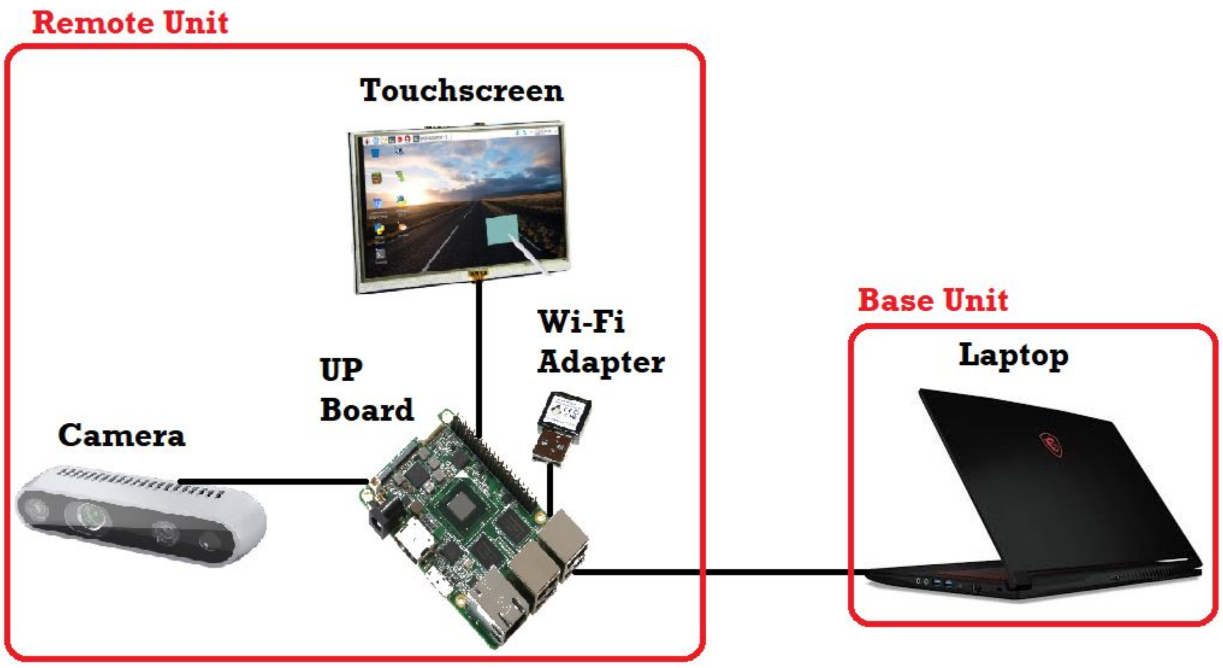
For the remote unit's map view request option, the 2D aerial view of the other remote unit's point cloud should be able to be updated at least every 2 seconds. This number is coming directly from the client.

3 Architecture Design

A significant portion of the architecture of the system was proposed by the client who is a researcher in this field. The UP board and camera were both provided by the client; the Wi-Fi adapter was chosen for its compatibility with the UP board. The software architecture has a front and back end, but unlike a standard application, it uses the ROS network to facilitate a direct connect for data transfer between the units.

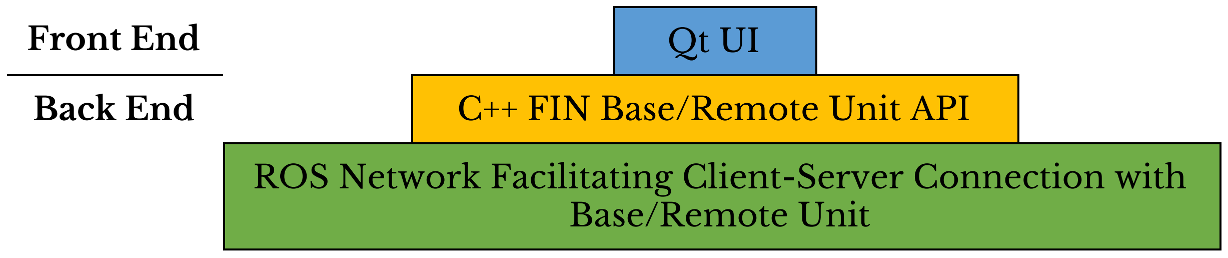
3.1 Hardware Architecture

The FIN system consists of two principal components which are the remote and base units. The remote unit UP board will capture visual data from the camera which is connected through a USB 3.0 port. It publishes this data as ROS messages through its Wi-Fi adapter (USB 3.0) over the point-to-point network to the laptop. The laptop is hosting the network; the remote units connect to the laptop's Wi-Fi signal. The laptop is subscribed to theses messages and processes the data for output in the FIN application. The Remote unit, through its touch screen (hdmi connection) will be able to request a 2D view of the point cloud map of another firefighter from the base unit; in this case the base unit is the publisher and the remote unit is the subscriber.



3.2 Software Architecture

The FIN system divides the system logic into three separate parts. The front end uses Qt to handle the GUI view. C++ is used to handle events in the UI and requests/responses to the ROS network. The ROS network manages communication with the other unit. The software architecture of the remote unit mirrors that of the base unit.


Front End
Back End
Qt UI
C++ FIN Base/Remote Unit API
ROS Network Facilitating Client-Server Connection with Base/Remote Unit

3.3 Network Architecture

The communication between the remote unit and the base unit in the FIN system uses point-to-point Wi-Fi to establish a connection and ROS to facilitate the data transfer. How the units communicate over the network is explained in section 4. **(See diagram 4.4.1)**

3.3 Performance

If a remote unit crashes, it is assumed that the system will be able to restart within 30 seconds after a request is sent from the base. This is based on a few timed tested done by the team to see how long Ubuntu takes to boot on the UP board, most tests were around 15-20 seconds.

The live video stream and point cloud map due to the nature of the ad hoc network connection and the processing required to generate the map will have a maximum delay of 5 seconds, any more than that and the system will become unreliable for real-time communication.

The 2D point cloud map request from the remote units should take less than 5 seconds to display the first image after the subscription request is sent. After that, the image should be able to refresh as fast as every 2 seconds. Both of the remote unit's will be able to make this request without putting too much strain on the system as a whole.

4 System Design

The FIN system has a specific group of requirements that were established in the associated software requirements specification document, from there a set of use-cases were formed in relation to each of these requirements. The use-cases in the next section, and their associated diagrams that follow provide a clear outline for the implementation of this system.

4.1 Use-Cases

The firefighters using this application will either be using the remote unit or the base unit. We will refer to those firefighters using the remote unit as the firefighter and those using the base unit as the operators, even though professionally they may have the same title.

FireFighter - Any person using the FIN system with the remote unit entering buildings on fire for the purpose of putting the fire out.

Operator - Any person(s) using the FIN system with the base unit working outside the location of the fire whose purpose is to navigate and communicate with the firefighters inside the structure.

4.1.1 UC-1: Video Feed Test on Remote Unit

Use Case ID:	UC-1		
Use Case Name:	Video Feed Test on Remote Unit		
Created By:	Nawar Mikha	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	12/5/19
Actors:	Firefighter		
Trigger:	Test button is selected		
Preconditions:	Remote GUI is displaying the main menu		
Postconditions:	If camera is receiving data, remote unit GUI will display that video feed. If the camera is not receiving data, the remote unit GUI will display an error message.		
Normal Flow:	1. Firefighter selects “Test” button 2. mainWindow object responds to click and calls RS2 API 3. RS2 API accesses camera data 4. RS2 API returns video feed to mainWindow object 5. mainWindow object displays video feed in GUI to		

	firefighter
Exceptions:	When the test button is selected, if the camera isn't connected, a message will be sent to the user indicating such.
Alternative Flows:	3. RS2 API cannot connect to camera 4. RS2 API has exception thrown to mainWindow 5. mainWindow object displays error message to firefighter about lack of camera connection
Frequency of Use:	Every instance in which the firefighter intends to test the camera connection. Presumably once per use.
Assumptions:	1. Remote unit executable is running 2. Camera is plugged into UP board

4.1.2 UC-2: Publishing Messages from Video to Topic

Use Case ID:	UC-2		
Use Case Name:	Publishing Messages from Video to Topic		
Created By:	Jordan Shimel	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	12/5/19
Actors:	Firefighter		
Trigger:	Firefighter presses the "Send" button on the remote unit GUI.		
Preconditions:	Remote unit GUI is displaying the main menu.		
Postconditions:	If the remote unit node is a publisher to that topic, a notification will tell the user that they are now publishing. If the remote unit node is not a publisher to the topic, a notification will tell the user there was an error.		
Normal Flow:	1. Firefighter presses "Publish" button on the GUI 2. mainWindow object responds to click and calls		

	RS2 API 3. RS2 API accesses camera data 4. RS2 API returns video feed and calls videoCapture object 5. videoCapture object converts raw frames to openCV images 6. Cv_bridge object converts the openCV images to ROS messages 7. ROS publisher node on remote unit publishes ROS image messages to topic 8. mainWindow object displays confirmation message to firefighter
Exceptions:	If the messages aren't being sent a test will catch the error and output an alert.
Alternative Flows:	7. ROS node is unable to publish to topic 8. mainWindow object displays error message to firefighter
Frequency of Use:	Every instance in which the firefighter intends to send camera data to the base unit. Presumably once per use.
Assumptions:	1. Remote unit and base unit executables running 2. Camera data is being received by the remote unit

4.1.3 UC-3: Subscribe to Messages from the Video Topic

Use Case ID:	UC-3		
Use Case Name:	Subscribe to Messages from the Video Topic		
Created By:	Nawar Mikha	**Last Updated By:**	Nawar Mikha
Date Created:	12/5/19	**Last Revision Date:**	12/5/19
Actors:	Firefighter		
Trigger:	Launching base unit		
Preconditions:	Remote unit is running and publishing data		

Postconditions:	GUI is displaying incoming data
Normal Flow:	1. Remote unit is publishing data 2. Base unit is subscribed to the correct topic and processes the ROS messages 3. The data is displayed in the GUI
Exceptions:	If the messages aren't being received, the application will still launch, but will say that no key frames are being received in the video stream.
Frequency of Use:	Once per use
Assumptions:	Data is being transmitted, and both units are properly connected to the same topic and there is enough bandwidth for the data stream.

4.1.4 UC-4: Base Unit Video Stream

Use Case ID:	UC-4		
Use Case Name:	Bse Unit Video Stream		
Created By:	Thomas Anter	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	12/5/19
Actors:	Operator		
Trigger:	Video data published to topic from remote unit		
Preconditions:	• Operator is using the base unit • Firefighter is using remote unit, receiving camera data, and publishing it to the ROS topic		
Postconditions:	Live video stream is displayed on the base unit showing the captured depth points		
Normal Flow:	1. Operator launches application 2. mainWindow object displays GUI on the base unit 3. Video data is published to the topic from the		

	remote unit 4. Operator can see video stream in GUI
Exceptions:	If the messages aren't being received, the application will still launch, but will say that no key frames are being received in the video stream. If the camera is moved to quickly, it will lose track of its frame of reference and output that its off track and trying to reinitialize.
Frequency of Use:	Once per use
Assumptions:	1. Base station laptop is booted up 2. Application is launched 3. Remote unit is booted 4. Video data is received by the remote unit

4.1.5 UC-5: Base Unit Top Down 2D Sparse Point Cloud

Use Case ID:	UC-5		
Use Case Name:	Base Unit Top Down 2D Sparse Point Cloud		
Created By:	Thomas Anter	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	12/10/19
Actors:	Operator		
Trigger:	Video data published to topic from remote unit		
Preconditions:	• Operator is using the base unit • Firefighter is using remote unit, receiving camera data, and publishing it to the ROS topic		
Postconditions:	2D, top down, navigation point cloud is displayed on the base unit GUI		
Normal Flow:	1. Operator launches application 2. mainWindow object displays GUI on the base unit		

	3. Video data is published to the topic from the remote unit 4. 2D sparse point cloud is displayed on the base unit GUI and the view perspective is from above
Exceptions:	If the messages aren't being received, the application will still launch, but will say that no key frames are being received in the video stream. If the camera is moved to quickly, it will lose track of its frame of reference and output that its off track and trying to reinitialize.
Frequency of Use:	Once per use
Assumptions:	1. Base station laptop is booted up 2. Application is launched 3. Remote unit is booted 4. Video data is received by the remote unit

4.1.6 UC-6: Base Unit 3D Sparse Point Cloud

Use Case ID:	UC-		
Use Case Name:	Base Unit 3D Sparse Point Cloud		
Created By:	Thomas Anter	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	10/16/19
Actors:	Operator		
Trigger:	A point on the map is selected and dragged by the cursor		
Preconditions:	Base station GUI displays a point cloud from one perspective		
Postconditions:	Point cloud map in the GUI has changed perspectives		
Normal Flow:	1. Operator drags point cloud window with cursor 2. mainWindow object responds to cursor drag and calls pointCloudWidget object		

	3. PointCloudWidget object responds to cursor parameters and changes perspective shown of point cloud in mainWindow object 4. mainWindow object displays point cloud in different perspective to operator
Exceptions:	
Frequency of Use:	Every event where the operator intends to change point cloud map perspective. Presumably multiple times per use (10+)
Assumptions:	1. Base station executable launched. 2. Point cloud for current camera view has been generated and is displaying.

4.1.7 UC-7: Base Unit 3D Dense Point Cloud

Use Case ID:	UC-7		
Use Case Name:	Base Unit 3D Dense Point Cloud		
Created By:	Thomas Anter	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	12/10/19
Actors:	Operator		
Trigger:	Video data published to topic from remote unit		
Preconditions:	• Operator is using the base unit • Firefighter is using remote unit, receiving camera data, and publishing it to the ROS topic		
Postconditions:	3D, colored, dense point cloud is displayed on the base unit GUI		
Normal Flow:	1. Operator launches application 2. mainWindow object displays GUI on the base unit 3. Video data is published to the topic from the remote unit 4. 3D, dense, colored point cloud is displayed on the base unit GUI		

Exceptions:	
Frequency of Use:	Once per use
Assumptions:	5. Base station laptop is booted up 6. Application is launched 7. Remote unit is booted 8. Video data is received by the remote unit

4.1.8 UC-8: Remote Unit Settings

Use Case ID:	UC-8		
Use Case Name:	Remote Unit Settings		
Created By:	Thomas Anter	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	10/18/19
Actors:	Firefighter		
Trigger:	"Settings" button is selected on the remote unit		
Preconditions:	Remote unit is displaying the main menu on its GUI		
Postconditions:	Remote unit displays configuration page. Frame rate and resolution may be adjusted		
Normal Flow:	1. Firefighter presses "Settings" button 2. mainWindow object responds to the button click and displays page for settings and configurations to firefighter 3. Firefighter changes fps and resolution parameters in settings page 4. mainWindow object responds to the changed parameters in GUI and changes those parameters in the application. 5. mainWindow object displays confirmation message to the firefighter		
Frequency of Use:	Every instance in which the firefighter intends to set or change the camera configuration. Presumably once per use.		

Assumptions:	1. Remote unit executable is running 2. Camera is connected to the UP board

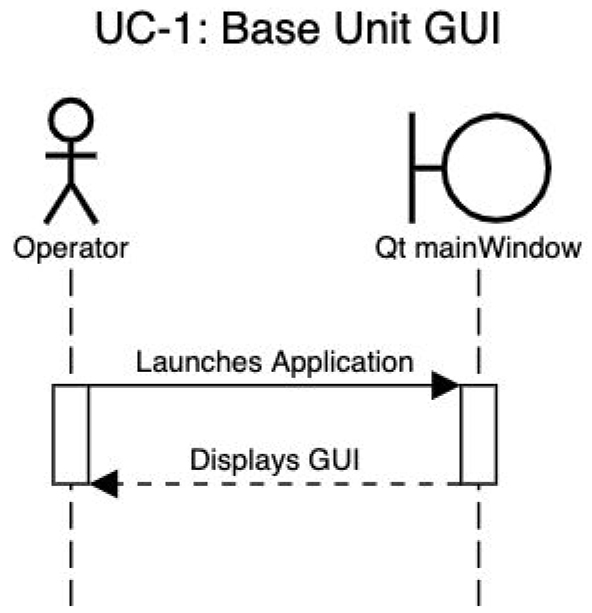
4.1.9 UC-9: Base Unit Settings

Use Case ID:	UC-9		
Use Case Name:	Base Unit Settings		
Created By:	Thomas Anter	**Last Updated By:**	Thomas Anter
Date Created:	10/15/19	**Last Revision Date:**	10/16/19
Actors:	Operator		
Trigger:	“Settings” button is selected on the base unit		
Preconditions:	Base unit is displaying the main menu on its GUI		
Postconditions:	Base unit displays configuration page. Remote unit may be restarted, point cloud to remote unit refresh rate and network settings may be adjusted		
Normal Flow:	1. Operator presses “Settings” button 2. mainWindow object responds to the button click and displays page for settings and configurations to the firefighter 3. Firefighter changes fps and resolution parameters in settings page 4. mainWindow object responds to the changed parameters in the GUI and changes those parameters in the application. 5. mainWindow object displays confirmation message to object		
Frequency of Use:	Once per use		
Assumptions:	Base unit executable is running		

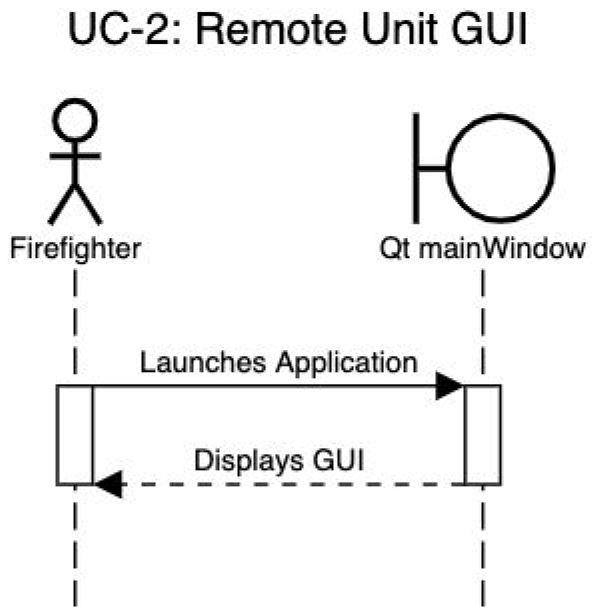
4.2 Sequence Diagrams

The following section contains the associated sequence diagrams for the use-cases that were outlined in the previous section.

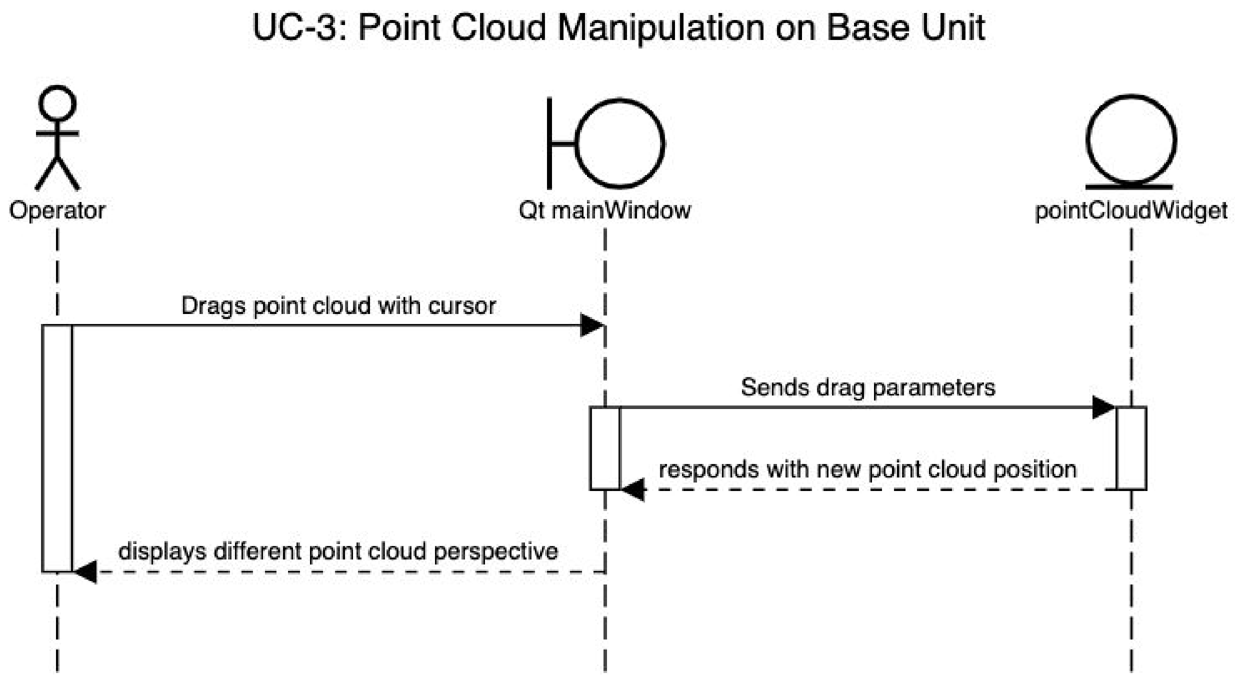
4.2.1 SD-1: Base Unit GUI



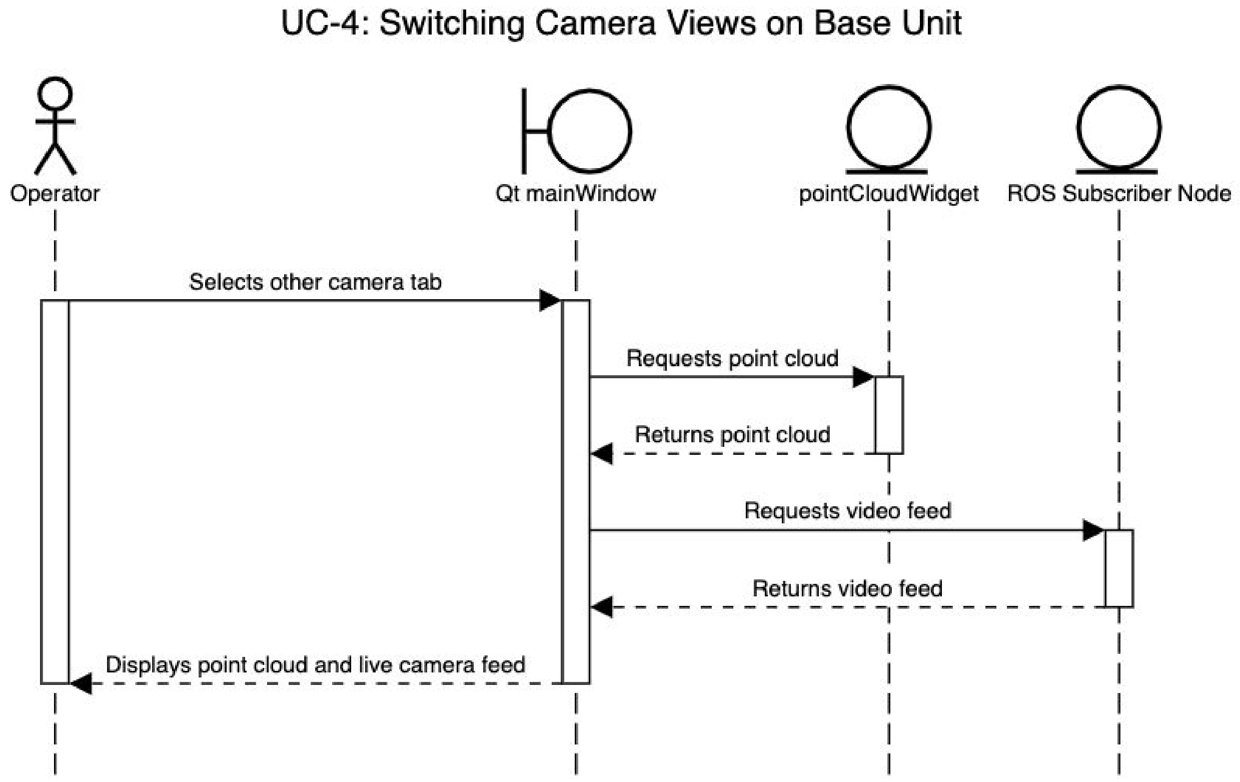
4.2.2 SD-2: Remote Unit GUI



4.2.3 SD-3: Point Cloud Manipulation on Base Unit

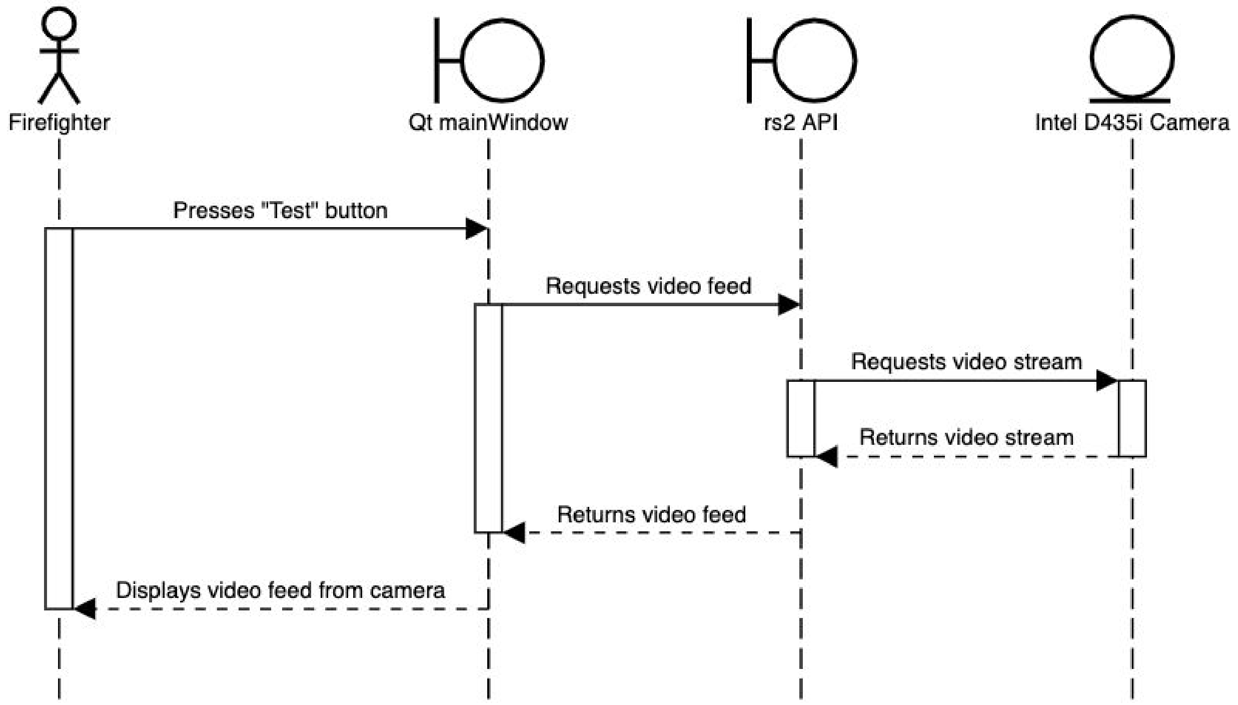

UC-3: Point Cloud Manipulation on Base Unit
Operator
Qt mainWindow
pointCloudWidget
Drags point cloud with cursor
Sends drag parameters
responds with new point cloud position
displays different point cloud perspective

4.2.4 SD-4: Switching Camera Views on Base Unit

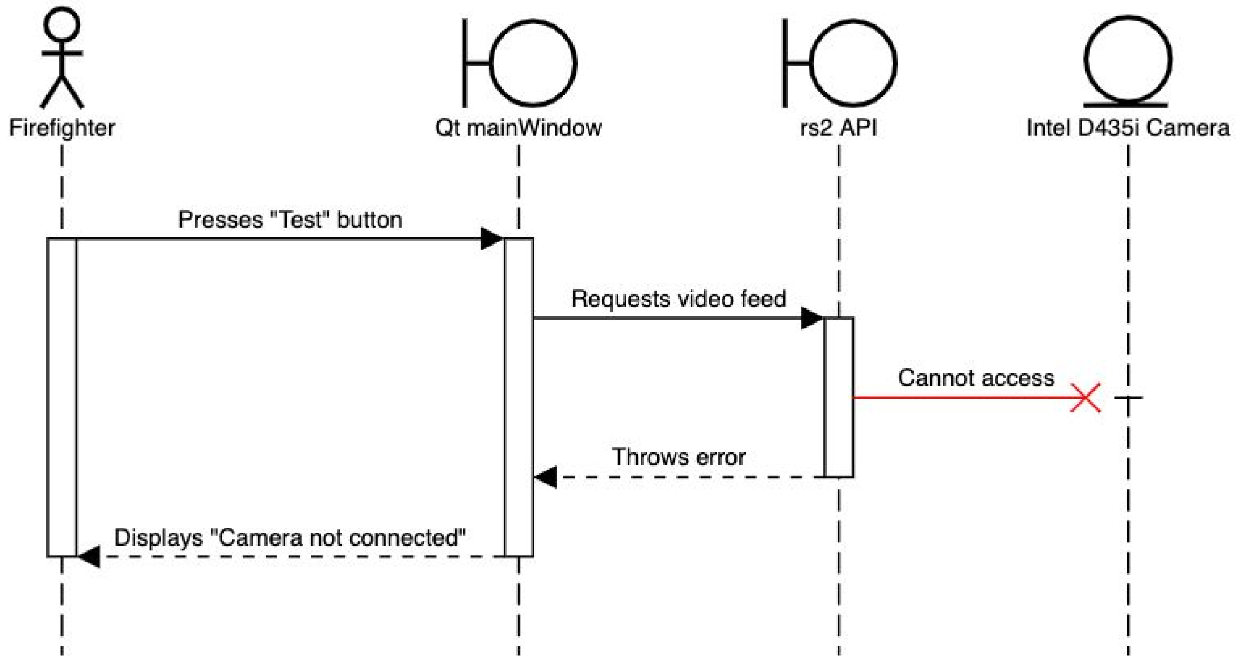


4.2.5 SD-5: Testing Video Feed on Remote Unit

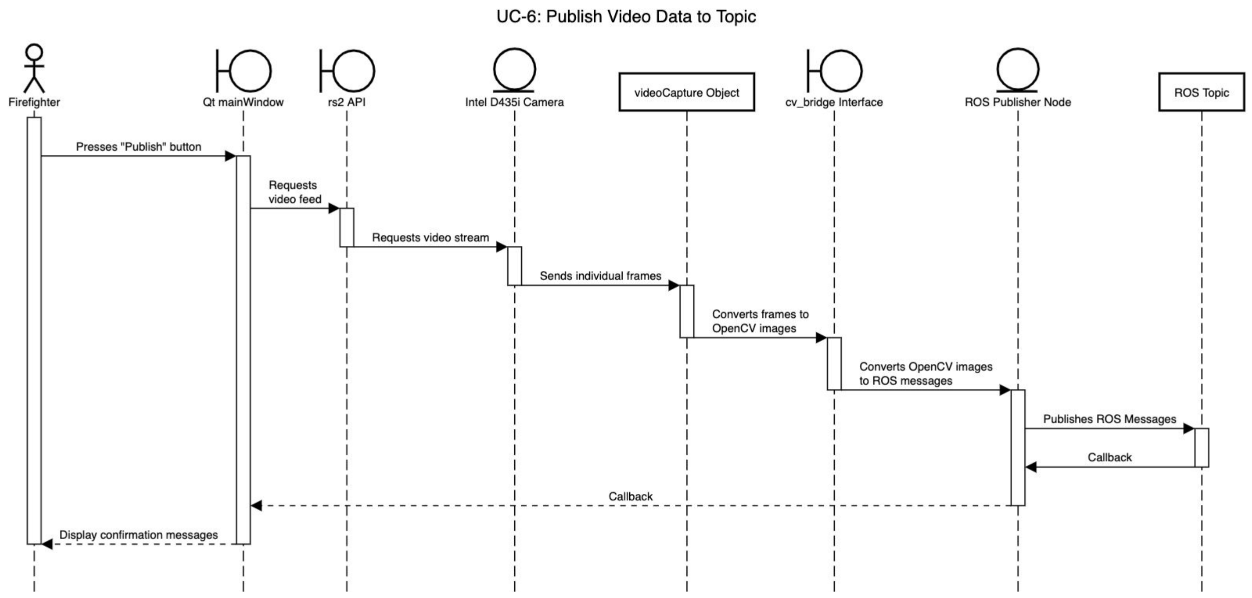
UC-5: Video Feed Test on Remote Unit

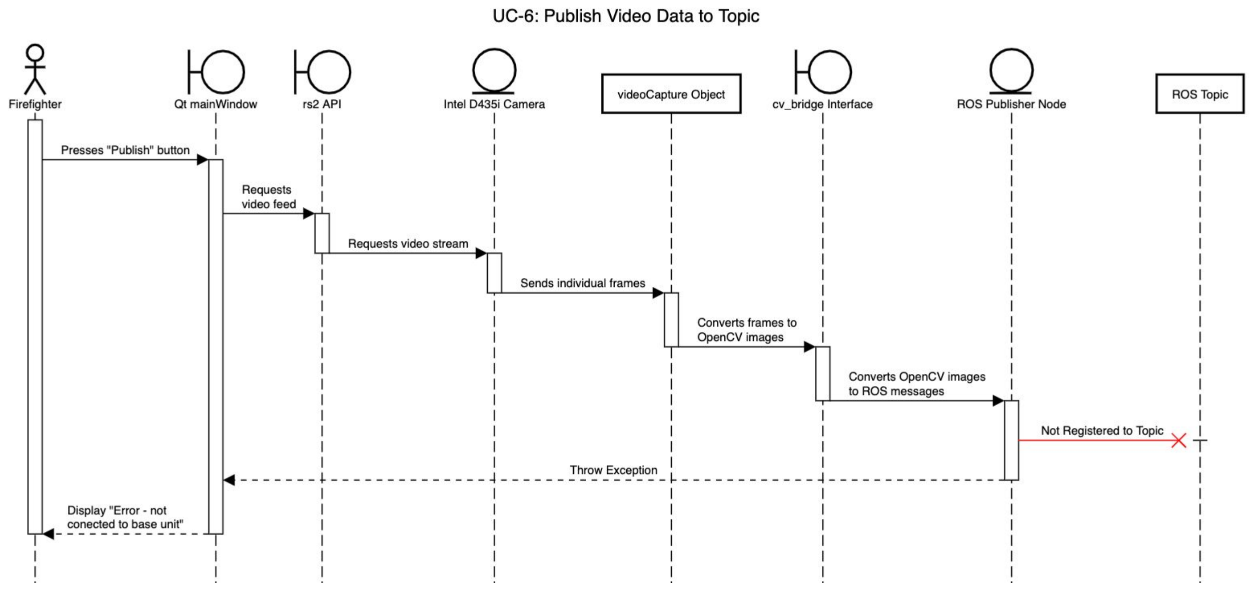


UC-4: Switching Camera Views on Base Unit



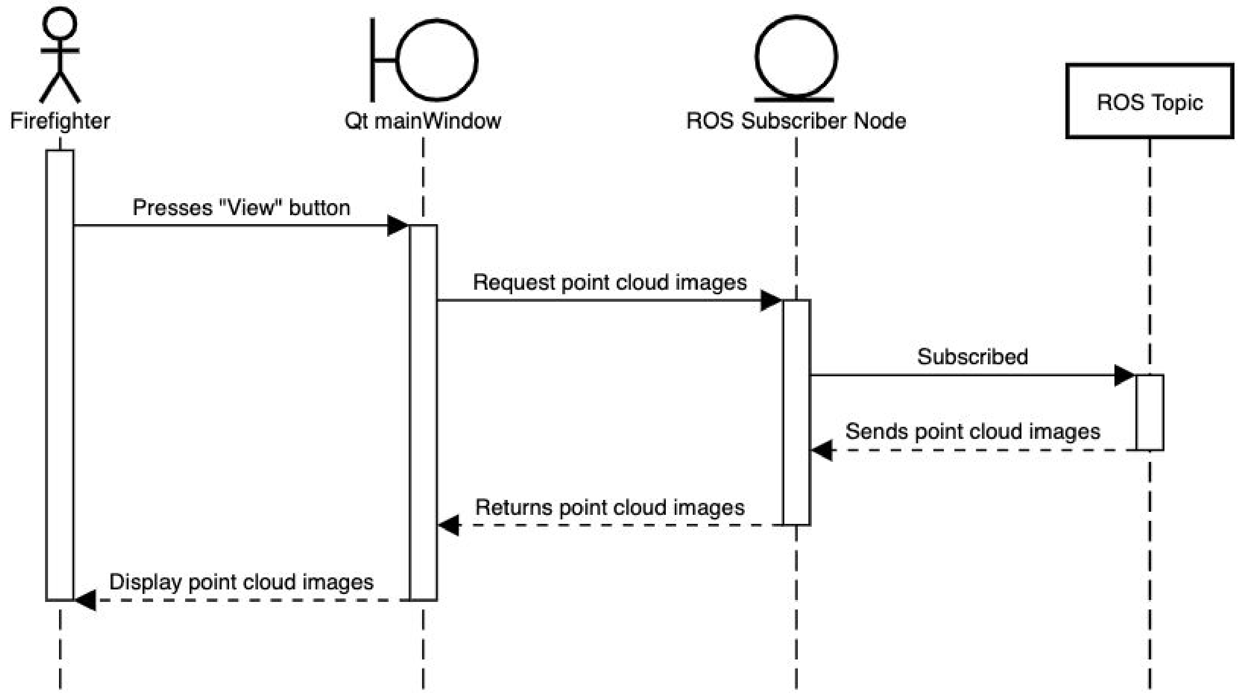
4.2.6 SD-6: Publishing Video Data to Topic



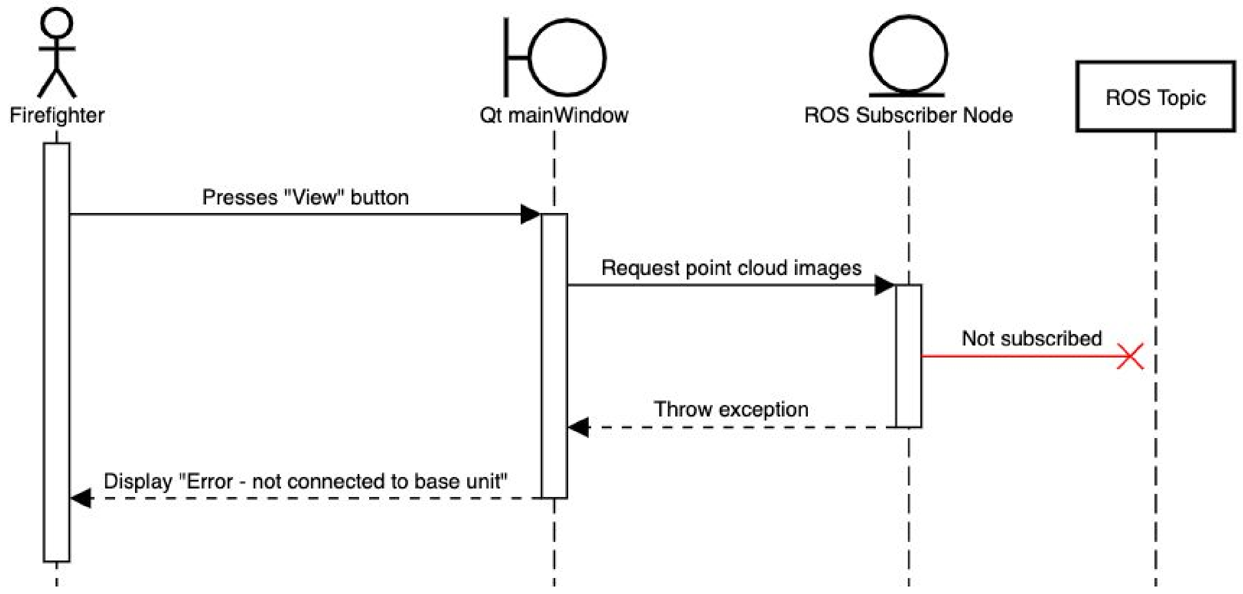


4.2.7 SD-7: Changing Camera View on Remote Unit

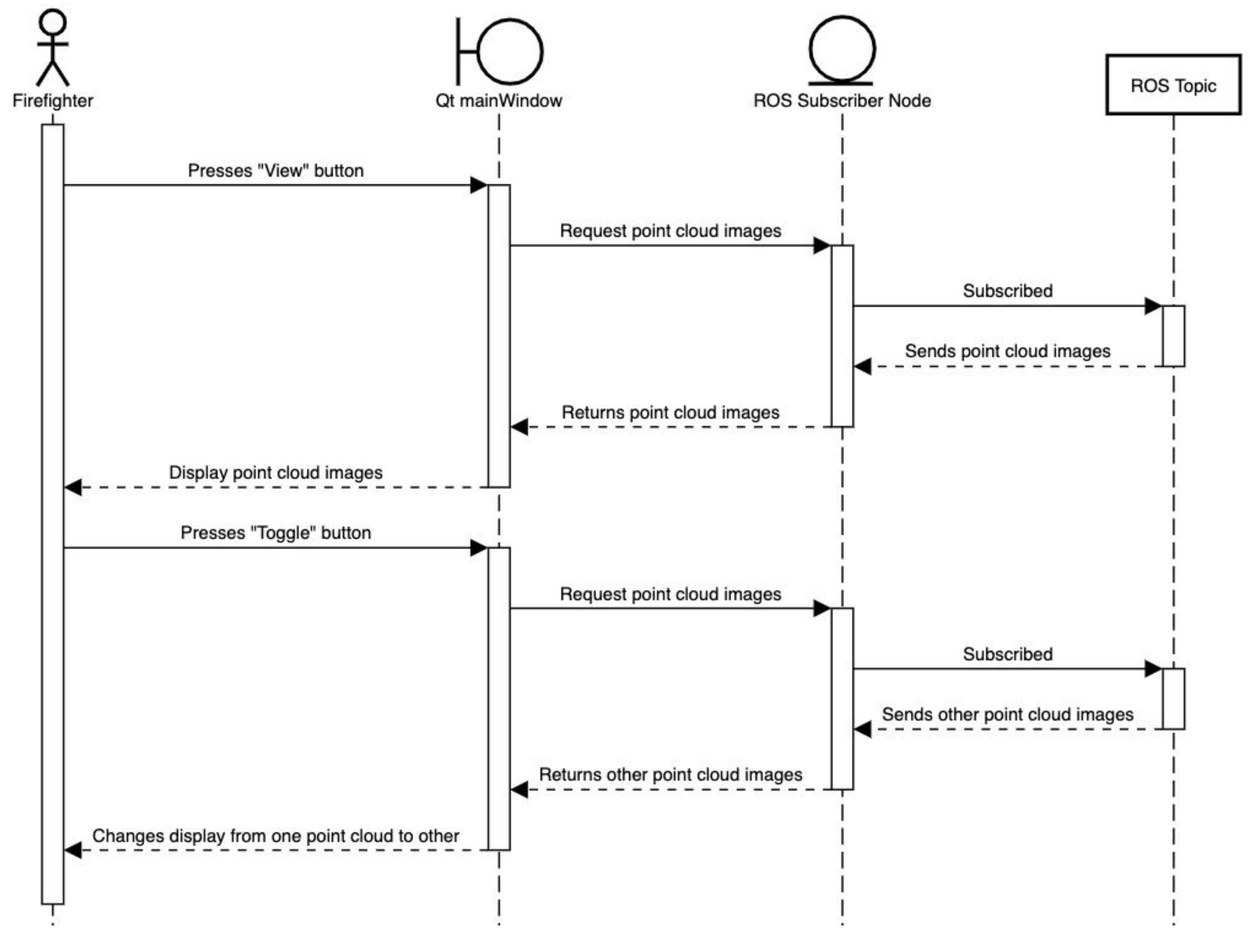
UC-7: Change Camera View on Remote Unit



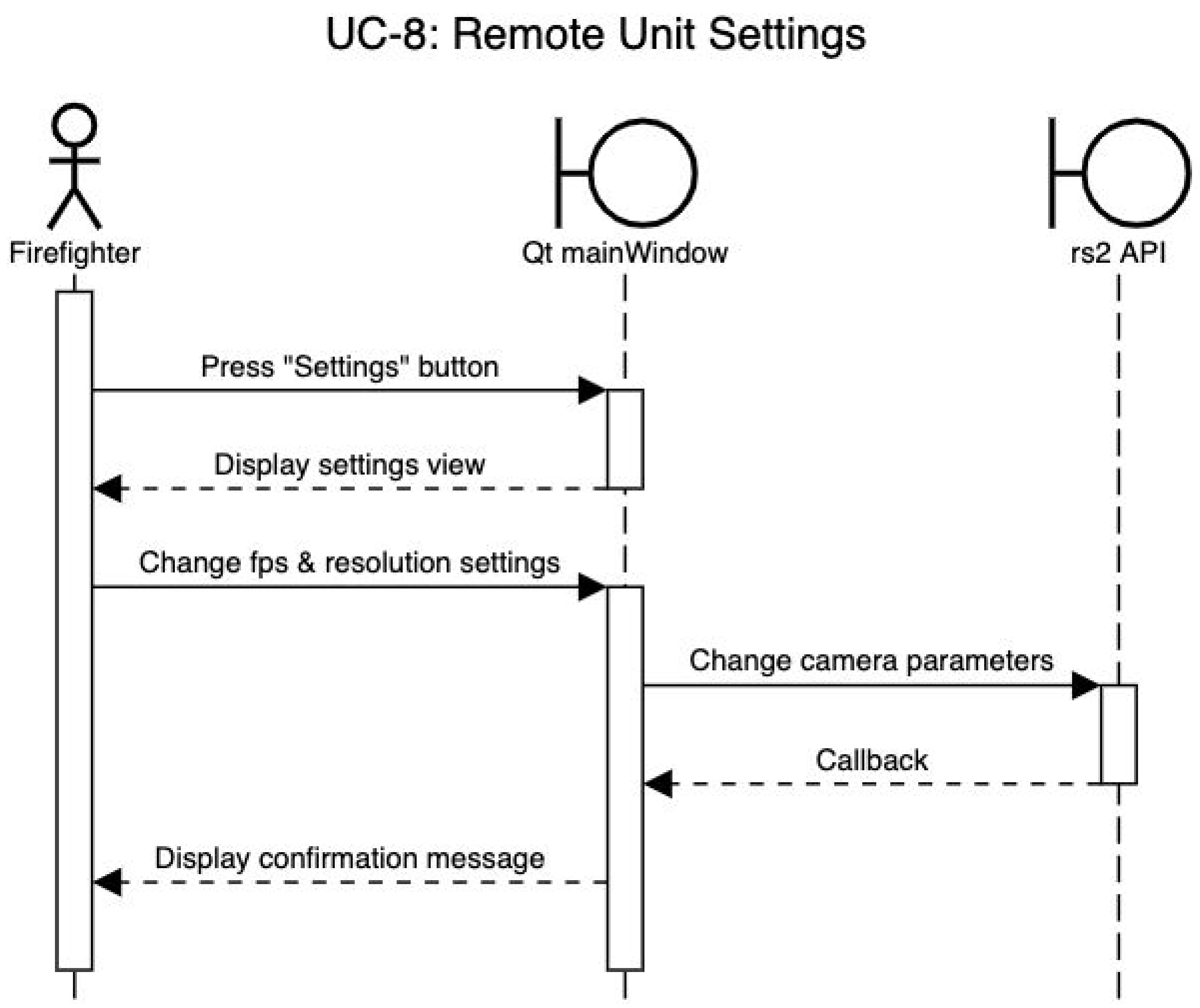
UC-7: Change Camera View on Remote Unit (Alternate Flow 1)



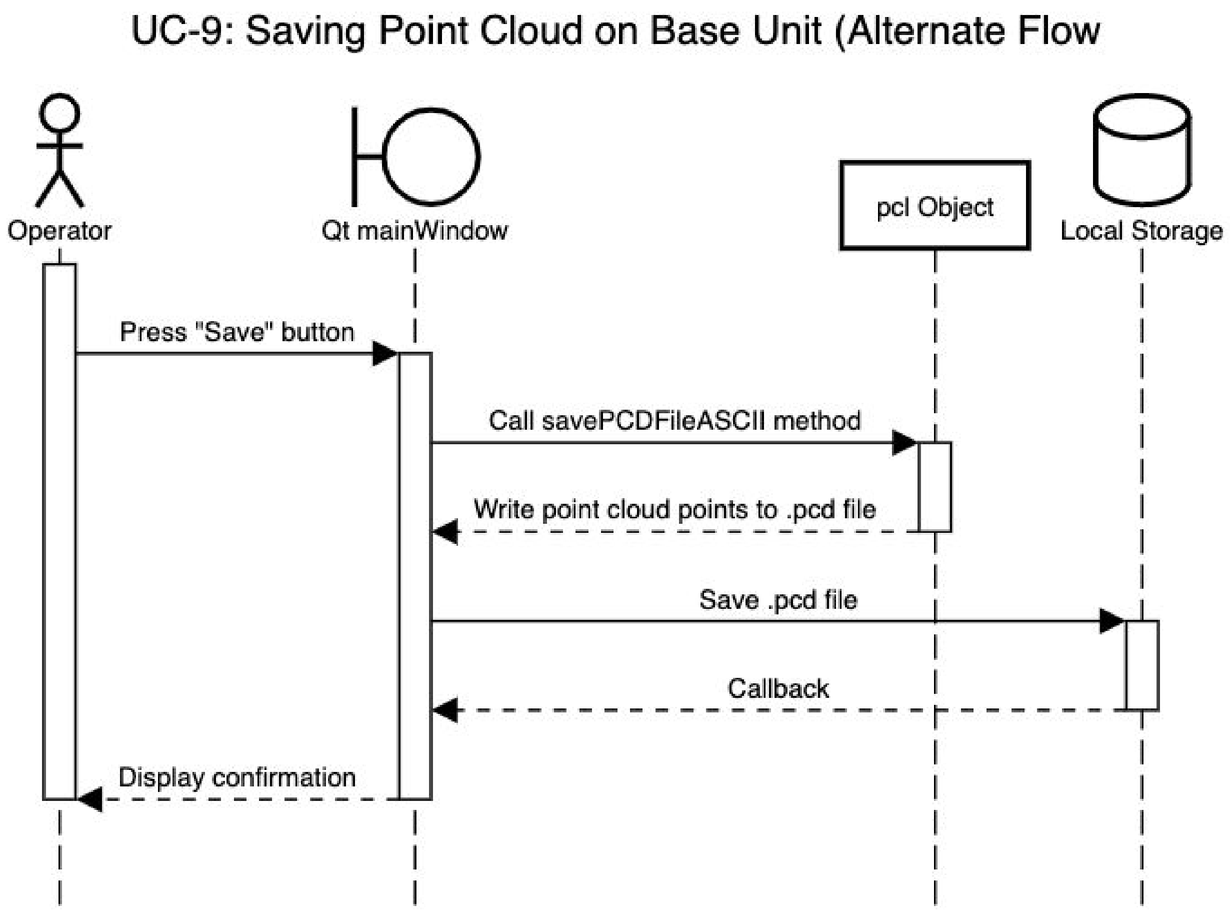
UC-7: Change Camera View on Remote Unit (Alternate Flow 2)


Firefighter
Qt mainWindow
ROS Subscriber Node
ROS Topic
Presses "View" button
Request point cloud images
Subscribed
Sends point cloud images
Returns point cloud images
Display point cloud images
Presses "Toggle" button
Request point cloud images
Subscribed
Sends other point cloud images
Returns other point cloud images
Changes display from one point cloud to other

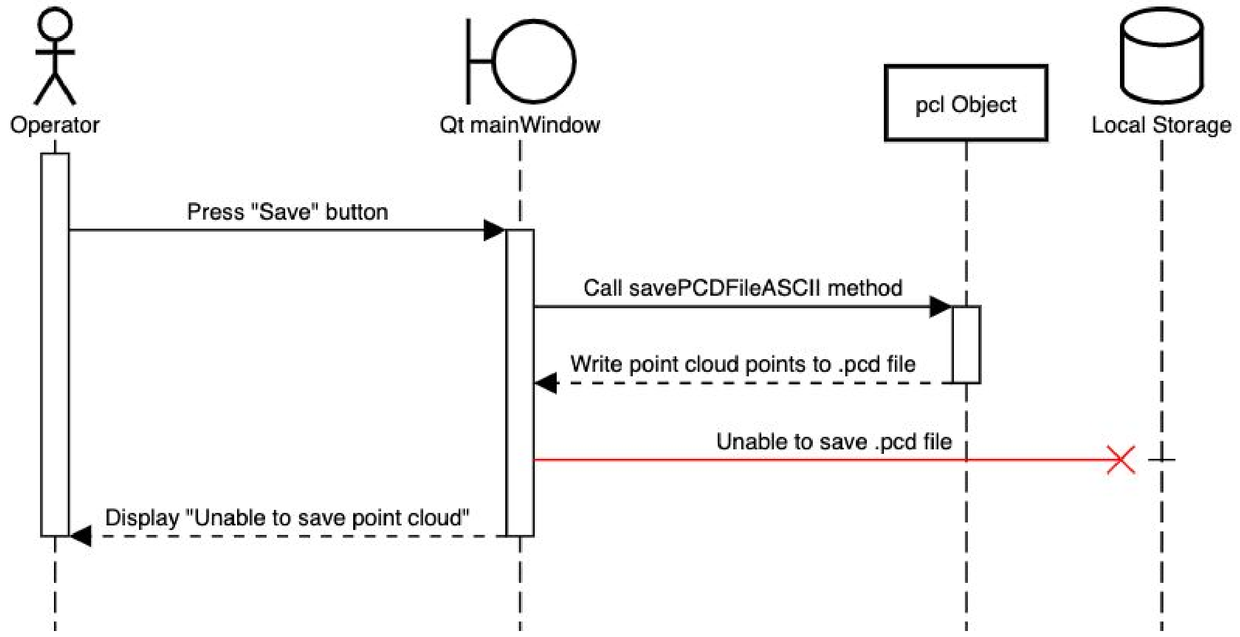
4.2.8 SD-8: Remote Unit Settings



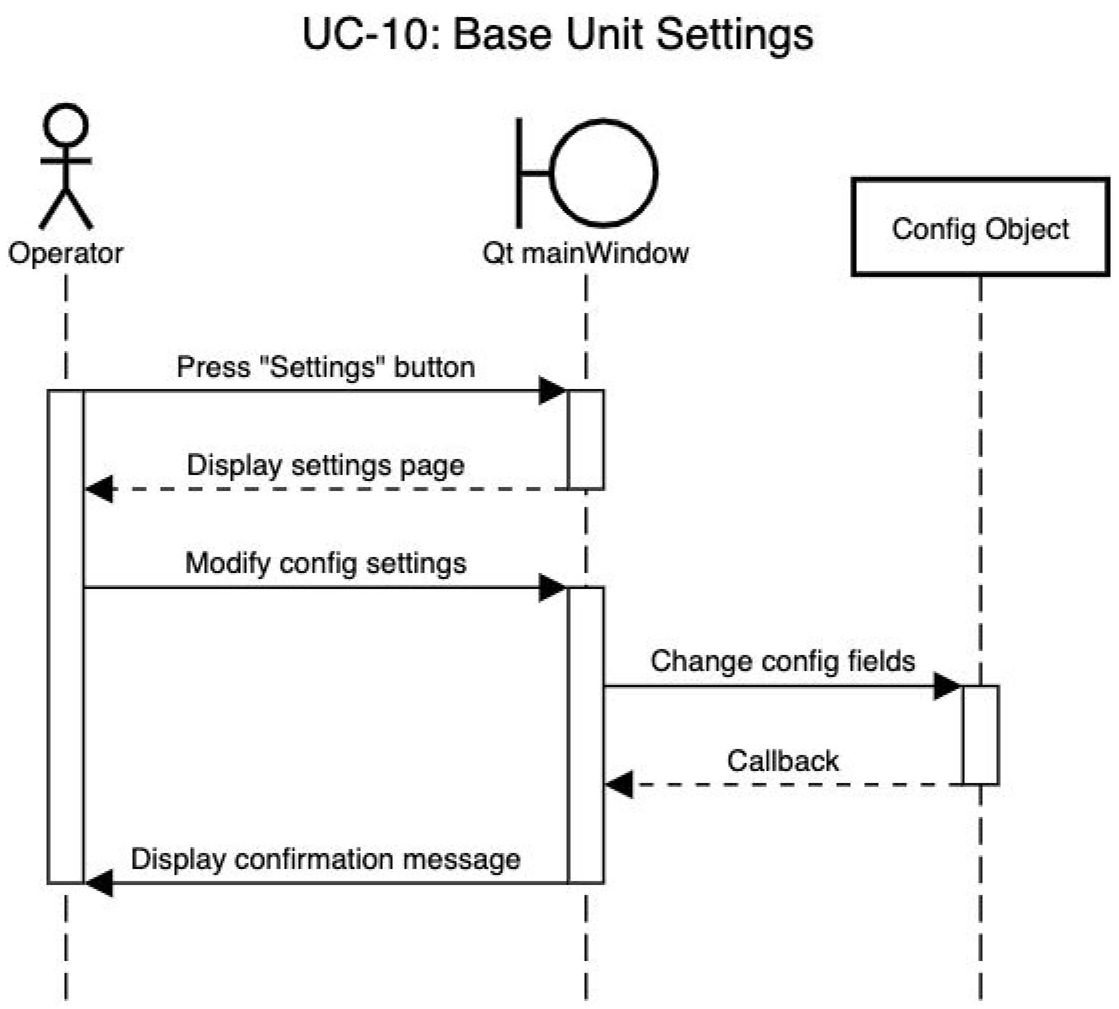
4.2.9 SD-9: Saving Point Cloud on Base Unit


UC-9: Saving Point Cloud on Base Unit (Alternate Flow
Operator
Qt mainWindow
pcl Object
Local Storage
Press "Save" button
Call savePCDFileASCII method
Write point cloud points to .pcd file
Save .pcd file
Callback
Display confirmation

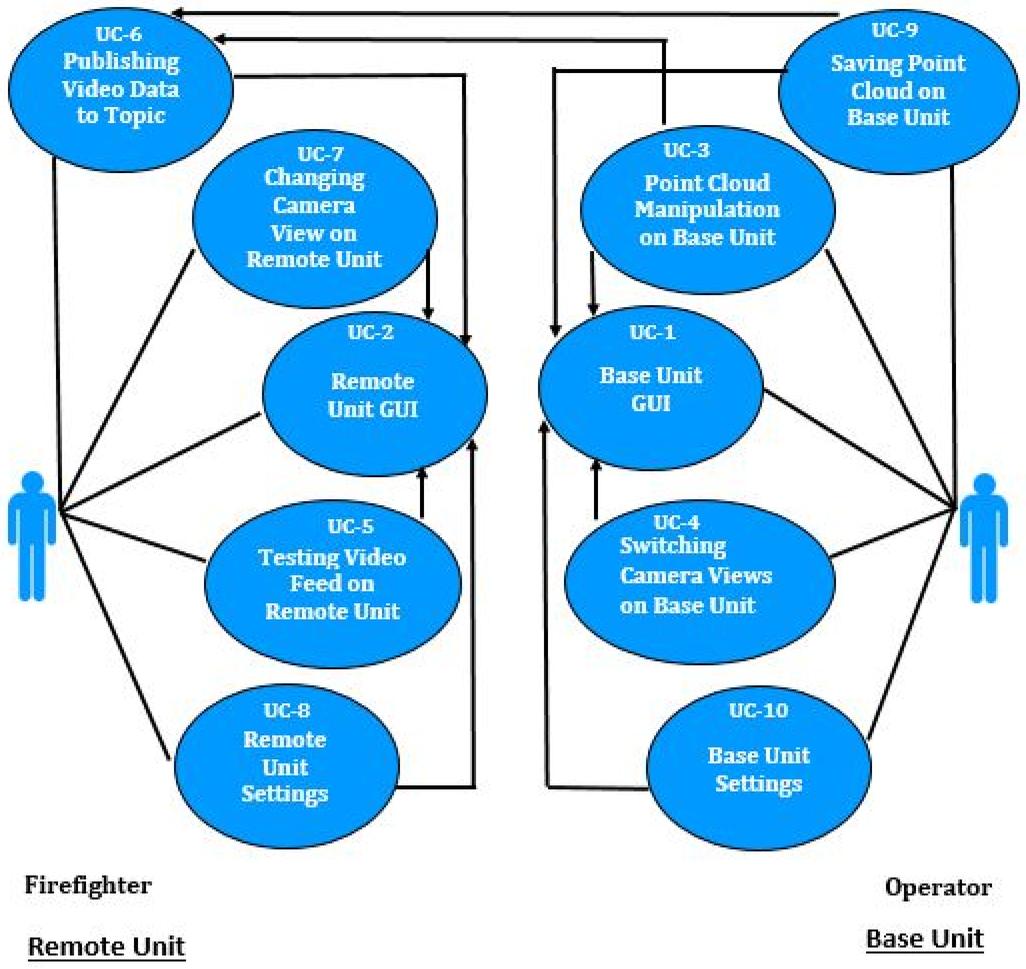
UC-9: Saving Point Cloud on Base Unit (Alternate Flow


Operator
Qt mainWindow
pcl Object
Local Storage
Press "Save" button
Call savePCDFileASCII method
Write point cloud points to .pcd file
Unable to save .pcd file
Display "Unable to save point cloud"

4.2.10 SD-10: Base Unit Settings



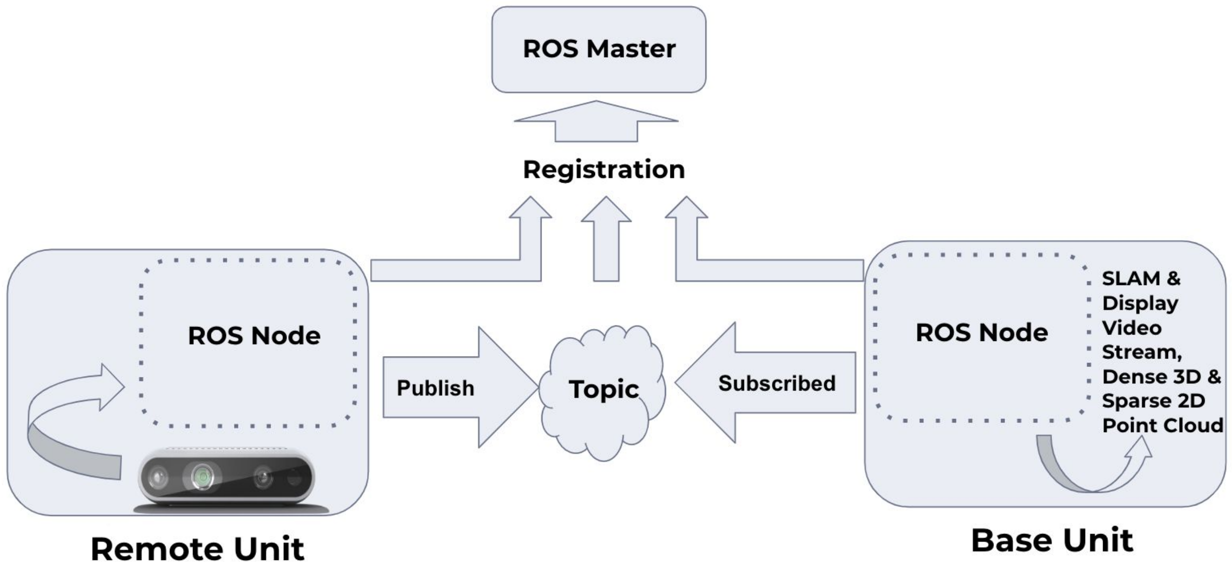
4.3 Use-Case Dependency Diagram


UC-6 Publishing Video Data to Topic
UC-7 Changing Camera View on Remote Unit
UC-2 Remote Unit GUI
UC-5 Testing Video Feed on Remote Unit
UC-8 Remote Unit Settings
UC-9 Saving Point Cloud on Base Unit
UC-3 Point Cloud Manipulation on Base Unit
UC-1 Base Unit GUI
UC-4 Switching Camera Views on Base Unit
UC-10 Base Unit Settings
Firefighter
Remote Unit
Operator
Base Unit

4.4 Data Flow Diagram

There will be three data flow diagrams. The first will examine the distributed FIN system as a whole. The second and third will be taking a deeper look into each individual unit and explain how the data flows and is manipulated. In the first two diagrams, the start-point of the flow of data will be symbolized with a picture of the Intel depth finding camera.

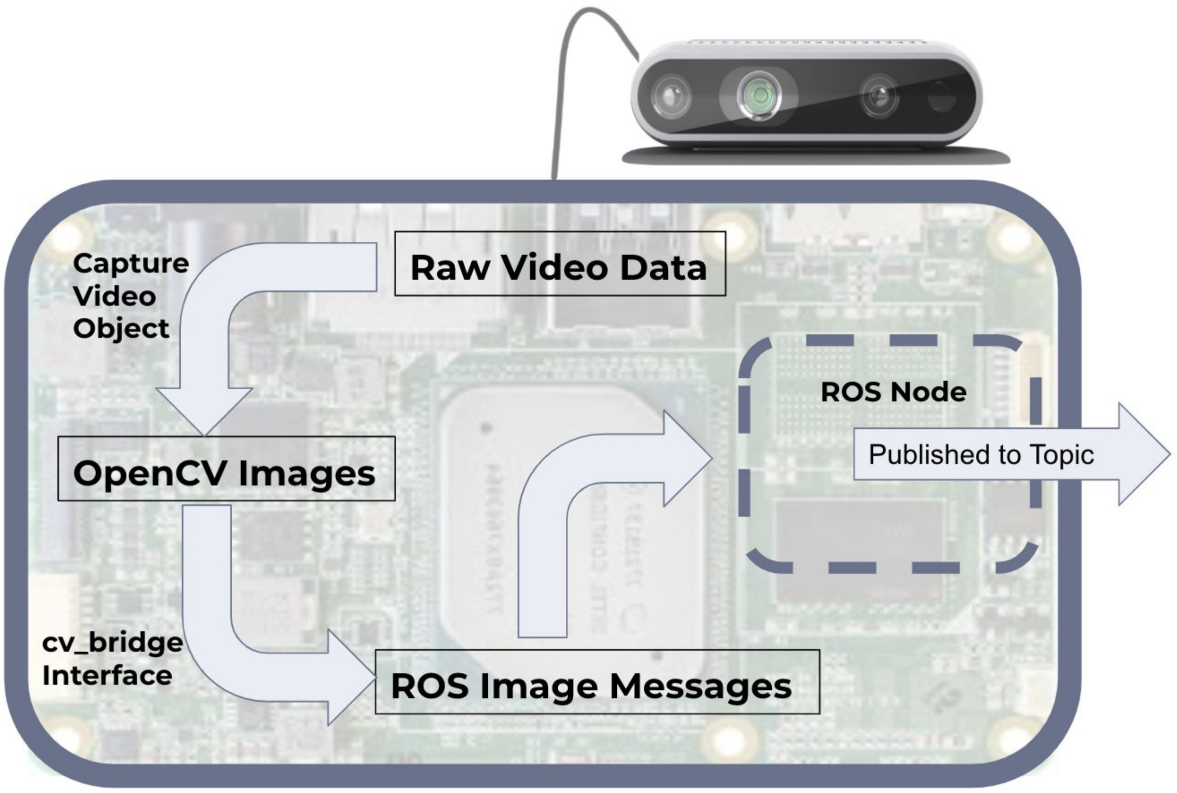
4.4.1 System Overview



This diagram depicts both the remote unit and the base unit. A ROS node is an executable that can communicate with other nodes. Each unit will run a node. A ROS message is the format in which data is sent through a ROS network. A topic is a bus that allows the flow of ROS messages from one node to another. Each node can subscribe or publish to a topic depending on the desired direction of the data flow. Each node and topic will be registered with the ROS master, which keeps track of which nodes are published or subscribed to which topics. This enables nodes to locate each other. In the remote unit, data received from the depth finding camera will be converted to ROS image messages.

Starting from the camera, the remote unit will receive the video data, convert it to ROS messages, publish it to a ROS topic. Since a node in the base unit is subscribed to that topic, it will receive those messages. This data transfer is represented the arrows interacting with the ROS topic cloud in diagram 4.4.1. In the base unit, the newly received messages are processed. It is at this point that the point cloud and live video feed will be displayed on the base unit. The processed data will end up in the form of point clouds.

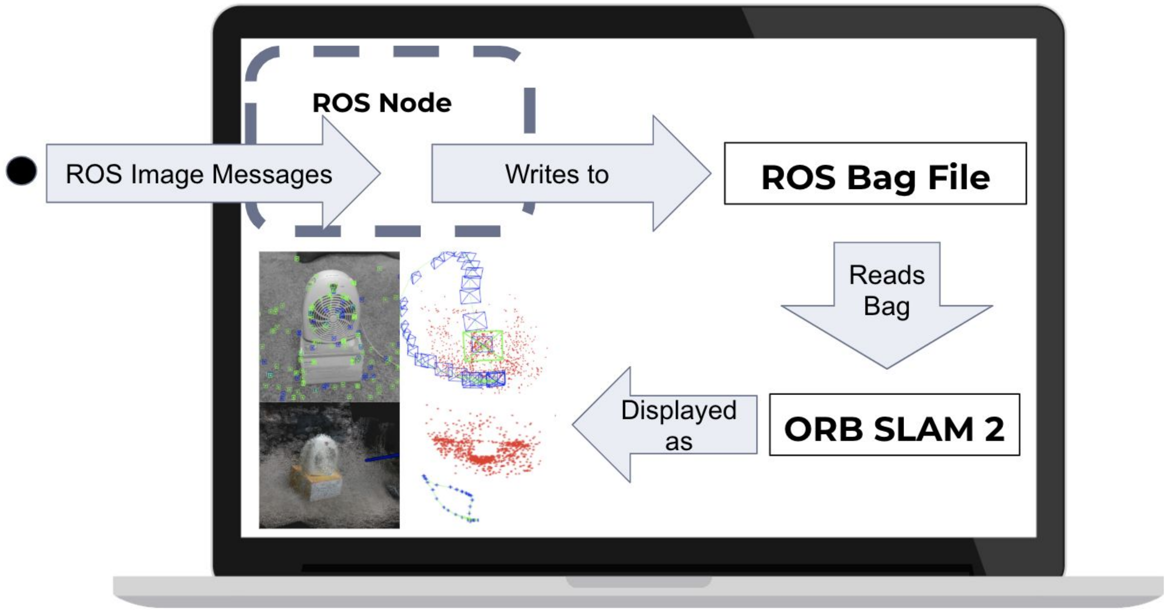
4.4.2 Remote Unit



The remote unit of the FIN system facilitates the data transfer of depth camera data; it is received as raw video data. The camera is connected to the UP board via USB. There will be a ROS node running on this unit. It publishes ROS messages to a topic to sends them to the base unit for SLAM.

Diagram 4.4.2 displays how the FIN system converts the raw video data to ROS image messages in order for them to be sent to the base unit. Two classes from OpenCV will be used in order to make this conversion. An object of class VideoCapture will process frames from the video data frame by frame and prime it as OpenCV images for the next step. A cv_bridge object acts as an interface to convert OpenCV images to ROS image messages. Once the frames have been successfully converted to ROS image messages, they will be sent sequentially using the top node running on the remote unit to a node running on the base unit.

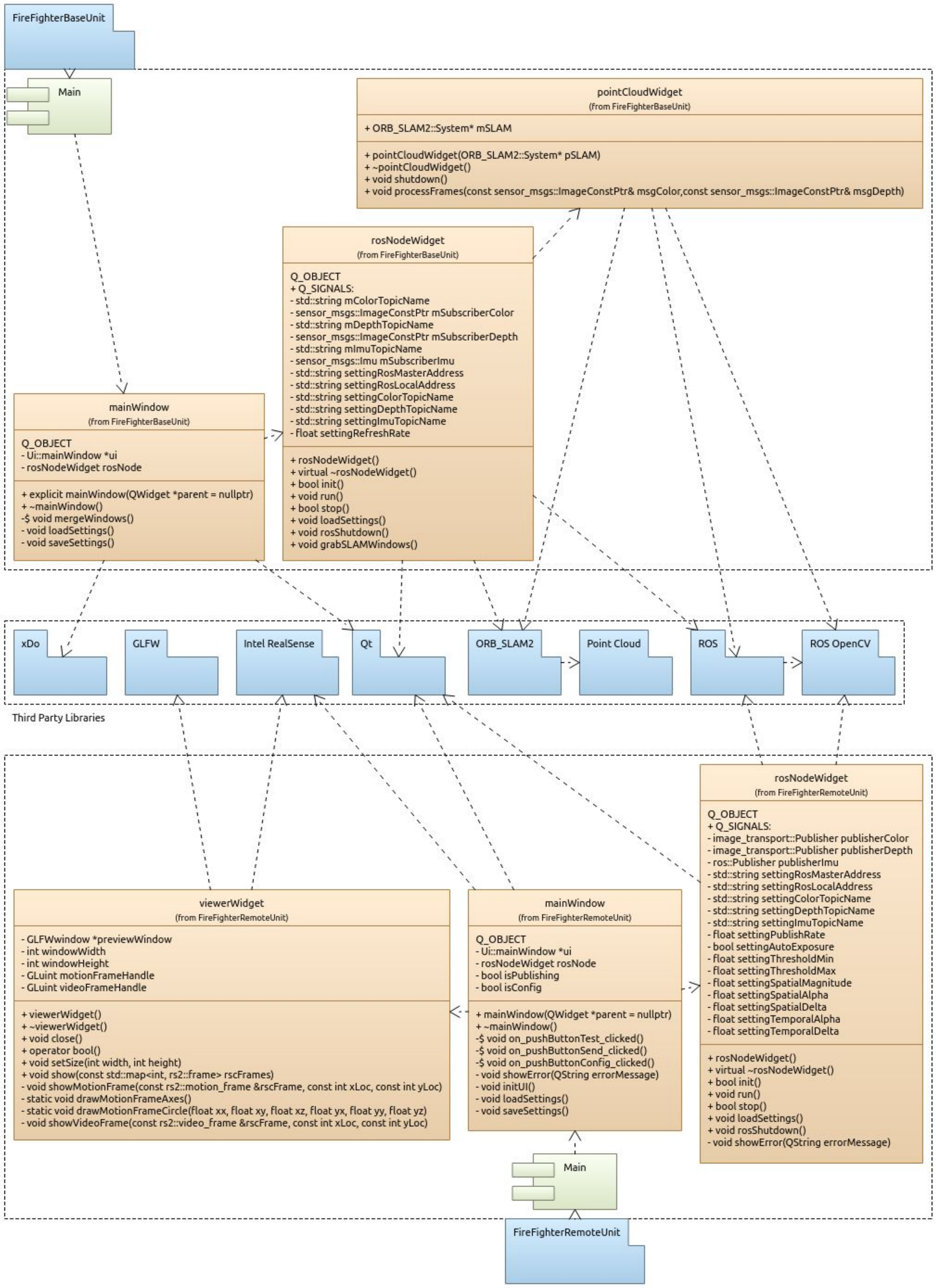
4.4.3 Base Unit



The base unit receives ROS image messages of video data, processes them with SLAM, and sends a point cloud back to the remote unit.

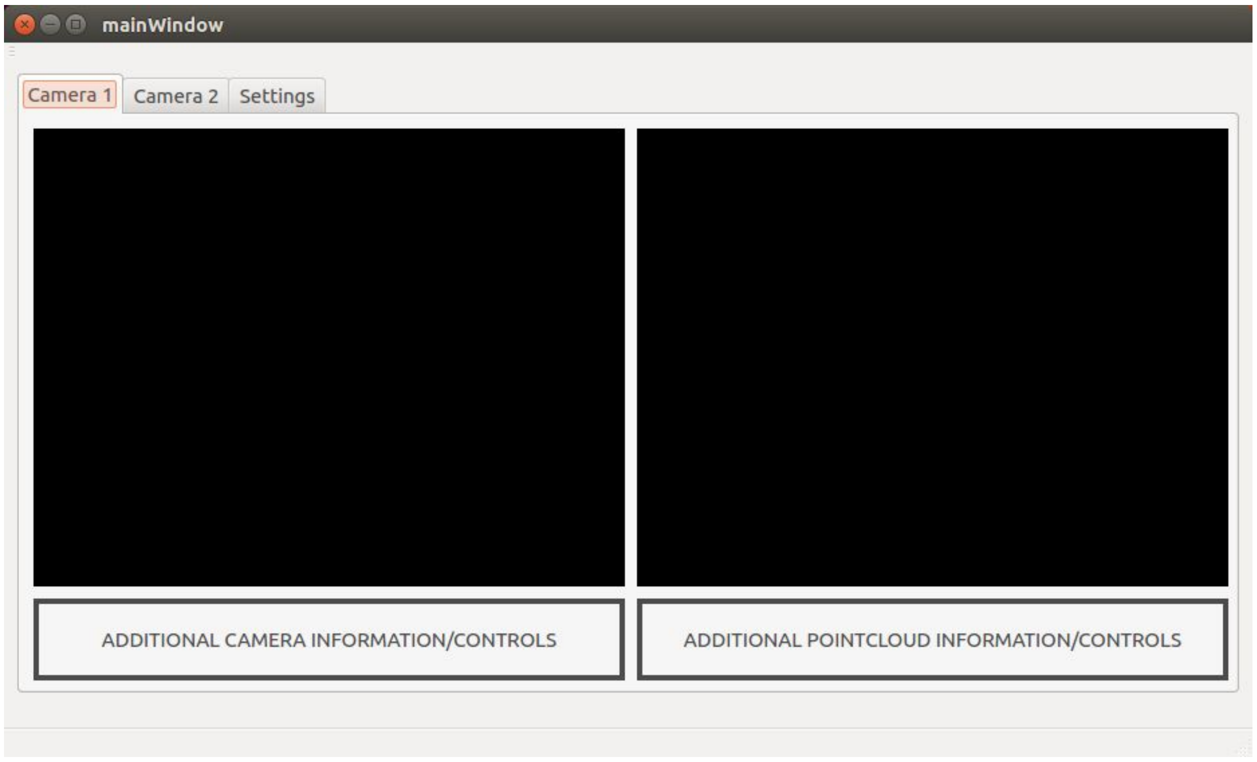
In diagram 4.4.3, data flow begins at the dot. The top node receives ROS image messages of video depth data which are then written to a ROS bag file, that bag file will be read simultaneously by ORB-SLAM2 to process that data and generate three point clouds. One point cloud will be sparse and 3D. Another will be sparse, 2D, and have a top down view for navigation. The third will be colored and dense for object recognition. There will be a video stream accompanying these point clouds.

4.5 Class Diagram

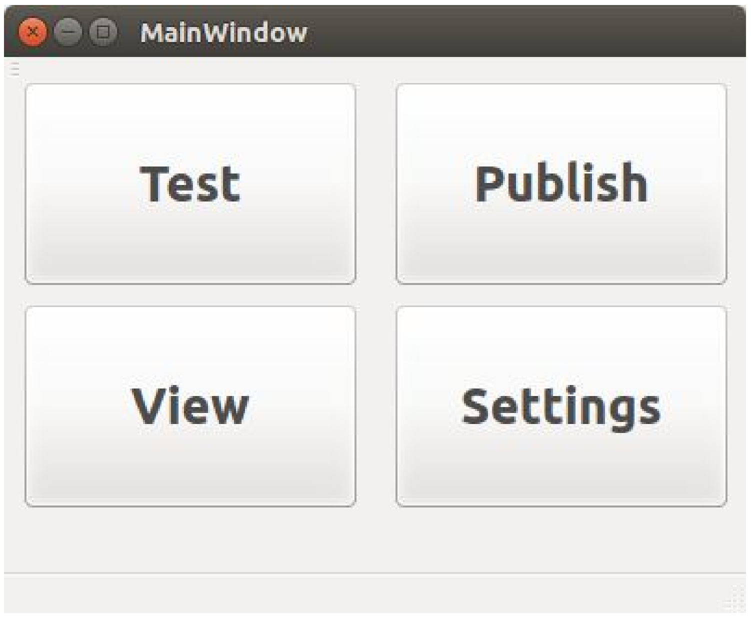

FireFighterBaseUnit
Main
pointCloudWidget
(from FireFighterBaseUnit)
+ ORB_SLAM2::System* mSLAM
+ pointCloudWidget(ORB_SLAM2::System* pSLAM)
+ ~pointCloudWidget()
+ void shutdown()
+ void processFrames(const sensor_msgs::ImageConstPtr& msgColor,const sensor_msgs::ImageConstPtr& msgDepth)
rosNodeWidget
(from FireFighterBaseUnit)
Q_OBJECT
+ Q_SIGNALS:
- std::string mColorTopicName
- sensor_msgs::ImageConstPtr mSubscriberColor
- std::string mDepthTopicName
- sensor_msgs::ImageConstPtr mSubscriberDepth
- std::string mImuTopicName
- sensor_msgs::Imu mSubscriberImu
- std::string settingRosMasterAddress
- std::string settingRosLocalAddress
- std::string settingColorTopicName
- std::string settingDepthTopicName
- std::string settingImuTopicName
- float settingRefreshRate
+ rosNodeWidget()
+ virtual ~rosNodeWidget()
+ bool init()
+ void run()
+ bool stop()
+ void loadSettings()
+ void rosShutdown()
+ void grabSLAMWindows()
mainWindow
(from FireFighterBaseUnit)
Q_OBJECT
- Ui::mainWindow *ui
- rosNodeWidget rosNode
+ explicit mainWindow(QWidget *parent = nullptr)
+ ~mainWindow()
-$ void mergeWindows()
- void loadSettings()
- void saveSettings()
xDo
GLFW
Intel RealSense
Qt
ORB_SLAM2
Point Cloud
ROS
ROS OpenCV
Third Party Libraries
rosNodeWidget
(from FireFighterRemoteUnit)
Q_OBJECT
+ Q_SIGNALS:
- image_transport::Publisher publisherColor
- image_transport::Publisher publisherDepth
- ros::Publisher publisherImu
- std::string settingRosMasterAddress
- std::string settingRosLocalAddress
- std::string settingColorTopicName
- std::string settingDepthTopicName
- std::string settingImuTopicName
- float settingPublishRate
- bool settingAutoExposure
- float settingThresholdMin
- float settingThresholdMax
- float settingSpatialMagnitude
- float settingSpatialAlpha
- float settingSpatialDelta
- float settingTemporalAlpha
- float settingTemporalDelta
+ rosNodeWidget()
+ virtual ~rosNodeWidget()
+ bool init()
+ void run()
+ bool stop()
+ void loadSettings()
+ void rosShutdown()
- void showError(QString errorMessage)
viewerWidget
(from FireFighterRemoteUnit)
- GLFWwindow *previewWindow
- int windowWidth
- int windowHeight
- GLuint motionFrameHandle
- GLuint videoFrameHandle
+ viewerWidget()
+ ~viewerWidget()
+ void close()
+ operator bool()
+ void setSize(int width, int height)
+ void show(const std::map<int, rs2::frame> rscFrames)
- void showMotionFrame(const rs2::motion_frame &rscFrame, const int xLoc, const int yLoc)
- static void drawMotionFrameAxes()
- static void drawMotionFrameCircle(float xx, float xy, float xz, float yx, float yy, float yz)
- void showVideoFrame(const rs2::video_frame &rscFrame, const int xLoc, const int yLoc)
mainWindow
(from FireFighterRemoteUnit)
Q_OBJECT
- Ui::mainWindow *ui
- rosNodeWidget rosNode
- bool isPublishing
- bool isConfig
+ mainWindow(QWidget *parent = nullptr)
+ ~mainWindow()
-$ void on_pushButtonTest_clicked()
-$ void on_pushButtonSend_clicked()
-$ void on_pushButtonConfig_clicked()
- void showError(QString errorMessage)
- void initUI()
- void loadSettings()
- void saveSettings()
Main
FireFighterRemoteUnit

4.7 User Interface Design

Base Unit GUI:


mainWindow
Camera 1
Camera 2
Settings
ADDITIONAL CAMERA INFORMATION/CONTROLS
ADDITIONAL POINTCLOUD INFORMATION/CONTROLS

Remote Unit GUI:



Appendix A: Definitions, Acronyms, Abbreviations

Term	Definition
API	Application program interface
Bag	A file format for storing ROS messages
Base Unit	Laptop with control software. “base station”, “main unit”, and “main station” will be used synonymously
Cvbridge	OpenCV library that creates an interface between openCV and ROS messages.
FIN	Firefighter indoor navigation
FFMPEG	Video capture and encoding package
GUI	Graphical user interface
IMU	Inertial Measurement Unit
mainWindow	Qt object for the main application window
Mbps	Megabits-per-second
Messages	Structured data file
Nodes	Executable that can communicate with other ROS nodes
OpenCV	Open computer vision
OpenGL	Open graphics library
ORB-SLAM2	Real-time SLAM library for Monocular, Stereo and RGB-D cameras
PCL	Point cloud library
Point Cloud	Collection of data points defined by a given coordinate system
Point Cloud Widget	Widget to manipulate the point cloud map
Priority	Scale: (1 - 3) (1 = low, 2 = mid, 3 = high)

Publisher	Executable that sends messages to topic over ROS
Remote Unit	UP board connected to a camera
ROS	Robotics operating system
Roscore	Or ROS Master, is a collection of nodes that are a prerequisite for ROS based systems.
RS2	Api’s from the RealSense2 library
QT	GUI development tool
SLAM	Simultaneous localization and mapping
Stereo Camera	Camera with two lenses that uses binocular vision to measure points of depth
Subscriber	Executable that receives messages on a topic over ROS
UP Board	Small computer
Ubuntu	Linux based operating system
Video Capture	OpenCV library that converts video frames to openCV images