Module Interface Specification

Cyclops Ride Assist: Real-time bicycle crash detection and blindspot monitoring.

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
2023-01-17	Aaron Li, Amos Cheung, Amos Yu, Brian Le, Manny Lemos	Document created
2023-02-02	Amos Cheung	Updated Timeline and Module Hierchary based on peer review
2023-03-22	Manny Lemos	Revision 1 Edit

2. Introduction

This document is the Modular Interface Specification of Cyclops Ride Assist (CRA) system. The purpose of this document is to outline the design specification for each component in CRA and serve as the basis for implementation work when building the system.

3. Purpose

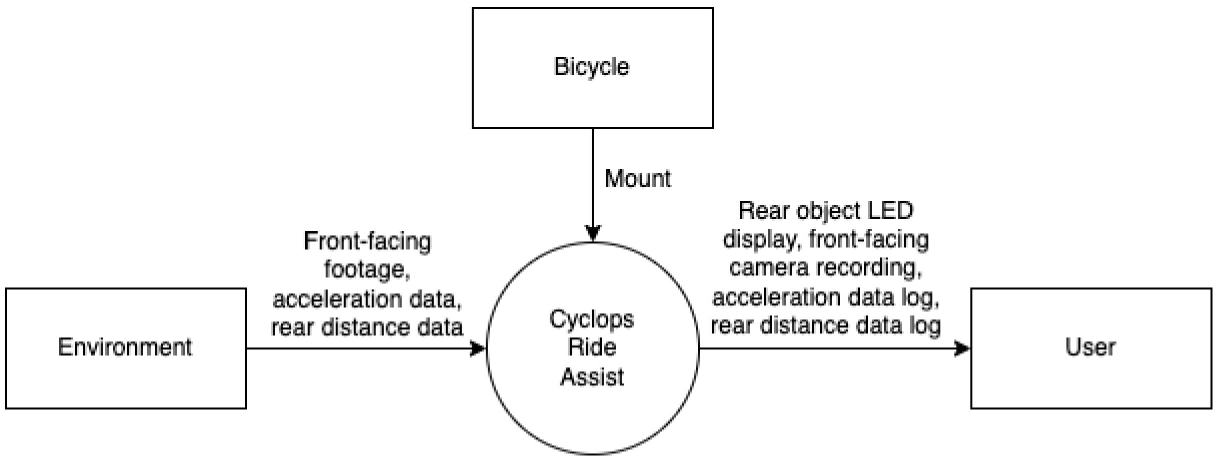
Cyclops Ride Assist (CRA) aims to provide cyclists with a simple-to-use ride monitoring system that decreases the risk of a crash, and provides valuable data in the event of a crash.

4. Scope

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.

4.1. System Context

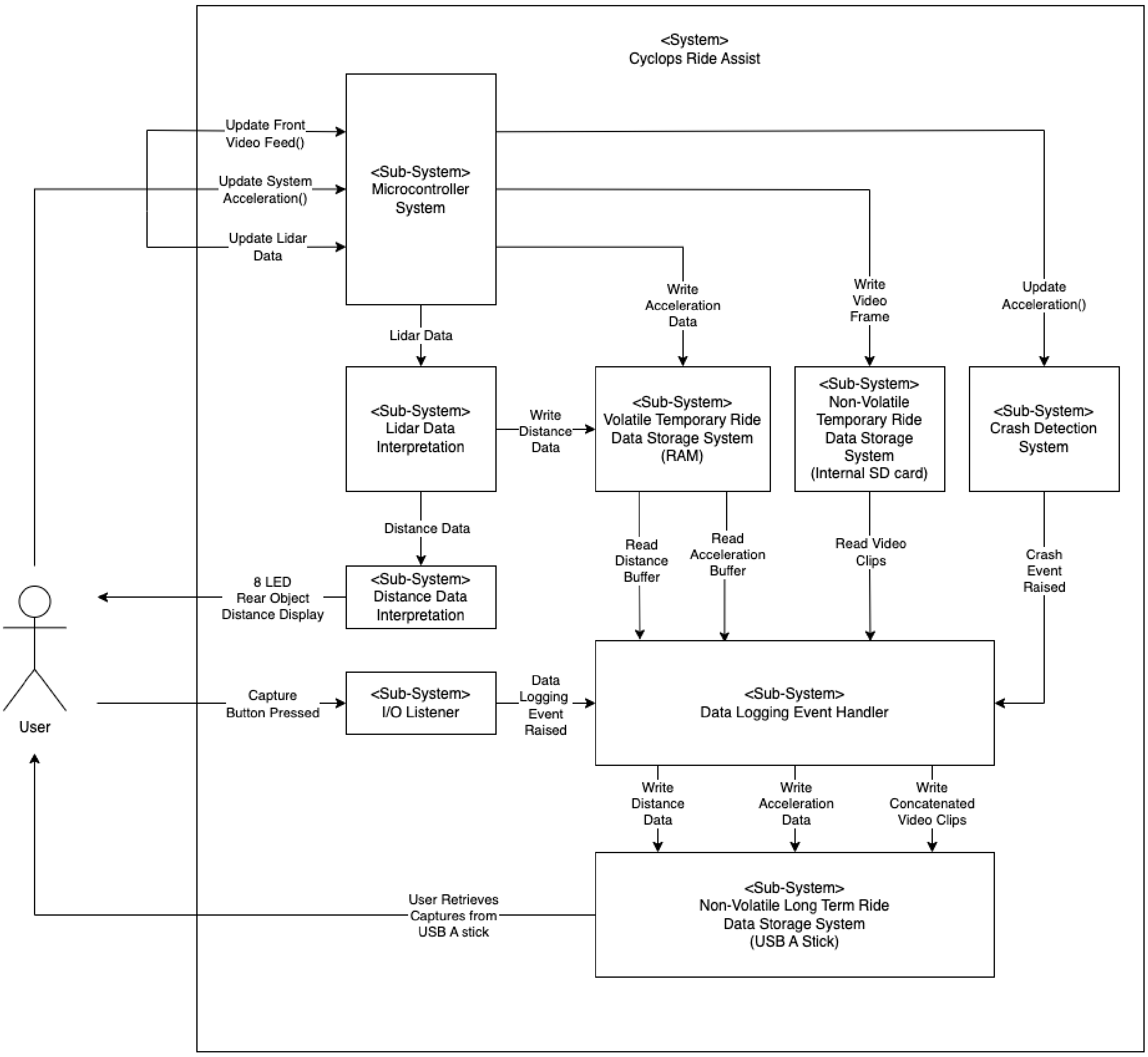
Figure 4.1.1: CRA System Context Diagram


Bicycle
Mount
Front-facing footage, acceleration data, rear distance data
Environment
Cyclops Ride Assist
Rear object LED display, front-facing camera recording, acceleration data log, rear distance data log
User

5. Project Overview

5.1. Functional Decomposition

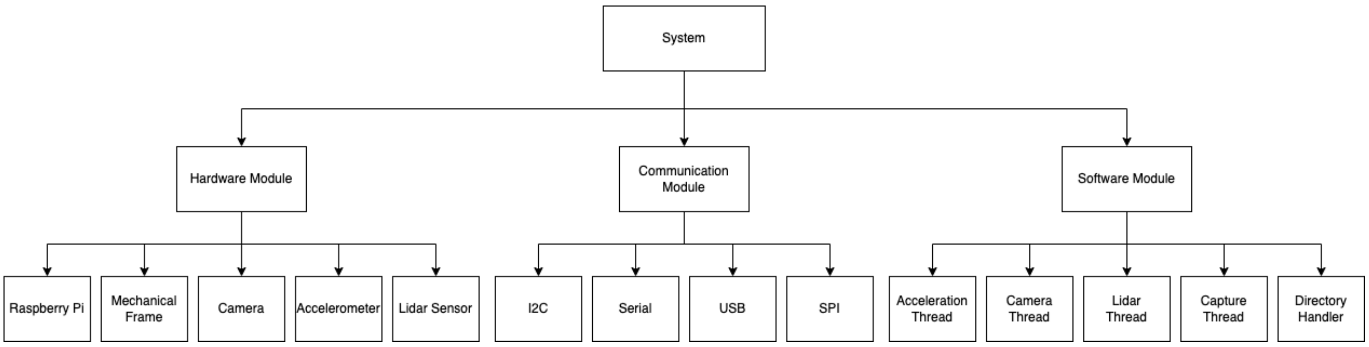
Figure 5.1.1: CRA Functional Decomposition Diagram



<System>
Cyclops Ride Assist
Update Front Video Feed()
Update System Acceleration()
Update Lidar Data
<Sub-System> Microcontroller System
Lidar Data
Write Acceleration Data
Write Video Frame
Update Acceleration()
<Sub-System> Lidar Data Interpretation
Write Distance Data
<Sub-System> Volatile Temporary Ride Data Storage System (RAM)
<Sub-System> Non-Volatile Temporary Ride Data Storage System (Internal SD card)
<Sub-System> Crash Detection System
Distance Data
Read Distance Buffer
Read Acceleration Buffer
Read Video Clips
Crash Event Raised
8 LED Rear Object Distance Display
<Sub-System> Distance Data Interpretation
User
Capture Button Pressed
<Sub-System> I/O Listener
Data Logging Event Raised
<Sub-System> Data Logging Event Handler
Write Distance Data
Write Acceleration Data
Write Concatenated Video Clips
User Retrieves Captures from USB A stick
<Sub-System> Non-Volatile Long Term Ride Data Storage System (USB A Stick)

5.2. Module Hierarchy

Figure 5.2.1: CRA Module Hierarchy Diagram



6. System Variables

6.1. Monitored and Controlled Variables

The following are a list of variables that are to be monitored.

Monitored Var	Monitored Type	Range	Units	Comments
poweroff_button_pushed	Boolean	N/A	N/A	GPIO event to respond to the press of the poweroff button.
capture_button_pushed	Boolean	N/A	N/A	GPIO event to respond to the press of the capture button.
capture	N/A	N/A	N/A	Video capture object which is queried for video frames, and monitored for errors.
accelerometer	N/A	N/A	N/A	Accelerometer sensor object which is queried for acceleration readings, and monitored for errors.
x	Size	[-160, 160]	m/s^2	The first acceleration vector provided by the accelerometer sensor.
y	Size	c	m/s^2	The second acceleration vector provided by the accelerometer sensor.
z	Size	c	m/s^2	The final acceleration vector provided by the accelerometer sensor.
norm	Size	c	m/s^2	Vector sum of the 3 dimensions of acceleration.
Lidar_serial	N/A	N/A	N/A	Lidar sensor object which is queried for distance readings, and monitored for errors.
distance	Size	[20, 800]	cm	Used to store the current distance retrieved from the Lidar sensor.

The following are a list of variables that are to be controlled.

Controlled Var	Controlled Type	Range	Units	Comments
poweroff_event	Boolean	N/A	N/A	Event which can be set to safely terminate all threads and power down the pi.
crash_event	Boolean	N/A	N/A	Event which is set based on accelerometer data parsed by a crash detection algorithm. Results in a 10-second wait followed by data logging.

Controlled Var	Controlled Type	Range	Units	Comments
c_capture_event	Boolean	N/A	N/A	Camera capture event which is set when data logging is intended to occur. Notifies camera thread to concatenate and log its clips.
a_capture_event	Boolean	N/A	N/A	Accelerometer capture event which is set when data logging is intended to occur. Notifies accelerometer thread to export its buffer of data.
l_capture_event	Boolean	N/A	N/A	Lidar capture event which is set when data logging is intended to occur. Notifies Lidar thread to export its buffer of data.
fps	Size	[1, 60]	Frames Per Second	Number of captures per second
frame_width	Size	[640, 1920]	Pixels	Number of pixel in capture width
frame_height	Size	[480, 1080]	Pixels	Number of pixel in capture height
video_length	Time	[0 - 60]	Seconds	The length in seconds of the requested video
accelerometer_sample_rate	Size	[10, 3200]	Samples Per Second	Number of accelerometer sensor readings per second
accelerometer_data	Entries	N/A	N/A	A FIFO queue which retains the last 60 seconds of accelerometer data.
Lidar_sample_rate	Size	[1, 250]	Samples Per Second	Number of Lidar distance sensor readings per second
baud_rate	Size	[9600, 921600]	Bits Per Second	Rate of data transfer in serial communication channel.
Lidar_data	Entries	N/A	N/A	A FIFO queue which retains the last 60 seconds of Lidar data.

6.2. Constants

Constant Var	Constant Type	Value	Units	Comments
g	Acceleration	9.81	m/s^2	Acceleration due to gravity

7. User Interfaces

7.1. Inputs

Input Name	Input Type	Range	Units	Comments
Power On Button	Physical	[0, 1]	N/A	Button that is used to power on the system and automatically run the CRA software.
Power Off Button	Physical	[0, 1]	N/A	Button that is used to power off the system.
Capture Button	Physical	[0, 1]	N/A	Button that is used to manually capture an event
Mount	Physical	N/A	N/A	Mount used to secure Cyclops to the bike.
Battery Port	Physical	N/A	N/A	Port for charging the battery bank.
SD Storage Device Port	Physical	N/A	N/A	Port for inserting the SD Card.
USB-A Storage Device Port	Physical	N/A	N/A	Port for inserting the USB-A stick.

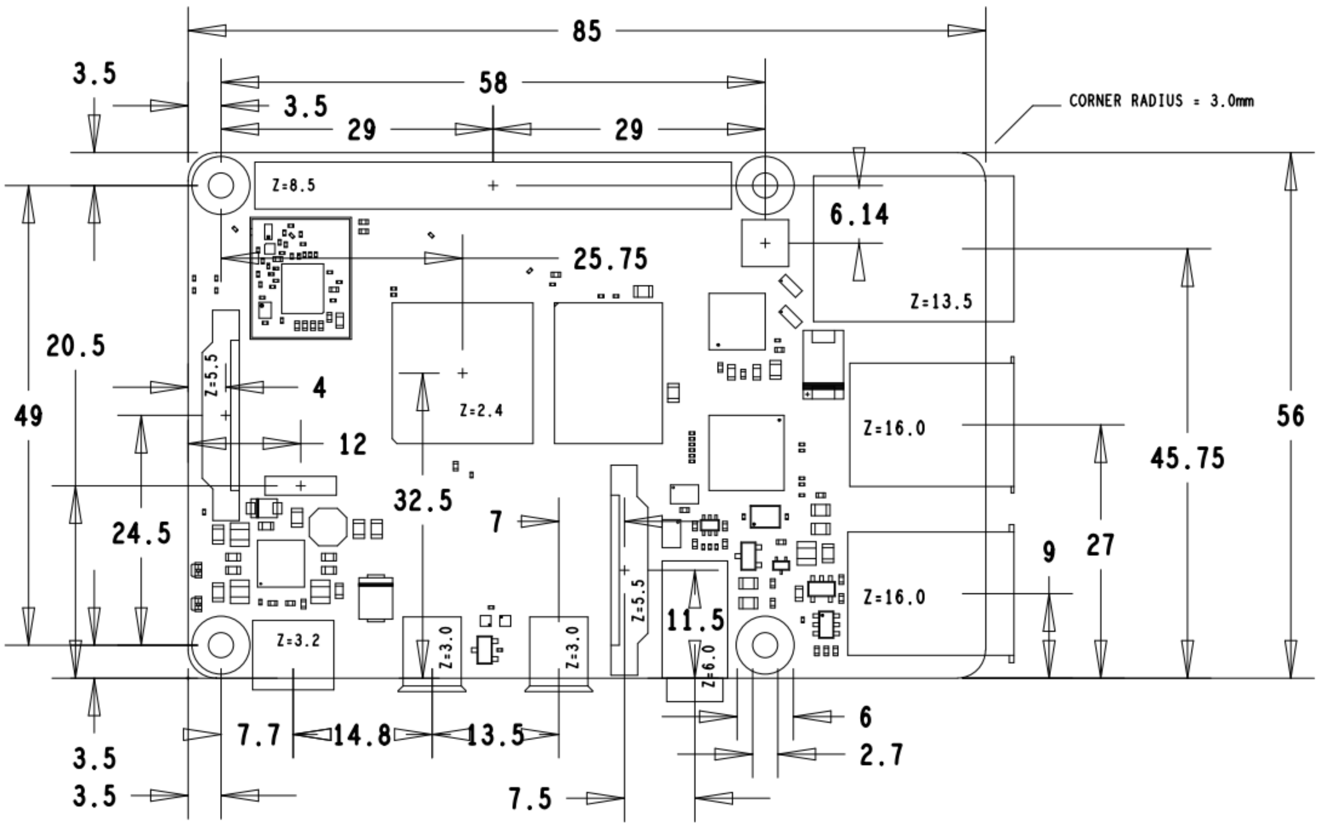
7.2. Outputs

Output Name	Output Type	Range	Units	Comments
LED Strip	Visual	[0, 8]	N/A	8 addressable RGB LEDs are used to communicate with the user. Their primary purpose is to display the distance of an object behind the user as a function of colour and number of LEDs turned on. Moreover, the LEDs also notify users of CRA powering on, detecting a crash, performing a data capture, and powering off.
USB-A stick	Physical	N/A	N/A	The USB-A stick is used to store capture folders containing the last 60 seconds of video capture, acceleration data, and rear object distance data.

8. Mechanical Hardware

8.1. Raspberry Pi Mechanical Specifications

Figure 8.1.1: Raspberry Pi Mechanical Drawing and Schematic [1]



The Raspberry Pi 4 Model B is the microcontroller that will be used for all software and processing. The mechanical specifications of the Raspberry Pi is as follows:

Raspberry Pi Mechanical Specifications	Value
Dimensions	85mm x 56mm
CPU	Quad core Cortex-A72 (ARM V8)
Ports	USB3, USB2, USB-C, HDMI, MicroSD
RAM	4GB
Operating Temperature	0-50C
Audio	4-Pole Stereo Audio
Video	Composite Video

The requirements traceability of the Raspberry Pi is as follows:

Module	Functional Requirements	Non-Functional Requirements
Raspberry Pi	CFR11	CNFR6, CNFR11, CNFR20, CNFR28, CNFR29, CNFR31, CNFR33, CNFR36, CNFR37

8.2. Polylactic Acid (PLA) Material

The mechanical chassis and frame will be created using Polylactic Acid (PLA) filament, molded through a Prusa i3 MK3s 3D printer.

The color of the PLA material was chosen to be white in order to reflect the NFRs outlined in the SRS document, specifically CNFR1.

The mechanical specifications of PLA are as follows [2]:

PLA Specifications	Value
Heat Deflection Temperature	52C
Density	1.24 g/cm3
Tensile Strength	50MPa

8.3. Mechanical Design

The mechanical chassis and frame is shown below in Figure 8.3. Each module outlined in the above sections will be fitted with a frame or mount to allow for ease-of-use and protection.

The mechanical design was created using PLA and the Prusa i3 MK3s 3D printer.

Figure 8.3.1: Raspberry-Pi Housing and Mount Assembly

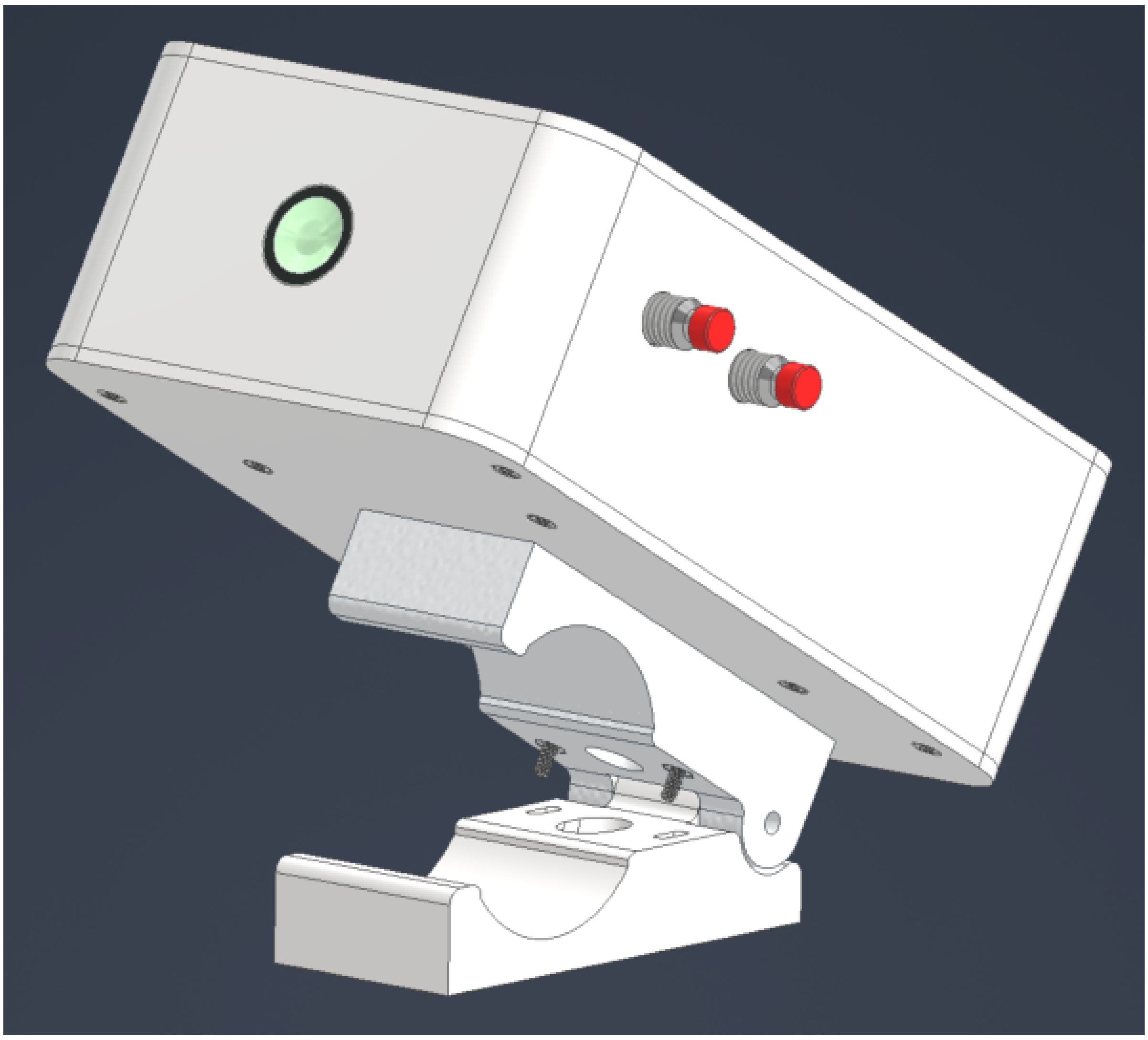
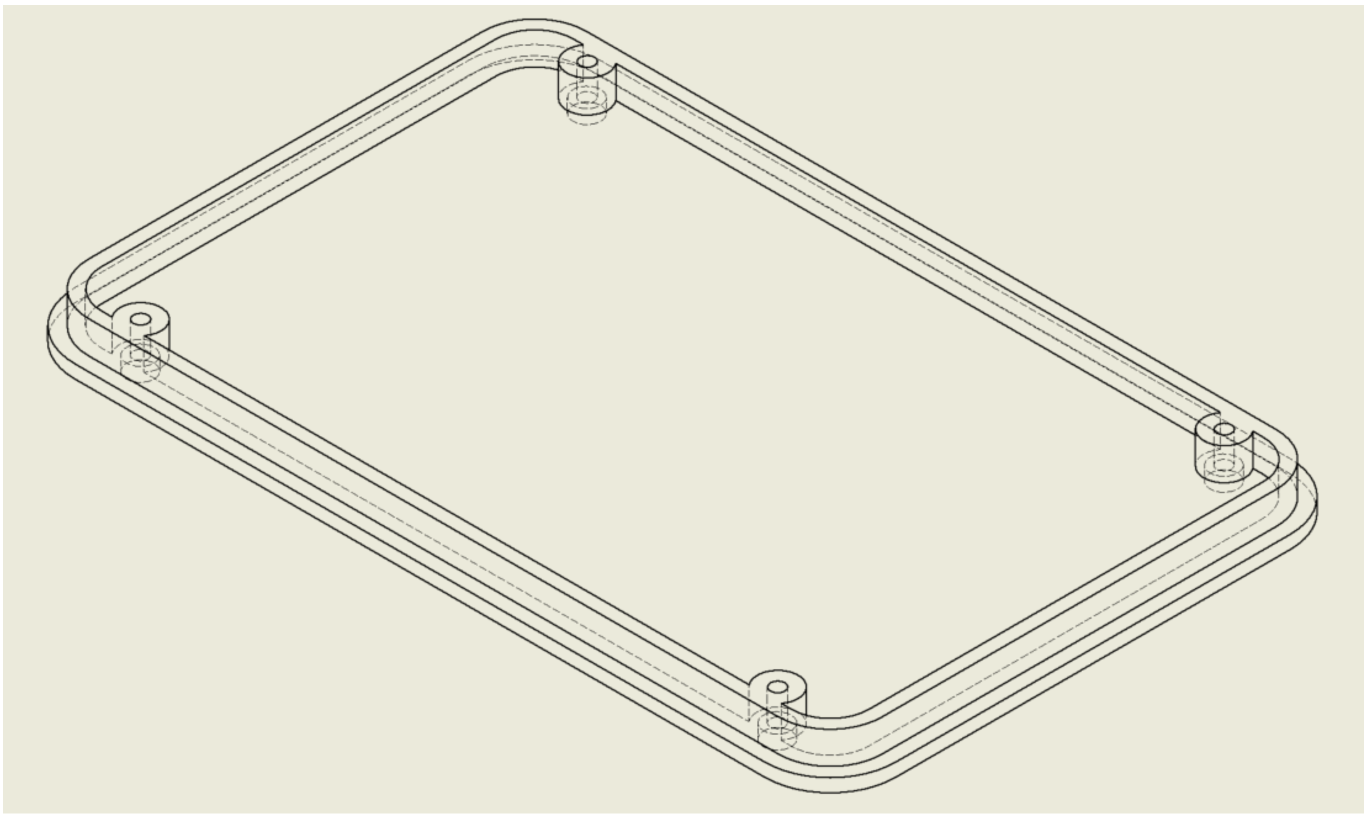
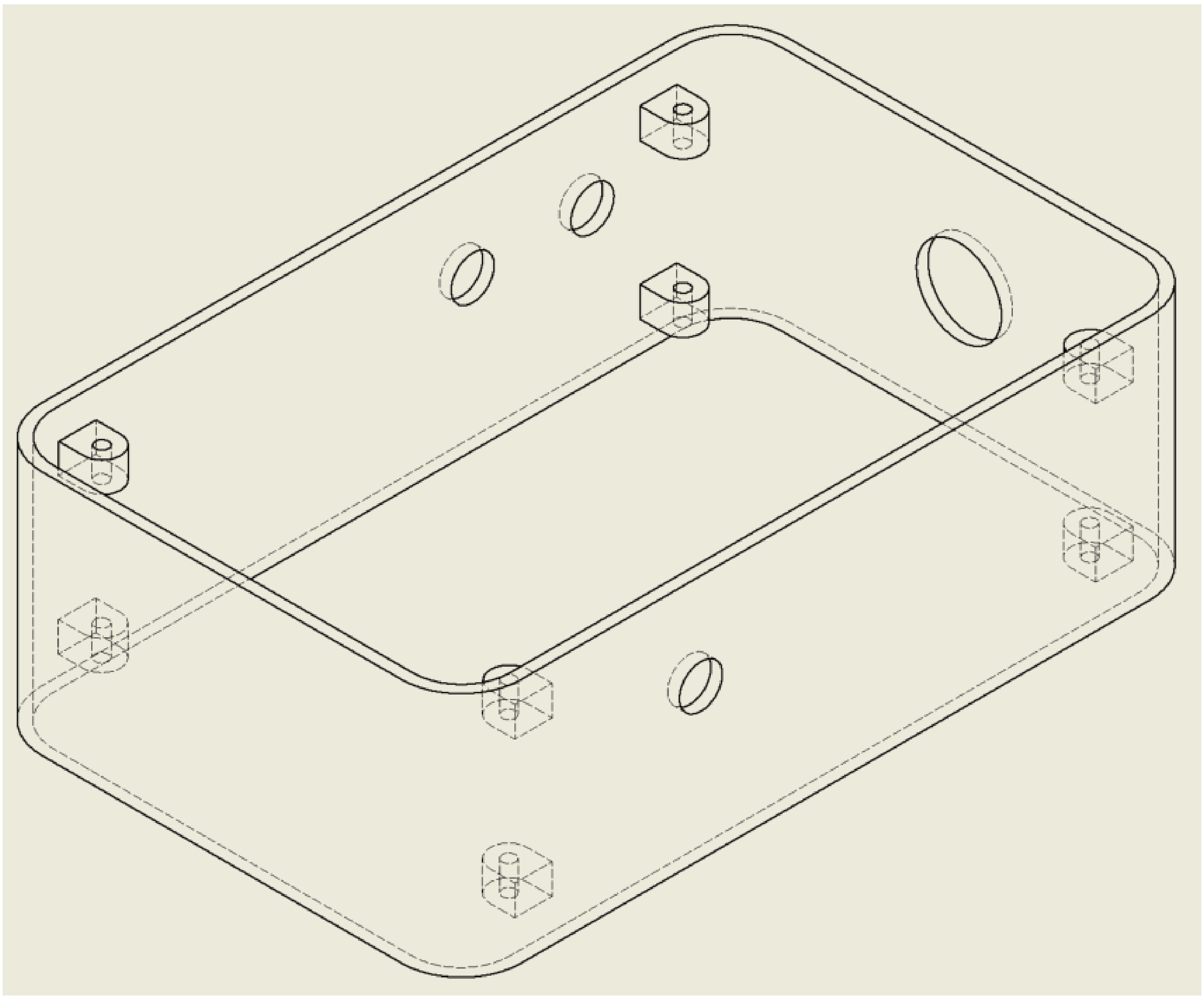


Figure 8.3.2: Raspberry-Pi Housing Top Component



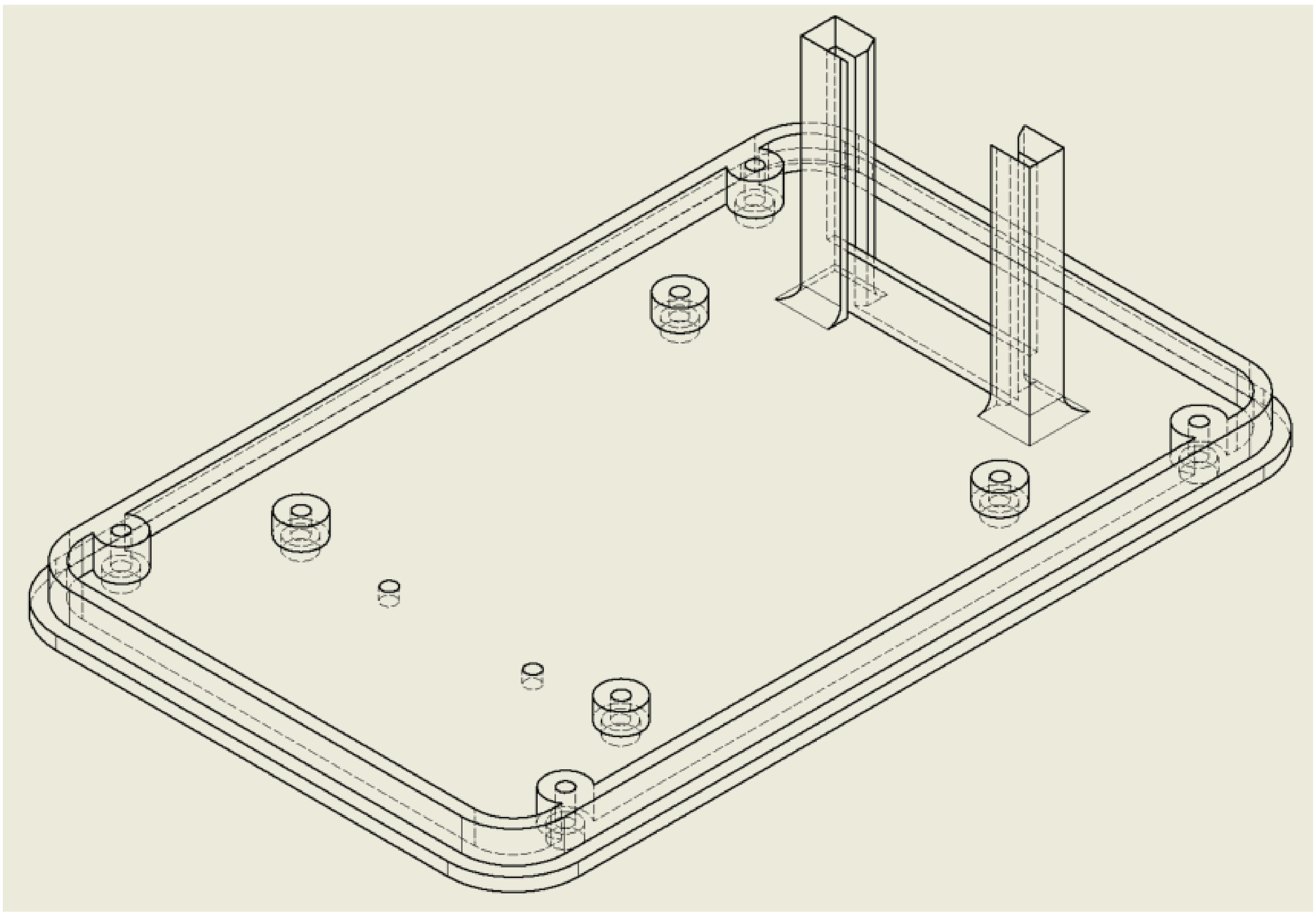
Raspberry-Pi Housing Top Component Specification	Value
Outer Dimensions	6.9 mm x 80 mm x 125 mm
Weight	8 grams
Material	PLA

Figure 8.3.3: Raspberry-Pi Housing Middle Component



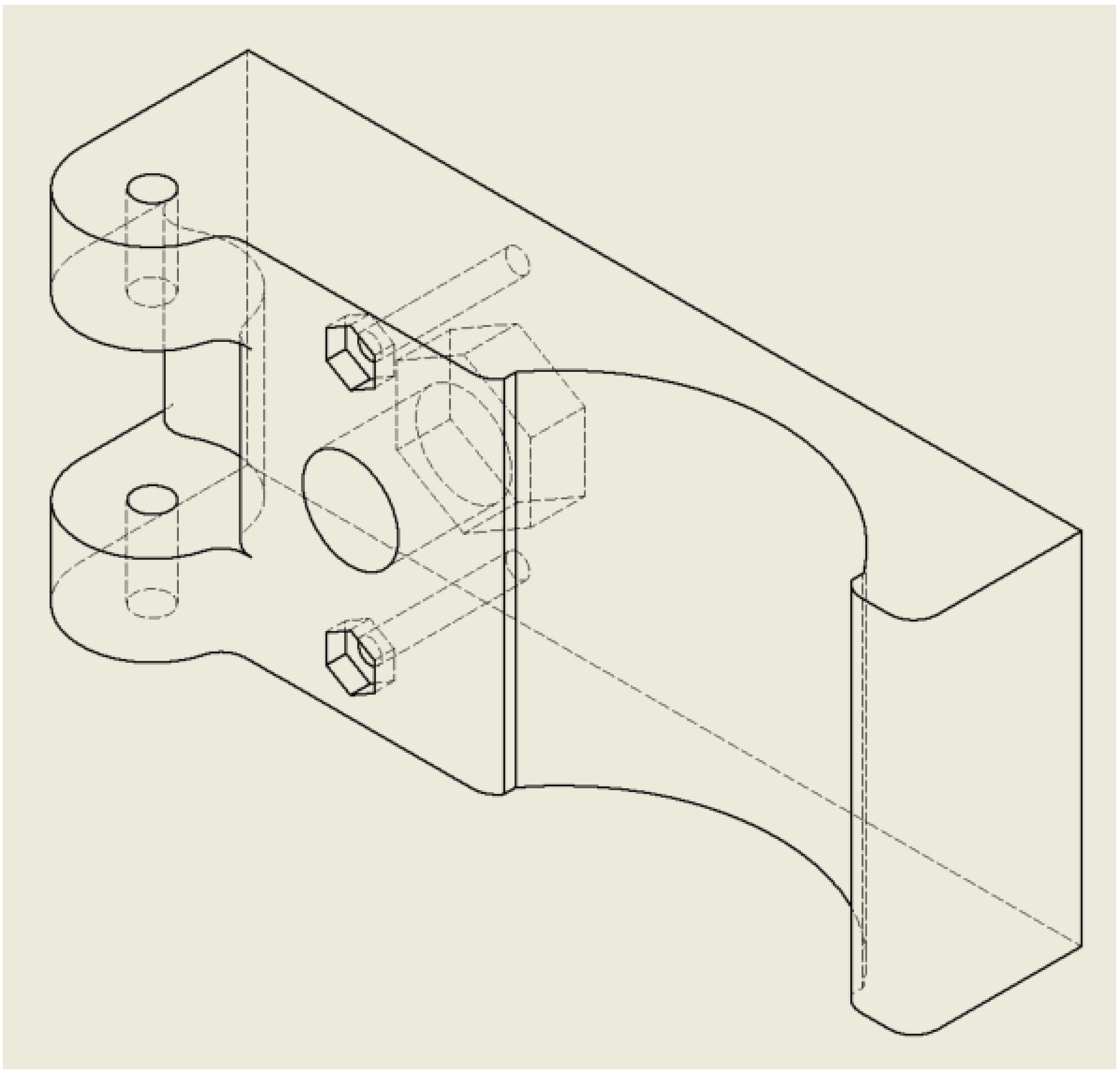
Raspberry-Pi Housing Middle Component Specification	Value
Outer Dimensions	41 mm x 80 mm x 125 mm
Weight	9 grams
Material	PLA

Figure 8.3.4: Raspberry-Pi Housing Bottom Component



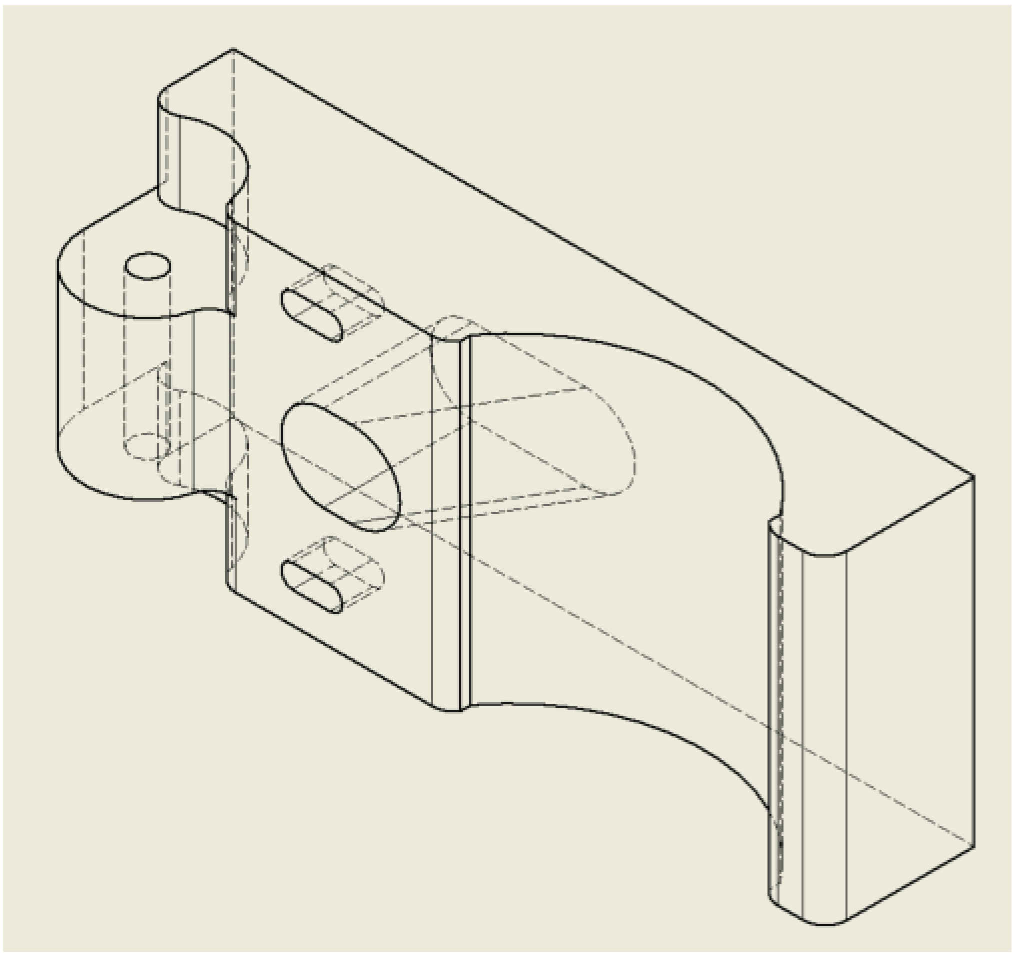
Raspberry-Pi Housing Bottom Component Specification	Value
Outer Dimensions	6.9 mm x 80 mm x 125 mm
Weight	8 grams
Material	PLA

Figure 8.3.5: Raspberry-Pi Housing Mount Top Component



Raspberry-Pi Housing Mount Top Component Specification	Value
Outer Dimensions	30 mm x 69.5 mm x 20 mm
Weight	8 grams
Material	PLA

Figure 8.3.6: Raspberry-Pi Housing Mount Bottom Component



Raspberry-Pi Housing Mount Bottom Component Specification	Value
Outer Dimensions	30 mm x 69.5 mm x 20 mm
Weight	8 grams
Material	PLA

Figure 8.3.7: Lidar Housing and Mount Assembly

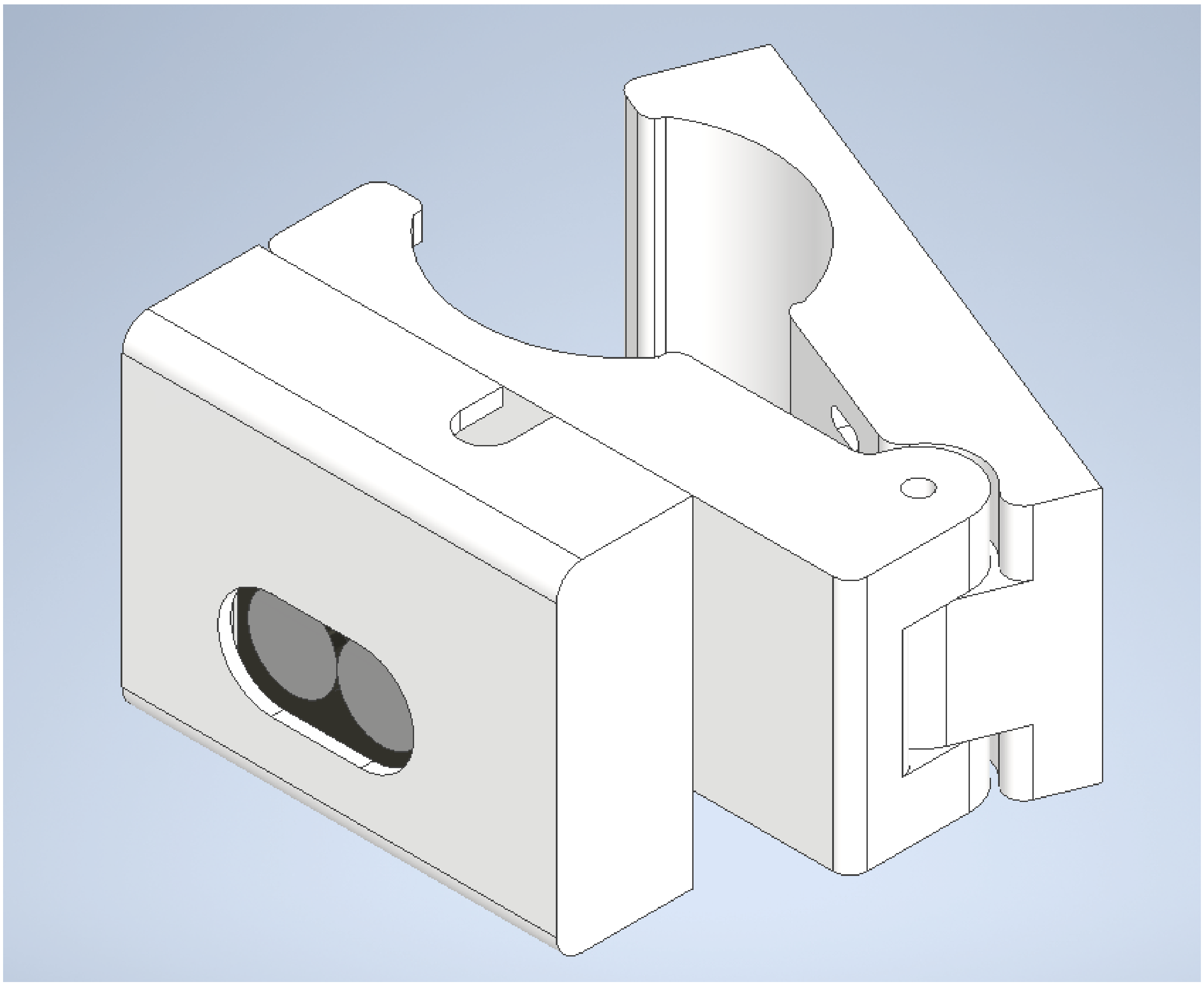
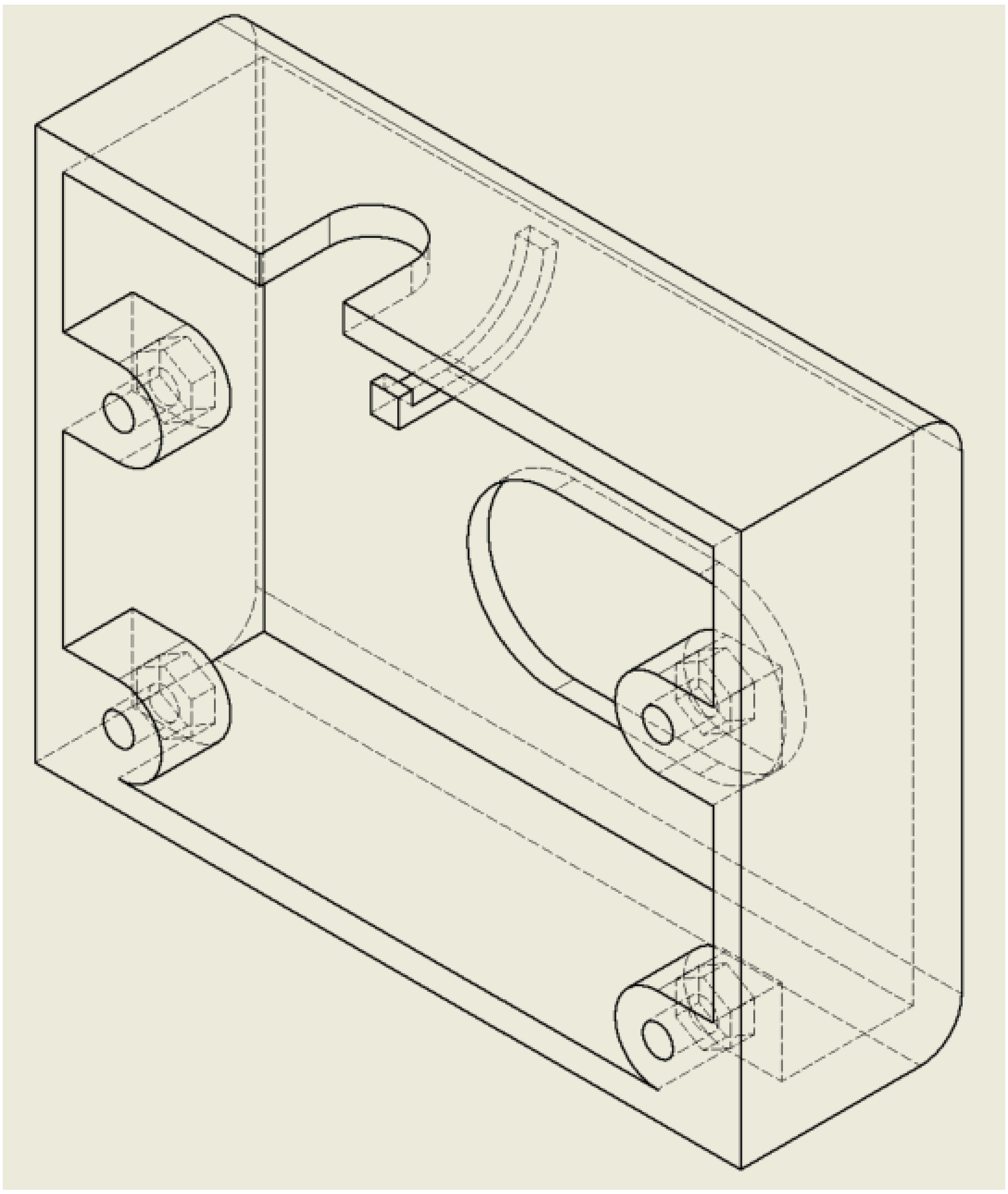
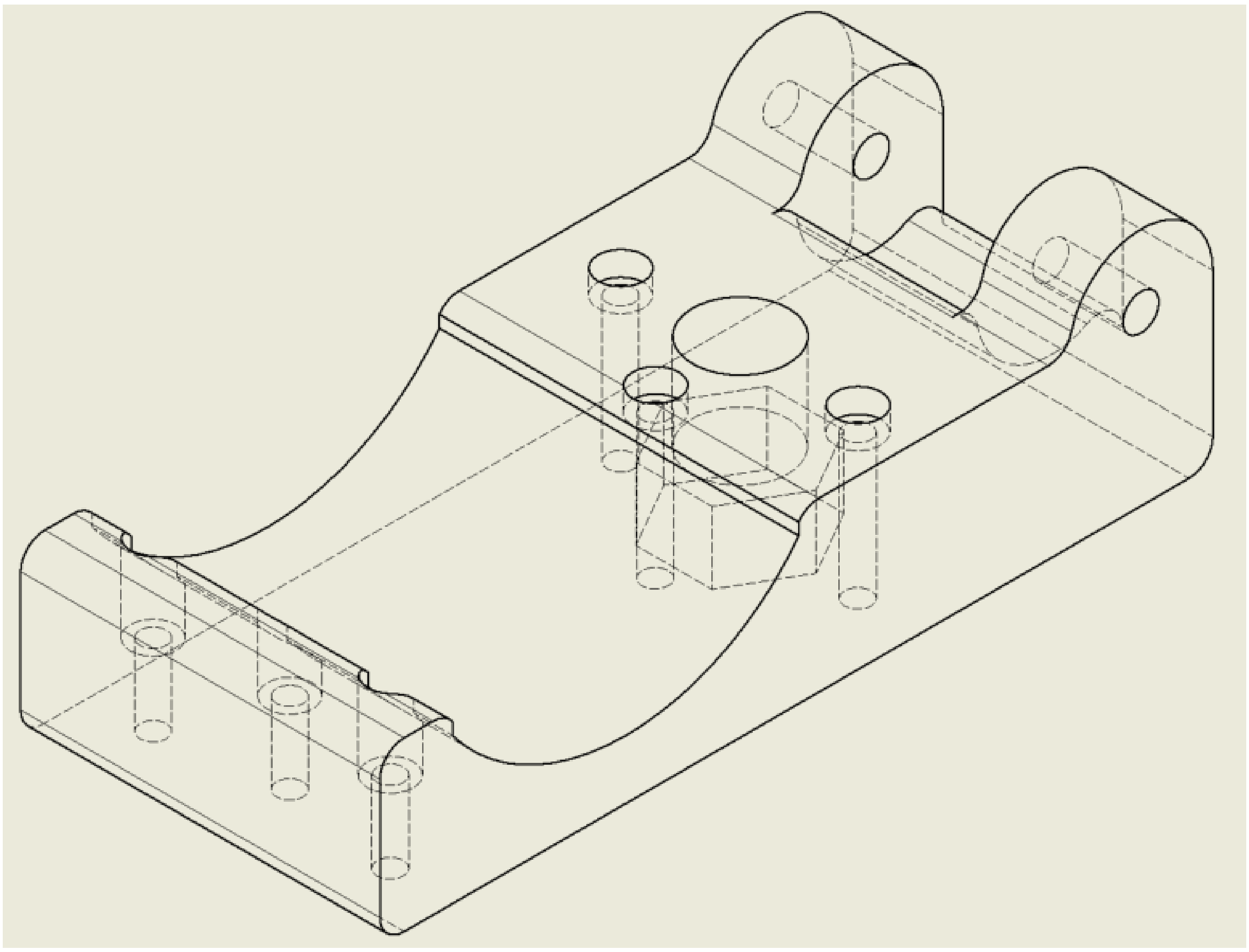


Figure 8.3.8: Lidar Housing Component



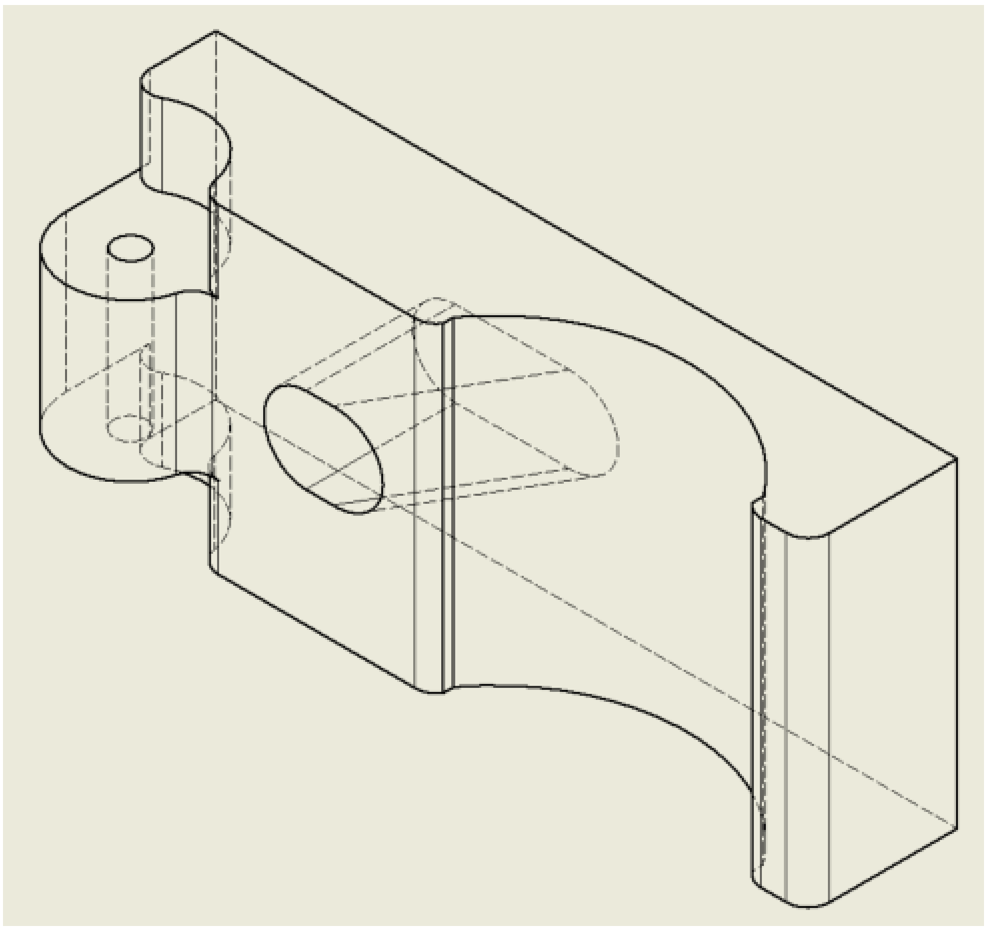
Lidar Housing Component Specification	Value
Outer Dimensions	40 mm x 51 mm x 16 mm
Weight	MASS grams
Material	PLA

Figure 8.3.9: Lidar Housing Mount Top Component



Lidar Housing Mount Top Component Specification	Value
Outer Dimensions	30 mm x 69.5 mm x 20 mm
Weight	8 grams
Material	PLA

Figure 8.3.10: Lidar Housing Mount Bottom Component



Lidar Housing Mount Bottom Component Specification	Value
Outer Dimensions	30 mm x 69.5 mm x 20 mm
Weight	8 grams
Material	PLA

The requirements traceability of the mechanical frames are as follows:

Module	Functional Requirements	Non-Functional Requirements
Mechanical Frame	n/a	CNFR1, CNFR2, CNFR3, CNFR4, CNFR5, CNFR9, CNFR11, CNFR14, CNFR21, CNFR22, CNFR23, CNFR28, CNFR29, CNFR32, CNFR33, CNFR35, CNFR40, CNFR48, CNFR49

9. Electrical Components

9.1. CRA Electrical Specifications

Figure 9.1.1: CRA Circuit Diagram

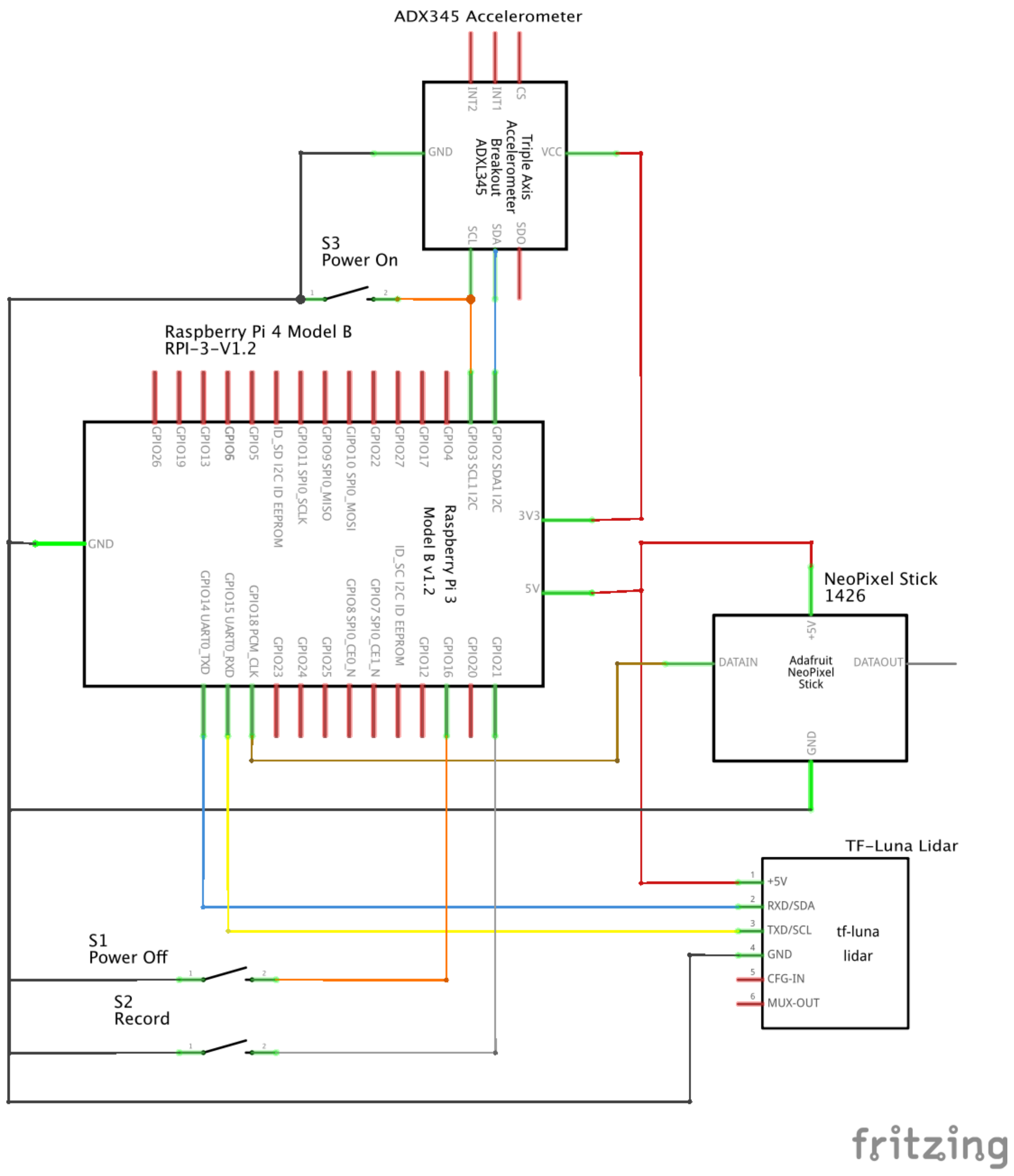
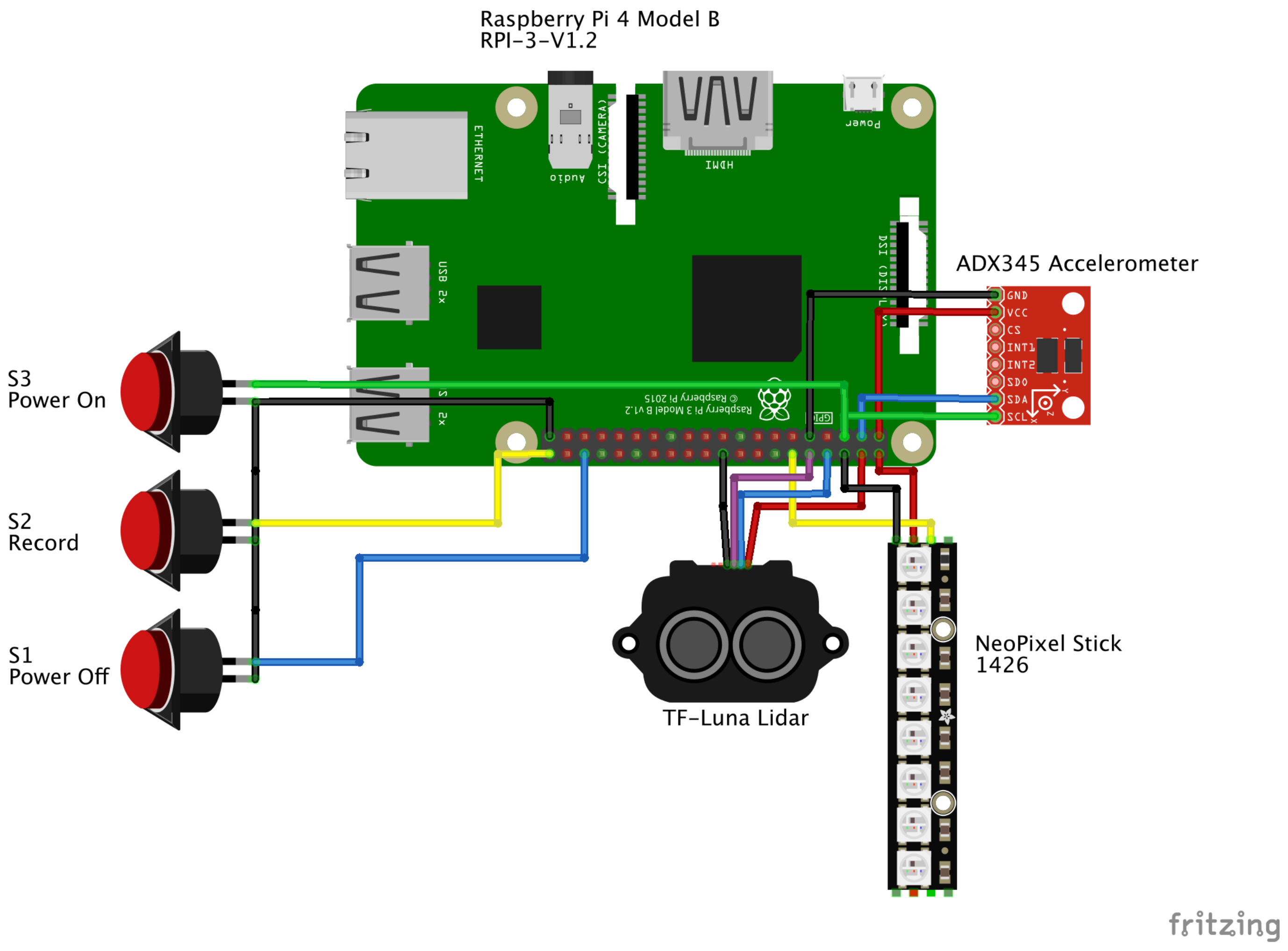


Figure 9.1.2: CRA Breadboard Schematic



9.2. Raspberry Pi Electrical Specifications

The Raspberry Pi 4's reduced electrical schematic can be seen below. Using various pins and ports provided, CRA will be able to accomplish what is set out in the scope.

Figure 9.2.1: Raspberry Pi 4 Model B Circuit Diagram [3]

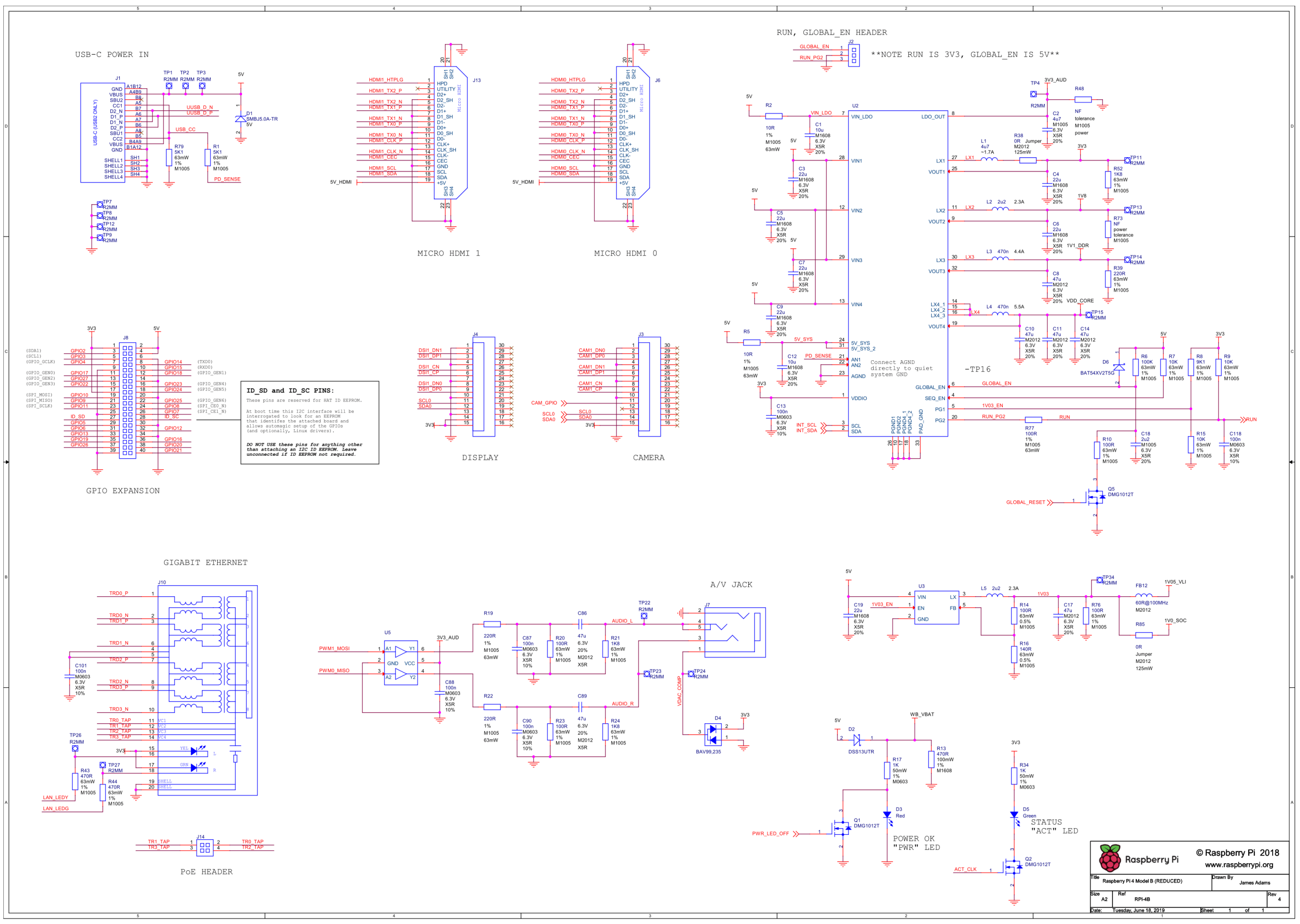
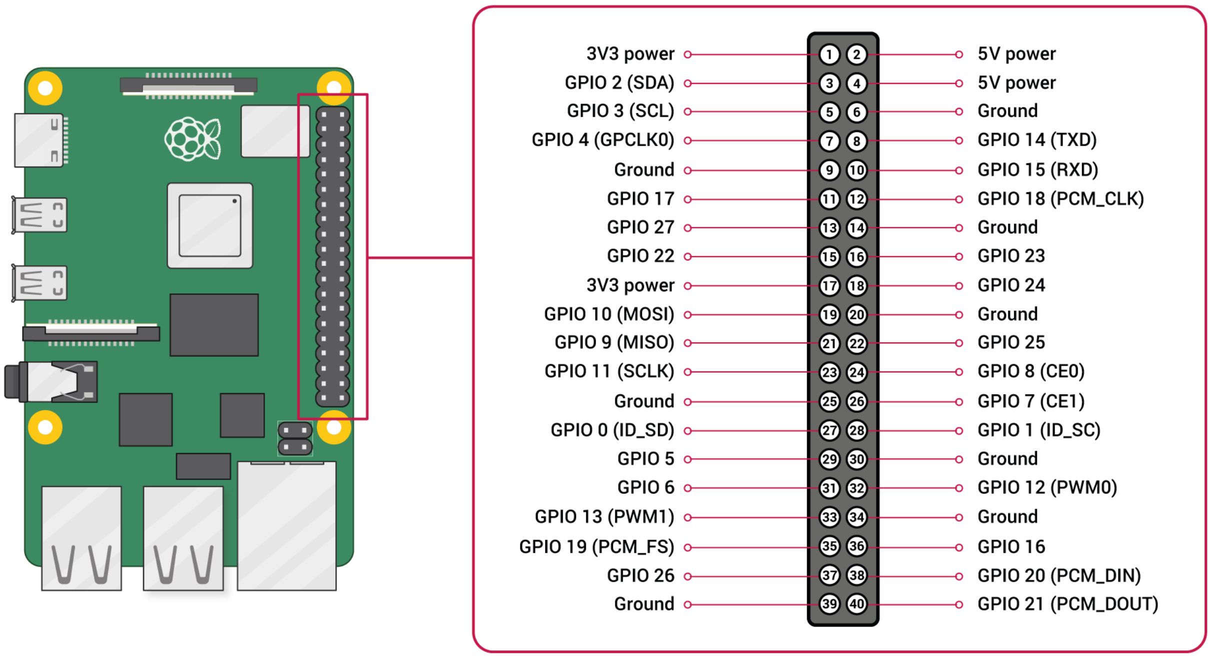


Figure 9.2.2: Raspberry Pi 4 Model B Pinouts



Raspberry Pi Electrical Specifications	**Value**
Microchip	64-bit SoC @ 1.5GHz

Raspberry Pi Electrical Specifications	Value
Pins	40 Pin GPIO Header
Voltage	5V DC via USB-C/GPIO Header
Amperage	3A

9.3. Accelerometer Specifications

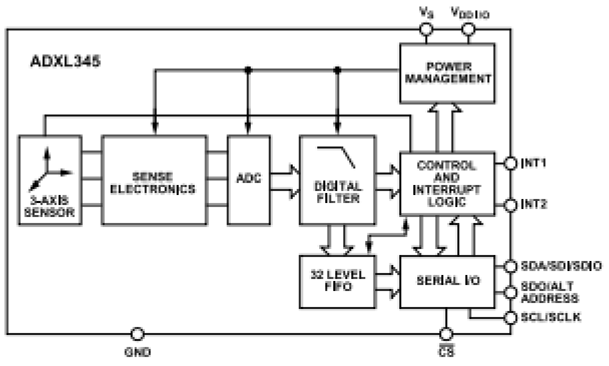
The accelerometer will be used to determine when a crash has occured. The accelerometer used is the ADXL-345 and the specifications are as follows, as outlined by Analog Devices [4].

Accelerometer Specifications	Value
Voltage Range	2.0V to 3.6V
Interfaces	SPI and I2C
Temperature Range	-40C to 85C
Axis	3-Axis (X,Y,Z)
Resolution	10-bit

The requirements traceability of the accelerometer is as follows:

Module	Functional Requirements	Non-Functional Requirements
Accelerometer	CFR3, CFR5	CNFR15, CNFR17, CNFR24, CNFR25, CNFR29, CNFR33, CNFR35

Figure 9.4.1: Accelerometer Sensor Diagram [4]



9.4. Lidar Sensor Specifications

The Lidar sensor will be used to determine the distance of objects approaching CRA from the rear. CRA makes use of the TF-Luna Lidar sensor and is specified in its documentation [5] by the following.

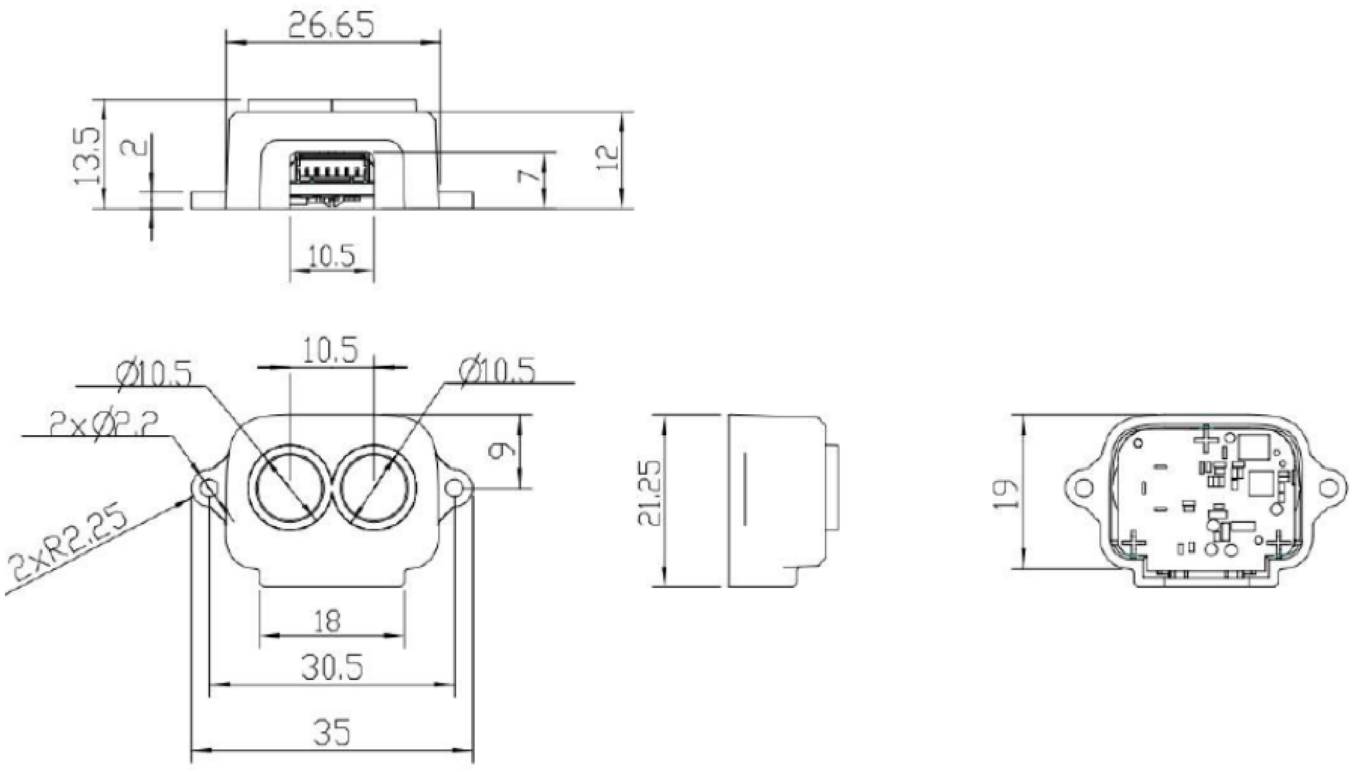
Radar Sensor Specifications	Value
Dimensions	45mm x 20mm x 15mm

Radar Sensor Specifications	Value
Operating Range	0.2-0.8m
Accuracy	±2%
Range Resolution	1cm
FOV	2°
Frame Rate	1-250Hz
Supply Voltage	5V
Average Power	350mW

The requirements traceability of the Lidar sensor is as follows:

Module	Functional Requirements	Non-Functional Requirements
Lidar sensor	CFR1, CFR2	CNFR2, CNFR4, CNFR15, CNFR18, CNFR29, CNFR33

Figure 9.5.1: Lidar Sensor Diagram [5]



9.5. Camera Specifications

The camera will be used to continuously record sequential clips such that the last 60 seconds of footage is always available to be permanently saved. The camera that is being used is the Ootoking 1080p webcam, that can be referenced on Amazon.ca [6].

Camera Specification	Value
Dimension	3cm x 4cm x 2cm

Camera Specification	Value
Weight	55g
Resolution	1080p

The requirements traceability of the camera is as follows:

Module	Functional Requirements	Non-Functional Requirements
Camera	CFR2, CFR6, CFR12	CNFR2, CNFR7, CNFR15, CNFR27, CNFR33, CNFR35

10. Communication Protocols

The requirements traceability of all communication protocols is as follows:

Modules	Functional Requirements	Non-Functional Requirements
USB, Wi-Fi, I2C	CFR9, CFR10	CNFR16, CNFR19, CNFR36, CNFR40, CNFR42. CNFR43, CNFR44, CNFR45

10.1. USB Protocol

In order to communicate and transmit data, the USB protocol will be used. The following ports will be mainly used for the following:

USB	Purpose
USB 3.0	The USB 3.0 ports will be used to connect a camera and a removable USB-A stick. During development other devices like keyboards, mice, external storage devices, and other computers will be connected to these ports to allow for greater ease-of-use and accessibility.
USB-C	The USB-C port will be used to power CRA via battery packs.

10.2. Wi-Fi Protocol

The Wi-Fi protocol is the 802.11ac protocol. This protocol allows for connections from frequencies of 2.4GHz or 5.0Ghz. This protocol will be used for the following.

Wifi Protocol	Purpose
802.11ac	This wireless protocol will allow the users to connect to the CRA remotely using SSH, VNC, and SFTP. Furthermore, this protocol allows users to connect their CRA to either a 2.4 or 5.0GHz wireless network.

10.3. I2C Protocol

The I2C protocol will be used for communication with the accelerometer.

I2C Protocol	Purpose
I2C	Enables peripheral chips like the accelerometer to communicate with the controller chip in a synchronous packet switched configuration.

10.4. Serial Protocol

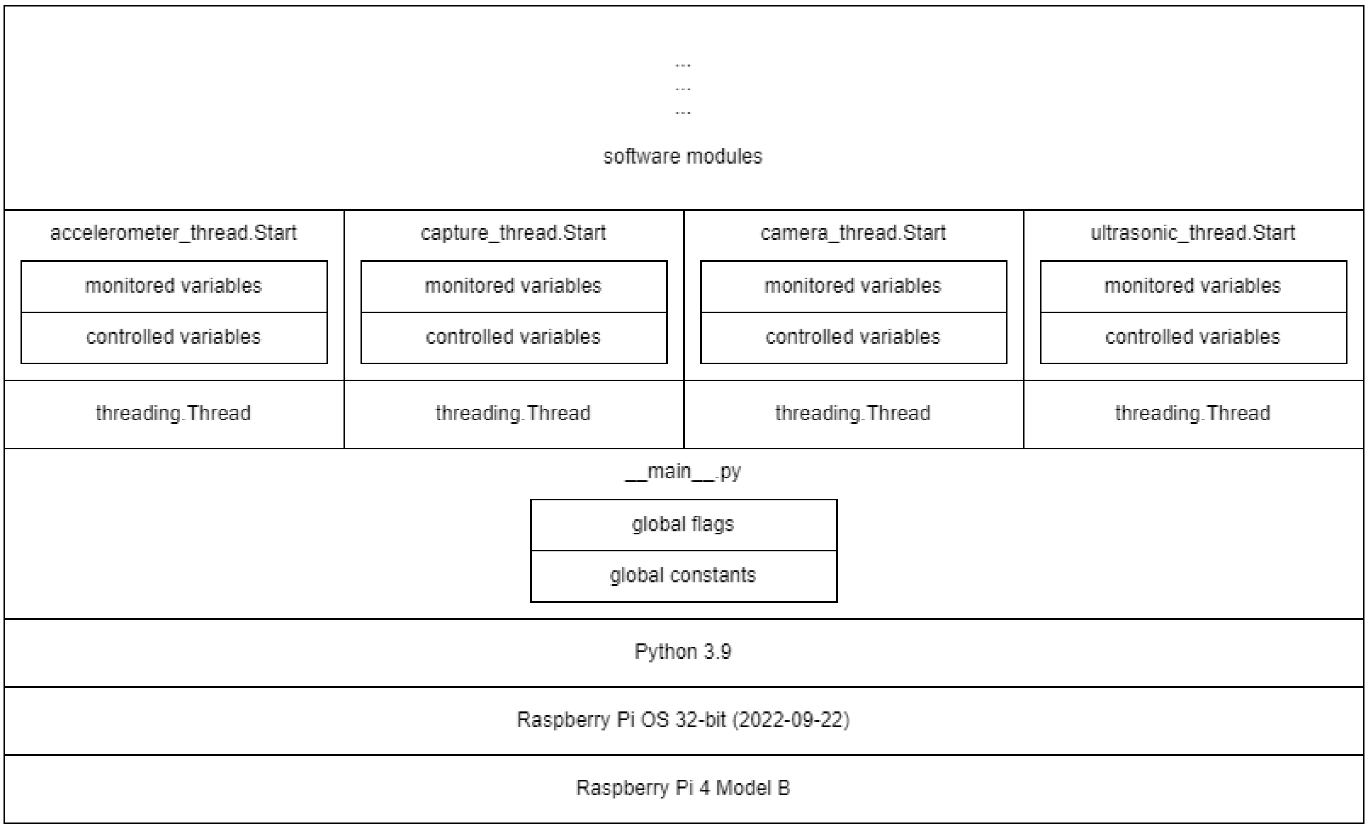
Bitwise continuous communication protocol which is used in two sensor applications of the CRA system.

Serial Protocol	Purpose
UART	Asynchronous communication interface used for short distance transmissions. The UART protocol will be used for communication with the Lidar distance sensor.
SPI	Synchronous communication interface used for short distance transmissions. The SPI protocol will be used for communication with the 8 LED pixel strip.

11. Software Modules

The main entry point of the program is the `__main__.py` script. All software modules are classes implemented in Python 3.9. Software modules are constructed and executed by threaded `Start` classes. Each module is instantiated by `__main__.py` and communicates with other threads via thread events. Upon startup of the Raspberry Pi, `__main__.py` should immediately begin execution. To shut down the Raspberry Pi, `__main__.py` should pass control to the OS to do a safe poweroff and terminate.

Figure 11.0.1: CRA Software Stack



11.1. video_capture.py

Module Implementation

Composed of a class Threaded_Video_Writing. Provides the functionality of grabbing frames from a camera, writing said frames to a temporary non-volatile storage location, and logging the last 60 seconds of footage to a permanent storage location.

Module Secrets

- Video capture instance.
- Frames of video clips.
- Video codec.
- Temporary non-volatile storage location
- Video Resolution
- Output file type.

Module Relationships

Receives the following Threaded_Video_Writing class construction parameters through `__init__()`.

Parameter	Type	Description
fps	integer	Number of frames requested from camera every second.

Likely Changes

- Reduce the number of input parameters to the instantiation of the Threaded_Video_Writing class. These inputs will be replaced by fixed parameters which are deemed most suitable.
  - This change was implemented
- Implement hardware accelerated video encoding.
  - This will likely enable higher resolution video capture
  - This change has not yet been implemented

Unlikely Changes

- Change the camera being used from a USB web-camera to the Raspberry Pi Camera Module.
  - Subsequently, a large portion of the code for the Threaded_Video_Writing class will have to be rewritten.
  - This change has not yet been implemented

Traceability

The requirements traceability of the video_capture.py class is as follows:

Module	Functional Requirements	Non-Functional Requirements
video_capture.py	CFR2, CFR8, CFR9, CFR12	CNFR6, CNFR12, CNFR15, CNFR16, CNFR19, CNFR26, CNFR27, CNFR30, CNFR37, CNFR38, CNFR39, CNFR43, CNFR44

11.2. acceleration.py

Module Implementation

A class named Acceleration. Provides functions for retrieving acceleration readings from the accelerometer, storing the past 60 seconds of readings to a queue, analyzing readings to determine is a crash has occurred, and logging the queue of readings to a non-volatile memory location in the form of a .csv file.

Module Secrets

- Amount of time to store acceleration readings for.
- Acceleration Readings.
- Acceleration scaling factors (calibrated).
- Algorithm for computing if a crash has occurred.
- Algorithm to translate queue of readings into .csv file.

Module Relationships

Receives the following Acceleration class construction parameters through `__init__()`.

Parameter	Type	Description
sample_rate	integer	Number of times per second to retrieve an accelerometer reading. This will determine the size of the required queue.
i2c	busio.I2C Class	Enables setup to accelerometer object by passing i2c as the communication protocol.

Receives incoming acceleration data through `read_data()`.

Parameter	Type	Description
x	float	x component of acceleration
y	float	y component of acceleration
z	float	z component of acceleration

The scaled available acceleration data is analyzed, and a determination is returned through `is_crashed()`.

Returns	Description
boolean	If a crash has been detected

Receives the current permanent storage location through `Dir_Handler()`.

| Parameter | Type | Description | |:--|:--|:--| | dir_handler | string | Absolute directory of the current permanent capture storage location. |

Likely Changes

- Adding the ability to log acceleration data in CSV format, so that data can be collected in the field and brought back to the lab for analysis.
  - This change has been implemented

- Crash detection algorithm will likely be updated and refined as manual testing continues.
    - This change has been implemented.

Unlikely Changes

- Functions related to the visualization of acceleration data are useful in testing, but may be removed to improve the performance of the embedded system.
    - This change has been implemented.

Traceability

The requirements traceability of the acceleration.py class is as follows:

Module	Functional Requirements	Non-Functional Requirements
acceleration.py	CFR3, CFR5, CFR7, CFR13	CNFR6, CNFR12, CNFR15, CNFR17, CNFR24, CNFR29, CNFR37, CNFR38, CNFR39

11.3. _*init*_.py - LED_Handler

Module Implementation A class named LED_Handler. Provides a startup, capture, dynamic capture error, dynamic rear object distance, and poweroff LED lighting animations for use with the 8 LED neopixel strip.

Module Secrets

- Rasberry Pi pin setups.
- Number of LEDs
- Distance Display Color Gradient
- Animation Implementations

Module Relationships

`error_capture_sequence()` method receives `debug` when called.

Parameter	Type	Description
debug	list	3 boolean values mapping to Accelerometer, Camera, and LiDar capture events respectively. Successful capture is indicated by False, and Failed capture by True.

`distance_display()` method receives `distance` when called.

Parameter	Type	Description
distance	integer	Distance in cm or objects in the rear.

Likely Changes

- Enhance Power on animation to show an initialization debug.
    - This change has not yet been implemented.

Unlikely Changes

- No unlikely changes

Traceability

The requirements traceability of the Led_Handler() class is as follows:

Module	Functional Requirements	Non-Functional Requirements
Led_Handler()	CFR1	CNFR6, CNFR8, CNFR12, CNFR13, CNFR18, CNFR37, CNFR38, CNFR39

11.4. lidar.py

Module Implementation A class named Lidar. Provides the functionality of reading data from the rear facing lidar sensor, translating this data into a distance reading, storing the past 60 seconds of readings to a queue, and logging the queue of readings to a non-volatile memory location in the form of a .csv file.

Module Secrets

- Rasberry Pi serial setup.
- Algorithm to translate raw lidar sensor data into a distance reading.

Module Relationships

Receives the following led class construction parameters through `__init__()`.

Parameter	Type	Description
sample_rate	integer	Number of distance sensor readings per second.
baud_rate	integer	Speed of communication over serial channel in bits per second.

Data is interpreted from the lidar distance sensor, and a distance is returned through `read_data()`.

Returns	Description
integer	distance in cm measured by the lidar distance sensor.

Receives the current permanent storage location through `Dir_Handler()`.

| Parameter | Type | Description | |:--|:--| | dir_handler | string | Absolute directory of the current permanent capture storage location. |

Likely Changes

- Some methodology to decrease noise.
  - Perhaps some distance average or data linearization.

Unlikely Changes

- No unlikely changes

Traceability

The requirements traceability of the lidar.py class is as follows:

Module	Functional Requirements	Non-Functional Requirements
lidar.py	CFR1	CNFR6, CNFR12, CNFR15, CNFR18, CNFR37, CNFR38, CNFR39

11.3. _*init*_.py - Dir_Handler

Module Implementation A class named Dir_Handler. Provides the functionality of mounting a USB stick if any are available, as well as determining or creating the most up-to-date directory to permanently store a capture file at.

Module Secrets

- Algorithm to mount a USB stick
- Algorithm to determine the most recent capture folder and the conditions under which a new folder must be created.
- Parent directory of folders which will be created.

Module Relationships

`locate_export_dir()` method receives `filename` when called.

Parameter	Type	Description
filename	string	Name of the file to be saved excluding timestamps.

Likely Changes

- Add intelligent USB stick mounting if a USB stick is detected.
  - This change has been implemented.

Unlikely Changes

- No unlikely changes

Traceability

The requirements traceability of the Dir_Handler() class is as follows:

Module	Functional Requirements	Non-Functional Requirements
Dir_Handler()	ADD	ADD

12. Timeline

Date	Task	Person	Testing
2023-01-22	Creation of crash detection algorithm.	Manny Lemos, Amos Yu	Automated testing to determine response under a wide range of accelerometer inputs.

Date	Task	Person	Testing
2023-01-25	Update design and 3D print hardware case to accommodate the camera, battery pack, and mounting bracket.	Aaron Li, Amos Cheung	Manual testing. Ensure everything fits as expected.
2023-01-28	Design and 3D print a mounting bracket to attach the hardware case to bicycle fork.	Aaron Li, Manny Lemos	Manual testing. Ensure the mounting bracket can support the weight of the loaded hardware case sufficiently. Ensure the mounting bracket will function for common bicycle forks.
2023-01-31	Swap breadboard and jumper wires for PCB and soldered connections.	Amos Cheung	Manual testing. Verify connections using multimeter.
2023-02-05	Integration of all software modules.	Brian Le, Amos Yu	Manual Testing. Mount the hardware to the bicycle and ensure it functions as detailed in the SRS Document.
2023-02-09	Rev0 Demonstration	Brian Le, Amos Yu, Amos Cheung, Aaron Li, Manny Lemos	Ensure all tasks in the timeline up to 2023-02-09 is completed and working as a whole.
2023-02-20	Implement Lidar Sensor	Amos Cheung, Amos Yu, Brian Le	Manual testing. Ensure lidar sensor functions as detailed in the SRS document.

13. Appendix

13.1. Reflection

Cyclops ride assist aims to fill the ride monitoring and crash avoidance gap in the cycling market. This solution takes the form of a camera, accelerometer, and distance sensor. These inputs are parsed by an embedded computer to provide users with warning lights when vehicles are approaching, and an automatically logged clip if a crash is detected.

13.1.1. Solution Limitations

Camera Resolution: The maximum camera resolution possible with the current implementation is 640p. This is due to the data transfer speeds being limited by the USB connection made between the camera and the Raspberry Pi. Using the Raspberry Camera Module is an alternative to the current implementation which would overcome the USB bandwidth issue and could increase the resolution to 1080p. This alternative comes with the downside of addition cost and recording software rewriting because the Pi Camera Module uses custom libraries.

Camera Position: With the current implementation the camera is seated atop the bicycle forks facing forward. This provides video capture of the volume of space in front of the cyclist. The limitation of this implementation is evident, there is no camera footage of the volume of space behind the rider. An alternative to the current state would be to add an additional camera with a view behind the cyclist. However, due to the current USB bandwidth issues with a single camera and the lack of support for dual Raspberry Pi Camera Modules this change is not feasible. A possible alternative would be changing the position of the current camera to face being the cyclist.

Battery Life: For cyclists who go on multi hour long bicycle rides, battery life may be of concern. The Raspberry Pi used to perform computations is not particularly battery efficient compared to more project specific embedded computers who do not have as much computational overhead. The capacity of the current battery pack being used is 10,000mah. To improve battery life, a higher capacity battery pack could be purchased. However, a higher capacity battery pack is almost certain to come along with the unwanted side effects of a larger size and heavier weight.

Ultrasonic Sensor: Prior to the Lidar sensor, an Ultrasonic distance sensor was used to measure the distance of rear objects. This Ultrasonic sensor was viable up to 2 metres. However, the effectiveness of this was extremely limited when detecting an object at the upper end of this range, or in an enclosed environment where ultrasonic signals bouncing off surfaces created noise. Temperature was also a major limiting factor as the accuracy was subject to change in temperatures of 5 - 19 degrees. One way we can look to improve on the performance and accuracy of our object detection would be to use a higher quality sensor for cyclops which would decrease these problems.

13.2. References

[1] "Raspberry Pi 4 Mechanical Drawing", 2018. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-mechanical-drawing.pdf

[2] "What is PLA?", 2023. [Online]. Available: https://www.twi-global.com/technical-knowledge/faqs/what-is-pla

[3] "Raspberry Pi 4 Electrical Schematic", 2018. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf

[4] "ADXL-345 Datasheet", 2023. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf

[5] "TF-Luna Product Manual", 2023. [Online]. Available: http://en.benewake.com/res/wuliu/docs/15978504093169389SJ-PM-TF-Luna%20A05%20Product%20Manual.pdf

[6] "1080p Webcam", 2023. [Online]. Available: https://www.amazon.ca/Microphone-Otooking-Streaming-Conferencing-Recording/dp/B08HYDZ6TN/ref=zg_bs_23883740011_sccl_10/140-3616671-2115546?psc=1

[7] K. Gross, “Ultrasonic sensors: Advantages and limitations,” MaxBotix Inc., 28-Oct-2020. [Online]. Available: https://www.maxbotix.com/articles/advantages-limitations-ultrasonic-sensors.htm/. [Accessed:

18-Jan-2023].