AutoAlert

Eunice Lee, Ankit Lenka, Emily Szabo
Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**With an attachable dashcam set up, AutoAlert brings advanced safety features only found in luxury cars to any car at a significantly cheaper price. With lane detection, traffic light change detection, forward car departure alerts, and forward collision warnings, our device bridges socioeconomic gaps to create a more accessible and safe driving space.

***Index Terms*—**Lane Detection (LD), Traffic Light Detection (TLD), Forward Car Departure (FCD), Forward Collision Warning (FCW)

I. Introduction

The vast majority of Americans drive frequently, and with the evolution of technology it has become increasingly important to be able to provide more effective driving safety features to increase safe driving. Many luxury car brands are very successful in this respect; features of interest they've developed include lane detection (LD), traffic light change detection alerts (TLD), forward car departure alerts (FCD), and forward collision warnings (FCW). These advances are amazing; however, only luxury cars implement these features, and this introduces a socioeconomic gap such that lower income drivers do not have access to more complex safety features.

AutoAlert's main focus is to bridge this socioeconomic gap by redesigning complex safety features at a cheaper price so they are accessible to lower income drivers. Specifically, we want to implement traffic light change detection alerts (which we also have to implement lane detection for in order to ensure the user is receiving notifications for the correctly-aligned traffic light respective to their lane), forward car departure alerts, and forward collision warnings. These features will be wrapped together into a dash cam device that you attach to your front windshield, so any car can use this as an add-on safety feature. The device will connect with your phone via Bluetooth and send audio alerts when a traffic light has changed from red to green, the car in front of you has started moving, and/or you are about to crash into a car in front of you.

There are currently no other car safety products that implement the extensive functionality our project aims to achieve at the price we have set (under $600). In this way, our product is filling a market gap to provide advanced car safety features for a low price.

II. Use-Case Requirements

The use-case requirements are split into 4 subsequent sections: LD, TLD, FCW, FCD.

LD, used solely for traffic light change detection, needs to be able to detect lane position to correctly determine the corresponding traffic light for the user. The LD must be able to detect correct lane position for up to 99% on good quality roads and 90% on poor quality roads, where lane position is defined as an integer value based on if there are lanes to the left or to the right of the driver's position. The success rate is further described in section VII, when discussing specific testing metrics. Additionally, the LD must be able to correctly identify lane position after changing lanes within 2 seconds.

TLD must correctly identify traffic lights and the alert must notify the driver in time when the light turns from red to green. 90% of traffic lights must be identified, due to approximately 90% of traffic lights being the standard three light traffic light in the United States. Drivers normally wait about 3 seconds before honking and the average human reaction to sound is 150ms, so the alert has an upper bound of 2.85 seconds. In order to give the user more time to react, the TLD alert must notify the driver within 2 seconds of the light changing.

FCD must alert the driver that the car in front of them has departed, only when the user is at a standstill. This notification must be given within 2 seconds of the car in front moving 10ft from its original position.

FCW must notify the driver to brake if the car in front of the driver suddenly stops for cars going from 3 miles per hour to 40 miles per hour. The system must be 99% accurate, precisely measuring distances up to 50m.

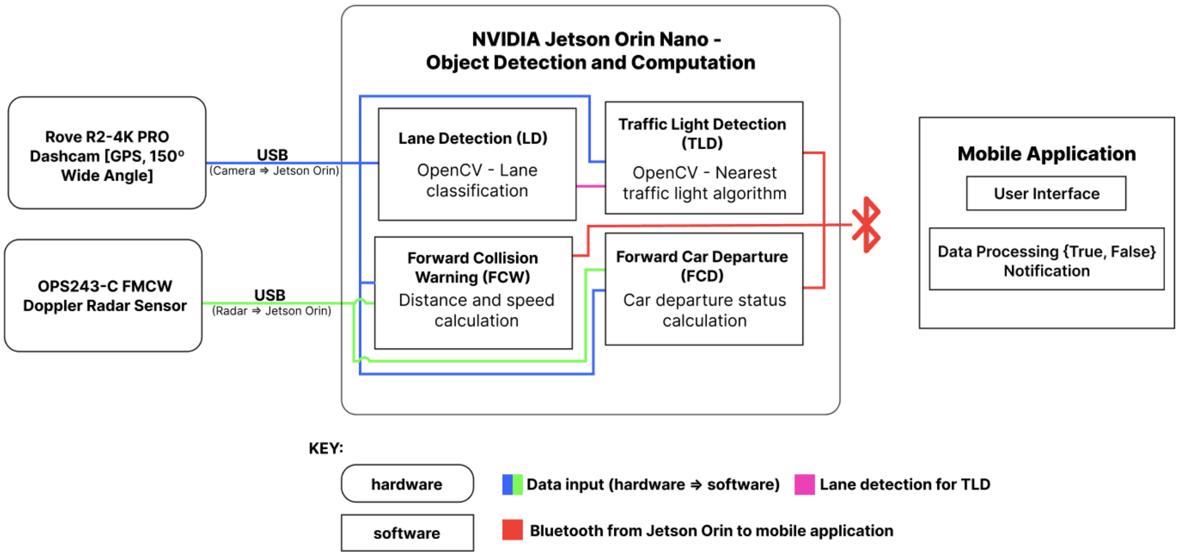

NVIDIA Jetson Orin Nano - Object Detection and Computation
Rove R2-4K PRO Dashcam [GPS, 150° Wide Angle]
USB
(Camera ⇒ Jetson Orin)
OPS243-C FMCW Doppler Radar Sensor
USB
(Radar ⇒ Jetson Orin)
Lane Detection (LD)
OpenCV - Lane classification
Traffic Light Detection (TLD)
OpenCV - Nearest traffic light algorithm
Forward Collision Warning (FCW)
Distance and speed calculation
Forward Car Departure (FCD)
Car departure status calculation
Mobile Application
User Interface
Data Processing {True, False} Notification
KEY:
hardware
software
🟦🟩 Data input (hardware ⇒ software)
🟪 Lane detection for TLD
🟥 Bluetooth from Jetson Orin to mobile application

Figure 1. Block diagram figure shows the connections and overall data input/output of each subsystem

III. Architecture and/or Principle of Operation

The architecture reveals the four main functionalities of the system. The data from the dashcam and the radar sensor are connected via USB 3.0, both powering the two devices and sending data to the processing component. Using the radar's API and image processing on the NVIDIA Jetson Orin Nano, four main components compute the data that it needs to make a decision on whether or not to alert the driver.

LD utilizes only the dashcam as an input, capturing snapshots of the road in front of the driver. After image processing and detecting lanes, further specified in Section VI, it sends data into TLD to help TLD determine the closest traffic light that is associated with the user. TLD, in addition to this new computed data, also takes in the dashcam's video as input. With both image and lane position, it calculates if the nearest traffic light has turned from red to green.

Both FCD and FCW utilize the dashcam and the radar as the input, factoring in image data of the road, the driver's current speed from the GPS on the camera, the forward object's speed, and the distance between the forward object and the current user's car. With these four components, these two subsystems are able to compute whether or not to alert the user.

The mobile application is connected to the hardware via bluetooth connection, where the application receives a boolean value. This value is a flag to notify the user, and comes from only three of the four subsystems: TLD, FCD, and FCW.

With this flow of information, all of the work that requires fast computation and decision making occurs on the hardware, and the user only interacts with the mobile application after the initial setup of the entire device. The separation of components between the mobile application and the hardware allows for both to be developed simultaneously. They can then be joined together at the end, interacting with each others' interfaces with enough abstraction to allow for smooth connection.

IV. Design Requirements

The maximum time the mobile application will need is 100ms, based on Swift metrics. For general bluetooth applications, a maximum of 300ms needs to be allocated. This allows for the data processing and computation section to take 1600ms max.

The dashcam has a 150º wide angle, 2160 UHD, and 30 FPS. With the assumption that color depth is 3 bytes/pixel, we can use this equation to determine the size of each snapshot take from the dashcam:

$$Data\ Size\ =\ Pixels\ \times\ Color\ Depth$$

The data size is (3840 x 2160) x (3 bytes/pixel) ~ 25 MB for a standard RGB image. USB 3.0 supports speeds of up to 5 Gbps, or 625 MB/s theoretically. To account for overhead, we are using the value of 400 MB/s for our data transfer rate. This allows for approximately 65ms to send 1 frame over USB 3.0. To detect lanes, another maximum of 100ms will be needed. Then, another 35ms are needed to compute the lane position integer. This totals to 200ms, but to allow for slack, the design requires this to occur in 400ms. As a result, a new lane position is fed into the TLD at a maximum of 400ms.

TLD requires data from both LD and the camera. Once the car comes to a stop, then TLD begins to compute data and detect the nearest traffic light every 400ms as well, similar to the lane position detection. After a change in the traffic light is detected, this new computation of alerting the notification can take up to 800ms max in order to meet the 2 second time constraint.

FCD is similar to TLD, but requires additional data from the radar. The data can be sent relatively quickly, within 100ms. With this additional data, given that the data from the radar and the camera are synchronized, there is up to 800ms max in order to meet the 2 second time constraint as well.

FCW requires both data from the radar and the camera, but instead of a 2 second time constraint, the system should compute and alert the user as soon as possible. If the radar is sampled every 100ms and the camera data can be abstracted every 400ms, we want to be able to do the other computation within 400ms.

V. Design Trade Studies

A. *Image Capturing Device*

When researching our image capturing device, many considerations came into play. We quickly decided to invest more time into the dashcam route than the traditional camera. Dashcams were designed with long term recording and night vision in mind, features that are often harder and more expensive to find on cameras. Additionally, dashcams often come with the hardware necessary to keep it continually powered off the car's battery whereas a camera would require us to find an additional power source for it.

Table I. Dashcam Tradeoffs

Model	**Price**	**Capabilities**
Azacvb DashCam for Cars	$69.99	Night Vision 170 Wide Angle
Pruveeo 360 Degree 4 Channel DashCam	$189.99	Night Vision 4 Cameras Built-in GPS
Rove R2-4K Pro DashCam	$129.99	Night Vision 150 Wide Angle Built-in GPS USB-C Data Output

Table I showcases some of the models we were most interested in, as well as their price and features. Our primary requirements involved price, night vision, built-in GPS, wide angle view, and wired connectivity. The night vision would allow for the system to operate in night time as well as harsher weather conditions such as rain and fog, while the wide angle view would capture more available and hopefully capture all necessary features for our object detection processes. Finally, the GPS was essential in tracking the vehicle's speed, otherwise, more research and computational effort would be spent calculating a method to obtain this information. Furthermore, because our project had to run in cars, we did not want to rely on wifi as a means of communication, thus requiring wired connectivity capability for our dashcam to be able to connect to our computer board. With this in mind, we quickly eliminated the Azacvb, which although was the cheapest, did not offer built-in GPS. Additionally, although the Pruveeo included multiple cameras, and had most of our features in mind, we could not ascertain whether or not it had wired capabilities, as it was listed nowhere on the specifications sheets and nowhere we could find in pictures. Rather than take the risk, we decided to move forward with the Rove, which although did not have as wide angle view as either the Azacvb or include multiple cameras like the Pruveeo, still offered all of our required features and at a reasonable price.

B. Distance and Speed Sensors

In addition to the dashcam, we also wanted a second source of data, specifically for forward obstacles. Initially a LiDAR was proposed for long range distance detection, however we quickly realized that it would require it to operate outside the car, which raised concerns about connectivity with the rest of our system as well as keeping the LiDAR protected from the elements. As a result, we refocused our efforts on radar sensors. Specifically, we focused on systems with a frequency modulated continuous wave (FMCW) sensor, operated at either the 24 GHz or 77 GHz ranges, and could detect a minimum of 50m. Normally, radar sensors use the doppler effect to calculate the speed of detected objects. FMCW is a specific technique that modulates the frequency wave as it is being sent out, thus the reflected wave gets received at a different frequency than it was emitted. The equation below represents this relationship, where *delta f* is the change in frequency, *t* is the chirp time, *c* is the speed of light, *bw* is the bandwidth, and *d* is the distance.

$$d = \frac{\Delta f \times t \times c}{2 \times bw}$$

Research indicated that normal car sensors were divided into two groups, short range and long range. When the industry shifted from 24 GHz to 77 GHz, the transition mainly benefited short range sensors, which helped with parking assist, lane assist, and better distinction of different objects at close range, while long range sensors mainly gained better adaptive cruise control. Due to these factors, we determined that for our use-case, either a 24 GHz or a 77 GHz was acceptable. Finally, in order to give ample time for a driver to break, we used the braking distance formula $d = \frac{v^2}{2ug}$, where *v* is initial velocity, *g* is acceleration due to gravity, *u* is coefficient of friction between the tires and the road, and *d* is the distance. In the worst conditions (i.e. slow driver reaction time, wet roads, etc), driving at 40 mph, requires at least 50 m of braking distance. Combining these strict requirements along with price restrictions, our team could only find 1 radar sensor that met all of them, the OPS243-C FMCW and Doppler Radar Sensor, which can detect the relative speed and distance of objects up to 60 m away, and up to 50 m detecting through glass.

C. Computer Board

Our plan was to spend the majority of our budget on our sensor and data capturing equipment, and rely on the current ECE inventory to supply us with our computer boards to run our detection processes. In particular, we were looking at the Kria KV260 Vision AI, the Nvidia Jetson Orin Nano Developer Kit. Our team did not believe that the raspberry pi computer boards would be suitable for the heavy ML object detection tasks our project requires and due to the additional cuda cores, improved architecture, and larger GB memory, the Jetson Orin Nano should outperform any of the Jetson Nanos. We ultimately decided to move forward with the Jetson Orin Nano for a few reasons. Due to the fact that we are relying on the ECE inventory to supply us, it meant that we had to compete with other groups for our components, as there was only 1 Kria KV260 Board, but multiple Jetson Orin Nanos. Additionally, the Jetson Orin Nano also came with bluetooth capability, which we realized we could use to connect with our mobile app, while the KV260 does not. Additionally, for further clarification as to why we did not run the software on our mobile app, we would still require a board to do some of the image processing work, as well as maintain a connection between the data and the phone, since otherwise, it would have been a trickier problem to connect the sensors to any mobile device directly. As such, the team decided to research specifically for computer boards with the capabilities of running our CV processes.

D. Power Supply

One of the benefits of using the dashcam was it greatly simplified our power requirements. There are 3 components that need to be supplied constant power, the computer board, the radar, and the dashcam. Our plan as of right now is to power the radar and dashcam from the Jetson Orin, and then to power the Jetson Orin from a car's cigarette port. However, due to components not yet arriving as of writing this report, whether or not this will fully satisfy our power requirements or if we will need to find a different source is still left untested. What we can confirm is that the dashcam can, and in fact must, be powered through the Jetson, however, we are still uncertain if the radar can power itself off the Jetson and if the cigarette port can provide enough voltage and wattage for the Jetson's 12V at ~10W efficiency. We still have enough budget to buy a suitable power brick.

E. *System Connections*

Our system relies on two data connection pathways, the captured sensor data to the Jetson, and the computed data from the Jetson to the mobile app. We will use USB and USB-C cables to transmit the data from the dashcam and radar to the Jetson. Having wired connections is a must. Dashcams only tend to support wireless wifi connections, which will not work for our project and our radar only supports USB connections, further necessitating wired connections. For the mobile app, we wanted users not to be distracted with their phones or worried about if it gets disconnected or gets shuffled around. As such, we are planning on using a wireless connection, specifically bluetooth, as it is a common capability between phones and the Jetson and does not require any internet connection for either.

F. *Mobile Application*

Our mobile application uses Swift for development. While using Swift means that our development will only be focused on iOS and we will not be developing for Android phones, we decided that Swift allows us to focus on performance on one subset of mobile phones. Running an iOS app built with Swift on an iPhone doesn't require a separate interpreter to run, while React Native could require one. In addition, having XCode for our Swift development allows us to have access to the powerful tool that includes debuggers. Swift also allows for an accurate native user interface (UI) and has many updated libraries to use.

G. *Software Libraries*

In terms of software libraries choices, there were two front runners, yoloV8 and openCV. YoloV8 uses a deep learning ML model and was specifically designed with object detection and tracking in mind, which will make up a large part of our software processes. OpenCV, on the other hand, is a more well-rounded library, offering support for image-processing and other CV applications. Additionally, openCV is less computationally intensive than yoloV8. As such, we plan to use both softwares throughout our project, balancing the more powerful but computationally expensive yoloV8 object detection with the image processing and counting of openCV.

VI. System Implementation

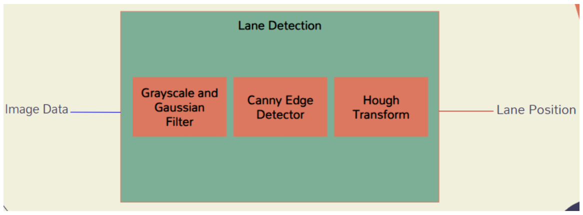
A. *Overall System Details*

The primary details of the system implementation revolve around each individual subprocess, their inputs, outputs, and the techniques and algorithms used throughout as part of that process. Each process should be assumed to be coded using openCV, yoloV8, or both. Furthermore, as seen in the overall system implementation shown in the Architecture section, the product will receive inputs from the dashcam and radar sensor and pass them along individually to each process they contribute towards. All major processes on the Jetson Orin Nano will be run in parallel and an important step in this is making sure that the data is synchronized both as it travels from the sensors to the major processes but also from the Jetson to the mobile application through bluetooth.

B. *Lane Detection Implementation*

The main focus of this particular subprocess is to detect straight line edges of the lane markers from images. From Image I, we can see that the only inputs are the images from the dashcam. Initially, the incoming frames will be grayscaled and a gaussian blur will be applied to smooth the image and remove the majority of noise. Afterwards, we will use the Canny Edge Detector and Hough Line Transform. The Canny Edge Detector applies an intensity gradient on the image based on the surrounding pixels and will specifically highlight the straight lines present, in this case, the lane markers. The Hough Line Transform will then detect the specific lane marker lines based on their slope, which help us determine left and right lanes. Afterwards, the process returns an integer that corresponds to the current lane the car is in.

Image I. Lane Detection Diagram

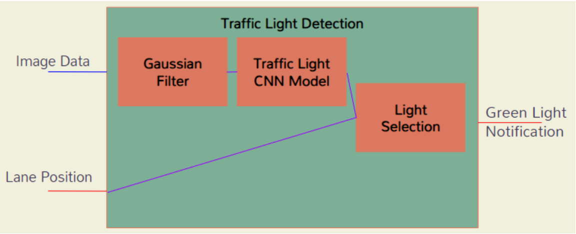


C. *Traffic Light Detection*

Once we have determined the lane in which the car exists, we feed this information into the traffic light detection algorithm. The process operates in two sections. The first is a state machine that keeps track what the vehicle has seen so far. As we can see in Image III, the states include no light detected, red light detected, and green light detected. Additionally, the process outputs a boolean value indicating to turn on the notification, specifically whenever there is a state transition from red light detected to green light detected.

The second is the actual object detection for the traffic lights. As shown in Image II, we will again apply a gaussian filter but instead feed it into a CNN model that will accurately detect the traffic lights through shape, particularly the circles of the lights. Afterwards, we use the previously calculated lane position to identify the corresponding traffic light to the lane that the vehicle is in. Finally, we will apply multiple color filters in order to determine the color of the light.

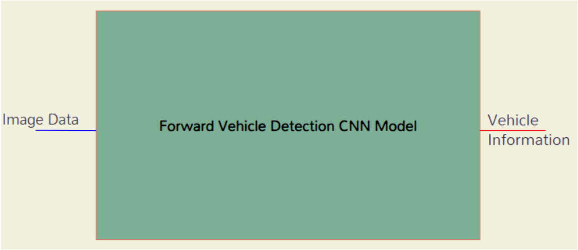
Image II. Traffic Light Detection Diagram



D. Forward Vehicle Detection

An important subsection of our system is detecting forward vehicles, which will be used in both our forward car departure and our forward collision warning. Similar to the traffic light detection, this will be done through a CNN model except this time, the object in mind are cars, specifically car rears. Certain features that will be usually in detecting accurate car rears are license plates, brake lights, and trunks. Furthermore, we can use the continuous stream of images to generate an estimate of the distance between cars and the relative speed. The process will output if a forward vehicle is detected, and the estimated speed and distance it is relative to our car. Image III depicts this subsection.

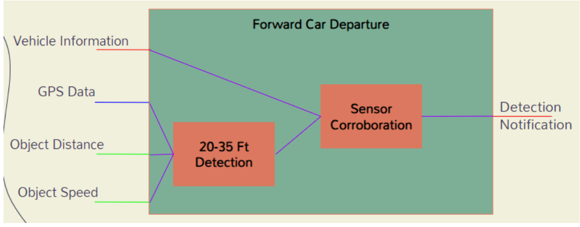
Image III. Forward Vehicle Detection Diagram



E. Forward Car Departure

One of the two processes the forward vehicle detection feeds into is the forward car departure notification. It also takes additional inputs from the dashcam GPS, and relative distance and speed from the data. The purpose of this double data source act is sensor fusion, which is combining data from multiple sensors in order to reduce uncertainty and variability. The radar data will better clarify the initial estimates provided by the forward vehicle detection system, and the forward vehicle detection will decrease the number of unnecessary notifications that might get triggered by other objects such as pedestrians, pets, or cars in other lanes. The GPS is important for providing absolute speed information, as this alert should only go off when the vehicle is stationary. When a forward car moves 20-35 ft away, the process will output a true boolean notification and will output false under all other conditions.

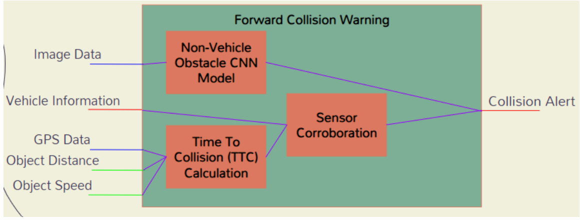
Image IV. Forward Car Departure Diagram



F. Forward Collision Warning

The final of the major four processes is the forward collision warning, and as shown in the Image V below, is also perhaps our most complex process. It takes inputs from the dashcam, GPS, vehicle detection output, and radar. The primary function of the forward collision warning runs similar to the forward car departure, except rather than give an alert when cars are moving away from our position, we use the same data to instead calculate Time-To-Collision (TTC) metrics which will then generate an alert to notify the user to slow down. In addition to the vehicle detection, the forward warning collision will also run its own object detection for bikes, pedestrians, and other immediate obstacles. A major difference from the forward car departure is where the two sensors worked to corroborate one another, creating a process where both sensors have to agree on the data in order to proceed, this is not necessarily the case for the forward warning collision. If only one of the sensors captures data that indicates an imminent collision, the process will still output a true boolean. That is not to say that it will run the data processing for the two sensors separately, it will still work together similarly to the forward car departure for calculating the TTC of forward vehicles, but that it is not necessary for it to agree. Furthermore, if we will reevaluate this decision if during testing, it generates an unacceptable level of false positives.

Image V. Forward Collision Warning Diagram



G. *Mobile Application*

The subsection of our system that the user solely interacts with is the mobile application. There are three main screens that the user will come across. The first is a setup/connection display, where the user will connect the hardware with their phone. The second page is a checkbox list of the three main features to be notified of: TLD, FCD, and FCW. Here, the user can determine which feature(s) they want, selecting up to 3 and a minimum of 1. There will also be a reminder to turn the volume up on the phone. After this, the user will then select a button to start the application and be directed to another screen. On this last screen, there will be a simple button that will stop the alert system and bring the user back to the second screen, but will be simple to keep the user free from distractions. From the hardware via bluetooth connection, the mobile application will decide if the boolean value should be processed based on what the user previously selected. If the boolean value is False, no further action will be taken. If it's true, then the phone will ring.

VII. Test, Verification, and Validation

While we develop each of our four features, LD, TLD, FCD, and FCW, we will do unit testing during the beginning stages of development. This unit testing starts with making sure we are able to smoothly receive data from our hardware (dashcam, radar) and utilize the interface with our NVIDIA Jetson Orin. Then, we will slowly build and test our software using data drawn from our dashcam and radar. After we have done sufficient unit testing, we will move on to integration testing.

We will do integration testing both by driving around in Pittsburgh and with RC cars. Our driving tests will help us test with high external validity; whereas, our RC car testing will help us ensure we are able to smoothly demonstrate our product.

A. *Tests for Camera Field of Vision*

In order to ensure that our camera is able to properly detect traffic lights, we will record videos while driving and then process the content with OpenCV to make sure that we have (1) a wide enough range of visual input to detect lane position and (2) have clear enough definition to be able to sense traffic lights.

B. *Tests for Lane Detection (LD)*

We will have four different testing situations for both driving and RC car testing: the car is surrounded by (1) no adjacent lanes, (2) only a left adjacent lane, (3) only a right adjacent lane, and (4) left and right adjacent lanes. Two edge cases we have identified are (1) switching lanes and (2) bad roads. We will be testing LD on Fifth Avenue, Washington Boulevard, and RIDC Park. Per each of the identified testing situations, we will sample our product output every 15 seconds for 10 minutes at a time. A test pass occurs when the lane detection output can correctly identify any adjacent lanes to the car within 2 seconds of the test start. A failure occurs when the adjacent lanes are incorrectly perceived. We will be taking # test passes / # total tests and producing accuracy percentages per testing situation. For our two edge cases, we will conduct the same testing and gather product output either after switching lanes or while driving on a bad road. Test passes and failures are defined the same.

C. *Tests for Traffic Light Detection (TLD)*

We will have three different testing situations for both driving and RC car testing: the car is waiting at a left turn light, the car is waiting at a typical go-straight light, and the car is waiting at a right turn light. We want to conduct at least 20 drive-throughs in each of these three situations. We will be testing on Forbes and Fifth Avenue. Per each testing situation, we will sample product output starting after the red light turns green. A test pass occurs when the traffic light notification chimes within 2 seconds of the traffic light changing from red to green. We will be taking # test passes / # total tests and producing accuracy percentages per testing situation.

D. Tests for Radar Range of Detection

To ensure that our radar can detect the 20ft - 50m distance range required by our quantitative design requirements, we will record data through a car windshield with our radar trying to detect objects as close as 20ft away and as far as 50m away to ensure that our range of detection is met.

E. Tests for Forward Car Departure

We will have two different testing situations for both driving and RC car testing: (1) stand still with the device and have the front car drive away and (2) move slowly with the device and have the front car drive away. We plan on testing this in parking lots and on Fifth Avenue. Per car departure, we will sample product output when the front car drives away. A test pass occurs when our device notifies us within 2 seconds of the forward car driving away. We will be taking # test passes / # total tests and producing accuracy percentages for each testing situation.

F. Tests for Forward Collision Warning

This feature is concerning to test (1) for driver safety and (2) for product safety because testing this on an actual road puts the driver at risk due to the risk of car collisions, and testing this with RC cars puts our device at risk to break. Because of this, we are only going to rigorously unit test this feature.

VIII. Project Management

A. Schedule

In Appendix A, you can see our GANTT chart schedule for the entirety of the semester. The most important parts lie in the development section.

During development, we will implement our product in the following order: (1) lane detection, (2) traffic light detection, (3) forward car departure, and (4) forward collision warning. Each member of our team will have individual ownership over one aspect of implementation as well as team membership with the other two members over two more aspects of implementation. In this way, we can grow as collaborators with our two other team members and grow independently during our individual contributions. Throughout development, we will all be testing our product together so we can still maintain synchronization on a weekly basis.

B. Team Member Responsibilities

Eunice is the leader in ordering and ensuring the delivery of all of our hardware required for implementation.

After our hardware has all been delivered, all three of us will work on ensuring that we understand how to interface with our devices. This entails understanding how to gather video data from our dashcam, gather radar data from our radar sensor, and spawn processes with our NVIDIA Jetson Orin.

Once we are familiar with how to work with our hardware, Eunice and Ankit will work together on implementing lane detection. During this time, Emily will work on forward car departure detection. Then, Eunice will tackle traffic light detection while Emily and Ankit implement forward collision detection. Lastly, Emily and Eunice will finalize the front end of the web application while Ankit connects the NVIDIA Jetson Orin to our mobile app via Bluetooth connection.

We will conduct testing as a team at each stage of development.

C. Bill of Materials and Budget

To show that we are under our $600 budget for our project, we have a chart of our materials and cost for each in Appendix B. Our current total cost is $521.

D. Risk Mitigation Plans

No one on our team has worked with an NVIDIA Jetson Orin before, but Ankit has been working with our TA Tjun Jet to get unblocked with a working interface with this hardware up and running.

No one on our team is familiar with using OpenCV to parse video input programmatically. To mitigate this, we have identified several OpenCV tutorials to help familiarize ourselves with usage. These include the OpenCV Bootcamp, OpenCV Real-Time Road Lane Detection, and Simple Lane Detection with OpenCV to help us get started with our development.

No one on our team has developed a mobile app using Swift before, but we are working on following documentation to install XCode to smoothly ramp up to mobile app development.

Our team has hardware experience, but we haven't specifically worked with Bluetooth before. To overcome this, we've identified an NVIDIA Jetson Orin Bluetooth Guide to help us implement this connection part of our project smoothly.

If any four of our features (LD, TLD, FCD, FCW) are not performing properly, we are planning on triaging the problem to see the severity. If the problem cannot be fixed by modifying the software, we will consider modifying our design's hardware.

IX. Related Work

Our project is originally inspired by Tesla's safety features, specifically their traffic light detection feature and their forward collision detection warning. These features are really great safety features; however, the vast majority of people are not able to afford Teslas, which is why our goal is to implement these features at a significantly lower price point.

X. Summary

In conclusion, our proposed project seeks to improve driver safety by creating an affordable device that provides crucial information to drivers in real time. The device has three primary goals: to detect the red to green light transition, detect when cars depart in front of the user vehicle when at a standstill, and to alert the user of any potential collision. Furthermore, the device will connect to the user's mobile device in order to give users better control over the notifications, while also not requiring them to use their phone during a drive.

Some of the upcoming challenges in implementation will be synchronizing the data to ensure that all processes are operating on the same data inputs at the same time, and to manage the multiple object detection in order to ensure that we can meet the timing metrics specified in the use-case and design requirements. Additionally, there are still important component requirements that will need to be tested in the beginning specifically around the power requirements, and the use of the smoker to power the entire system.

References

[1] Mananchaya Thawonsawat, Chanchai Thongsopa, Samran Santalunai, and Thanaset Thosdeekoraphat, "Detecting Targets by 24GHz FMCW Radar Technique", School of Electronics Engineering, Institute of Engineering, Suranaree University of Technology, https://omnipresense.com/wp-content/uploads/2023/04/SEATUC2023-Paper-6295.pdf

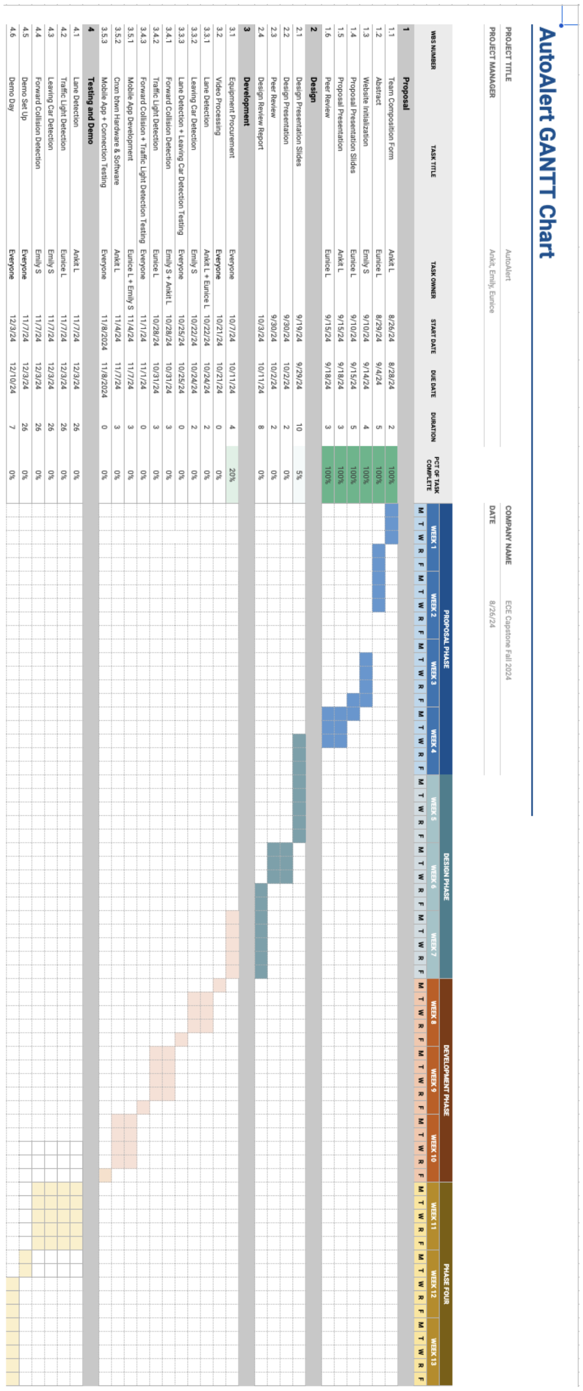
[2] Karthik Ramasubramanian, Kishore Ramaiah, Artem Aginskiy, "Moving from legacy 24 GHz to state-of-the-art 77 GHz radar", Texas Instruments, Dallas, Texas, 2017, https://www.ti.com/lit/wp/spry312/spry312.pdf

[3] *FMCW radar sensor: How it works. FMCW, CW & Pulse radar*. OndoSense. (2024, September 30). https://ondosense.com/en/radar-know-how-optimal-use-of-radar-sensors/fmcw-radar-sensor-basics/#:~:text=CW%20radar%3A%20no%20frequency%20modulation&text=In%20contrast%20to%20FMCW%20radars,mainly%20used%20for%20speed%20measurement

[4] OpenCV Documentation, https://docs.opencv.org/4.x/index.html

APPENDICES

A. Appendix A: GANTT Chart

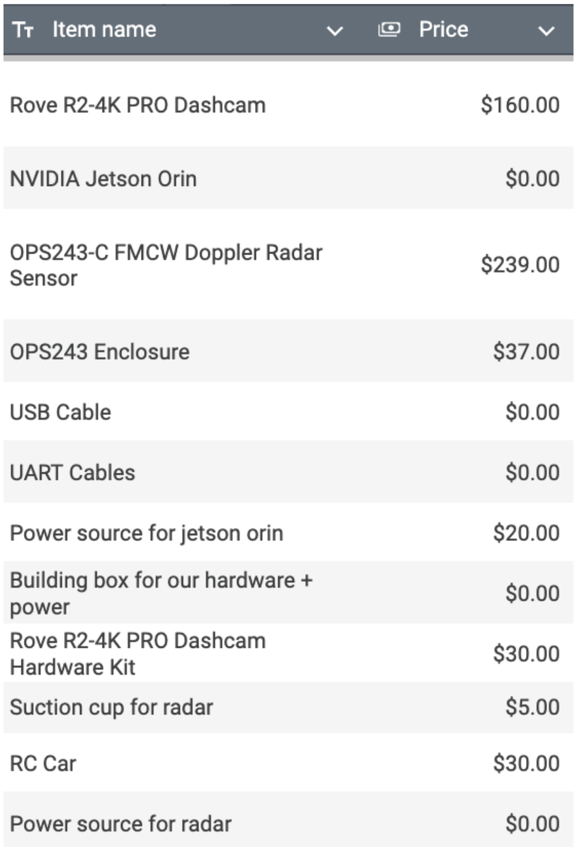


AutoAlert GANTT Chart

PROJECT TITLE	AutoAlert	COMPANY NAME	ECE Capstone Fall 2024
PROJECT MANAGER	Ankit, Emily, Eunice	DATE	8/26/24

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Proposal**					
1.1	Team Composition Form	Ankit L	8/26/24	8/28/24	2	100%
1.2	Abstract	Eunice L	8/29/24	9/4/24	5	100%
1.3	Website Initialization	Emily S	9/10/24	9/14/24	4	100%
1.4	Proposal Presentation Slides	Eunice L	9/10/24	9/15/24	5	100%
1.5	Proposal Presentation	Ankit L	9/15/24	9/18/24	3	100%
1.6	Peer Review	Eunice L	9/15/24	9/18/24	3	100%
2	**Design**					
2.1	Design Presentation Slides		9/19/24	9/29/24	10	5%
2.2	Design Presentation		9/30/24	10/2/24	2	0%
2.3	Peer Review		9/30/24	10/2/24	2	0%
2.4	Design Review Report		10/3/24	10/11/24	8	0%
3	**Development**					
3.1	Equipment Procurement	Everyone	10/7/24	10/11/24	4	20%
3.2	Video Processing	Everyone	10/21/24	10/21/24	0	0%
3.3.1	Lane Detection	Ankit L + Eunice L	10/22/24	10/24/24	2	0%
3.3.2	Leaving Car Detection	Emily S	10/22/24	10/24/24	2	0%
3.3.3	Lane Detection + Leaving Car Detection Testing	Everyone	10/25/24	10/25/24	0	0%
3.4.1	Forward Collision Detection	Emily S + Ankit L	10/28/24	10/31/24	3	0%
3.4.2	Traffic Light Detection	Eunice L	10/28/24	10/31/24	3	0%
3.4.3	Forward Collision + Traffic Light Detection Testing	Everyone	11/1/24	11/1/24	0	0%
3.5.1	Mobile App Development	Eunice L + Emily S	11/4/24	11/7/24	3	0%
3.5.2	Cnxn btwn Hardware & Software	Ankit L	11/4/24	11/7/24	3	0%
3.5.3	Mobile App + Connection Testing	Everyone	11/8/2024	11/8/2024	0	0%
4	**Testing and Demo**					
4.1	Lane Detection	Ankit L	11/7/24	12/3/24	26	0%
4.2	Traffic Light Detection	Eunice L	11/7/24	12/3/24	26	0%
4.3	Leaving Car Detection	Emily S	11/7/24	12/3/24	26	0%
4.4	Forward Collision Detection	Emily S	11/7/24	12/3/24	26	0%
4.5	Demo Set Up	Everyone	11/7/24	12/3/24	26	0%
4.6	Demo Day	Everyone	12/3/24	12/10/24	7	0%

B. Appendix B: Bill of Materials and Budget



Item name	Price
Rove R2-4K PRO Dashcam	$160.00
NVIDIA Jetson Orin	$0.00
OPS243-C FMCW Doppler Radar Sensor	$239.00
OPS243 Enclosure	$37.00
USB Cable	$0.00
UART Cables	$0.00
Power source for jetson orin	$20.00
Building box for our hardware + power	$0.00
Rove R2-4K PRO Dashcam Hardware Kit	$30.00
Suction cup for radar	$5.00
RC Car	$30.00
Power source for radar	$0.00