Verification and Validation Report

Cyclops Ride Assist: Real-time Monitoring System.

Team 9
Aaron Li (lia79)
Amos Cheung (cheuny2)
Amos Yu (yua25)
Brian Le (leb7)
Manny Lemos (lemosm1)

Table Of Contents

List of Tables

List of Figures

1. Revision History

Table 1.1: Revision History

Date	Developer(s)	Change

Date	Developer(s)	Change
March 1, 2023	Aaron Li, Amos Cheung, Amos Yu, Brian Le, Manny Lemos	Document created
March 6, 2023	Brian Le	Updated sections
March 6, 2023	Amos Cheung	Updated sections
March 7, 2023	Aaron Li	Updated sections
April 3, 2023	Amos Cheung	Changed name/description of some of the sections. Refer to issue #322 for details

2. Symbols, Abbreviations, and Acronyms

Name	Description
CRA	Abbreviation of Cyclops Ride Assist.
Cyclist	A person who operates a bicycle as a means of transportation.
User	A person who will operate the final product. See Cyclist.
Client	See user.
SRS	Software Requirements Specification
Video feed	A cycle of clips using a fifo queue implementation. Can be concatenated to form the last buffer_time seconds of video.
Lidar	Light Detection and Ranging. A range sensing method that uses a pulsed laser.
LED stick	An array of 8 individually addressable RGB LEDs.
α_x	Measures acceleration parallel to the path of the bicycle.
α_y	Measures acceleration perpendicular to the path of the bicycle along the plane of the ground.
α_z	Measures acceleration in the vertical direction.
norm	A measure of the absolute value of the acceleration vector.
tilt	A measure of acceleration vector in the xy plane.
f_sampling_cd	Sampling frequency of the crash detection.
f_sampling_bd	Sampling frequency of the rear vehicle detection.
buffer_time	Seconds of video to save after a crash occurs.

3. General Information

3.1. Purpose

The purpose of this document is to outline the steps taken for validation, verification, and testing of the functionality of Cyclops Ride Assist (CRA).

3.2. Scope

This document details the testing procedures and pass-fail requirements that ensure each sub-system in CRA functions independently and as a complete and fully integrated product. Software and hardware components alike will both be subject to Black box testing.

3.3. Background

CRA will be an easily mountable, and quick to set up system that adds modern car safety features onto any bike. These features include rear vehicle detection and alert, a continuous loop of the last 60 seconds of camera, accelerometer, and Lidar data, and crash identification and response. CRA is aimed at cyclists of all levels that frequently traverse road and gravel terrains. CRA is not designed to be used on extreme terrain such as downhill mountain biking.

4. Functional Requirements Evaluation

The testing of the functional requirements is important to ensure that the system, and the many components that it is composed of, functions as designed. The methodology for the functional requirements is to begin with unit testing of the individual components, then move to integration testing.

Traceability	Relevant Components	Test Plan and Test Factors
CFR1	Rear Vehicle Detection	The tests that focus on this requirement demonstrated that the system is able to accurately convert a distance signal input into an LED output. For the testing of this requirement to pass, it was critical that both the lidar and the LEDs are functioning properly independently of each other. When testing the lidar, it was important to consider the length of the cable that connects the lidar to the microcontroller. Having such a long cable can cause inaccuracies in signal transmission. Testing the lidar also involved making sure that the sensor could return accurate distance measurements over the full range of operation of the device. The LEDs also needed to be tested to make sure that each individual LED on the LED stick could be toggled on or off, and the color modified as desired. To test the integration of the full rear vehicle detection system, it was important to test that the rate at which the LEDs turn on was proportional to the rate at which an object was being closer in the lidar's range of operation. Testing involved both using code to test as well as field testing, to ensure that this functional requirement worked accurately and continuously from the input to the output.

Traceability	Relevant Components	Test Plan and Test Factors
CFR2	Video/Data Logging	The test plan for video recording demonstrated that the system can collect uninterrupted video footage. This sort of testing focuses on the camera equipment itself as well as the code that uses it. Factors to consider included making sure that the camera was mounted securely to the housing, and verifying that the soldering of the camera cable has good contact and no shorting. When testing the software for the video logging, the camera was connected to a laptop and it was demonstrated that it worked as intended. This same test was then transferred to the microcontroller to make sure that it could provide the same performance on the embedded system.
CFR3	Crash Detection	A factor that was kept in mind when testing the accuracy of the acceleration data returned by the accelerometer was that the data returned was not in m/s^2. Thus, it was also necessary to test the function used to scale the data coming in to be in SI units. Knowing also that the accelerometer is three-axis, it was necessary to test all three axes independently to measure any cross-sensitivity that might be present. It was also important to test the frequency at which the crash detection polls data. Testing revealed an optimal f_sample_cd sampling frequency that was a balance between resource usage and data reliability. Testing involved both using code to test as well as field testing. Field testing looked like subjecting the accelerometer to various impacts in different directions, and also running tests to ensure that, when a user is cycling under normal conditions, the jitter of the acceleration data that is being measured is kept below a threshold.
CFR5	Crash Detection	The tests that focus on this requirement were designed to demonstrate that the crash detection component is able to differentiate between a user that is cycling versus a user that is still. Testing revealed that the system is not capable of achieving this, as there were too many factors at play. For example, in order for the acceleration data of an idle bike versus a moving bike to be differentiable, it had to be assumed that the bike was on its kickstand in order to remove the factor of any movement that the user might introduce. Although this test did not pass, it was shown that the functionality of the crash detection in

Traceability	Relevant Components	Test Plan and Test Factors
CFR6	Rear Vehicle Detection	When testing the lidar sensor, a factor that was considered was the type of material of the surface that the signal would reflect off of. The exterior of vehicles is usually made of a smooth metal with a coating to finish, which was shown to reflect the lidar signal reliably. Although vehicles were the primary target for the rear vehicle detection, testing factored in other common surfaces that might be seen by cyclists on the road, such as concrete, wood, and plastic, as well as objects with bizarre shapes, such as other bicycles. It was also important to test the frequency at which the rear vehicle detection polls distance data. Testing revealed an optimal f_sample_bd sampling frequency that was a balance between resource usage and data reliability. Unit testing for the software portion of this functional requirement involved feeding mock data to the rear vehicle detection function, then verifying that the algorithm could recognize that an object was approaching (i.e. the distance was steadily decreasing).
CFR7	Crash Detection	Testing the crash detection algorithm involved verifying the functionality of each of the three stages of the algorithm, then making sure that the algorithm was able to transition from one stage to the next in response to the acceleration data coming in. The three stages are called the Normal, Climax, and Inverse states, and to enter each stage, the acceleration data needed to exhibit certain behavior. If the algorithm got to the Inverse state, it was determined that a crash had occurred. When testing the robustness of the algorithm, atypical crash datasets were provided to see if a crash would be detected. By this approach, the algorithm could be continuously improved and refined until all the datasets would be detected. The same testing process was carried out for false positives. Testing involved using software testing suites as well as field testing, to ensure that the algorithm could match performance in a real-time environment.
CFR8	Crash Detection, Video/Data Logging	The testing done for this requirement focused on the length of the video exported by the system. Other factors to consider included the memory and CPU usage that the buffer process consumed on the microcontroller. A software testing suite was used to verify that the buffer process was safely creating and dumping video fragments, and could stitch those fragments together into a full video of maximum length buffer_time upon request. The suite also tested the length of video outputs that are requested one right after another. On top of using the suite, manual testing revealed the importance of maintaining a steady, expected framerate. This was found to be important because it was necessary to ensure a video that was not sped-up or slowed-down, and would also not sporadically speed up and slow down.

Traceability	Relevant Components	Test Plan and Test Factors
CFR9	Video/Data Logging	It was important to test the functional requirement of CRA to be able to save the video footage to disk in order to define how the microcontroller behaves when the external storage is either not plugged in or out of memory. The idea is to validate that the user experience is not affected when the external storage component is found to be in an invalid state. A factor to consider was the fact that a user could choose to plug in their own external storage instead of using the one provided. Another factor was the fact that a user might be using the external storage for other things as well, and thus the CRA should not interfere with other files on it. Field testing looked like leaving the external storage unplugged or plugging in several different external storages at different storage capacities.
CFR11	Power	The objective behind testing the power component was to make sure that the battery is able to sustain the microcontroller under normal operating conditions, and to define the behavior of CRA in the cases when the power supplied by the battery is insufficient. A part of the testing plan involved measuring the battery life of the system from a full charge, when it is turned on, and when it is turned off. This testing revealed ways that the microcontroller process could be optimized to consume less power, and gave the product a specification of battery life that can be provided to the user. Furthermore, the testing plan needed to decide at what point the user needs to be reminded of low battery, and at what point the battery can no longer sufficiently power CRA.
CFR12	Video/Data Logging	The testing done for this requirement focused on the continuous, uninterrupted execution of the video buffer process while another a previous video was being logged. For the testing of this requirement to pass, it was required that there be a negligible, ideally zero, amount of lag occur when a video logging action is called. This testing was first done manually in the lab, performed over SSH or with the CRA plugged into an external monitor. Then, the testing was transitioned to the field, where the system was subjected to normal operating conditions. In the case of the field, the process could not be monitored in real time, so the test was verified by looking for any time discrepancies in the generated logs.

Traceability	Relevant Components	Test Plan and Test Factors
CFR13	Crash Detection	The testing done for this requirement focused on the continuous, uninterrupted execution of the crash detection process while another crash was being logged. For the testing of this requirement to pass, it was required that there be a negligible, ideally zero, amount of lag occurring when the crash detection algorithm would detect a crash. This testing was first done manually in the lab, performed over SSH or with the CRA plugged into an external monitor. Then, the testing was transitioned to the field, where the system was subjected to normal operating conditions. In the case of the field, the process could not be monitored in real time, so the test was verified by looking for any time discrepancies in the generated logs.

5. Non-Functional Requirements Evaluation

The testing of the non-functional required both qualitative and quantitive tests. Usability surveys, rankings, and timing tests were all used in conjunction with one another to determine if all the requirements had been satisfied. In order to ensure that the CRA is of the highest standards, the team ensured that all testing was done in a robust and repeatable way. All test specifications and structure can be found in the VnVPlan.

5.1. Look and Feel Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR1, CNFR2, CNFR3, CNFR4, CNFRST1	The main goal of testing the Look and Feel Requirements is to ensure that the CRA is well adapted to a certain style and appearance that was outlined in the SRS document. Due to certain factors such as societal or personal factors, it was key to ensure that the look and feel of the CRA would not become a hindrance to its users. The test plan was to first conduct a visual inspection test and to create several usability surveys for all patrons to fill out in order to determine whether or not the product was satisfactory according to the noted requirements.	Usability tests were completed by different users and the overall score was averaged out to be an 8.5. Positive feedback came from the minimalistic design, colour choice, and size. However, there was criticism in terms of the overall rectangular appearance and choice of material.

5.2. Usability and Humanity Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR5 - CNFR14, CNFRST2 - CNFRST6	The main goal of testing the Usability and Humanity Requirements is to ensure that the CRA can be used by a variety of users and stakeholders with ease as was outlined in the SRS document. Since the system may be considered complicated to non-technical or technical users, it was imperative to make sure that the product was as simple as possible. Therefore, two categories were to be tested: hardware and software.	Overall, four different tests were completed to ensure that the Usability and Humanity Requirements were fulfilled. For further details and implications of the test, the VnV Plan can be referenced. For CNFRST2, the average time was measured to be around 16.3s. Positive feedback was given for the design of the locking mechanism while negative feedback was given for the weight. For CNFRST3, the average time it took to open the application after a power cycle was approximately 30.7s. This was not satisfactory to the developers as it was above the threshold set (RESPONSE_TIME_MILLISECONDS) given and so further implementation and testing will be done to ensure a shorter time. For CNFRST4 and CNFRST5, the manual was spoken instead to different users using non-technical language. The overall score was 8.1 out of a possible 10. Feedback included further instructions and explanations of the system. These tests were combined for simpler testing. For CNFRST6, the audiovisual cue was output almost immediately which was <1 second. This is within the response time desired.

5.3. Performance Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR15 - CNFR33, CNFRST7 - CNFRST21	The main goal of testing the performance requirements is to ensure that the CRA is performing up to expected standards. The testing plan included all the different modules of the system. This would include the hardware (The chassis, the camera, the microcontroller, the storage, etc.) as well as the software (Algorithms, crash detection, LiDAR, video logging, etc). These tests also ran simultaneously with various CFRST tests.	Please refer to the Functional Requirements Evaluation in Section 4 or the Testing listed below in Unit 6 for further details on the testing plan for performance. Furthermore, the details of each test can be found in the VnVPlan document. For CNFRST7, the average of the recorded tests was 30.7s. This was the same test as CFNRST3. For CNFRST8, the average of the recorded tests was 16.5s. This is within our allowable range as prescribed through UPLOAD_TIME_SECONDS. For CNFRST9, the average of the recorded tests was <1s. This is within our allowable range. For CNFRST10, the average of the recorded tests was 84%. This is within our allowable range but can be improved on to ensure better vehicle detection. For CNFRST11, the developers tested on their own personal bicycles with no hindrance. Furthermore, after conversing with users, it was deemed that there was no hindrance to the pedaling, braking, or other normal functions on their bicycle. For CNFRST12, there was no leaking or sounds to be heard. Over the course of twenty trials, there was no indication of any failure or degradation. For CNFRST13, through our software unit testing, it was deemed that all calculations were within the three decimal range provided. For CNFRST14, through our software unit testing, it was deemed the speedometer and accelerometer were running accurately within 2%. For CNFRST15, the average of the recorded tests was within 1s. This is within our allowable range. For CNFRST16, the average of the recorded tests was 10.3s. This is within our allowable range. For CNFRST17, a drop test and crash test was performed twenty different times. After further inspection, it was deemed that there was no damage to the CRA. For CNFRST18, the average of the recorded tests was 5.1s. This is within our allowable range. For CNFRST19, the system was left on for various periods of time while running solely on battery. At the end of the tests, it was found that the battery could last for approximately 30 minutes before depleting. This result was not satisfactory and will require further research. For CNFRST20, a survey was collected with feedback. Many responses did not have any concerns with the CRA; however, some wanted further extensibility in terms of ports, cameras, battery. For CNFRST21, the test was intended to be continuous. Upon writing this document, there has been no documentation of any damage or wear to the product.

5.4. Operational and Environmental Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR34 - CNFR38, CNFRST22 - CNFRST24	The main goal of testing the operational and environmental requirements is to ensure that the CRA would be able to operate in various conditions and situations. The testing plan included placing the CRA in various locations with different climates, placing the CRA in different positions, as well as interfacing the CRA with different OS and hardware.	For CNFRST22, the CRA was brought outside and all intended functionalities were working as expected. For CNFRST23, the CRA was connected to a variety of systems and computers and all intended functionalities were working as expected. For CNFRST24, the repository was able to be accessed publicly by the developers as well as other users and stakeholders through a link.

5.5. Maintainability and Support Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR39 - CNFR42, CNFRST25	The main goal of testing the maintainability and support requirements is to ensure that the CRA will be widely available and accessible to all. The testing plan included ensuring that the code repository was fully public and able to be accessed by any on various sorts of computers and systems.	For CNFRST25, different errors like unplugging, missing hardware, and recreated bugs were created and the system did not work as intended. Furthermore, the system would alert the user that it was not working as intended.

5.6. Security Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR43, CNFR44, CNFR45, CNFRST26, CNFRST27	The main goal of testing the security requirements is to ensure that the CRA. The testing plan included two plans that would test the access and the privacy to the loaded storage. It was ensured that the footage created would not be uploaded to any unwanted platforms or the cloud.	For CNFRST26, the SD card was placed into various systems and computers and all files were readable. A conversion was done between the Pi video file and an .mp4. For CNFRST27, no videos were sent to the cloud through the wireless connection provided to the CRA. All files were found to be local.

5.7. Cultural and Political Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR46, CNFR47, CNFRST28	The main goal of testing the cultural and political requirements is to ensure that the CRA will not offend stakeholders, users, or others. This testing plan was included to ensure that the product would not infringe on Equity, Diversion, and Inclusion of any users. The tests included a usability survey given to users to gain their feedback.	For CNFRST28, the manual was indeed in English.

5.8. Legal Requirements

Requirement Traceability	Test Plan and Test Factors	Results and Evaluation
CNFR48, CNFR49, CNFRST29, CNFRST30	The main goal of testing the legal requirements is to ensure that the CRA is compliant and is not breaking any laws in various countries, mainly Canada. The testing plan included group meetings and further research that would be compiled into a document.	For CNFRST29, various bicycle brands were researched and tested. Furthermore, many users possess bicycles and so testing was also done on a multitude of these vehicles. For CNFRST30, the CRA did not infringe on any of the legal or compliance requirements in Canada. Therefore, it was noted that the product would be allowed.

6. Testing

Testing was structured based on how the systems and subsystems were defined for the software of Cyclops. With the nature of how each major safety feature is composed of both electrical and software components, either white box or black box testing was utilized. If the unit required a real-world environment to fully work, black box testing was chosen and if the unit involved more algorithmic components, white box testing was chosen.

6.1. Rear Vehicle Detection

The main goal of testing the rear vehicle detection would be to test the functionality of our lidar sensor as well as the algorithm used to convert the distance-based data into a visual representation (LED)'s. The visual representation of LEDs is configured such that each LED will represent a specific level/proximity to the rider of a vehicle or object. Our plan in unit testing would be to ensure that the LEDs are functioning from each extremity (close and far) ensuring that there is a visible object in the sensor's field of vision at all times.

Table 1.2: Rear Vehicle Detection Tests

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
BD1	Object should be held 0 - 1 meters to the rear sensor	CFR1, CFR2, CFR6, CFR10, CFR12, CNFR12, CNFR25	Lidar sensor feed	LED stick cells 1 - 5 should all be lit up	LED stick cells1 - 5 were all lit up	Pass
BD2	Object will be 2 - 3 meters from the rear sensor	CFR1, CFR2, CFR6, CFR10, CFR12, CNFR12, CNFR25	Lidar sensor feed	LED stick cells 1 - 3 should all be lit up	LED stick cells 1 - 2 were all lit up	Pass
BD3	Object will be greater than 5 meters from the rear sensor	CFR1, CFR2, CFR6, CFR10, CFR12, CNFR12, CNFR25	Lidar sensor feed	LED stick cells should all be off	LED stick cells were all off	Pass
BD4	Rear Vehicle detection should have < 1 second in latency for updating the LEDS. We will hide an object away from the Ultrasonic sensors field of vision and then suddenly hold it in the view to see its reaction time	CFR1, CFR2, CFR6, CFR10, CFR12, CNFR15	Lidar sensor feed	LED stick cells should light up within 1 second once an object is put in the sensors field of vision	LED stick cells light up within 1 second once the object is put in the sensors field of vision	Pass

6.2. Crash Detection

The crash detection unit of CRA consists of 2 main components which are the accelerometer inputs and the algorithm to determine a crash. The algorithm is implemented through trial and error to approximate a valid threshold of disturbance towards the unit for the program to have recognized a crash. The test plan is to create a real world environment where the rider and bike will go through common obstacles and routines that are similar to that of the field. Varying obstacles shall be used to create a variety of test cases to test the system boundaries. The accelerometer is the main electrical component in this subsystem so we will also add test suites to ensure that reads are properly made and logged. Simulated tests will be done in a safe and risk free manner in which we are generating enough force for CRA to recognize a crash.

Each test performed for the crash detection unit will have CRA 2 main units already mounted on the bike handle and rear seat respectively.

Table 1.3: Crash Detection Tests

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
CD1	Rider will ride in a straight line on a flat road for 10 seconds	CFR3, CFR4, CFR5, CFR7	Accelerometer electrical input, front camera feed	No crashed detected	No crashed detected	Pass
CD2	Rider will ride in a straight line over grass or a bumpy path for 10 seconds	CFR3, CFR4, CFR5, CFR7	Accelerometer electrical input, front camera feed	No crashed detected	No crashed detected	Pass
CD3	Rider will ride in a straight line for 5 seconds and then “crash”	CFR3, CFR4, CFR5, CFR7	Accelerometer electrical input, front camera feed	Crash detection algorithm should prompt a video clip to be logged	Video clip was logged and stored on SD as well as crash information of moments before	Pass
CD4	Rider will hold the bike in their hand and let the bike drop/fall over	CFR3, CFR4, CFR5, CFR7	Accelerometer electrical input, front camera feed	No crashed detected	No crashed detected	Pass
CD5	Rider will hold the bike and jolt it forward and backwards for 5 seconds	CFR3, CFR4, CFR5, CFR7	Accelerometer electrical input	α_x readings should update to reflect the sudden change in acceleration	α_x readings updates to reflect the sudden change in acceleration	Pass
CD6	Rider will hold the bike and jolt it side to side (left to right) for 5 seconds	CFR3, CFR4, CFR5, CFR7	Accelerometer electrical input	α_y readings should update to reflect the sudden change in acceleration	α_y readings updates to reflect the sudden change in acceleration	Pass

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
CD7	Rider will hold the bike and jolt it up and down for 5 seconds	CFR3, CFR4, CFR5, CFR7	Accelerometer electrical input	α_z readings should update to reflect the sudden change in acceleration	α_z readings updates to reflect the sudden change in acceleration	Pass
CD8	Create another crash after recovering from the prior crash.	CFR12, CFR13	Accelerometer electrical input	CRA should resume crash detection and recording video and data	CRA resumes crash detection and recording video and data	Pass

6.3. Video/Data Logging

The video and data logging unit testing will ensure the functionality of the logging remains active during the time the system is running, saves logged values upon crash, and resumes its logging after the crash. Logged clips and values should remain consistent and have little to no latency. The test suite for this unit will also account for the accuracy and rate in which data has been logged when the unit itself experiences varying changes in movement.

Each test will be performed by having the CRA unit experience and trigger a crash.

Table 1.4: Video/Data Logging Tests

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
VDL1	Take note of event that happened in the field of view of vfront 60 seconds prior to crash	CFR2, CFR7, CFR8	Vfront input	The events of the video clip should be exactly the same as what was viewed 60 seconds prior	The events of the video clip are the exact same as what was viewed 60 seconds prior	Pass

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
VDL2	Prior to the crash, jolt CRA forward and backwards to log and verify the changes in α_x have been logged	CFR3	Vfront input, Accelerometer electrical input, front camera feed	The past 60 seconds of events for α_x should be logged and stored within the CSV data log	The past 60 seconds of events for α_x are logged and stored within the CSV data log	Pass
VDL3	Prior to the crash, jolt CRA forward and backwards to log and verify the changes in α_x have been logged	CFR3	Vfront input, Accelerometer electrical input, front camera feed	The past 60 seconds of events for α_y should be logged and stored within the CSV data log	The past 60 seconds of events for α_y are logged and stored within the CSV data log	Pass
VDL4	Prior to the crash, jolt CRA forward and backwards to log and verify the changes in α_x have been logged	CFR3	Vfront input, Accelerometer electrical input, front camera feed	The past 60 seconds of events for α_z should be logged and stored within the CSV data log	The past 60 seconds of events for α_z are logged and stored within the CSV data log	Pass

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
VDL5	When CRA is powered on, video and data logs should be stored on a new file	CFR7, CFR8	Power button, Vfront input, Accelerometer electrical input, front camera feed	The next crash is detected and should be logged, verify that the file is a fresh file from the previous crash log	A crash was detected and the file is a fresh file from the previous crash log	Pass
VDL6	When CRA detects a crash, the next crash after should have video and data logs stored on a new file	CFR7, CFR8, CNFR29	Power button, Vfront input, Accelerometer electrical input, front camera feed	The next crash is detected and should be logged, verify that the file is a fresh file from the previous crash log	A crash was detected and the file is a fresh file from the previous crash log	Pass
VDL7	CRA video and data logs should be responsive and have latency of < 1 second. We will display the constant data outputs being read in by CRA and note if the values are changing within a reasonable time	CNFR15, CNFR17	Vfront input, Accelerometer electrical input, front camera feed	The values of the acceleration should update and reflect a drop within 1 second of use dropping the unit	The values update within 1.5 second of dropping the unit	Fail

6.4. Power

CRA has configured the power button such that both the crash detection and rear vehicle detection should automatically start running once it has been pressed. The system also has configured the button to log the last bit of data once the power has been turned off (button is pressed again).

Table 1.5: Power Tests

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
P1	CRA should automatically have the scripts of CRA's crash and rear vehicle detection ran once powered on	CFR11, CNFR6	Power button	Video and data should begin to be read in and LEDs should light up to indicate power on	Video and data begins to be read in and LEDs light up when powered on	Pass
P2	CRA should automatically clip and log the past 60 seconds of CRA's crash and rear vehicle detection data once powered off	CFR11, CNFR6	Power button	Data should be logged and saved within CSV locally when CRA has been powered off	Data is logged and saved within CSV locally when CRA was powered off	Pass

6.5. Mounting

The physical system of the CRA consists of two separate housing units that each attach to different parts of the bike. The main housing unit mounts to the handlebars, while the lidar housing unit mounts to the seat post. Both units use the same mounting hardware. Testing will ensure that each mounting system can reliably secure the CRA to the bicycle in a variety of situations.

Table 1.6: Mounting Tests

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
M1	The main housing is resistant to shaking when mounted to the handlebars of the bicycle	CNFR5, CNFR21	High-frequency, low-amplitude shaking over an extended time period	The housing stays in place	The housing stays in place	Pass
M2	The main housing is resistant to impact when mounted to the handlebars of the bicycle	CNFR5, CNFR21	A single, high-amplitude impact	The housing stays in place	The housing stays in place	Pass

Test Number	Description	Requirement Referenced	Inputs	Expected Outputs	Actual Outputs	Result
M3	Securely mount the main housing the handlebars of the bicycle	CNFR13, CNFR21	Several bike handlebars of different diameters	The housing mounts securely	The housing mounts securely	Pass
M4	The lidar housing is resistant to shaking when mounted to the seat post	CNFR5, CNFR21	High-frequency, low-amplitude shaking over an extended time period	The housing stays in place	The housing stays in place	Pass
M5	The lidar housing is resistant to impact when mounted to the seat post	CNFR5, CNFR21	A single, high-amplitude shock	The housing stays in place	The housing stays in place	Pass
M6	Securely mount the lidar to the seat post	CNFR13, CNFR21	Several seat posts of different tube diameters	The housing mounts securely	The housing mounts securely	Pass

7. Changes Due to Testing

The Rev0 demo provided the cyclops team with an opportunity to showcase a functional prototype of the ride monitoring system. However, the presentation's weak points paired with feedback from the Teaching Assistants provided valuable insight that has shaped our final product.

The most glaring failure of the demo was the poor performance of the ultrasonic sensor. The signal noise from operating in an enclosed space meant that the distance being detected was marginally accurate at best, and catastrophically dangerous to users at worst. Immediately following the demo more reliable alternatives were researched. It was determined that the technology at the forefront of the automotive industry, lidar distance sensing, was the only suitable option. In the following weeks, the switchover to lidar was implemented in code, wiring schematics, and mounting. The change proved fruitful, the outputs were significantly more accurate and less susceptible to noise. Moreover, the upper limit of distance sensing was drastically increased from 2 metres to 8 metres providing a much larger window to warn users of incoming vehicles.

Another problem identified by the Teaching assistants was our hardware housing’s striking resemblance to the humble construction block commonly known as a brick. This issue was multifold; the design was not only sharp and unattractive but also too large. This problem was considered at length following the presentation and led the Cyclops team to take on a complete redesign of the internal circuitry and hardware layout. This effort yielded a hardware housing with an approximate 50 percent reduction in volume and a less boxy design which favoured rounded corners.

While digging into technical specifications and the real-world usage of the ride-assist system, the Teaching Assistants identified battery life as a potential source of trouble. At that time, the unit could run for

approximately 30 minutes on a 10,000 mAh battery. Thanks to this insight the Cyclops team made it a priority to begin researching battery-saving methods which could be applied to the raspberry pi development board used. While still an ongoing effort, this issue has begun to be addressed. With all things considered, the largest contributor to poor battery life is the overhead associated with running unneeded functions of the operating system. Namely; the ethernet and power LEDs were disabled, wifi and Bluetooth were disabled, and HDMI output was disabled. Moreover, the LEDs used were exchanged for a more power-efficient led strip. Reaching beyond the raspberry pi, the problem was also addressed at its root - the battery. The size of the battery has been increased. The summation of these changes aims to see battery life improved to a minimum of 2 hours.

8. Code Coverage Metrics

Code coverage is determined by whether a specific sub system and code was exercised or not during the execution of our test suite. For the code coverage, although there exists a lot of unit testing extensions that measure out these metrics for us, it is more accurate and effective for us to calculate these metrics ourselves due to the nature that the majority of our testing is done in a real world environment and limited use of PyTest. Coverage was collected my monitoring each segment of the system while running through all of the mentioned unit tests. We monitor and have set breakpoints for functions, statements, and conditions ensuring they run through during utilization of that specific sub system. It is also noted that we will not be running unit testing for each of the respective constructor (init.py) files for each of the subsystems. Metrics that we are including in our coverage report includes:

- Function coverage: How many functions defined have been called
- Statement coverage: How many of the statements in the in the program have been executed
- Condition coverage: How many of the boolean sub-expressions have been testing for a true and a false value
- Line coverage: How many lines of source code have been tested

Table 1.7: Code Coverage Metrics

File	Statements		Condition		Functions		Lines	
acceleration_plot.py	100%	38/38	100%	1/1	100%	4/4	100%	79/79
acceleration.py	100%	35/35	N/A	N/A	100%	9/9	100%	91/91
video_capture.py	93%	65/70	80%	8/10	100%	4/4	100%	159/159
led.py	100%	14/14	100%	2/2	100%	3/3	100%	36/36
ultrasonic.py	100%	30/30	100%	3/3	100%	5/5	100%	82/82

9. Appendix

9.1. Reflection

The VnV Plan put about a solid foundation upon which the Cyclops team worked to verify and validate the numerous intricate components of the ride assist system. The testing identified in the VnV Plan was a response to the fundamental features required by the implementation chosen. Those features are rear vehicle detection, crash detection, video and data logging, and some baseline usability and stylistic requirements.

As a result of tackling broader requirements in testing specifications as opposed to generating specific tests for specific implementations, the VnV Plan proved to be a robust document that withstood the changes made due to testing. Some of those changes include switching from ultrasonic to lidar distance sensing and decreasing the size of the system. In the first instance clearly predicting what was required was driven by the overarching concept of the system; improving user safety by indicating the distance of objects behind the rider. This brings about an important concept that the Cyclops team learned and will be sure to use in future Verification and Validation Planning; Test cases should be broad enough that implementation shifts will not render them obsolete, yet specific enough that every requirement is sure to be met under their cumulative rule.

Another vital component of specifying testing requirements is the idea of a feasible workload. This concept was considered carefully in the construction of the VnV plan and remains a cornerstone of our testing specifications. Striking a balance between a robust set of tests that fulfill all of our system's requirements while remaining feasible for the members to perform is a tricky task that must be navigated with care. Over the course of the VnV plan and subsequent testing specifications, the Cyclops team formed smaller subgroups to form testing teams. These testing teams worked together to share the load of testing, hold each other accountable to schedules, and vouch for one another when workloads were too high. To maximize efficiency the Cyclops team learned that tests should be designed to maximize their coverage and overlap as minimally as possible with other tests.

Selecting the right tests to prove the required characteristics is yet another vital component of testing design. The Cyclops team came to understand that having the ability to take a step back from the specific test being designed and consider how the project will operate as a whole was the key to this undertaking. To succeed the team must be on the same page when it comes to overarching goals that must be fulfilled and how the product will be used. For example, when designing testing for the hardware housing it is extremely important not to treat the housing as simply a piece of plastic operating in a vacuum. Testing designers must consider the hardware housing operating in the context of the project riding atop the handlebars of a typical bicycle, outside in the elements, moving at speed, and experiencing bumps and jolts. These considerations are what make or break a testing case and demonstrate project vision.