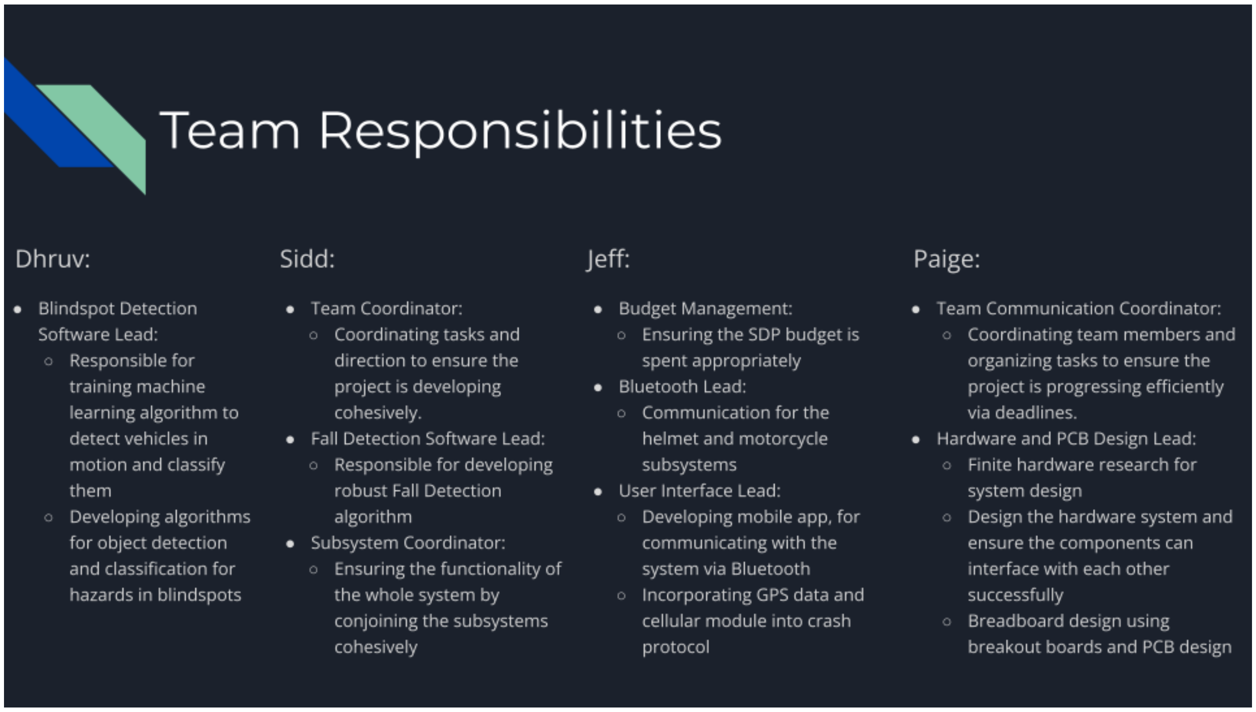
ECE 415: Senior Design Project I
Midway Design Review

Handout

Team 21
Siddhanta Shreshtha, Dhruv Vikram Krishna, Paige Wadas, Jeff Wang

Table of Contents (In Order of Slides)

3) Team Responsibilities



Team Responsibilities

Dhruv:

- Blindspot Detection Software Lead:
  - Responsible for training machine learning algorithm to detect vehicles in motion and classify them
  - Developing algorithms for object detection and classification for hazards in blindspots

Sidd:

- Team Coordinator:
  - Coordinating tasks and direction to ensure the project is developing cohesively.
- Fall Detection Software Lead:
  - Responsible for developing robust Fall Detection algorithm
- Subsystem Coordinator:
  - Ensuring the functionality of the whole system by conjoining the subsystems cohesively

Jeff:

- Budget Management:
  - Ensuring the SDP budget is spent appropriately
- Bluetooth Lead:
  - Communication for the helmet and motorcycle subsystems
- User Interface Lead:
  - Developing mobile app, for communicating with the system via Bluetooth
  - Incorporating GPS data and cellular module into crash protocol

Paige:

- Team Communication Coordinator:
  - Coordinating team members and organizing tasks to ensure the project is progressing efficiently via deadlines.
- Hardware and PCB Design Lead:
  - Finite hardware research for system design
  - Design the hardware system and ensure the components can interface with each other successfully
  - Breadboard design using breakout boards and PCB design

---

5) Problem Statement

A majority of products for motorcyclists are tailored for convenience rather than safety. Typically riders must manually check their blindspots via side-mirrors and shoulder checks. This becomes a hazard for riders because they become distracted when doing so. After vehicle accidents, there is no efficiency of contacting emergency services. Our solution will address this issue by alerting riders with the necessary information on the road and after an accident to better ensure their safety.

---

6) Project Goal

To create a two part device that is attachable to a motorcycle and helmet. The system should notify riders of oncoming vehicles within their blindspots to increase rider safety, and alert emergency services if the rider gets into a crash.

8) Initial Deliverables

- Blindspot Subsystem
  - Detect moving vehicles by classifying objects within the blindspots
  - Transmit object detection data from the motorcycle to the helmet over Bluetooth
  - Configure lighting system to indicate direction and proximity using the detection data
- Fall Detection Subsystem
  - Compare both data transmitted from the motorcycle's gyroscope/accelerometer with the helmet's accelerometer for fall/accident
  - Implement fall detection protocol by testing visual alert notification with cut off button in the helmet

9) Updated Deliverables

- Blindspot Detection System
  - Detect and classify vehicles within frame
  - Flag vehicles in the blindspots
- Fall Detection System
  - Compare the data transmitted from the motorcycle's gyroscope/accelerometer with the helmet's accelerometer to determine if the crash protocol should be initialized
  - Implement fall detection protocol by testing visual alert notification with cut off button in the helmet
  - Implement crash protocol cutoff sequence using push button in the helmet, with LED notification
- Mobile Interface
  - Prototype interface with rudimentary login/signup, verification and user profile page

10) Challenges Faced

YOLOv3 & OpenCV Benchmarks

Benchmarks of YOLOv3 on OpenCV 3.4.2, running 1080x720 pixel video:

- Average FPS of 4.77
    - Max: 6.21 | Min: 3.32
- GPU Utilization: ~42%
    - Checked using nvidia-smi tool

YOLOv5s & PyTorch Improvements

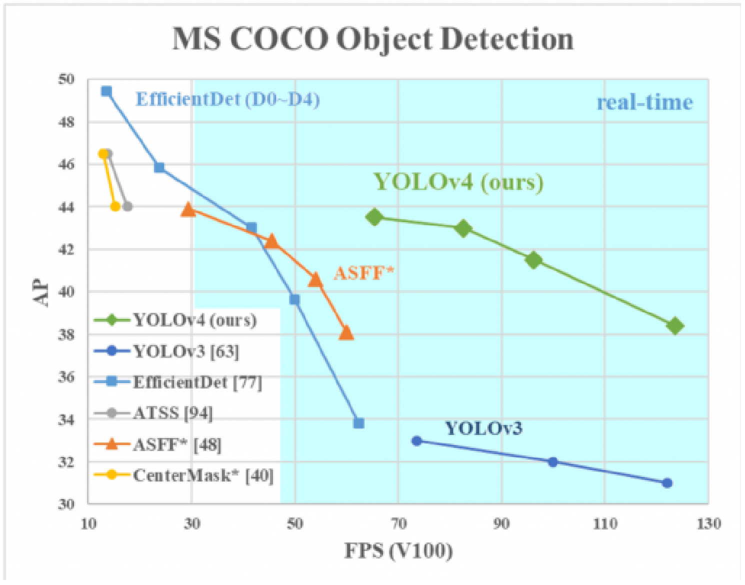


Figure 1: Graph indicating the FPS vs Average Precision in YOLOv4 and YOLOv3. FPS increase in YOLOv4 is upto 12% with an improvement in mean average precision by 10%.[1]

[1] https://ai-pool.com/a/s/yolov3-and-yolov4-in-object-detection

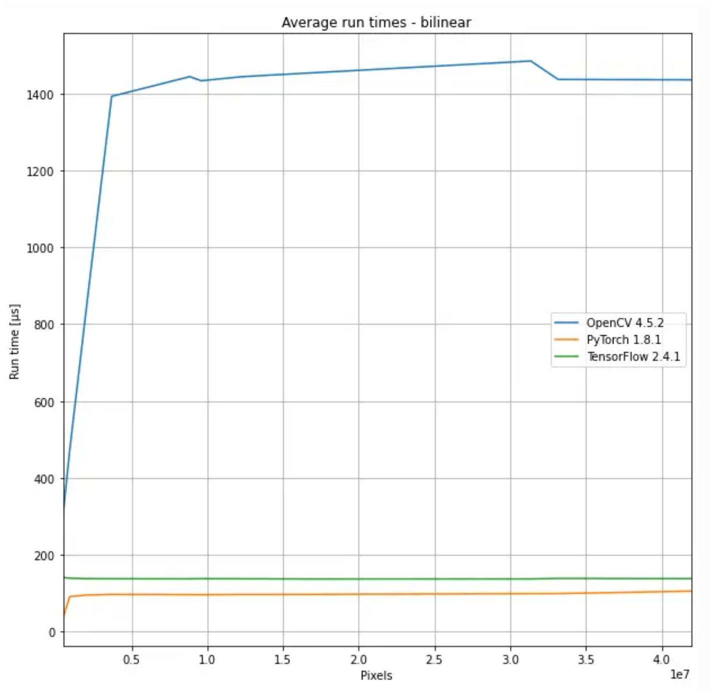


Figure 2: Image resizing runtimes across OpenCV, PyTorch & TensorFlow[2]

12) Hardware Flowchart

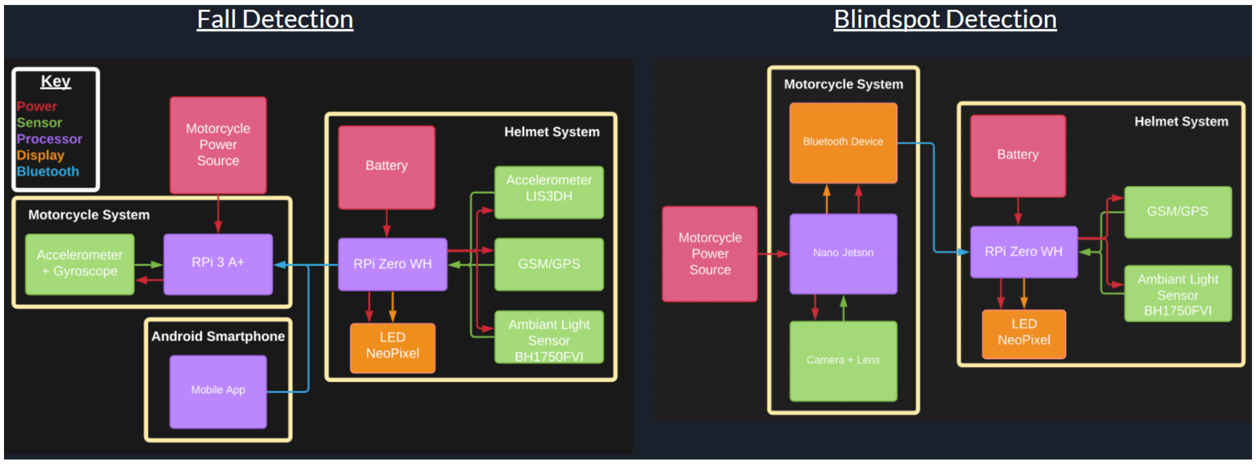


Figure 3: Hardware flowchart diagram detailing hardware in all subsystems

2 https://www.simonwenkel.com/2021/04/20/Resizing-images-on-GPU-OpenCV-PyTorch-TensorFlow.html

13) Hardware Used

Helmet

- Raspberry Pi Zero W H
- Adafruit LIS3DH Triple-Axis Accelerometer IC
- Adafruit NeoPixel LED Side Light Strip
- microSDHC Card 32GB
- Battery power bank
  - Lithium Polymer, output 5V/3.4A(max) → 12Wh Capacity

Motorcycle

- Raspberry Pi 3 A+
- NVIDIA Jetson Nano
- Adafruit MPU6050 6-DoF Accel and Gyro Sensor - STEMMA QT Qwiic
- 6mm 3MP Wide Angle Lens for Raspberry Pi HQ Camera
- Raspberry Pi high quality camera 12MP

14) Updated Hardware Diagram

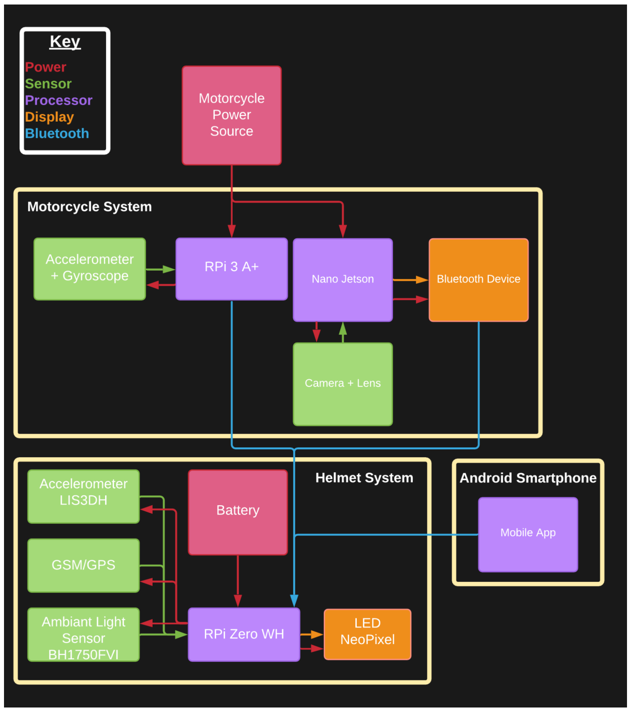


Figure 4: Hardware diagram displaying how the components are connected as a whole

15) PCB Design

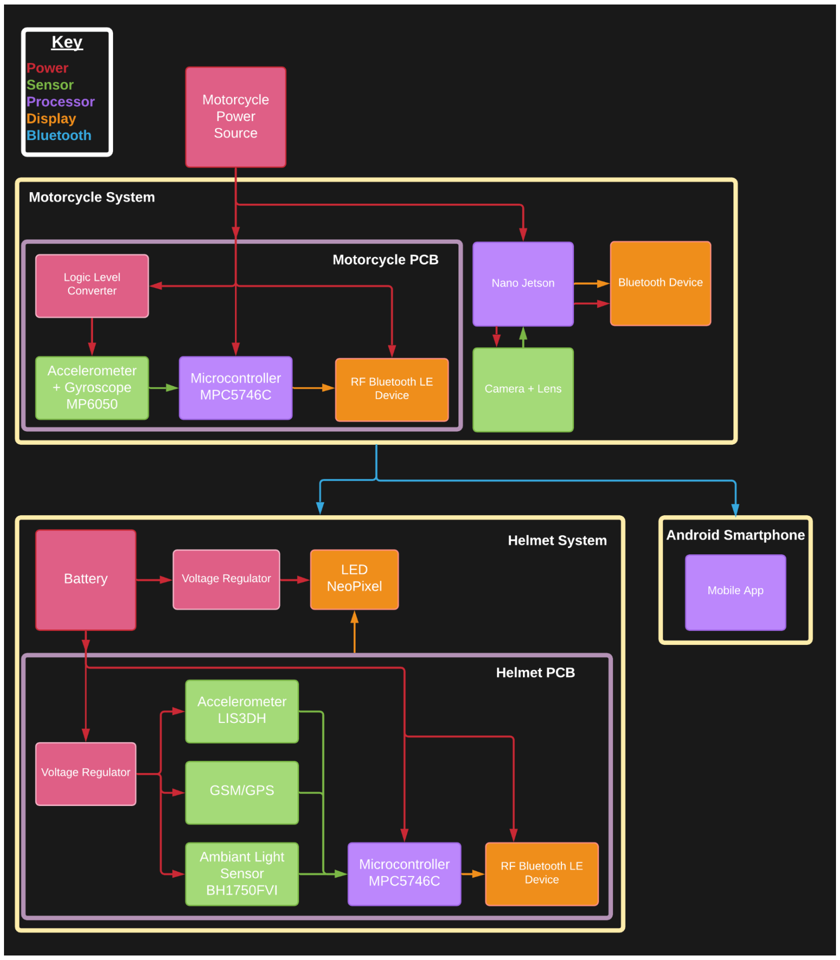


Figure 5 : PCB diagram showing the connections of the major components

16)Software Flowchart

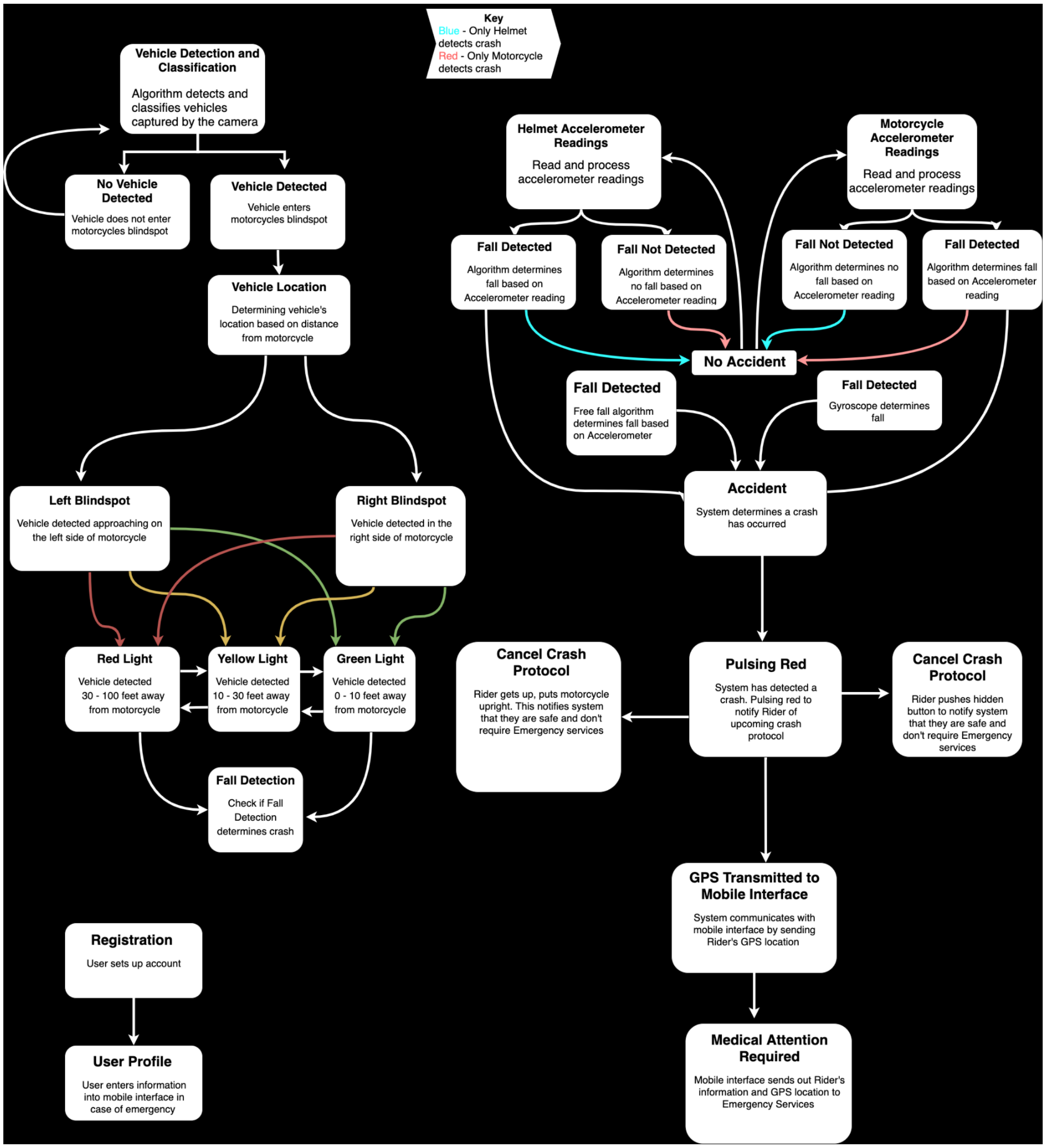

Key
Blue - Only Helmet detects crash
Red - Only Motorcycle detects crash
Vehicle Detection and Classification
Algorithm detects and classifies vehicles captured by the camera
No Vehicle Detected
Vehicle does not enter motorcycles blindspot
Vehicle Detected
Vehicle enters motorcycles blindspot
Vehicle Location
Determining vehicle's location based on distance from motorcycle
Left Blindspot
Vehicle detected approaching on the left side of motorcycle
Right Blindspot
Vehicle detected in the right side of motorcycle
Red Light
Vehicle detected 30 - 100 feet away from motorcycle
Yellow Light
Vehicle detected 10 - 30 feet away from motorcycle
Green Light
Vehicle detected 0 - 10 feet away from motorcycle
Fall Detection
Check if Fall Detection determines crash
Registration
User sets up account
User Profile
User enters information into mobile interface in case of emergency
Helmet Accelerometer Readings
Read and process accelerometer readings
Motorcycle Accelerometer Readings
Read and process accelerometer readings
Fall Detected
Algorithm determines fall based on Accelerometer reading
Fall Not Detected
Algorithm determines no fall based on Accelerometer reading
Fall Not Detected
Algorithm determines no fall based on Accelerometer reading
Fall Detected
Algorithm determines fall based on Accelerometer reading
No Accident
Fall Detected
Free fall algorithm determines fall based on Accelerometer
Fall Detected
Gyroscope determines fall
Accident
System determines a crash has occurred
Cancel Crash Protocol
Rider gets up, puts motorcycle upright. This notifies system that they are safe and don't require Emergency services
Pulsing Red
System has detected a crash. Pulsing red to notify Rider of upcoming crash protocol
Cancel Crash Protocol
Rider pushes hidden button to notify system that they are safe and don't require Emergency services
GPS Transmitted to Mobile Interface
System communicates with mobile interface by sending Rider's GPS location
Medical Attention Required
Mobile interface sends out Rider's information and GPS location to Emergency Services

Figure 6: Software flowchart of the two systems; demonstrates the software logic of the two systems.

17) Blindspot Flowchart

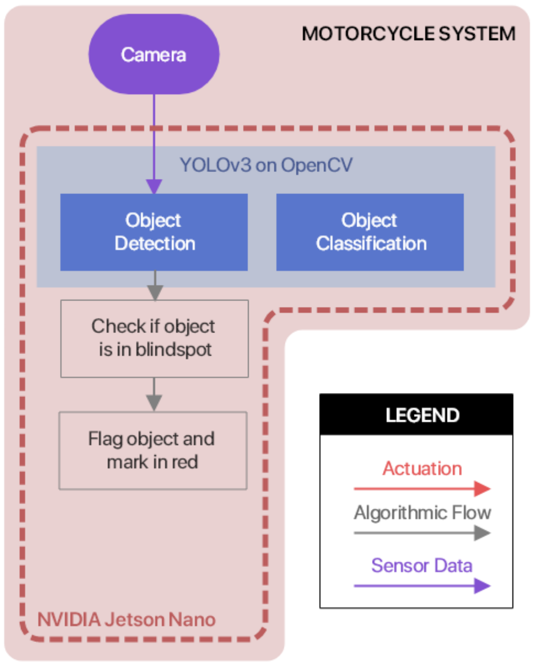


Figure 7: Diagram to show the current software implementation of Blindspot Detection system.

Determining Blindspot Region

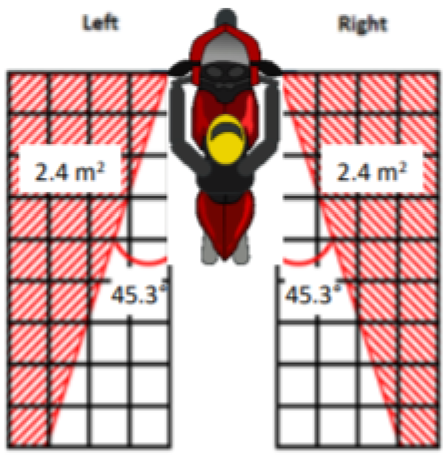
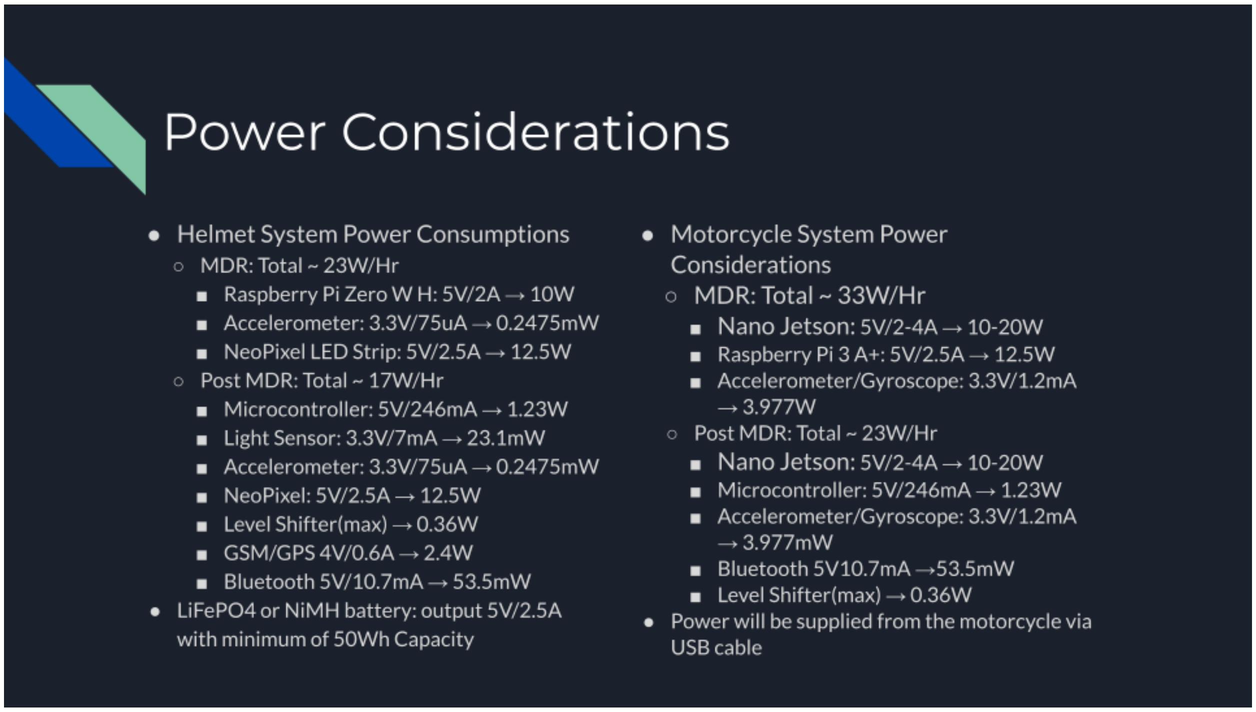


Figure 8: Diagram showing the average blindspot angle, derived from a paper by the Motorsport Technology Research Unit (MoTech) and the Malaysian Institute of Road Safety Research (MIROS)[3]

[3] M S M Hashim et al 2019 IOP Conf. Ser.: Mater. Sci. Eng. 670 012075 doi:10.1088/1757-899X/670/1/012075

18) Power Considerations



Power Considerations

- Helmet System Power Consumptions
  - MDR: Total ~ 23W/Hr
    - Raspberry Pi Zero W H: 5V/2A → 10W
    - Accelerometer: 3.3V/75uA → 0.2475mW
    - NeoPixel LED Strip: 5V/2.5A → 12.5W
  - Post MDR: Total ~ 17W/Hr
    - Microcontroller: 5V/246mA → 1.23W
    - Light Sensor: 3.3V/7mA → 23.1mW
    - Accelerometer: 3.3V/75uA → 0.2475mW
    - NeoPixel: 5V/2.5A → 12.5W
    - Level Shifter(max) → 0.36W
    - GSM/GPS 4V/0.6A → 2.4W
    - Bluetooth 5V/10.7mA → 53.5mW
- LiFePO4 or NiMH battery: output 5V/2.5A with minimum of 50Wh Capacity
- Motorcycle System Power Considerations
  - MDR: Total ~ 33W/Hr
    - Nano Jetson: 5V/2-4A → 10-20W
    - Raspberry Pi 3 A+: 5V/2.5A → 12.5W
    - Accelerometer/Gyroscope: 3.3V/1.2mA → 3.977W
  - Post MDR: Total ~ 23W/Hr
    - Nano Jetson: 5V/2-4A → 10-20W
    - Microcontroller: 5V/246mA → 1.23W
    - Accelerometer/Gyroscope: 3.3V/1.2mA → 3.977mW
    - Bluetooth 5V10.7mA →53.5mW
    - Level Shifter(max) → 0.36W
- Power will be supplied from the motorcycle via USB cable

20) Software Specifications - Blindspot Detection

Full System Flowchart | Blindspot Detection

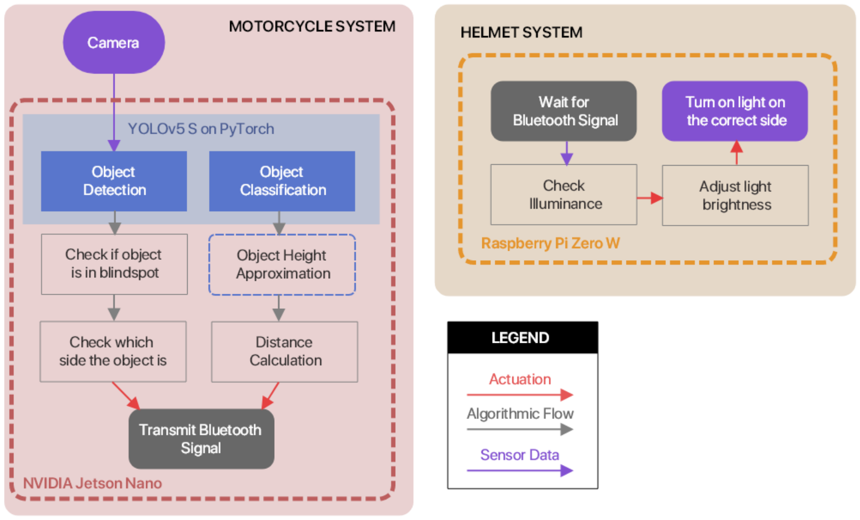


Figure 9: Diagram to show the planned software flow of Blindspot Detection system.

FPS Requirements

To analyze the required FPS for our algorithm, we look at some of the faster consumer vehicles in the market:

Vehicle	Length	Top Speed
Tesla Model S Plaid (car)[4]	16.5 ft	200 mph
Kawasaki Ninja 650 (motorcycle)[5]	7 ft	131 mph

Speed Analysis:

To travel 31.5 ft at 200 mph, it takes 0.108 seconds.
To travel 22 ft at 131 mph, it takes 0.115 seconds.

This means that the FPS > 1/0.108
FPS > 9.25

Therefore, the minimum FPS to cover the top speeds of some of the fastest consumer vehicles is 10.

Note: this is assuming that at the end of the previous frame, the vehicle is at the edge of the blindspot, therefore this calculates the minimum FPS required for the worst case scenario.

[4] https://www.tesla.com/models
[5] https://www.kawasaki.com/en-us/motorcycle/ninja/sport/ninja-650

21) Software Specifications - Fall Detection

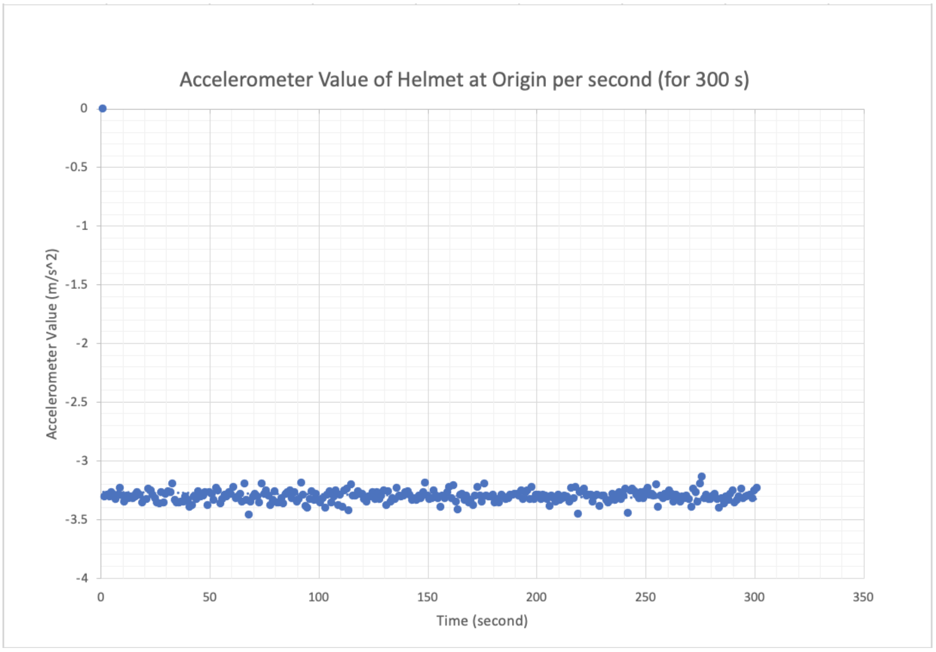


Figure 10: Accelerometer attached to the helmet is turned on for 300s and left at origin. The data indicates that the accelerometer on the helmet is capturing very little noise. The origin value for the Helmet's Fall Detection system was chosen to be -3.3 m/s^2

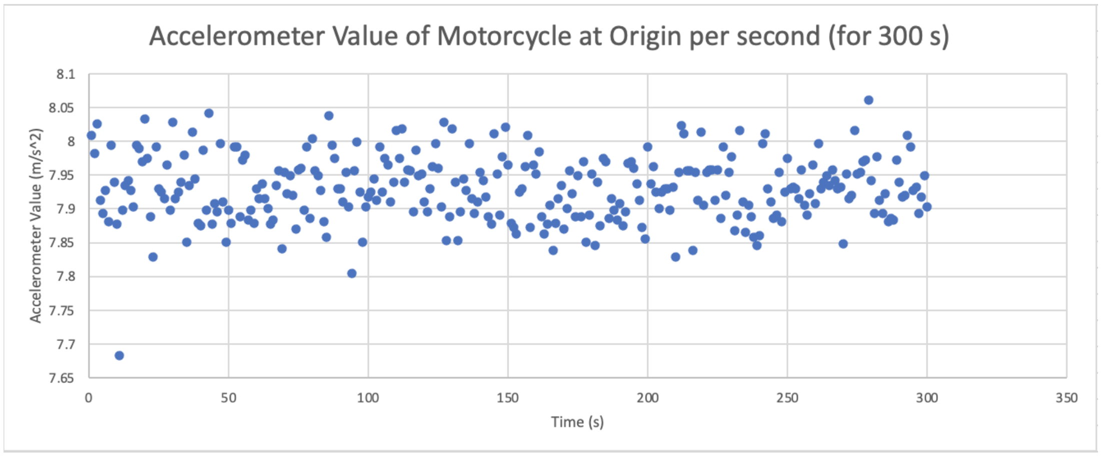


Figure 11: Accelerometer attached to the motorcycle is turned on for 300s and left at origin. Motorcycle engine vibration is simulated which results in the data indicates experiencing mild noise. The origin value for the Motorcycle's Fall Detection system was chosen to be 7.9299 m/s^2

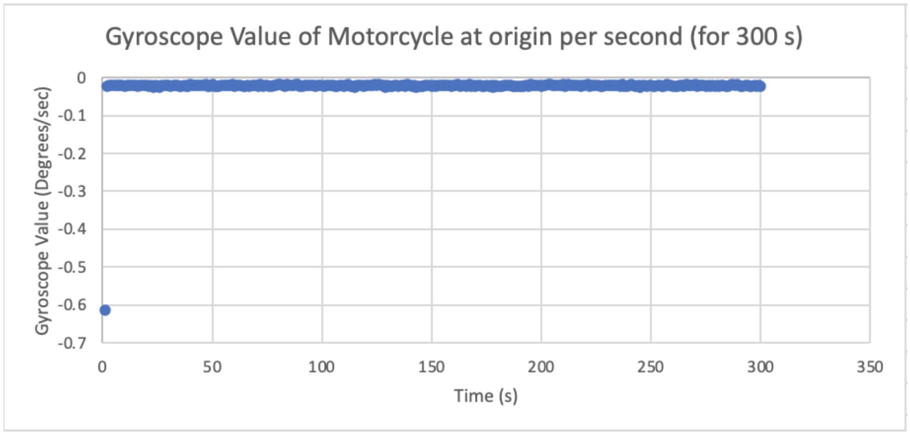


Figure 12: Gyroscope attached to the motorcycle is turned on for 300s and left at origin. Motorcycle engine vibration is simulated, however the data indicates minimal noise. The origin value for the Motorcycle's Fall Detection system was chosen to be -0.023 degrees/sec.

---

22) Bluetooth Specification

- Connected through the Serial Numbers:
  - "B8:27:EB:EE:05:DC" of the raspberry pi 0 W
  - "B8:27:EB:F9:4A:75" of the raspberry pi 3 A+
- Connection through Pybluez in python
- Low Latency where it is less than 0.005 seconds to connect between the pi 0 and the pi 3

Raspberry Pi 0W built-in 4.1 bluetooth
Raspberry Pi 3 A+ built-in 4.2 bluetooth/BLE
IoT Capabilities:

- Low-power IP (IPv6/6LoWPAN)
- Bluetooth Smart Internet Gateways (GATT)

With BLE 4.2 Bluetooth Smart sensors can transmit data over the internet.
Security:

- LE Privacy 1.2
- LE Secure Connections

With new, more power efficient and highly secure features, BLE 4.2 provides additional benefits allowing only trusted owners to track device location and confidently pair devices.
Speed:

- 250% Faster
- 10x More Capacity

Over 4.1 BLE

---

23) Mobile App Specifications

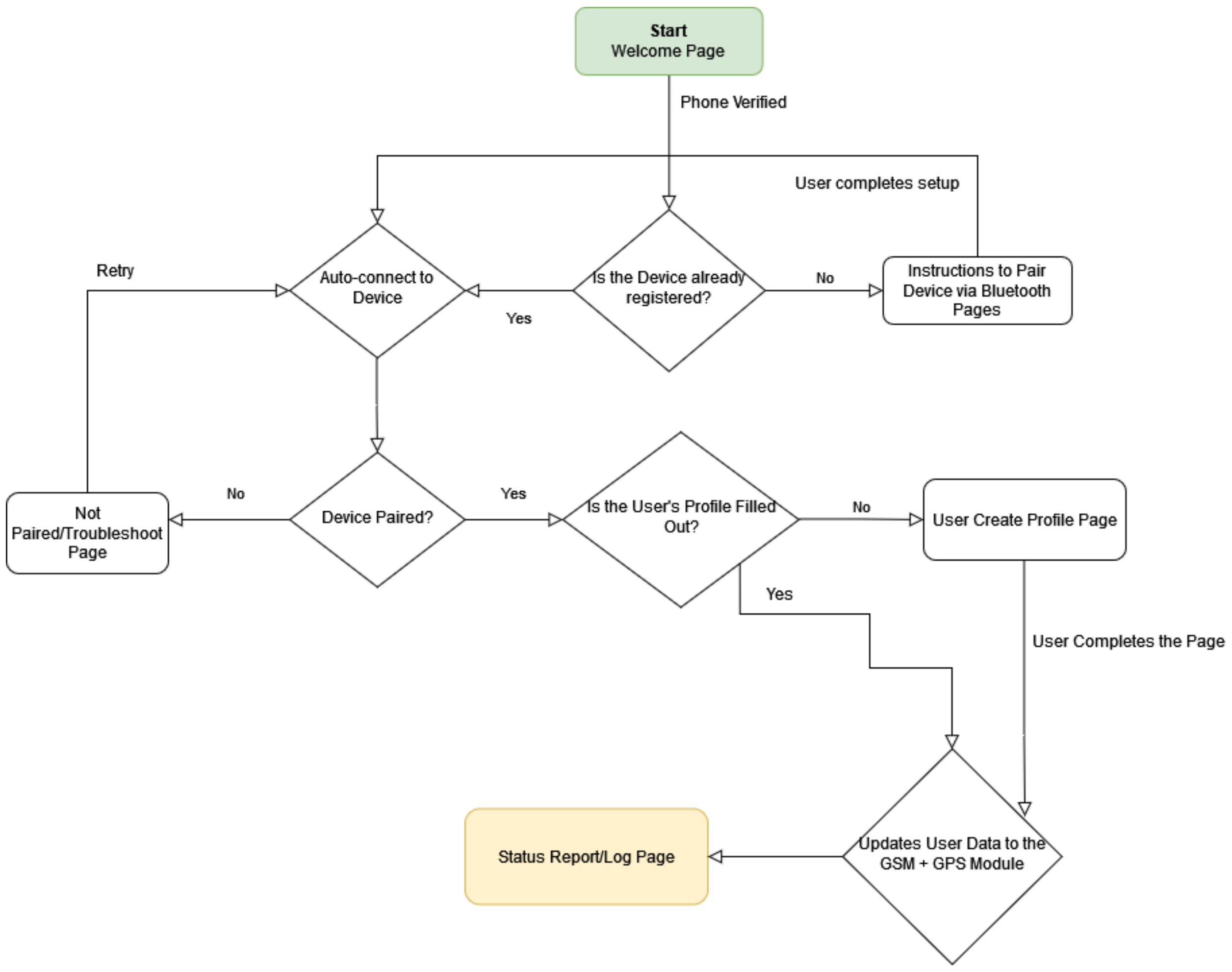

Start
Welcome Page
Phone Verified
User completes setup
Auto-connect to Device
Is the Device already registered?
Instructions to Pair Device via Bluetooth Pages
Retry
Yes
No
Not Paired/Troubleshoot Page
Device Paired?
Is the User's Profile Filled Out?
User Create Profile Page
User Completes the Page
Updates User Data to the GSM + GPS Module
Status Report/Log Page

Figure 13: Phone application flowchart

---

25) Blindspot Detection Verification

Test Footage Specifications

- 18 feet lane width
    - Highway lane width specified by MassDOT[6]

[6] MassDOT section 15.2.3 and 527 CMR 10.03

- 160 feet road length
- Equipment:
    - Camera specifications:
        - 12 MP sensor
        - 120 degree Horizontal Field of View
        - F/2.4 aperture
    - Bicycle to simulate a motorcycle
        - Video recorded at 4 feet height
    - Video specifications
        - Resolution: 1920x1080
        - FPS: 30
        - Duration: 8 min 41 sec

---

26) Blindspot Detection Demo Video

Test Results

- Detection rate:
    - 90.96% detection rate in full lane (160 ft)
    - 100% detection rate within 120 ft mark
        - Car height >= 3% of frame height (32.4 px)
- Classification rate (after detection):
    - 87.51% classification rate as car
        - Misidentifying large car as truck
    - 100% classification rate as ‘some vehicle type’
- Blindspot flag rate:
    - 100%; successfully flags the cars passing through the blindspot 26/26 times

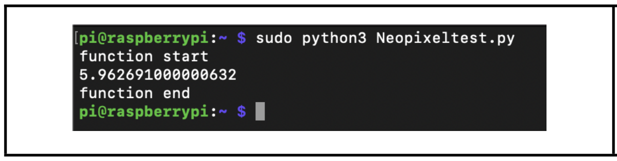
---

27) Blindspot Direction Verification & Demo

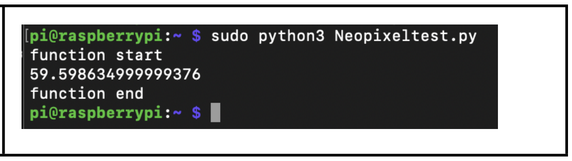
Test Results

- Communication rate:
    - 0.00019534 seconds to send and receive data

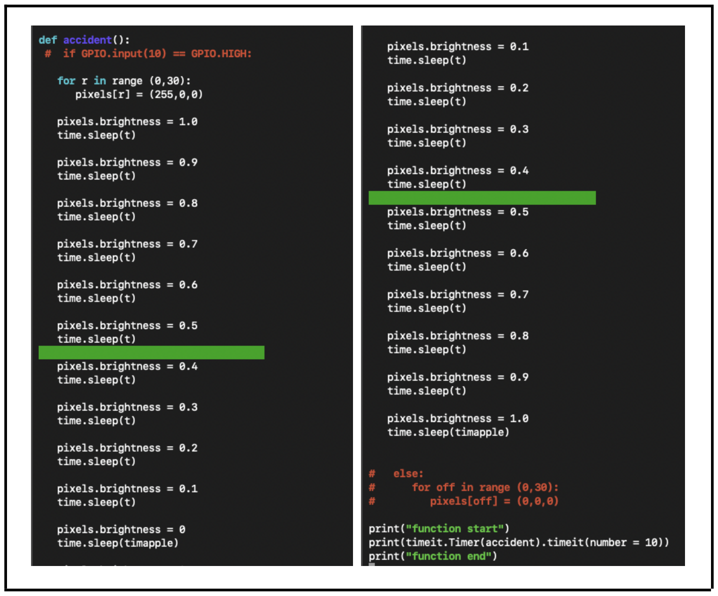
28) Fall Detection Verification & Demo



```
pi@raspberrypi:~ $ sudo python3 Neopixeltest.py
function start
5.962691000000632
function end
pi@raspberrypi:~ $
```



```
pi@raspberrypi:~ $ sudo python3 Neopixeltest.py
function start
59.598634999999376
function end
pi@raspberrypi:~ $
```



```
def accident():
 #  if GPIO.input(10) == GPIO.HIGH:

   for r in range (0,30):
      pixels[r] = (255,0,0)

   pixels.brightness = 1.0
   time.sleep(t)

   pixels.brightness = 0.9
   time.sleep(t)

   pixels.brightness = 0.8
   time.sleep(t)

   pixels.brightness = 0.7
   time.sleep(t)

   pixels.brightness = 0.6
   time.sleep(t)

   pixels.brightness = 0.5
   time.sleep(t)

   pixels.brightness = 0.4
   time.sleep(t)

   pixels.brightness = 0.3
   time.sleep(t)

   pixels.brightness = 0.2
   time.sleep(t)

   pixels.brightness = 0.1
   time.sleep(t)

   pixels.brightness = 0
   time.sleep(timapple)
```

```
   pixels.brightness = 0.1
   time.sleep(t)

   pixels.brightness = 0.2
   time.sleep(t)

   pixels.brightness = 0.3
   time.sleep(t)

   pixels.brightness = 0.4
   time.sleep(t)

   pixels.brightness = 0.5
   time.sleep(t)

   pixels.brightness = 0.6
   time.sleep(t)

   pixels.brightness = 0.7
   time.sleep(t)

   pixels.brightness = 0.8
   time.sleep(t)

   pixels.brightness = 0.9
   time.sleep(t)

   pixels.brightness = 1.0
   time.sleep(timapple)


#   else:
#      for off in range (0,30):
#         pixels[off] = (0,0,0)

print("function start")
print(timeit.Timer(accident).timeit(number = 10))
print("function end")
```

Figure 14a, 14b: Specification on the speed at which the red pulsing occurs.
Left demonstrates the speed for one cycle (where t = 0.2 s and timapple = 1s).
Right demonstrates the speed for 10 cycles (where t = 0.2 s and timapple = 1s).

30) CDR Deliverables

Verification Plan

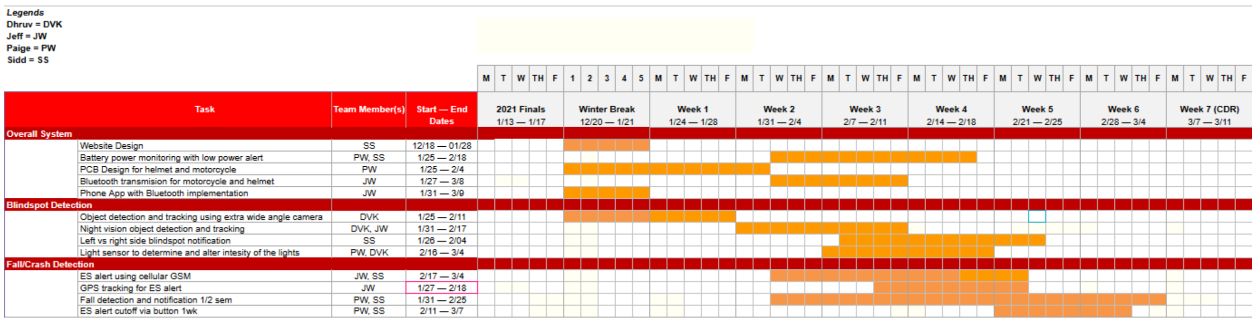
Requirement	**Verification**	**Description**
System must execute algorithm at at least 10 frames per second	Analysis	Verified by computing the average number of frames processed per second
System must detect and classify all vehicles within 120 ft of the system	Test	Run algorithm with road footage, test if all vehicles within 120 ft of the camera are being detected and classified
System must detect all vehicles in the blindspot region of the video frame	Demonstrate	Run algorithm in front of vehicles moving into the blindspot of the motorcycle & show that all the vehicles get flagged within the blindspot
System must detect and classify all vehicles within 80 ft of the system in low-light environment	Test	Run algorithm with low-light footage, test if all vehicles within 100 ft of the camera are being detected and classified
System must determine the proximity of vehicles within the blindspot region and classify into one of three categories: very near (0-10 feet), near (10-30 feet), far (30-100 feet)	Demonstrate	Run algorithm in front of vehicles at various ranges within the blindspot of the motorcycle & show that the proximity of all the vehicles is determined accurately
System must be able to notify the rider what side, left or right, a vehicle is approaching from	Demonstrate /Test	Run algorithm to track the location of oncoming vehicles entering blindspot, and display LED Strip will light up for the corresponding side
System must be able to accurately initialize crash protocol	Test	Run algorithm to detect 100% of crashes for crash protocol to be initialized
System must be able to terminate the crash protocol through user input	Demonstrate	When the rider presses the hidden button or when the rider and motorcycle are upright, the algorithm must terminate the crash protocol every time.

31) Budget

Item	Cost	Quantity	Total		Item	Cost	Quantity	Total
Motorcycle:					**Helmet**			
HQ Camera	$50.00	1	$50.00		Light Sensor	$4.50	1	$4.50
Wide Angle Lens	$25.00	1	$25.00		Accelerometer	$4.95	1	$4.95
Accelerometer & Gyroscope (MEMS)	$6.95	1	$6.95		Raspberry Pi Zero W	$14.00	1	$14.00
Raspberry Pi 3 A+	$25.00	1	$25.00		SD Card	$9.99	1	$9.99
USB V4.0 Bluetooth Adapter	$13.95	1	$13.95		LED RGB Strip	$17.95	1	$17.95
					Level Shifter	$3.95	3	$11.85
		M Subtotal	$120.90		GSM + GPS	$0.00	1	$0.00
					Helmet	$69.99	1	$69.99
Shipping:								
Adafruit	$11.67		**Subtotal**	$254.13			H Subtotal	$133.23
Amazon	$32		**Total**	$298.00				

Item	Cost	Quantity	Total
Future:			
PCB	$100.00	TBH	$100
Cellular Model w/ GPS	$50.00	1	$50.00
Batteries	$5	4	$20
Subtotal	$120		

32) Gantt Chart



Legends
Dhruv = DVK
Jeff = JW
Paige = PW
Sidd = SS

	Task	Team Member(s)	Start — End Dates
Overall System			
	Website Design	SS	12/18 — 01/28
	Battery power monitoring with low power alert	PW, SS	1/25 — 2/18
	PCB Design for helmet and motorcycle	PW	1/25 — 2/4
	Bluetooth transmision for motorcycle and helmet	JW	1/27 — 3/8
	Phone App with Bluetooth implementation	JW	1/31 — 3/9
Blindspot Detection			
	Object detection and tracking using extra wide angle camera	DVK	1/25 — 2/11
	Night vision object detection and tracking	DVK, JW	1/31 — 2/17
	Left vs right side blindspot notification	SS	1/26 — 2/04
	Light sensor to determine and alter intesity of the lights	PW, DVK	2/16 — 3/4
Fall/Crash Detection			
	ES alert using cellular GSM	JW, SS	2/17 — 3/4
	GPS tracking for ES alert	JW	1/27 — 2/18
	Fall detection and notification 1/2 sem	PW, SS	1/31 — 2/25
	ES alert cutoff via button 1wk	PW, SS	2/11 — 3/7