DriveWise

Authors: Sirisha Brahmandam, Yasser Corzo, and Elinora Stoney

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—DriveWise is a system that assists drivers with becoming safer on the road. It works by keeping track of a user's eye movements, blinks, mouth movements, and overall face position to check for signs of inattention and sleepiness. It will then notify them accordingly as well as keeping logs of their driving history. We plan on using cameras with computer vision to track the driver's face as well as an accelerometer to keep track of the car's movements. This device aims to provide more driver-focused feedback as many modern safety features in cars don't currently do that. It will also be able to be easily integrated into most cars so anyone can use the device. The domains in the project will be software, hardware, and signals and systems.**

***Index Terms*—computer vision, driving assistant, embedded system, eye tracking, facial detection, machine learning, web application**

I. Introduction

Close to a million car accidents occur annually due to distracted driving in the United States alone, with young and inexperienced drivers particularly at risk. Most cars on the road today don't give real-time feedback about a driver's attention levels. Even vehicles with preventative features are not particularly driver-focused, reporting on symptoms (ex. irregular car movement) rather than the cause of distracted driving. DriveWise is an automated driver-feedback system that can be integrated easily into any semi-modern car. The device uses facial detection to identify when a driver is inattentive to the road or showing signs of drowsiness. Real-time audio feedback will be provided to the driver alerting the driver of their distractedness. The device will also work in the cases of low-light conditions and an obstruction to the eyes (hat or sunglasses). Similar facial detection devices for drivers exist (Cipia, Smart Eye, Eyeware Tech, and Seeing Machines), but they are marketed towards car manufacturers rather than individuals. DriveWise will be able to easily integrate into any car that the user currently owns.

II. Use-Case Requirements

To ensure that DriveWise adequately addresses our user's needs, we consider the following use-case requirements.

Clear and concise feedback must be provided when the driver reaches two seconds of inattention. We define inattention as either looking away from the road or exhibiting signs of drowsiness. More specifically, (1) looking anywhere other than the windshield or rear/side view mirrors and (2) yawning, head-bobbing, or blinking at a rate higher than 10 per minute. We chose two seconds as our time limit because the US National Highway Traffic Safety Administration (NHTSA) [5] has determined it is unsafe to look away from the road for greater than two seconds. Clear feedback in these cases will provide the user with an opportunity to notice and correct their behavior.

Inattention detection accuracy should be at least 90% in ideal conditions and 85% in non-ideal conditions. We define ideal conditions as high back lighting and no obstruction of the driver's eyes. In contrast, non-ideal conditions include instances of low lighting and when the driver is wearing sunglasses or a hat. We are considering these non-ideal conditions to more accurately reflect the driving experience. We chose these percentages because similar research papers on facial detection for inattention using convolutional neural networks (CNNs), as we are using, have reported approximately 95% accuracy for our defined ideal conditions and 90% accuracy for non-ideal conditions [6].

The computation time for detecting inattention should be less than 1 second. Because the NHTSA determines 2 seconds [5] as the dangerous limit for inattention, we want our computation time to be less than one second to ensure there is enough remaining time to deliver feedback before the two second threshold is reached.

The frame rate for the video camera should be approximately 5 frames per second (fps). We are using CNNs to classify whether the video stream of the driver's face is showing signs of inattention. An fps of five will allow enough time to use CNNs to give a more accurate classification.

Users should be able to view all of their past data. This is motivated by the fact that users want to observe and monitor their progress to motivate further safe driving practices. We will accomplish this with a web application that the user can log into to view all of their past data.

User data should be private and not shared with other users or insurance companies. In this way we can protect user privacy concerns and prevent negative insurance implications, such as higher rates due to higher inattention levels while driving.

The device should be able to plug into a power source in the user's car. Direct powering from the car avoids the burden of charging in advance or changing batteries and also allows the device to be easily integrated into any semi-modern car. We define any semi-modern car as one with a 12V auxiliary power outlet or any kind of USB power port.

Feedback should not be given while the car is stationary. The user does not need to pay as much attention to the road while the car is parked or at a red light and any audio feedback given during this time would be annoying to the user.

Lastly, **the device should not obstruct the user's view of the road**, as this would jeopardize driver safety.

III. Architecture and/or Principle of Operation

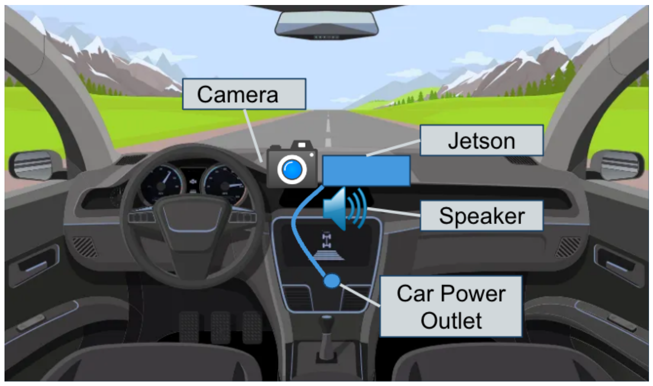


Fig. 1. This is a diagram of the layout of the larger system components on a user's car dashboard.

The device is powered by the 12V USB-A port in the user's car, connected to the Jetson via a USB-A to USB-C adaptor. Note that a different adaptor could be used to allow the device to be powered by an auxiliary power outlet instead, but the cars that we are testing the device in have a USB-A port. The Jetson houses all of the machine learning and computer vision capabilities, as well as the classification and feedback logic.

When the car is powered on, the device will immediately turn on and the camera will begin recording a video of the driver's face. However, the device will not be enabled to provide feedback until the accelerometer (connected to the Jetson via I2C) detects that the car is moving.

All of the computer vision and machine learning algorithms will be stored in the Jetson. Connected to a live camera feed, frames will be sent (once the car is in motion) to the OpenCV DNN module and its artificial neural network where, along with the landmark detector, will help classify whether a driver is distracted or sleepy through eye and mouth detection. As a result, if a driver is determined to be distracted or sleepy, audio feedback will be relayed to the driver through the speaker connected to the Jetson.

At periodic intervals of every two minutes, the Jetson will send classification data in JSON format via cellular data (over the cellular dongle) to our web application for the user to view when they log in. The web application itself will be hosted on an AWS EC2 instance, with the frontend programmed using React. If the EC2 instance is not large enough to hold the data for our MVP, we will also use an AWS S3 bucket as data storage.

IV. Design Requirements

We require the audio feedback to be less than one second long as, according to the National Highway Traffic Safety Administration (NHTSA) [5], looking away from the road for more than two seconds is considered unsafe. With the feedback being given immediately in under one second, this will give the driver enough time to hopefully adjust their behaviors accordingly. As a result, we want to make sure the feedback is clear and concise to make it clear to the driver that changes to their driving behavior are needed.

We require our facial detection algorithm to be unfazed by large head movement. This is necessary for our use case requirement of having the inattention detection accuracy be at least 90% in ideal conditions as having our facial detection algorithm fazed by sudden head movements will affect the overall accuracy of our detection model. This is especially important since similar research studies using CNNs produced 95% accuracy in ideal conditions, ensuring our product does actually promote safe driving practices in an accurate manner.

We require GPU parallelization to be compatible with the Jetson and our facial & landmark detection algorithms be less than one second. This helps ensure we meet the use case requirement of having the computation time for detection attention to the road and drowsiness be less than one second. Our justification for this requirement is the NHTSA's research concluding that inattention of the road for more than two second is unsafe [5], therefore having our computation time be less than one second gives enough time for the system to give feedback in under two seconds, ensuring the driver immediately changes behaviors as soon as the two second limit is reached.

We require our camera capture video at a frame rate of at least five fps. This ensures we reach the use case requirement of having a frame rate of five fps because we want to use CNNs as they provide a more accurate facial detection when the user's faces are at different angles. However, computation does take longer so a lower fps is needed to reach our computation time.

For the web app we want to host it on an AWS EC2 instance with an AWS S3 which will be used for additional data storage. This is required to give the ability to users to view all their past data in a web app to motivate further safe driving practices.

Web app accounts should have a login and be linked to a specific DriveWise device. This is required to address the use case requirement of having user data being private (i.e., not viewable by anyone other than the user). This is needed

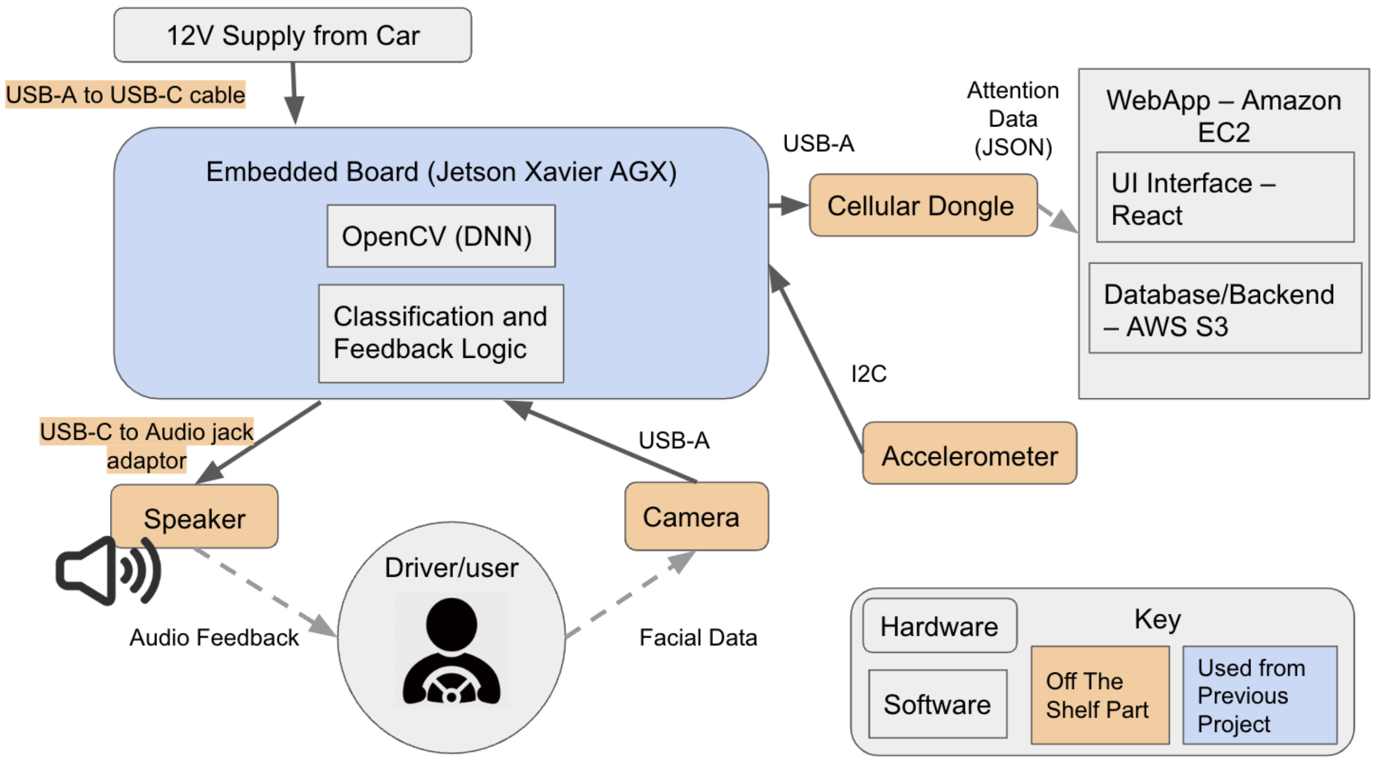


Fig. 2. This is the functional block diagram for the device.

to address user privacy concerns as not keeping this data private may have negative insurance implications. We are only considering the case that there is one driver per DriveWise device.

The Jetson Xavier must be powered by a 12V auxiliary port or USB A or C port in the vehicle. This addresses the use case requirement of having the device being able to be plugged into the power source of a user's vehicle. This also eases the user's experience with DriveWise by helping the user avoid the time and burden of recharging and replacing batteries as well as making the device compatible with any semi-modern car.

The device should be under 100x100x150mm as this will not significantly obstruct the driver's view through the windshield in order not to jeopardize the driver's safety.

V. Design Trade Studies

A number of design trade-offs have been made thus far regarding the design process of our system. Specifically, they have been made for the hardware component selection, the computer vision algorithms, the scope of the project, calibration of the device, and the integration of a web application with our physical device and how communication between the two will occur.

A. Hardware Component Selection

Embedded Board: For our embedded board, we chose the NVIDIA Jetson AGX Xavier. For machine learning and computer vision applications, NVIDIA Jetsons are often used due to their relatively compact size and high computation power. Since purchasing our own Jetson for the project would take up a significant portion of our budget, we decided to borrow one from the 18-500 class part inventory. We had the choice between a variety of Jetsons (Xavier NX, AGX Xavier, Nano, and TX2) and since price was no longer an issue, we chose the model with the highest computation power (most CPU cores and GPU frequency) to ensure that we are able to meet our design requirement of completing computation/classification in under 1 second. The Jetson AGX Xavier has 8 CPU cores and a 512-core GPU, compared to the 6 CPU cores and 384-core GPU of the Jetson NX Xavier, which is the next closest in computation power. If during testing the Jetson AGX Xavier does not operate well with the relatively low power source available from the car (12V), our backup plan is to switch to one of the Jetson Nanos available to borrow from the class parts inventory, which would have less computation power but require a lower wattage to run.

Camera: Two viable options at our disposal were the C920 Logitech Webcam and the ELP 5 USB camera as these were the two cameras mostly used in CV projects online. After further research we decided on the ELP camera because it had the best characteristics for a real-time computer vision camera: low latency, adequate low-light, and not a high frame rate. The ELP camera has a frame rate of 30 fps and latency of around 105 ms while the C920 camera has a frame rate of 60 fps and a latency of around 250 to 300 ms. In order to meet our use case requirements of having our audio feedback given in less than 1 second, a low latency is required from our camera. Therefore the ELP 5 camera is a better option.

B. *Facial Detection Models*

Facial detection is a crucial aspect of this project as it allows us to detect a driver's face, crucial in detecting facial features to determine whether a driver is drowsy or distracted. Different face detection models in Python exist with strengths and weaknesses unique to each. Therefore, it was critical to choose the best overall model for this project. After researching, we were left with two models, OpenCV's DNN and Dlib frontal face detector, as the most viable options available to us.

Dlib is a C++ toolkit binded to run in python containing machine learning algorithms used to detect faces. The frontal face detector works using features extracted from Histogram of Oriented Gradients (HOG) which are then passed into an SVM [1].

OpenCV's deep neural network is a Caffe model based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture as its backbone [4].

From our preliminary testing we found that Dlib's frontal face detector was inaccurate when it came to large angled head movements, specifically when it comes to side faces. OpenCV's DNN module was much more accurate. In addition, the frame rate with Dlib was much lower at 5.41 fps while OpenCV's DNN was much higher at 12.95 fps [1]. All in all OpenCV's DNN performed much better than Dlib which is why we've decided to use DNN over Dlib. As a result of this decision, we decided to use CNN-facial-landmarks [3] as a landmark detector that estimates the location of 68 points that map to facial structures on a user's face instead of the keypoint facial landmark detector from Dlib.

C. *Web Application*

Initially, we were considering displaying driving attention to users on either a phone or web application. While a phone app may be more convenient for the user, we settled on a web app because our team has more experience with programming web apps using React during prior internships and classes at CMU. To host the web app, we chose AWS as the cloud platform to use because we are more familiar with it than Microsoft Azure or Google Cloud.

Another design decision we had to make regarding the web application was how we would send driver attention data to it from the user's DriveWise device. Because the device must operate in a moving vehicle that does not have its own WiFi network, we considered two options for sending the data: connecting the Jetson over WiFi to a hotspot on the user's phone and connecting to the web app or connecting directly to the web app using cellular data. Each of these would require the addition of a card or dongle for WiFi or cellular data. We settled on the cellular data dongle option because this allows the Jetson to connect directly to the internet without any additional steps on the user's end.

D. *Scope*

In regard to the scope of the project, there have been several changes along the way. The most notable ones have been when discussing how the device would operate under ideal and nonideal conditions. We had initially hoped to have the device work to a near perfect accuracy under ideal conditions, which we classified as with good lighting (not extremely bright or direct sunlight and not during the night) and with no obstructions that would drastically alter the computer vision algorithms (items like sunglasses and hats which could prevent the camera from picking up on a driver's eyes), and a little more less than perfect accuracy under nonideal conditions. However, when experimenting with the computer vision algorithms, the difficulty of tracking a user's face under nonideal conditions became very evident and we opted to go about those situations through detecting a person's head position instead. This new method does require more time, so it has become a bit of a stretch goal to include the nonideal conditions. Very early on in the discussion of what our project would entail, we also considered adding features such as detection of hands on a steering wheel and making sure that the driver doesn't get too close to objects in front of the car, but we decided to narrow the scope and focus on making the facial tracking with computer vision as accurate as possible. The only new feature we added was an accelerometer to determine if the car is at a stop or not because we don't want the system to give the auditory feedback if the car isn't moving.

VI. System Implementation

A. *Calibration*

The software components of this project consists of two cycles. When a driver first uses this device, they will be required to complete a calibration process, which they are guided through with audio instructions played over the device's speaker. Afterwards, the driver will be classified as being asleep or distracted using computer vision and machine learning algorithms. This process is summarized in Figure 4.

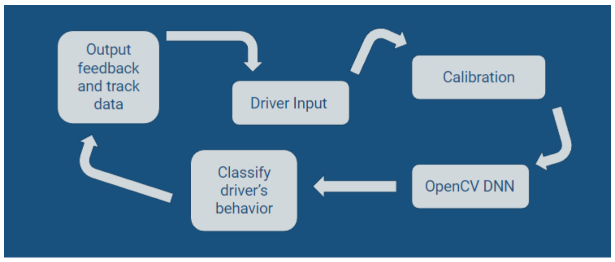


Fig. 3. Software Specification Diagram

The calibration process entails the device reading a driver's facial dimensions to help better determine whether a driver is distracted or sleepy. In particular, we are measuring the position of the driver's pupils when looking at the road and the distance between the lips when the mouth is closed as this will serve as a point of reference for a non-distracted/sleepy driver. The calibration process will also help to ensure that adequate background light is available as this will ensure that our computer vision and machine learning algorithms can correctly determine whether a driver is distracted or sleepy.

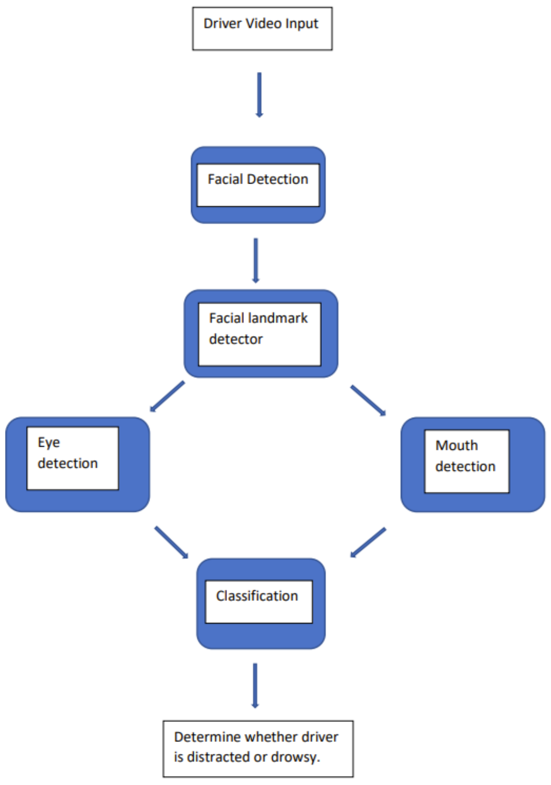


Fig. 4. Block Diagram for Computer Vision Algorithm

B. *Computer Vision*

The computer vision aspect of this project will allow us to detect whether a driver is distracted or sleepy. A vital aspect of this approach is detecting the mouth and eyes of a driver as these facial structures will be used to determine driver awareness. This is a two step press: localizing the face in the image and detecting key facial structures. We accomplish facial localization through face detection and detecting key facial structures through the use of facial landmarks.

In order to obtain facial detection, we use OpenCV and deep learning [4]. In particular, we are using OpenCV's "deep neural networks" (DNN) module which supports a number of deep learning frameworks such as Caffe, TensorFlow, and PyTorch. OpenCV's deep learning face detector is based on the Single Shot Detector (SSD) framework with a ResNet base network.

When using OpenCV's DNN module, a model is required to be imported from various frameworks compatible with DNN. The framework we will use in this project is the Caffe framework. When using OpenCV's DNN module with Caffe models, two sets of files are required: .prototxt and .caffemodel. Prototxt files define the model architecture (i.e. the layers themselves in the neural network) and .caffemodel files contain the weights for the actual layers. Both of these files allow us to read and load a network model from disk by utilizing cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"]) which explains why both files are required when using models trained using Caffe for deep learning. This returns an artificial neural network where we can pass our images as passed as a blob image (through cv2.dnn.blobFromImages['path to image'] where mean subtraction and scaling are used to preprocess these images and prepare them for classification) to obtain detections and predictions of a driver's face. It is important to note that we aren't training a neural network-rather, we are making use of a pre-trained network. Therefore we are just passing the image through the network (i.e. forward propagation) to obtain the result (no back-propagation). Our predictions are represented by localized faces though drawn bounded boxes around a user's face.

Facial landmarking is used to localize and label facial regions such as the eyes and mouth. Therefore, we use this method for eye and mouth detection. In particular, we are using a CNN-facial-landmark facial [2], a facial landmark detection based on convolutional neural network by Yin Guobing. It is built with a TensorFlow model [3] and is trained on five datasets (i.e., pre-trained network), exported as an API. This model gives 68 landmarks that map to 68 (x, y) coordinates of important facial structures on faces. By passing the TensorFlow built landmark model along with the predicted face detection rectangles to a keypoint detector, we can detect our desired key facial structures in an image.

In order to detect the eyes and mouth we must get the (x, y) coordinates from the facial landmark model that corresponds to these facial structures. In addition, we create a black mask using NumPy of the same dimensions as our webcam frame. Using the stored (x, y) coordinates of the eyes and mouth we draw these points on the mask using the OpenCV method .fillConvexPoly. Doing this returns a black mask where the eye and mouth areas are drawn in white. Using bitwise operations on images (provided by methods from OpenCV), we can apply the mask on the image to segment out the eyes and mouth. In addition, thresholding is used to create a binary mask. To obtain a more precise mouth and eye detection, thresholding processing steps namely erosion, dilation, and median blur will be used. With respect to the eye, we detect the eyeballs and find its center. We can then track the movements of the eyeball in real time and see whether the center of the eyeball has deviated from the position stored from the calibration process, indicating that the driver is looking away from the road. Note that this eye tracking method continues for each frame in the video. With respect to the mouth, this method requires us to calculate the distance of the lips and compare this distance with the closed mouth distance recorded in the calibration step. When a driver yawns, their mouth will be open and will have a lip distance greater than that of a closed mouth, indicating the driver is sleepy.

C. Sensor Input and Feedback

The device will take and process input from two different sensors: an accelerometer and a camera. The accelerometer will report data on the speed of the car and will be essentially used as an on/off switch for the auditory feedback component of the device. Once the accelerometer has detected that the car is moving forward, the feedback logic in the Jetson will be able to be triggered when the user has been classified to show signs of inattention. The accelerometer is used in this way to account for situations in which the driver may be safely looking away from the road while parked, stopped at a red light, or reversing (in which case the user could be looking backwards over their shoulder). The camera acts as another sensor, providing images at a rate of 5fps to be processed as described earlier in the Computer Vision section.

Audio feedback is provided through the speaker plugged into the Jetson. For each of the feedback cases (looking away from the road for two seconds and exhibiting signs of drowsiness) we will have pre-made sound files that the Jetson will trigger to play through the speaker when the classification logic determines that the corresponding conditions have been met.

D. Web Application

We are using React.js to program the frontend of the web app. We will be hosting the web app itself on an AWS EC2 instance and using an AWS S3 bucket for additional data storage. For setting up accounts and login capabilities we will be using Firebase.

When visiting the DriveWise web application, users will first see the login page shown in Fig. 5. If the user does not have an existing account, they can click the "Register" button to be taken to the registration page, shown in Fig. 6. To create a new account, the user must make a username and password, as well as provide the device ID from their DriveWise device, assuming that each device will have an ID printed visibly on it.

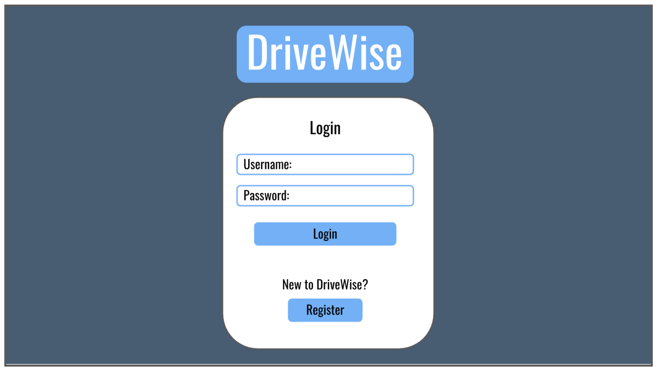


Fig. 5. Web Application: Login Page

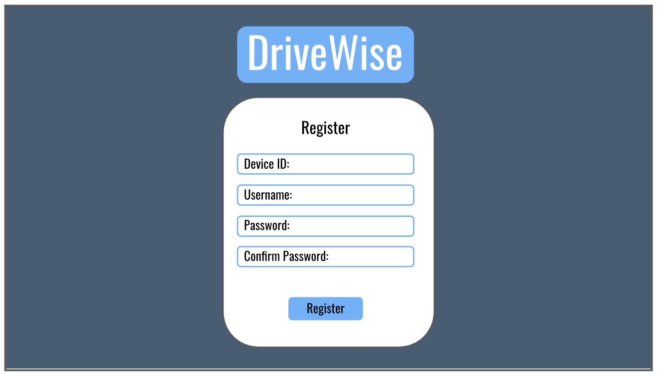


Fig. 6. Web Application: Registration Page

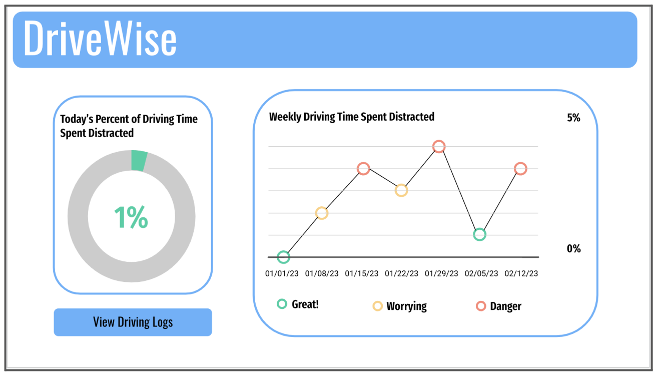
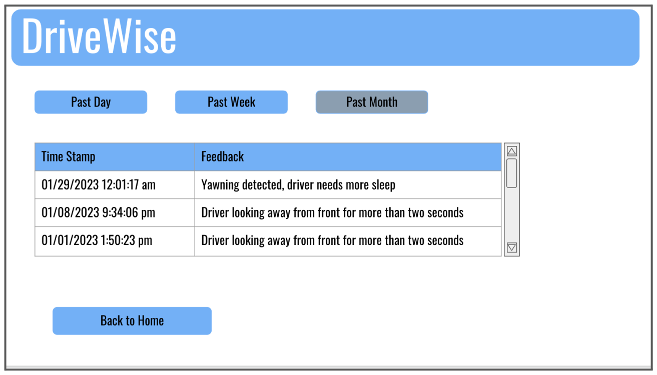


Fig. 7. Web Application: Statistics Page

After logging in or creating an account, the user will be redirected to the statistics page (i.e. home page). The statistics page will display easy-to-understand visuals containing the data received from the Jetson to tell a user how their driving is. The chart on the left shows the percentage of time the user spent driving in an inattentive or sleepy state. The graph on the right shows the quality of their driving over the past week so they can see trends in driving behavior. There is also a link to the user's driving logs for further detailed information.



Time Stamp	Feedback
01/29/2023 12:01:17 am	Yawning detected, driver needs more sleep
01/08/2023 9:34:06 pm	Driver looking away from front for more than two seconds
01/01/2023 1:50:23 pm	Driver looking away from front for more than two seconds

Fig. 8. Web Application: Data Log Page

By clicking the "View Driving Logs" button on the statistics page, the user is redirected to the data log page, shown in Fig. 8 above, displaying each instance that the DriveWise device has flagged as unsafe driving with a timestamp and the corresponding feedback. The user can select whether to view their data logs from the past day, week, or month.

VII. Test, Verification and Validation

A. *Classification and Feedback Are Each Under One Second*

In order to meet the safety guidelines, we want the audio feedback to happen within one second so that the total time taken for computation, classification, and feedback is under two seconds which allows the driver to have time to correct their behavior. This also takes into account how we want the GPU parallelization that is compatible with the Jetson and the facial and landmark detection algorithms to run under one seconds as well so that computation time for detecting attention to the road and drowsiness is under one second, which in total adds up to under the two second limit. We plan to test this by measuring how long it takes from start to end to produce a result and doing this for several trials. We also want to run the computation for one minute to obtain the average frames per second.

B. *Detection Accuracy of 90% for Ideal Conditions/85% for Nonideal Conditions*

Our next design requirement is that the facial detection algorithm is unfazed by head movements so it can meet the use case requirement of an inattention detection accuracy of at least 90% in ideal conditions and 85% in nonideal conditions. We want to test the device's accuracy in these ideal conditions by replicating the scenarios for ideal and nonideal conditions (which can be referenced in the project's scope) with three separate users and doing driving tests to record the correctness of the feedback.

VIII. Project Management

A. *Schedule*

The updated Gantt chart of the schedule is located in the appendix. It contains each part of the project and how the tasks are broken down. We divided it by each member of the team. The red boxes represent all of us, the purple is Elinora, the blue is Sirisha, and the green is Yasser.

B. *Team Member Responsibilities*

In terms of division of labor, we have given each person two areas to focus on, one being a primary and the other being a secondary. This division was based on each person's strengths, so there will also be a good amount of overlap. Mainly Yasser will be focusing on the software in regards to machine learning and computer vision. Elinora will be working on hardware and the user interface part of the software. Sirisha will be working on the user interface part of the software as well as the machine learning part of the software, and she will also provide assistance with hardware as needed.

We all are equally contributing to the work for the presentations and reports for this project as well as the integration and testing of the hardware with the software and user testing.

Fig. 9.

C. *Bill of Materials and Budget*

Table I. Bill of Materials

Component	Cost	Status
NVIDIA Jetson Xavier AGX	$1,200-$2,000 x1	Borrowed from department for free
K-Tech Mini Portable Speaker	$14.99 x1	Ordered from Amazon, has been delivered
ZENVAN USB-C to Headphone Jack	$7.98 x1	Ordered from Amazon, has been delivered
ELP 2 megapixel Hd Free Driver USB Camera	$40.99 x1	Ordered on Amazon
USB-A to USB-C Charging Cables	x1	Already had this part

HiLetGo MPU-6050 MPU6050 6-axis Accelerometer Gyroscope Sensor	$9.99 x1 (pack comes with 3)	Ordered on Amazon
ZTE MF833V 4G LTE USB Modem Dongle	$54.99 x1	Order pending
Total Budget	$128.94 used	$471.06 left over

D. *Risk Mitigation Plans*

Our first potential risk is having malfunctioning hardware components. We will mitigate this by repurchasing the necessary part with our budget, which is currently high at $471.06 still since our most expensive component was able to be borrowed from the ECE department. We also took this possible risk into account very early on and started to test each of our components individually to allow for enough time for a new part to arrive in case something needed to be reordered.

Another risk we could run into is that OpenCV DNN GPU Parallelization isn't compatible with the Volta GPU. We would have to switch to using Dlib instead of Open CV DNN if it isn't fast enough since there is a tradeoff between speed and accuracy between the two options for facial detection.

We could potentially also face issues with sending data form the Jetson to the web application over cellular data using the dongle. If this approach is ineffective, our mitigation plan is to purchase a WiFi dongle that is compatible with our Jetson and connect it to a hotspot on the user's phone in order to access the internet and send data to the web application.

Our last main risk is that the calibration process may be unclear for first-time users, or the device use in general. We aim to mitigate this by doing extensive user testing and changing the audio responses from the Jetson during the calibration process and the feedback as we see fit according to the user's feedback.

IX. Related Work

There have been a few products created on the market similar to what we are proposing to implement. Companies like Cipia, Smart Eye, Eyeware Tech, and Seeing Machines have created quite comprehensive products that track many aspects of a driver's behaviors, and some even go as far as to track how other people in the car are behaving. From research on these companies and their products, they all seem more focused on commercial applications rather than the average daily driver, which is what DriveWise aims to do.

We have also come across research in this field that has aimed to do similar things as us, particularly in the area of tracking a driver's eye in ideal and nonideal conditions.

We have some ideas for future work on DriveWise if time permits or if we have the chance to work on it even past this course. These ideas include adding detection of hands on a steering wheel and some way to remind driver's to check their blind spots when changing lanes.

X. Summary

We hope to encourage safe driving practices through the use of a system that detects driver's inattention to the road and drowsiness through computer vision algorithms and a feedback system that alerts drivers of bad driving practices through the use of a speaker. Upcoming challenges that we anticipate in this project is the high possibility of drivers not being in ideal conditions as not having enough background light could affect the accuracy of our computer vision algorithms, thus skewing our testing and validation results.

Glossary of Acronyms

CV	Computer Vision
NHTSA	National Highway Traffic Safety Administration

References

[1] Vardan Agarwal, *Face Detection Models: Which to Use and Why?*, July 2, 2020, [Online]. Available: https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c

[2] Vardun Agarwal, *Facial Landmark Detection for Occluded Angled Faces*, August 5, 2020. Available: https://towardsdatascience.com/robust-facial-landmarks-for-occluded-angled-faces-925e465cbf2e

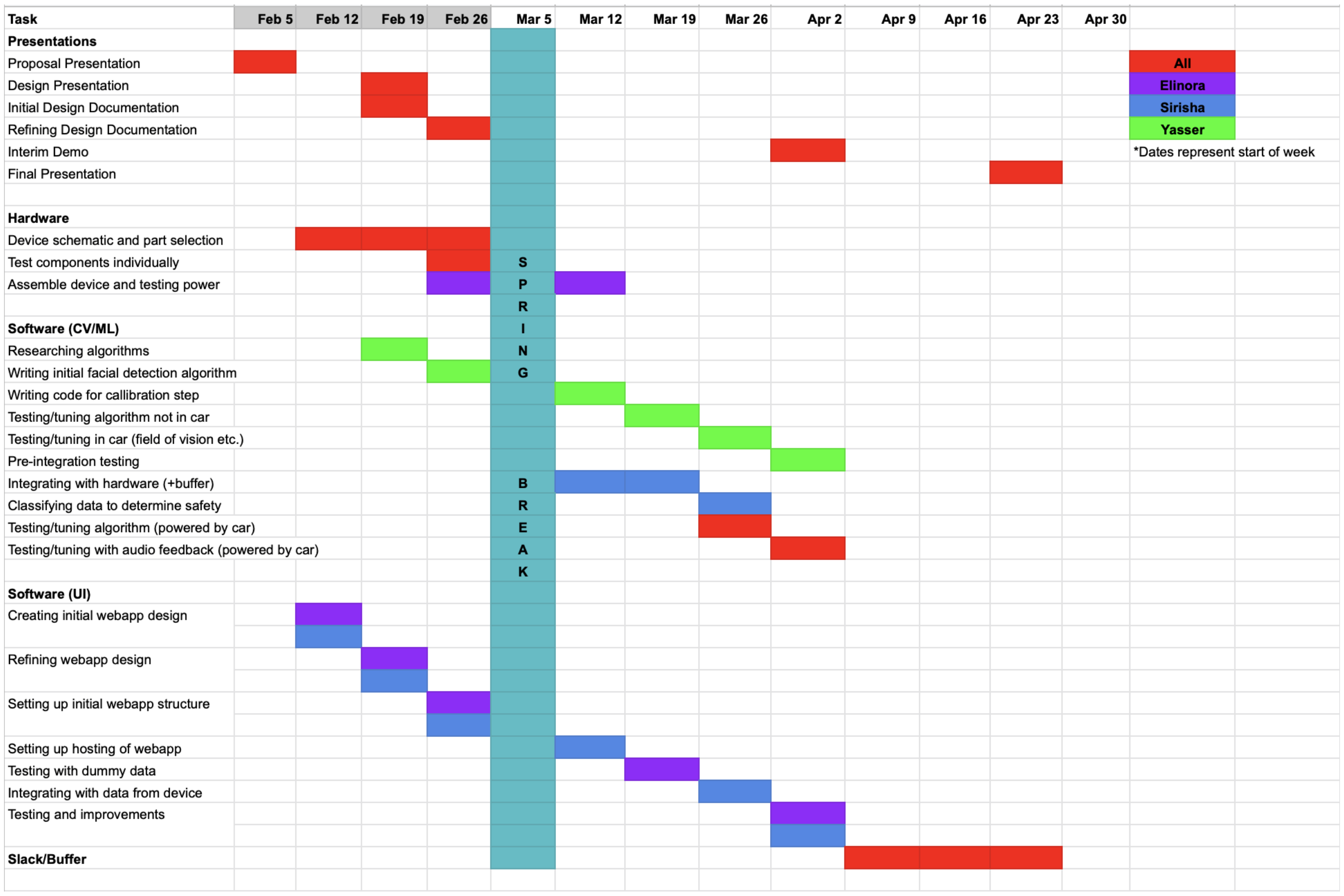
[3] Yin Guobing, *cnn-facial-landmark*, April 22, 2019. Available: https://github.com/yinguobing/cnn-facial-landmark

[4] Adrian Rosebrock, *Deep Learning with OpenCV*, August 21, 2017. Available: https://pyimagesearch.com/2017/08/21/deep-learning-with-opencv/

[5] Eric A. Taub, *2-Second Rule for Distracted Driving Can Mean Life or Death*, September 27, 2018. Available: https://www.nytimes.com/2018/09/27/business/distracted-driving-auto-industry.html#:~:text=The%20National%20Highway%20Traffic%20and,a%20time%2C%20the%20agency%20says.

[6] Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa Al-Khalifa, Moez Krichen, et al.. "Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application." ICIoT 2020, 2020, Doha, Qatar. pp.237-242, ff10.1109/ICIoT48696.2020.9089484ff. ffhal02479367f

APPENDIX



Task	Feb 5	Feb 12	Feb 19	Feb 26	Mar 5	Mar 12	Mar 19	Mar 26	Apr 2	Apr 9	Apr 16	Apr 23	Apr 30	
Presentations														
Proposal Presentation	🟥													🟥 All
Design Presentation			🟥											🟪 Elinora
Initial Design Documentation			🟥											🟦 Sirisha
Refining Design Documentation				🟥										🟩 Yasser
Interim Demo									🟥					*Dates represent start of week
Final Presentation												🟥		
Hardware														
Device schematic and part selection		🟥	🟥	🟥										
Test components individually				🟥	S									
Assemble device and testing power				🟪	P	🟪								
					R									
Software (CV/ML)					I									
Researching algorithms			🟩		N									
Writing initial facial detection algorithm				🟩	G									
Writing code for callibration step						🟩								
Testing/tuning algorithm not in car							🟩							
Testing/tuning in car (field of vision etc.)								🟩						
Pre-integration testing									🟩					
Integrating with hardware (+buffer)					B	🟦	🟦							
Classifying data to determine safety					R			🟦						
Testing/tuning algorithm (powered by car)					E			🟥						
Testing/tuning with audio feedback (powered by car)					A				🟥					
					K									
Software (UI)														
Creating initial webapp design		🟪												
		🟦												
Refining webapp design			🟪											
			🟦											
Setting up initial webapp structure				🟪										
				🟦										
Setting up hosting of webapp						🟦								
Testing with dummy data							🟪							
Integrating with data from device								🟦						
Testing and improvements									🟪					
									🟦					
Slack/Buffer										🟥	🟥	🟥		

Fig. 10. Gantt chart with milestones and team responsibilities