Testing Document

Firefighter Indoor Navigation

Client: Mostafa Daneshgar Rahbar

GTA: Samira Taghavi

Team Members: Thomas Anter
Nawar Mikha
Jordan Shimel

Revision History

Date	Version	Authors	Comments
11/21/19	Version 1.0	Thomas Anter Nawar Mikha Jordan Shimel	First Draft

Table of Contents

1 Introduction

This document is intended primarily for the quality assurance group within the development team. They will be responsible for testing the Firefighter Indoor Navigation (FIN) system before its final submission. Functional Testing is outlined first, followed by non-functional, then system integration, and lastly user acceptance testing. Each section covers various aspects of the testing being conducted; which are as follows: testing approach, pass/fail criteria, entry/exit criteria, suspension/resumption criteria, risks and issues, and what items are being tested.

The bulk of the testing will be done manually by the software testers through interactivity with the GUI. Some of the testing will also occur through the command line terminal, Since there is no database associated with this system, there won't be any database queries to test.

Once the application is fully built, the code base will be refactored before it is ready for the testing phase. It is important to note that testing will also be conducted on multiple machines by multiple testers to ensure the quality of the product.

1.1 Purpose

The main purpose of this document is to ensure that the system is working properly without defects so that the customer is happy with the final product. It outlines the instructions on how to go about testing the various components of the systems for the software testers. Which will ensure a more effective and efficient result for this phase of the development cycle. Each use case from the design document may have 1 or more test cases within this document. Any problems that are discovered during the testing period will need to be addressed before the project can be delivered.

2 Functional Testing

This portion of the document is concerned with the functional requirements and their related use cases from the requirements and design document associated with the FIN system. Testing of this section relates to the actual features of the application's that were formulated by the client and the development team in coordination.

2.1 Approach

The bulk of the testing will be done manually through interacting with the GUI or entering terminal commands and checking the results. The app has built in error handling to prevent crashes when an error is encountered, which will assist the testers for testing fail cases. This portion of the testing will be done first as it is the most fundamental to the overall function of the applications.

2.2 Pass / Fail Criteria

The applications will run as expected for passing conditions, for failures, alerts will be displayed letting the user know that there is a problem. For example, if a user tries to send camera data from the remote unit while the camera isn't fully plugged in, they will receive a message saying "No RealSense device detected". This method will make testing a lot simpler to conduct.

2.3 Entry / Exit Criteria

While some casual testing is done during development, the entire system is expected to be built and refactored before the dedicated testing phase will begin. Before exiting, all of the pass and fail tests will be conducted, with any and all issues being resolved and retested before the product will be ready for the next phase of testing and eventually for release.

2.4 Suspension / Resumption Criteria

Any time a failed test is conducted and it doesn't return an error message to the user testing will be suspended and the code in question will be rewritten to output the error and then tested again. If any major errors in the code base are found, then the development team will have to meet to discuss how to go about solving the problem before edits to the source code will be made.

2.5 Risks / Issues

Because there are so many system and library requirements for the base unit, testing at some point will have to be done a machine that wasn't being used for the development process. This is to ensure the correctness of the of the build environment as well as the installationation and instruction manuals (incase anything was missed during production) so that the end user will be able to install and run the final product without any problems.

2.6 Items to be Tested

The test cases for the functional requirements being tested are: the remote unit GUI, remote unit video feed, remote unit configuration, video data publishing, base unit GUI, base unit configuration, point cloud manipulation, 2d point cloud rasterization, and saving the point cloud on the base unit . They are listed in greater detail below.

2.6.1 Remote Unit GUI

Test Case Name	Base Unit GUI
Test Case ID	FTC - 1
Use Case Dependencies	N/A

Priority	5
Preconditions	1. Firefighter Indoor Navigation software installed 2. Build Environment Created 3. Display connected to the UP board via HDMI 4. UP board connected to a power supply and booted
Postconditions	1. Remote Unit GUI is visible to the operator
Test Steps	1. In the linux terminal, navigate to where the applications executable is saved in the build folder. 2. Execute the application with `./FirefighterRemoteUnit`
Expected Results	Remote unit application executes and the GUI displays with two small top buttons "Test" and "Config" and a large button underneath for "Send"

2.6.2 Testing Video Feed on Remote Unit

Test Case Name	Testing Video Feed on Remote Unit
Test Case ID	FTC - 2
Use Case Dependencies	UC-1: Remote Unit GUI
Priority	2
Preconditions	1. Remote unit has been booted 2. Intel RealSense camera is connected to UP Board via USB 3. Remote unit main menu displayed
Postconditions	1. Remote unit main menu displayed 2. New window is launched, specifications in expected results
Test Steps	1. Press "Test" button on the remote unit GUI
Expected Results	• Either an alert or a new window will be displayed to the user

	• If the video feed test is successful - firefighter notified by a new window popping up with camera data, including: color, depth and imu (inertial measurement unit) • If the video feed is unsuccessful - firefighter notified with message “No RealSense device detected”

2.6.3 Remote Unit Configuration

Test Case Name	Remote Unit Configuration
Test Case ID	FTC - 3
Use Case Dependencies	UC - 1: Remote Unit GUI
Priority	1
Preconditions	Remote unit GUI main menu shown
Postconditions	Remote unit GUI configuration menu shown
Test Steps	Firefighter presses the “Config” Button, a new menu with several options will display. Changing the ros master ip and local ip addresses will provide the ability to change or test the connection with the base unit.
Expected Results	New configuration inputs will be saved

2.6.4 Publishing Video Data to Topic

Test Case Name	Publishing Video Data to Topic
Test Case ID	FTC - 4
Use Case Dependencies	UC - 1: Remote Unit GUI UC - 2: Testing Video Feed on Remote Unit
Priority	4

Preconditions	Remote unit GUI main menu is shown; video data received from camera
Postconditions	Remote unit GUI main menu is shown; video data stream is published to ROS topic
Test Steps	Firefighter presses “Send” button
Expected Results	• If publish successful - the send button will turn green and the test and config buttons will disappear • If publish unsuccessful - firefighter notified with message “No RealSense device detected”

2.6.5 Base Unit GUI

Test Case Name	Base Unit GUI
Test Case ID	FTC - 5
Use Case Dependencies	UC-1: Remote Unit GUI UC-2: Testing Video Feed on Remote Unit UC-3: Publishing Video Data to Topic
Priority	5
Preconditions	1. Firefighter Indoor Navigation software installed 2. Build Environment Created
Postconditions	1. Base Unit GUI is visible to the operator
Test Steps	1. In the linux terminal, navigate to where the applications executable is saved in the build folder. 2. Execute the application with `./FirefighterBaseUnit`
Expected Results	Base unit application executes and GUI displays with widgets for point cloud, video stream visible, and system configuration information.

2.6.6 Base Unit Configuration

Test Case Name	Base Unit Configuration
Test Case ID	FTC - 6
Use Case Dependencies	UC - 5: Base Unit GUI
Priority	2
Preconditions	1. Base unit executable launched 2. Base unit GUI displayed to operator
Postconditions	Base unit GUI displayed to operator
Test Steps	Operator changes configuration parameters on the base unit GUI and presses “Save”
Expected Results	Message displayed to Operator “Config Saved”; Base unit configuration parameters are saved

2.6.7 Point Cloud Manipulation on Base Unit

Test Case Name	Point Cloud Manipulation on Base Unit
Test Case ID	FTC - 7
Use Case Dependencies	UC - 1: Remote Unit GUI UC - 2: Testing Video Feed on Remote Unit UC - 4: Publishing Video Data to Topic UC - 5: Base Unit GUI
Priority	3
Preconditions	1. Image data sent to the remote unit 2. Point cloud generated and displayed on the base unit
Postconditions	Point cloud display changed after manipulation
Test Steps	1. Test zoom functionality with the scroll widget on the mouse

	2. Test panning with by clicking the left button on the mouse and moving either vertically, horizontally, or diagonally 3. Test rotation in all 6 degrees of freedom by clicking on the right button on the mouse and dragging across the screen. 2 dimensions are accessible with just the right click, the 3rd will require both the left and right mouse buttons to be clicked.
Expected Results	Point cloud perspective/zoom changed with mouse drag/scroll

3 Non-Functional Testing

Scalability is not included in this iteration of the project, since only one remote unit will be in testing at a time. Security is not addressed as the client stated it was a non-concern for the project at this time. Portability is not tested as this system is only designed for a single build environment. Maintainability will not be included with the traditional tests as this document and the others are in a sense, the subjects for the test, along with the code clarity. The test for these are the grades received in the course, and peer review. In this section, the non-functional requirement's test cases are outlined; which include efficiency, performance, reliability, and availability. Testing will be done by multiple testers to get more accurate outputs.

3.1 Approach

Individual testers will use statements within the code that get the time difference between the start and the end of a certain event. These metrics will be compared to the standards described in the requirements document; any numbers that aren't within range will have to be addressed. Other metrics, such as system boot time will have to be calculated manually. Test results must be within range as this is a real-time system, where unnecessary delays can have disastrous consequences. Non-functional tests will be conducted at least 10 times each by an individual tester and the average of the results will be used for evaluation. Each tester on the development team will have to have positive results before a test can be marked as passed.

3.2 Pass / Fail Criteria

If a given test's results are within an acceptable range, then that test will be marked as passed. Otherwise, if the result falls outside of the acceptable range, then the individual tester or the team as a whole will have to determine what changes need to be made to the code base to fix the problem.

3.3 Entry / Exit Criteria

Entry will begin when a certain process is initiated, such as the send button being selected from the remote unit which will trigger a timestamp marking the start of the image message transfer. Continuing with this example, the exit condition here is when the image message is received by the base unit. This will then trigger another timestamp. The exit condition will occur regardless if the test passes or fails. For testing purposes, a single function will calculate the difference in timestamps for all of the different items being measured and display them on the screen. Once all of the testing is completed, and the nonfunctional tests have all been shown to produce positive results, this function will be removed from the source code.

3.4 Suspension / Resumption Criteria

Fail criteria as described above will trigger a suspension event. As the development phase of the project will already be complete, the functionality being tested is expected to work. How well it's working is what is being tested; unmet standards will result in a suspension. Resumption will only occur when the errors in the code are resolved by the tester(s). If any major errors in the code base are found then the development team will have to meet to discuss how to go about solving the problem before edits to the source code will be made.

3.5 Risks / Issues

It will be important for multiple testers to conduct the non-functional test cases for accuracy of data, as different computers are likely to produce different results for these tests. It is expected that all of the machines that the testing will be conducted on are capable of meeting the standards for these tests, so if problems are encountered on any one system, then the architecture and implementation of the code are likely the root of the problem, not the hardware. A major issue that has already been discovered during the implementation phase, was that the wifi adapter on the remote unit was causing a bottleneck in the bandwidth between the two units, this is currently being resolved as the development phase is nearly complete. If another hardware related bottleneck is discovered, this may cause some trouble with the final deadline nearing.

3.6 Items to be Tested

The test cases for the non-functional requirements being tested are: the speed of image message transfer, boot time of the distributed system, the frame discrepancy between the point cloud map and the video stream, remote unit video test speed, mean time between failures, and down time. They are listed in greater detail below.

3.6.1 (Performance) Image Message Speed

Test Case Name	Speed of ROS Image Transfer Between Units
Test Case ID	NFTC - 1
Priority	3
Preconditions	1. Both remote and base units are booted 2. Camera has been connected to remote unit via USB 3. Video feed on remote unit has been tested

	4. Time logs are placed where the first message is being sent from the remote unit to a topic, and at the location on the base unit where the first message is received.
Postconditions	Depth and video data are received by the base unit
Test Steps	1. Firefighter presses “Send” button to publish video and depth data. 2. Time logs are compared and evaluated
Expected Results	Duration of the message transfer time determined by the difference of both time-stamped logs is under one second and is then output to the tester(s).

3.6.2 (Performance) Distributed System Boot Time

Test Case Name	Boot Time
Test Case ID	NFTC - 2
Priority	4
Preconditions	1. Devices for both the base and remote units are booted 2. Camera has been connected to the UP board
Postconditions	1. Both base unit and remote unit applications are launched 2. Video data is sent from remote unit to base unit 3. Data received by base unit is being processed 4. Video stream is displayed on base unit 5. Point cloud is displayed on the base unit
Test Steps	1. Firefighter launches remote unit executable 2. Operator launches base unit executable 3. Firefighter and operator point publishing/subscribing configuration parameters to point to the same topics (IMU_TOPIC_NAME, COLOR_TOPIC_NAME, DEPTH_TOPIC_NAME) 4. Firefighter presses “Test” button to confirm that

	camera is collecting data 5. Firefighter presses “Send” button to publish that data to topic
Expected Results	Video stream and point cloud are displayed on the base unit, confirming that data is flowing as expected

3.6.3 (Performance) Frame Discrepancy Between Point Cloud and Video Stream

Test Case Name	Frame Discrepancy Between Point Cloud and Video Stream
Test Case ID	NFTC - 3
Priority	2
Preconditions	1. Devices for both the base and remote units are booted 2. Camera has been connected to the UP board 3. Both applications are launched 4. Data is sent from remote unit to base unit with ROS 5. Point cloud is generating on the base station 6. Video stream and point cloud are being displayed on the base station
Postconditions	Point cloud and video stream remain visible on the base station
Test Steps	1. Firefighter will move camera quickly in the opposite direction to capture depth points that are outliers of the already created points. 2. Operator will time the difference between the time the new angle of the camera pivot is visible in the video stream and those corresponding depth points added to the point cloud.
Expected Results	The duration calculated in test step two: < 1 second = Passing > 1 second = Failing

3.6.4 (Performance) Video Test Speed

Test Case Name	Video Test Speed
Test Case ID	NFTC - 4
Priority	2
Preconditions	1. Remote unit executable launched 2. Camera is connected to UP board via USB 3. Remote unit main menu is displayed
Postconditions	Message is displayed to firefighter regarding status of video data collection
Test Steps	Firefighter presses “Test” button on remote unit main menu and records time until resulting message is displayed on the screen
Expected Results	Time taken to display the message: < 1 second = Passing > 1 second = Failing

3.6.5 (Reliability) Mean Time Between Failures

Test Case Name	Mean Time Between Failures
Test Case ID	NFTC - 5
Priority	3
Preconditions	1. Devices for both the base and remote units are booted 2. Camera has been connected to the UP board 3. Both applications are launched 4. Data is sent from remote unit to base unit with ROS 5. Point cloud is generating on the base station 6. Video stream and point cloud are being displayed on the base station
Postconditions	Video stream or point cloud generation have terminated or paused

Test Steps	Timestamp the event of the preconditions are done and timestamp the event of the post condition. Repeat this test 20+ times to get a reasonable mean of this duration
Expected Results	Mean time between failures: > 30 min = Passing < 30 min = Failing

3.6.6 (Availability) Down Time

Test Case Name	Down Time
Test Case ID	NFTC - 6
Priority	*5*
Preconditions	1. Devices for both the base and remote units are booted 2. Camera has been connected to the UP board
Postconditions	Case 1: Video stream and point cloud are displayed on the base unit, confirming that data is flowing successfully Case 2: Base unit failed to display either the point cloud or video stream
Test Steps	1. Firefighter launches remote unit executable 2. Operator launches base unit executable 3. Firefighter and operator point publishing/subscribing configuration parameters to point to the same topics 4. Firefighter presses “Test” button to confirm that camera is collecting data 5. Firefighter presses “Send” button to publish that data to topic 6. Log result as true if point if in postcondition case 1, false for postcondition case 2 7. Repeat this test 50 times
Expected Results	< 1 false output = Passing > 1 false output = Failing

4 System Integration Testing

This section is concerned with the FIN system as a whole, and that all of its different software and hardware components are communicating properly with each other. Several third party software packages are used in these applications; such as: orb-slam2, pangolin, opengl, opencv, ros, point cloud library, and more. In terms of hardware, there are also several constituents, the primary two being the base and remote units. The remote unit being the more probable source of occasional problems, since it's a mobile device, loose physical connections are more likely to occur between the different parts on this unit during routine use. The connection between the two units for both wireless and wired communication is also encompassed in the integration testing phase.

4.1 Approach

The "big bang" approach will be used for the integration testing rather than a top down or bottom up approach. This means the system will be tested as it works together as one unit, rather than testing the components individually. The reason for this is to avoid mocks for data flow. There are a lot of dependencies for the FIN system, and individual unit testing will require mocking its requirements. Testing of this system will be more efficient conducted on the system as a whole. The system will be tested in sequence, starting as the data flow does, with the camera, and finishing with the base unit GUI. The assumption before this test is conducted is that both units have been properly launched, all attachments are connected, and configurations properly set; with the distributed system either being directly connected or in range for point to point wifi. With new additions/changes to the FIN system, this test will be used as regression testing, to confirm that no code was broken.

4.2 Pass / Fail Criteria

If the system is operating normally as expected under routine use, then it can be considered a passing condition; because if the two streams are being displayed in the base unit application, then that means that everything has been integrated properly as this is the last step in the data flow and the system overall. If the point cloud is not generated for any reason or and sort of custom error message is displayed, then there is a disconnection somewhere in the system. Several problems may need to be resolved, such as: the camera may need to be unplugged and plugged back in, or there may be a problem with the linkage to a dependency, or the configurations may have been entered incorrectly.

4.3 Entry / Exit Criteria

Entry will occur as soon as the remote unit is launched and starts sending information. Exit criteria will be when the camera data is displayed in the base unit GUI and the user can interact with it. Crashes will also qualify as exit criteria, but as failure instances of the test.

4.4 Suspension / Resumption Criteria

Any Crashes during testing will result in suspensions. The third party messages displayed in the terminal or the custom ones from the applications themselves will be used by the testers to diagnose the problems and solve them. During development, often times segmentation faults have occurred (and were previously addressed), so more may be likely to appear during integration testing.

4.5 Risks / Issues

If the various packages used by the FIN system aren't organized or integrated well this may cause problems for future users when they install the applications, but have problems because the applications can't find the necessary third party libraries. When integration testing, the tester will start from a fresh environment on a virtual machine and install the application and associated packages to guarantee any users or legacy developers will be able to use or built on the programs respectively.

4.6 Items to be Tested

The system integration test is broken up into three subtests, in the order of the data flow. Beginning with the actual check for the incoming camera data on the remote unit. This test is built into the system for the user and is a prerequisite for all later testing. From there both of the two streams are checked on the base unit (the point cloud and the direct video frames). The latter two being the final stages of the data flow, ensuring integration among all of the system's components.

4.6.1 Camera Data Reception

Test Case Name	Camera Data Reception
Test Case ID	SITC - 1
Priority	3
Test Steps	User presses "Test" button on remote unit GUI
Expected Result	A new window with depth, color, and imu data from the camera is displayed.
Assumption	This confirms data is flowing properly to this point, otherwise there is a bug

4.6.2 Point Cloud Display

Test Case Name	Point Cloud Generation and Display
Test Case ID	SITC - 2
Priority	5
Criteria	User presses "Send" button on the remote unit GUI
Expected Result	Point cloud appears and grows in the left widget on the base unit GUI as more data is collected over time.
Assumption	• Data is being collected by a camera, converted

	to a ROS message • ROS message is being sent over point to point wifi • ROS message interpreted on the base unit • Point cloud is generated on the base unit Confirmation that data is flowing correctly to generate a point cloud

4.6.1 Video Stream Display

Test Case Name	Point Cloud Generation and Display
Test Case ID	SITC - 2
Priority	5
Criteria	User presses "Send" button on the remote unit GUI
Expected Result	Video stream from camera begins in right widget on the base unit GUI and objects in the view of the camera are clearly visible, with edge detection indicators.
Assumption	• Data is being collected by acamera, converted to a ROS message • ROS message is being sent over point to point wifi • ROS message interpreted on the base unit • Video Stream is generated on the base unit Confirmation that data is flowing correctly to generate a point cloud

5 Client Acceptance Testing

At this point, all of the testing by the development team will have been completed. All that is remaining is for the client to conduct his own testing to determine if the project has met all of his standards and requirements.

Any possible problems will have to be discussed with the team on how they can be resolved. No additional requirements may be added at this point.

5.1 Approach

Our client has clear expectations for the FIN system and those expectations are described in detail in the SRS document. The client acceptance testing will reference those requirements communicated to the team. There were a number of original requirements and a few added on as the process went on. As there was turnover on the team and requirements changed, the team has adapted to the new requirements and the most updated of those will be explained in this section. There will be simple pass/fail criteria, as it is up to the client to see if the implementation of the SRS was satisfactory.

5.2 Pass / Fail Criteria

While the requirements are quite clear in the SRS, this portion is still somewhat subjective; while the client had agreed on a clear set of requirements, he may decide that he would like some minor things changed to better suit his needs. It behooves the development team to see that the client is happy with the overall product he is receiving and to try and accommodate minor changes before the deadline. If all of the requirements have been met and the client is satisfied, then this test is passed; otherwise any small changes will have to be made accordingly.

5.3 Entry / Exit Criteria

Again, this portion of testing will only be initiated once the testing team has finished their phase of testing. Once the client states that the FIN system is completed and everything is functioning as required, then the criteria to exit the test will have been met and the actual development process will be complete.

5.4 Suspension / Resumption Criteria

Any items the client deems as failures will result in suspensions with the teams having to resolve them. The client will then resume testing for a second iteration, and the process will continue until there are no more suspensions and all of the tests have passed.

5.5 Risks / Issues

The team has maintained constant communication with the client throughout the development process to ensure that the applications are exactly what was envisioned by the client. However, with software development there is always the chance that the client is unhappy with the product and will ask for major changes or find some problems. Any defects will have to be addressed, but new functionality will not be accepted, as there is no time to implement any with the deadline for the applications completion date rapidly approaching.

5.6 Items to be Tested

The test that the client will be testing are: the remote unit hardware, remote unit software and environment, remote unit GUI, distributed system network, base unit hardware, base unit software and environment, and the base unit GUI. They are listed in greater detail below.

5.6.1 Remote Unit Hardware

Test Case Name	Remote Unit Hardware
Test Case ID	CATC - 1
Priority	2
Criteria	1. Intel Realsense D435i camera used 2. UP board is used for computation

5.6.2 Remote Unit Software and Environment

Test Case Name	Remote Unit Software and Environment
Test Case ID	CATC - 2
Priority	3
Criteria	1. Intel RealSense sdk for camera API 2. Linux OS 3. OpenCV to interface with ROS

5.6.3 Remote Unit GUI

Test Case Name	Remote Unit GUI
Test Case ID	CATC - 3
Priority	2
Criteria	1. Clear and easy layout to use for firefighter on mission 2. Language appropriate for firefighter to understand usage of the applications 3. Professional color scheme, fonts, design, etc.

5.6.4 Distributed System Network

Test Case Name	Distributed System Network
Test Case ID	CATC - 4
Priority	4
Criteria	1. ROS used for data transfer between remote and base units a. Nodes executed on each unit b. Nodes interact by publishing/subscribing

	to topic 2. Distributed System is capable of using point to point wifi to enable ROS data transfer

5.6.5 Base Unit Hardware

Test Case Name	Base Unit Hardware
Test Case ID	CATC - 5
Priority	2
Criteria	1. Laptop is used to launch Firefighter Base Unit application 2. Laptop chosen to facilitate the point cloud generation to the best of the team's abilities

5.6.6 Base Unit Software and Environment

Test Case Name	Base Unit Software and Environment
Test Case ID	CATC - 6
Priority	5
Criteria	1. ORB-Slam 2 software used for SLAM a. Creation of dense 3d point cloud b. Capture top angle of 2d point cloud c. Use video stream provided by ORB Slam displaying depth points 2. Linux OS 3. OpenCV to interface with ROS 4. Qt used for application development, GUI

5.6.7 Base Unit GUI

Test Case Name	Base Unit GUI

Test Case ID	CATC - 7
Priority	5
Criteria	1. Dense 3d point cloud displayed and manipulation possible (Angle change, zoom, pan, etc.) a. Position and orientation of the camera displayed b. Camera path shown 2. 2d top angle point cloud captured and displayed 3. Video stream of camera displayed 4. Option for operator to modify calibration settings