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

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

The purpose of this project is to create a system that helps cyclists to have a more secured experience especially on roads without bike lanes.

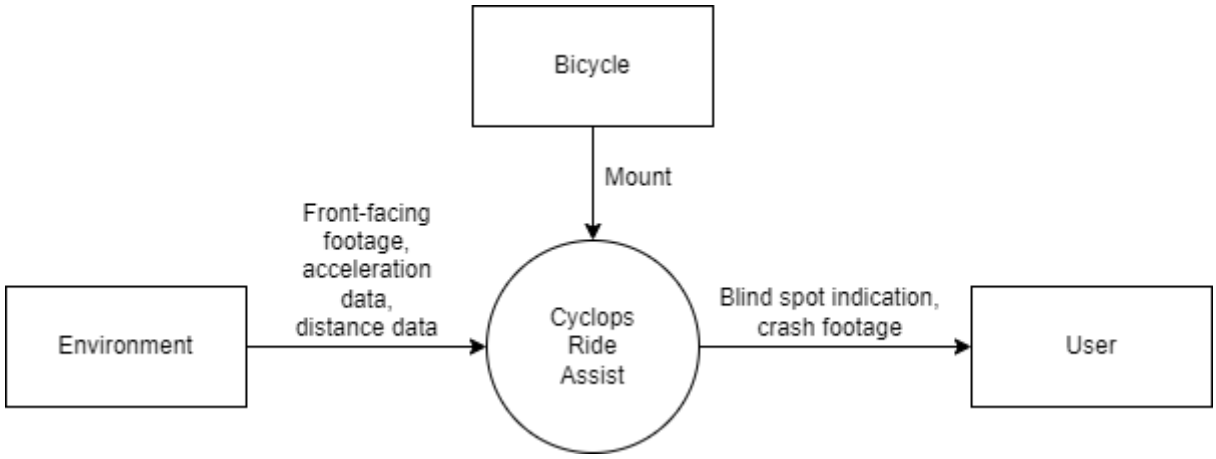
Cyclops Ride Assist (CRA) is going to be an all-in-one, easily mountable, and quick to setup system that adds modern car safety features onto a bike, such as blind spot detection and crash detection. The system will also have a built in headlamp that will illuminate during night time, not just for the cyclist to see but also for cars around to realize the bike.

4. Scope

CRA is going to be a bike assist system with convenient mounting, accurate crash detection, video buffering and saving, reliable blindspot monitoring and a user controlled headlamp that helps cyclist to have a peace of mind while riding on the road. Although CRA is primarily targeted towards road cyclists, it will able be useful for cyclists who ride on mountains or trails.

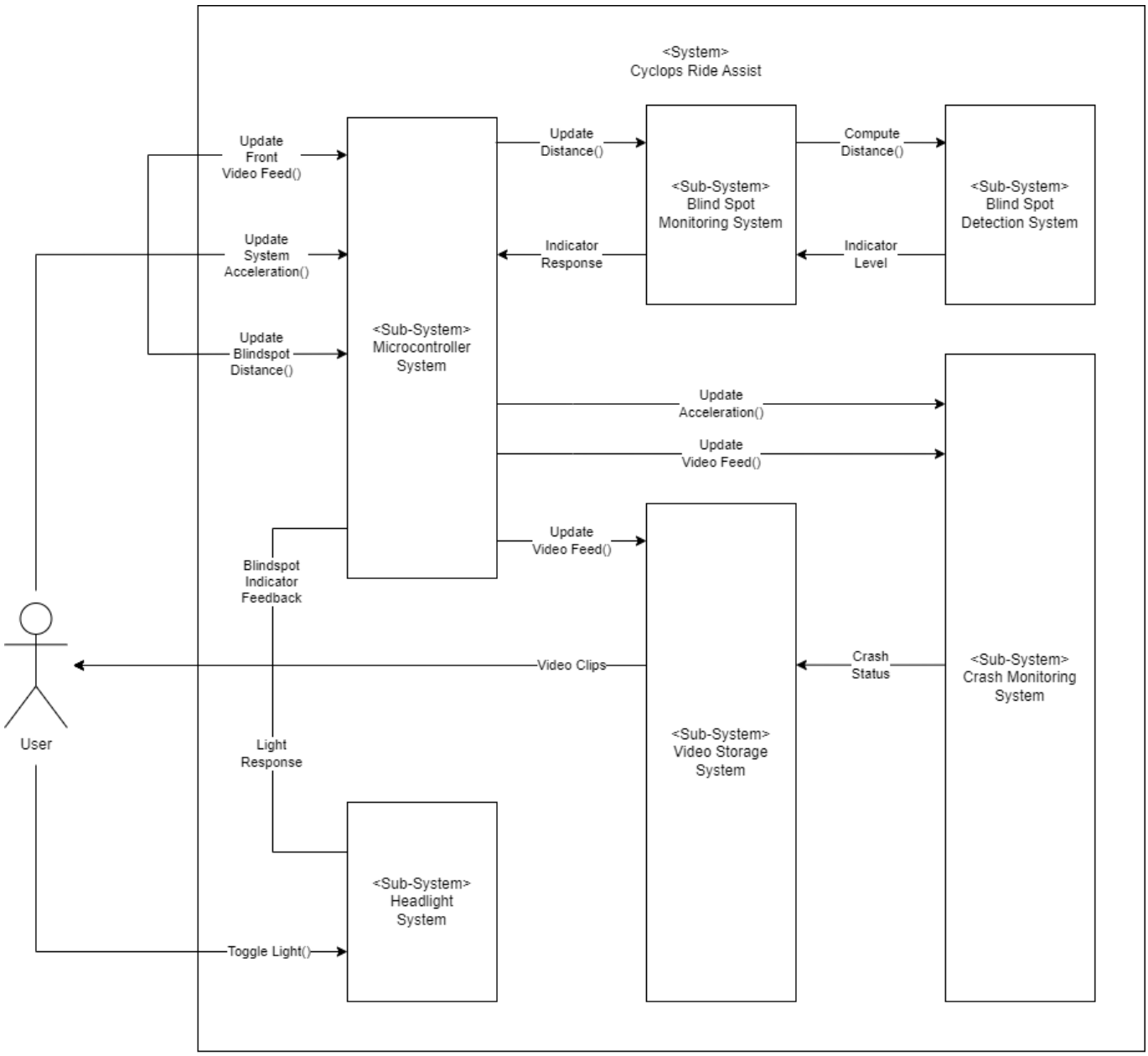
4.1. System Context

Figure 4.1.1: CRA System Context Diagram



5. Project Overview

Figure 5.1: CRA Functional Decomposition Diagram



6. System Variables

6.1. Monitored and Controlled Variables

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

Monitor Var	Monitor Type	Range	Units	Comments
distance_cm	Distance	[0, 4000]	cm	Distance to closest obstacle
curr_frames	Frequency	[0, 30]	FPS	The rate at which the video buffer can sample a frame for video feed
run_buffer	Boolean	N/A	N/A	Boolean of if the buffer should run or not
average_of	Acceleration	[-16, 16]	G's	Takes a rolling average of the xyz acceleration points
avg_x	Acceleration	N/A	m/s^2	Acceleration in the x plane
avg_y	Acceleration	N/A	m/s^2	Acceleration in the y plane
avg_z	Acceleration	N/A	m/s^2	Acceleration in the z plane
avg_norm	Acceleration	N/A	m/s^2	Normal of the accelerations

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

Controlled Var	Controlled Type	Range	Units	Comments
lock	Mutex	N/A	N/A	Mutex
data_points	Size	TBD	N/A	Maximum number of data points to show on the plot and kept track of
GPIO	Boolean	N/A	N/A	Toggle for LEDs to display sensor data
frame_width	Size	[640, 1920]	px	Capture resolution
frame_height	Size	[480, 1080]	px	Capture resolution
video_length	Time	[0 - 60]	Seconds	The length in seconds of the requested video

6.2. Constants

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

Constant Var	Constant Type	Value	Units	Comments
c_s	Speed	343	m/s	Speed of sound

7. User Interfaces

7.1. Inputs

Input Name	Input Type	Range	Units	Comments
Power Button	Physical	[0, 1]	N/A	Button that is used to power on the system and automatically run the blindspot/crash detection scripts
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
Storage Device Port	Physical	N/A	N/A	Port for inserting the SD Card

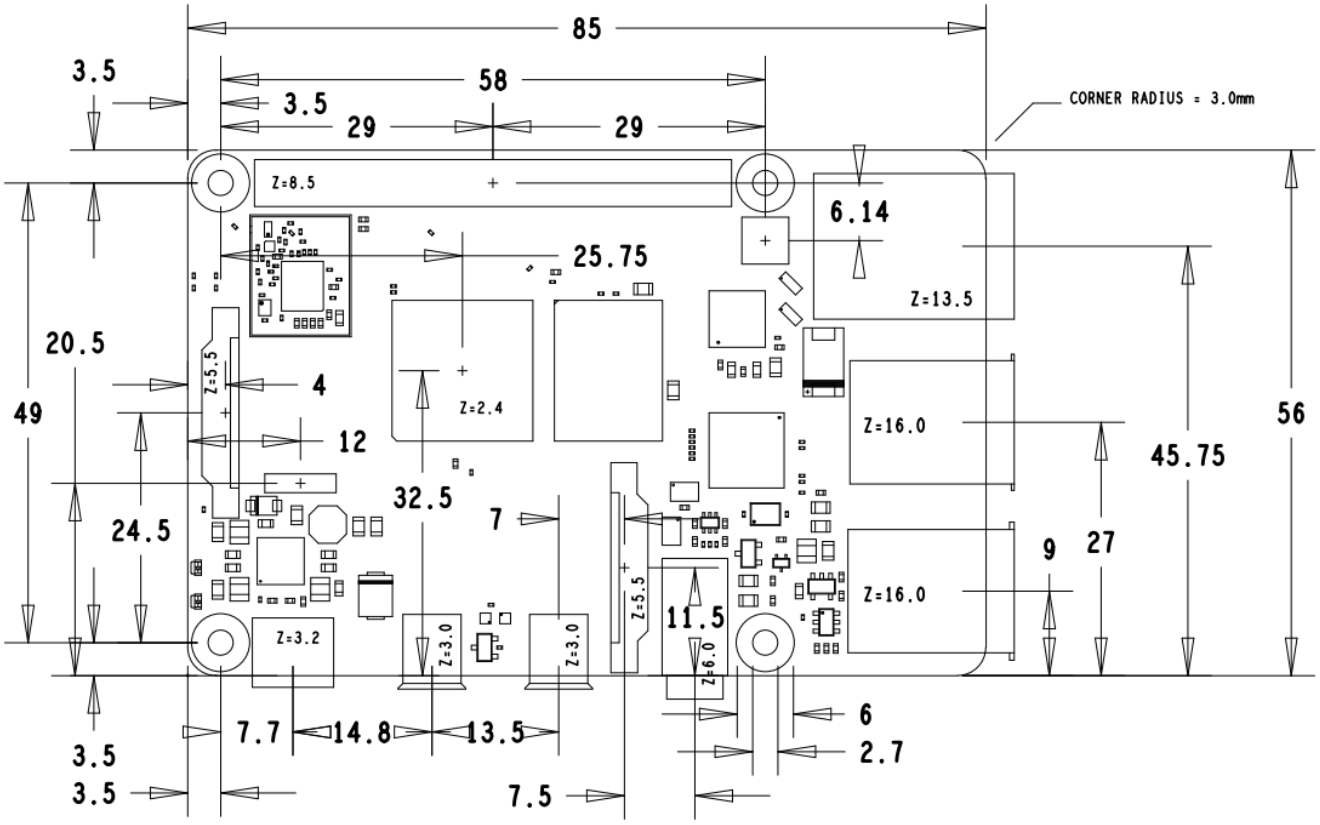
7.2. Outputs

Output Name	Output Type	Range	Units	Comments
LEDS	Visual	[0, 5]	N/A	5 LEDS are used to indicate the distance of an object in your blindspot as well as to signify that the cyclops is on an running
SD	Physical	N/A	N/A	SD slot is used to store the video buffer when a crash has been detected

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]:

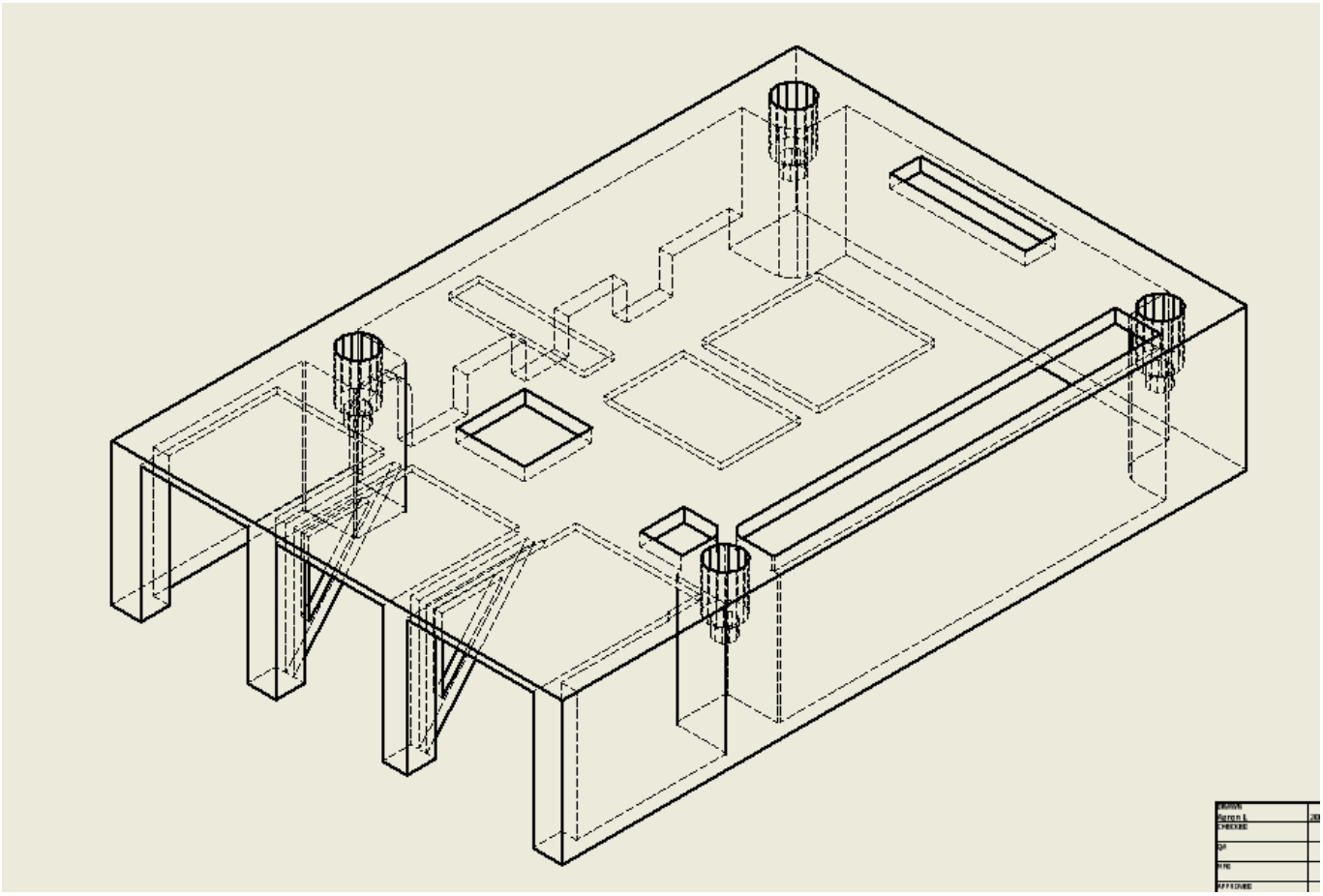
PLA Specifications	Value
Heat Deflection Temperature	52C
Density	1.24 g/cm3
Tensile Strength	50MPa

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.

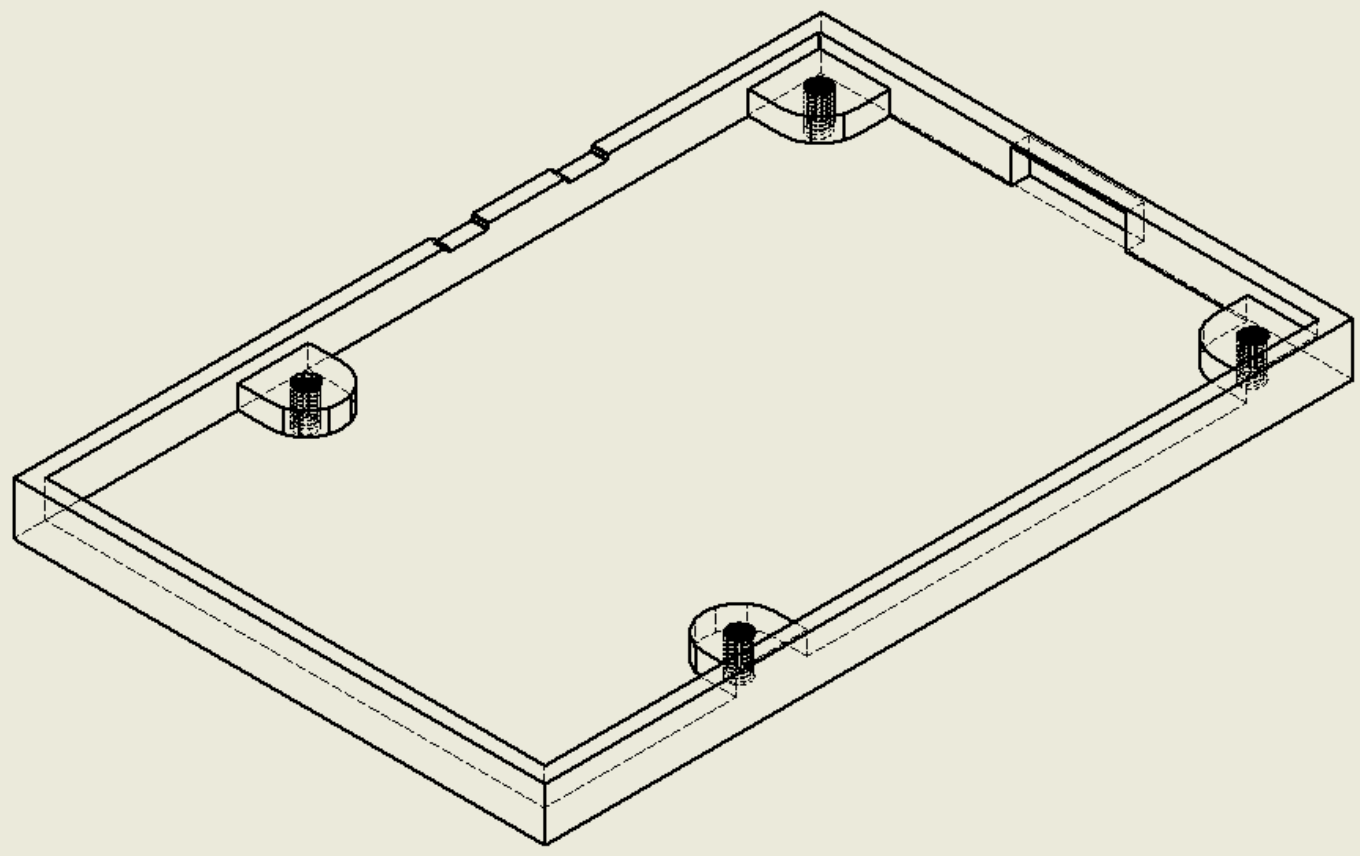
The mechanical design was created using PLA and the Prusa i3 MK3s 3D printer.

Figure 8.3.1: Mechanical Housing Drawing Top



Mechanical Frame Top Component Specification	Value
Dimensions	91.25mm x 60mm x 19mm
Weight	23grams
Material	PLA

Figure 8.3.2: Mechanical Housing Drawing Bottom



Mechanical Frame Bottom Component Specification	Value
Outer Dimensions	91.25mm x 60mm x 6mm
Weight	15 grams
Material	PLA

The requirements traceability of the mechanical frame is 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

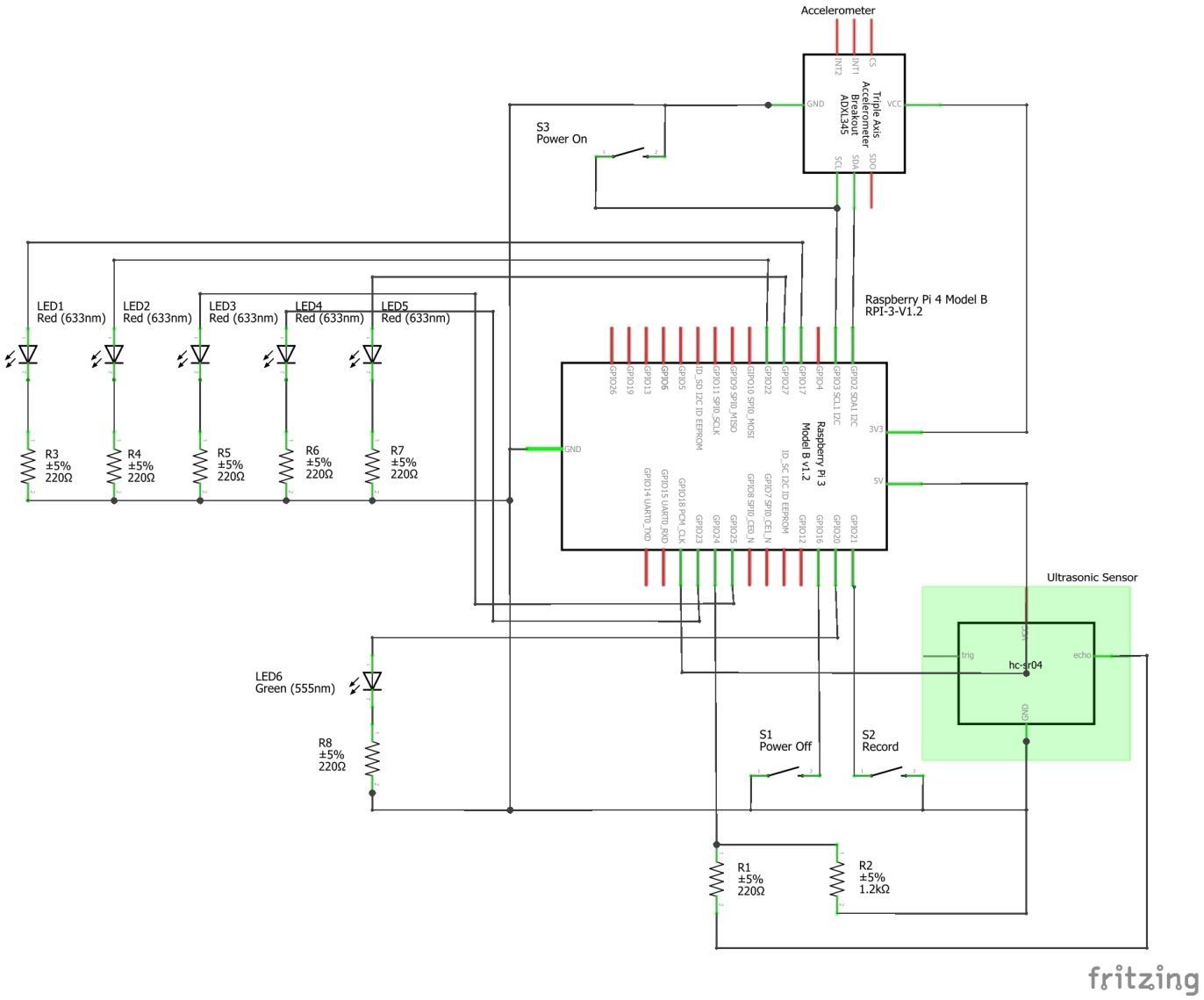
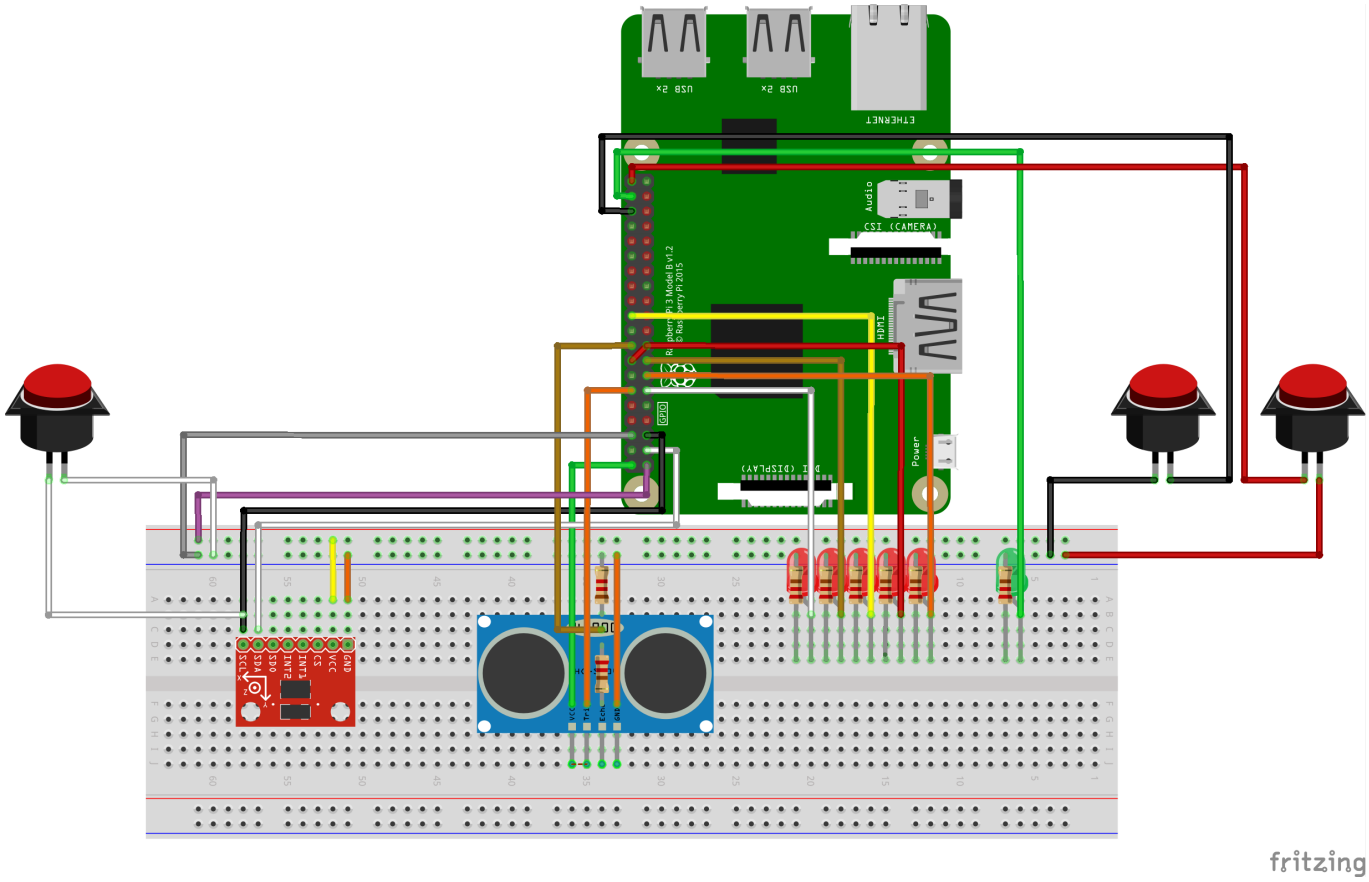


Figure 9.1.2: CRA Breadboard Schematic



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]

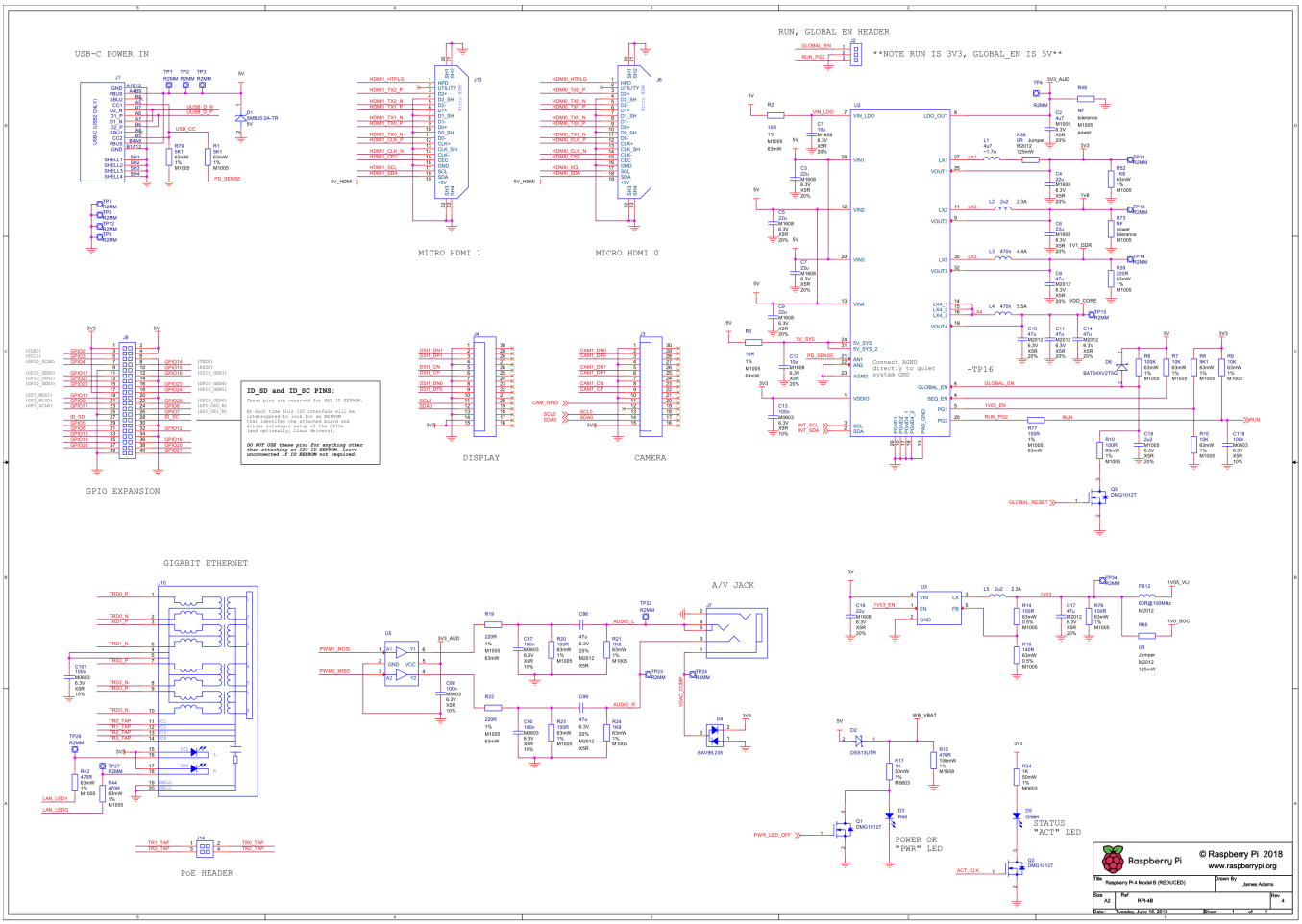
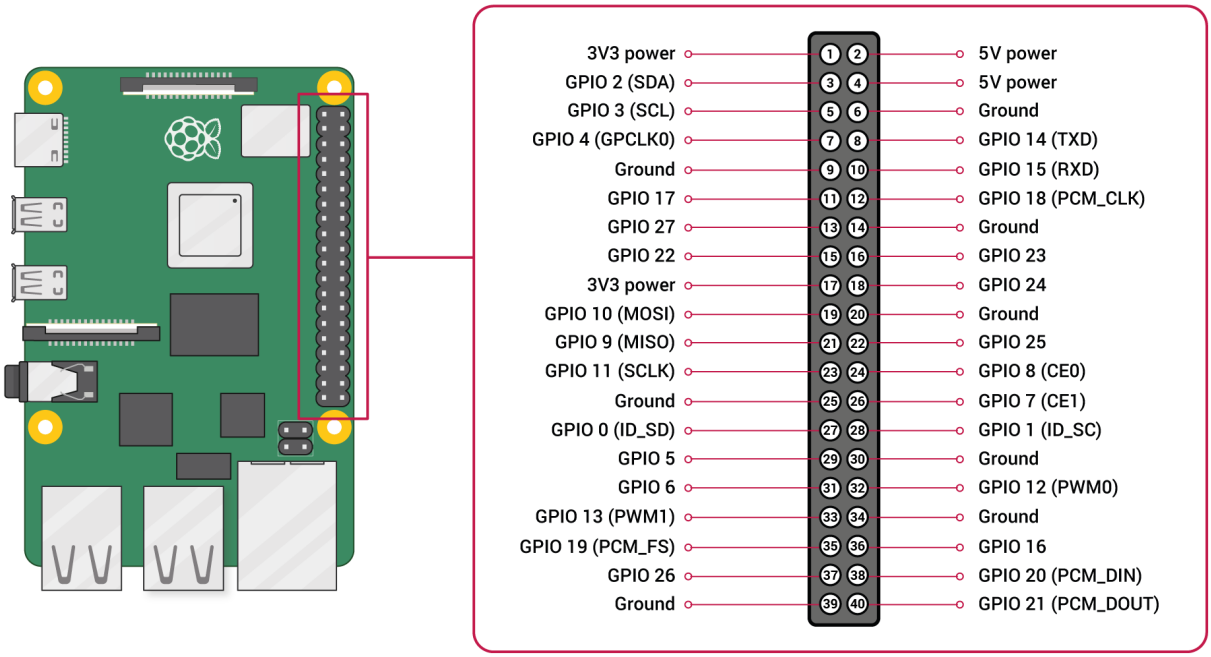


Figure 9.2.2: Raspberry Pi 4 Model B Pinouts



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. Printed Circuit Board (PCB) Specifications

The printed circuit board will be used to combine all the electrical hardware with its respective software interfaces. Using soldering techniques, each wire and resistor will be soldered to its respective hole. The PCB specifications are listed below, referenced from Elegoo [4].

PCB Specifications	Value
Dimensions	5cm x 7cm
Thickness	1.6mm
Material	Glass Giber FR4
Hole-Pitch	2.54mm
Hole-diameter	1mm

The requirements traceability of the printed circuit board is as follows:

Module	Functional Requirements	Non-Functional Requirements
Printed circuit board	CFR3, CFR11	CNFR2, CNFR4, CNFR11, CNFR29, CNFR31, CNFR33, CNFR40

9.4. Accelerometer Specifications

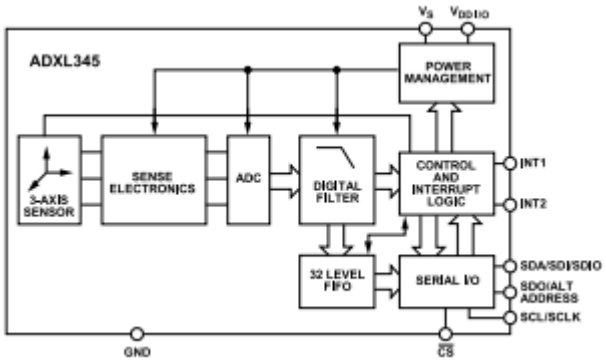
The accelerometer will be used to determine when a crash has occured. The accelerometer used is the ADXL-345 and the specifications are as follows, as outlined by Analog Devices [5].

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 [5]



9.5. Radar Sensor Specifications

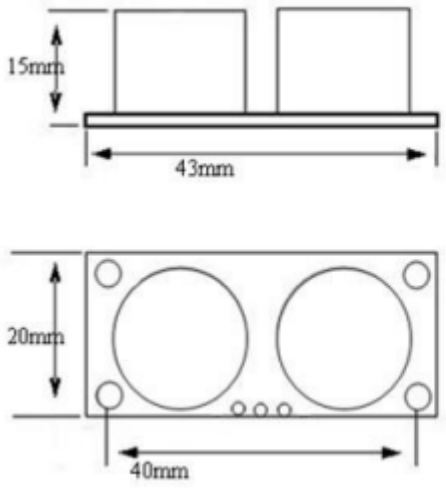
The radar sensor will be used to sense when a car is a certain distance away from a vehicle or large object. The radar sensor that is being used is the HC-SR04. The specifications will be as follows, as noted by Elec Freaks [6].

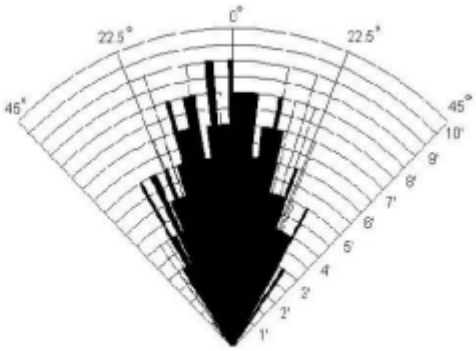
Radar Sensor Specifications	Value
Dimensions	45mm x 20mm x 15mm
Voltage	15 mA
Frequency	40Hz
Maximum Range	4m
Minimum Range	2cm
Measuring Angle	15deg
Trigger Input Signal	10us TTL pulse
Echo Input Signal	Trigger Input Signal + Proportional Range

The requirements traceability of the radar sensor is as follows:

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

Figure 9.5.1: Radar Sensor Diagram [6]





Practical test of performance,
Best in 30 degree angle

9.6. Camera Specifications

The camera will be used to record the crash footage for a period of time. The camera that is being used is the Ootoking 1080p webcam, that can be referenced on Amazon.ca [7].

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

9.7. Headlamp Specifications

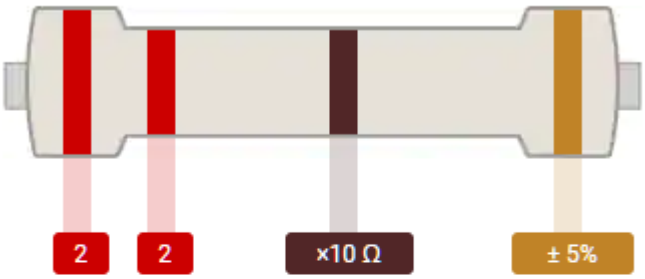
The headlamp will be a forward facing LED with its own battery source. The requirements traceability of the headlamp is as follows:

Module	Functional Requirements	Non-Functional Requirements
Headlamp	CFR14	CNFR1, CNFR7, CNFR8, CNFR13, CNFR35, CNFR40, CNFR48, CNFR49

9.8. Resistor Specifications

Resistors will be used to ensure that the electric current is controlled and that any voltage spikes will not damage the components located on the CRA. The resistors are 4-band resistors and are of 220 Ohms and 1.2k Ohms as shown by Digikey [8].

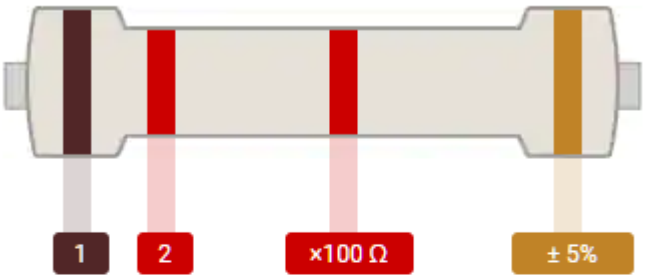
Figure 9.7.1: Resistor with 220 Ohms



Resistor value:

220 Ohms 5%

Figure 9.7.2: Resistor with 1.2k Ohms



Resistor value:

1.2k Ohms 5%

10. Communication Protocols

The requirements traceability of all communication protocols is as follows:

Modules	Functional Requirements	Non-Functional Requirements

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 to external cameras or other devices like keyboards, mice, external storage devices, and other computers to allow for greater ease-of-use and accessibility.
USB-C	The USB-C port will be used to charge the CRA when the battery falls below the desired power percentage.

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 through various methods including SSH. Furthermore, this protocol allows users to connect their CRA to either a 2.4 or 5.0GHz wireless network.

10.3. I2C Protocol

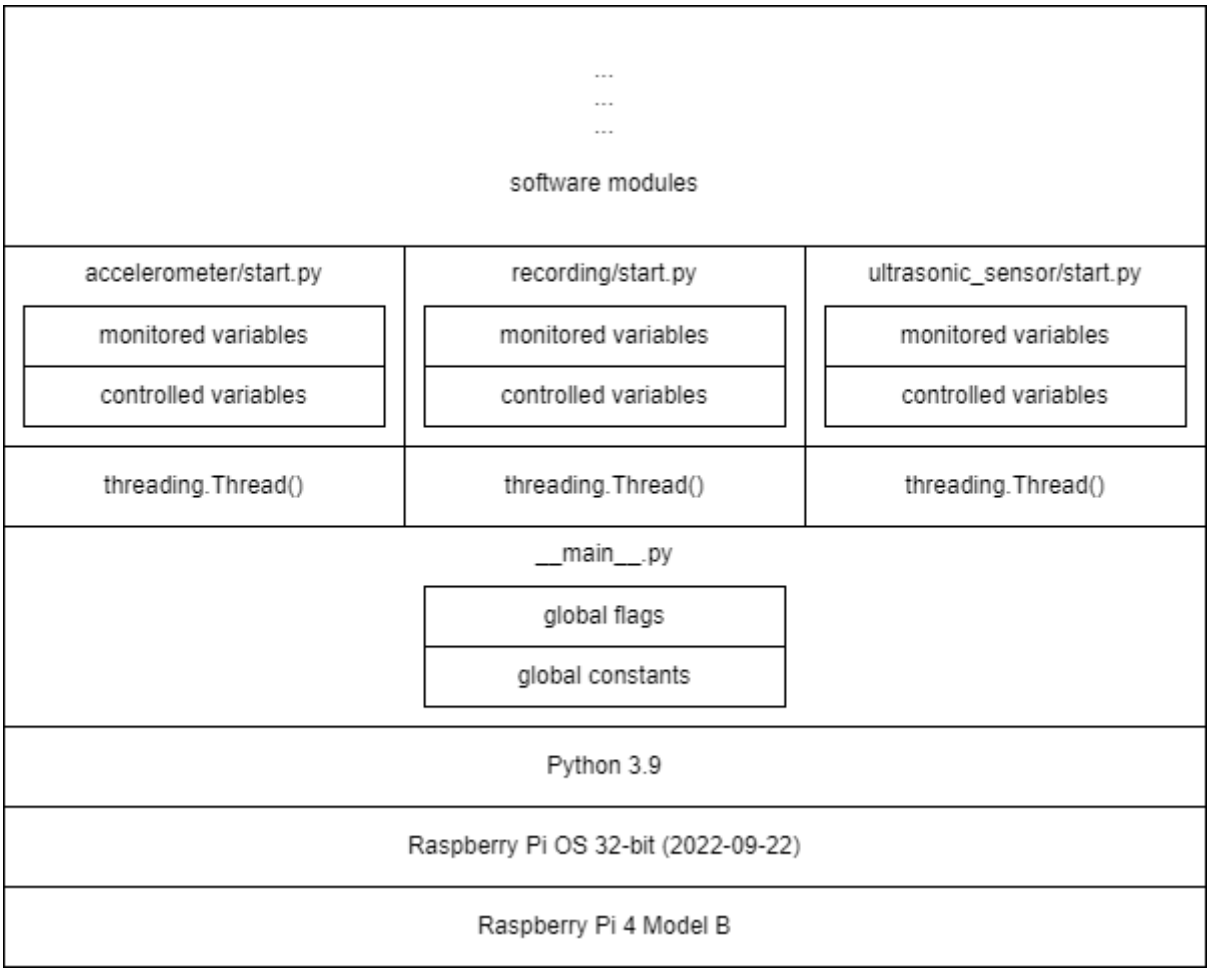
The I2C protocol will be used for communication with the accelerometer.

I2C Protocol	Purpose
802.11ac	This wireless protocol will allow the users to connect to the CRA remotely through various methods including SSH. Furthermore, this protocol allows users to connect their CRA to either a 2.4 or 5.0GHz wireless network.

11. Software Modules

All software modules are classes implemented in Python 3.9. Software modules are constructed and executed by threaded `start.py` classes. The `__main__.py` class should immediately begin running upon startup of the Raspberry Pi OS.

Figure 11.0.1: CRA Software Stack



11.1. video_buffer.py

Module Implementation

Composed of a class Buffer. Provides the functionality of starting, stopping, and logging a video clip.

Module Secrets

- Video capture instance.
- Frames of video clips.
- Video codec.

Module Relationships

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

Parameter	Type	Description
video_length	integer	The length in seconds of the requested video.
num_partitions	integer	[2->10] The number of clips which will be combined to form the output video. As num_partitions increases; video concatenation time increases, disk storage decreases.
fps	integer	Number of frames requested from camera every second.
save_directory	string	The full directory you wish to save recordings at

Parameter	Type	Description
temp_directory	string	The full directory used to temporarily store .mp4 clips
camera_id	integer	-1 -> Automatically detect camera, >=0 -> Manually identify by webcam index
resolution	integer	-1 -> Automatically detect resolution, 0 -> 640x480, 1 -> 1280x720, 2 -> 1920x1080

Likely Changes

- Reduce the number of input parameters to the instantiation of the Buffer class. These inputs will be replaced by fixed parameters which are deemed most suitable.

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 Buffer class will have to be rewritten.

Traceability

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

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

11.2. acceleration_plot.py

Module Implementation

A class named Acceleration_Plot. Provides functions for analyzing and visualizing acceleration data.

Module Secrets

- Algorithm for computing if a crash has occurred.
- Implementation details of the live plot.

Module Relationships

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

Parameter	Type	Description
data_points	integer	Maximum number of data points to show on the plot
average_of	integer	Takes a rolling average of the xyz acceleration points

Receives incoming acceleration data through `update()`.

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

Computes, using the available acceleration data, and returns if a crash has occurred through `is_crash_detected()`.

Returns	Description
boolean	If a crash has been detected

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.
- Crash detection algorithm will likely be updated and refined as manual testing continues.

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.

Traceability

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

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

11.3. led.py

Module Implementation A class named led. Provides the functionalty of mapping Rasperry Pi pin to LEDs and also turning on/off the individuals LED.

Module Secrets

- Rasberry Pi pin setups.
- Implementation of turning the LEDs on/off.

Module Relationships

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

Parameter	Type	Description

Parameter	Type	Description
pins	integer	Pin number on the Rasberry Pi for connecting the LEDs

Turning on the LEDs based on the percentage it is at through `percentage_high()`.

Parameter	Type	Description
percent	integer	Percentage of LEDs that should be turned on

Likely Changes

- No likely changes

Unlikely Changes

- No unlikely changes

Traceability

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

Module	Functional Requirements	Non-Functional Requirements
led.py	CFR1	CNFR6, CNFR8, CNFR12, CNFR13, CNFR18, CNFR37, CNFR38, CNFR39

11.4. ultrasonic_sensor.py

Module Implementation A class named ultrasonic_sensor. Provides the functionalty of measuring the distance of the object using ultrasonic sensor.

Module Secrets

- Rasberry Pi pin setups.
- Calculation for the distance of object from ultrasonic sensor.
- Time it takes for the ultrasonic sensor to pikc up the object.

Module Relationships

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

Parameter	Type	Description
pin_trigger	integer	Pin number on the pi corresponding to the trigger pin on the sensor
pin_echo	integer	Pin number on the pi corresponding to the echo pin on the sensor
distance_min	integer	The closest distance that the ultrasonic sensor should detect
distance_max	integer	The furthest distance that the ultrasonic sensor should detect
unit	string	Unit of the min and max distances (ex. cm)

Likely Changes

- Number of LEDs.
- Distance unit.
- Maximum and minimum distance the ultrasonic sensor should detect.

Unlikely Changes

- No unlikely changes

Traceability

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

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

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.
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.

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: For the blindspot detection, our ultrasonic sensor is viable up to 4 meters. [9] However, the effectiveness of this is then limited when detecting an object from range. Temperature is also a major limiting factor as accuracy can be changed 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 can 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] "Elegoo Double Sided PCB Board Kit", 2023. [Online]. Available: https://www.elegoo.com/en-ca/products/elegoo-double-sided-pcb-board-kit

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

[6] "HC-SR04 Datasheet", 2023. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[7] "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

[8] "4 Band Resistor Color Code Calculator", 2023. [Online]. Available: https://www.digikey.ca/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

[9] 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].