Capstone Project Phase B

StayAwake – Driver Drowsiness Detection System

Supervisor: Aliza Attaly

Project Number: 22-2-D-13

Students:

Arik Zagdon Arik.zagdon@e.braude.ac.il

Dvir Ben-Simon Dvir.ben.simon@e.braude.ac.il

TABLE OF CONTENT

Abstract. The majority of car accidents are mainly caused by human mistakes, in addition, drowsiness and fatigue of drivers are among the significant causes of road accidents. We attempt to overcome this scenario by developing a car system that analyzes driver fatigue levels predicting drowsiness cases to notify the driver before an accident can occur as well as detects falling asleep incidents and warn the driver in real-time. our project includes an image processing procedure to recognize the driver's face and behavior and an algorithm that will analyze driver image frames and extract unique features of drowsiness.

StayAwake source code: *https://github.com/arikz-tech/StayAwake*

Keywords: DeepLearning, DrivingSafety, FaceDetection, DrowsinessDetection, ImageProcessing

1. INTRODUCTION

Nowadays, there is an increase in vehicle uses on the road. Human life is getting busier causing drivers to be drowsier while driving. Consequently, there is increasing in car accidents. Each year we were exposed to 100,000 crashes, 71,000 injuries, and 1550 fatalities because of drowsy driving these fatalities constitute 2.4% of the total fatalities in accidents, according to the NSC [1]. Our working development aims to propose a new system to efficiently track eyes and mouth movements using image processing algorithms, examine that driver's behavior, and notify him by recognizing his drowsiness and fatigue levels. The system can evaluate fatigue levels by using three main parts. Monitor Blink per minute (BPM), Yawning detection, and snoozing. As [2] indicates that fatigue is associated with increased blink frequency. As a result, our system can determine how many blink drivers had per minute and take it into account on drowsy evaluation. In addition, the duration of eye blink can define drowsiness as [2] mentions. The average duration of a single blink is 0.1-0.4 seconds [3]. Additionally, one single yawning can indicate that the driver is drowsy. Our system includes the following tools: a camera, mini pc, speaker, and monitor. the majority of the project will be the software that's running on the mini pc. The camera monitors the driver's face the whole trip. The software algorithm analyzes driver drowsiness features the whole monitored time and saves this data. If the algorithms recognize exceptional fatigue levels it notifies the driver. In addition, if the system notice that the driver fell asleep it starts hazard mode and sounds instantly alarm. At the end of travel, the system produces a report of the driver's fatigue levels as a graph.

Drowsiness features:

- Blinking, number of blinking in one minute determines if you are in a sleepy zone.
- Yawning detection.
- Snoozing detecting duration of single blink.

Hazard mode:

- a closed eye for more than 2 seconds.
- head falling.

2. RELATED WORK

In fact, driver drowsiness detection techniques have been researched intensively in recent years. Researchers have proposed various measures to detect drowsiness signs. The techniques are divided into three major categories: physiological-based detection, vehicle-based detection, and image processing detection. In the physiological field, we see various methods that include physiological sensors: Electroencephalography (EEG), Electrocardiography, Heart rate variability(HRV), etc., and combine these sensor data with machine learning algorithms. In [4] they describe the LSTM model that uses EEG data they got 94.31% accuracy, but these EEG sensors require you to wear a helmet. In [5] they made multivariate statistical process control to detect abnormalities in HRV and compare their algorithm with EEG learned model. Their accuracy was excellent, but they used a small dataset that include only twelve persons. In the vehicle-based detection field, we found systems that monitor and analyze vehicle behavior, like steering wheels angle and lateral distance. In [6] they used steering wheel angles and speed movements to identify the driver's drowsy while he rotates the steering wheel. In the image processing detection field, there is a variety of methods that can be applied. Using eye mouth and head tilt detections. In [7] their system identifies the face, and eyes and localizes pupil and iris boundary, they used the circular Hough Transform on the extracted eye region through the projection functions to locate the iris position. In [8] they used Convolutional Neural Network (CNN) detects drowsy features yawing and eye movements, they achieved an accuracy of 98.81% they had 322 videos from the YawdDD dataset and approximately 9 hours of the dataset including normal driving, yawning, slow blink rate, falling asleep, burst out laughing from NthuDDD dataset.

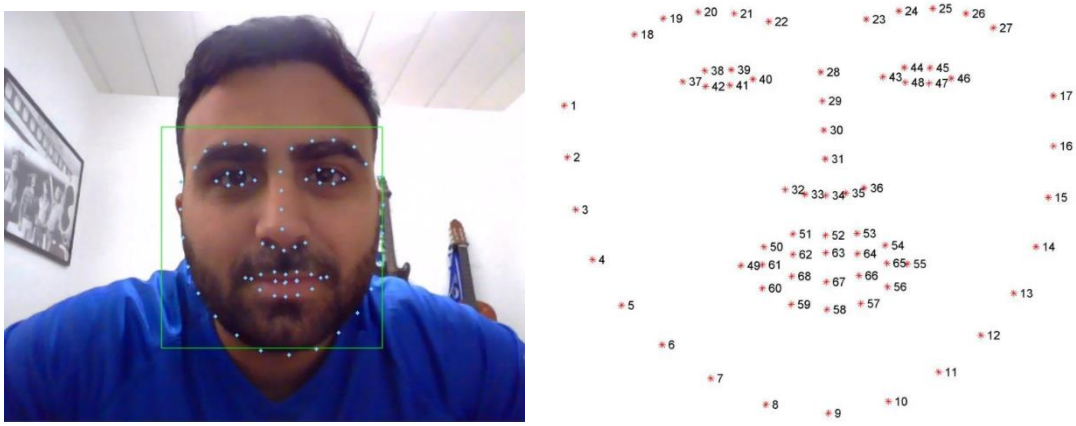
3. BACKGROUND

3.1. Dlib

"Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real-world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high-performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge." [9]

3.2. Facial Points Detection

Facial landmarks are regions of interest that can uniquely identify different components of the face, such as eyes, eyebrows, lips, and nose. These landmarks can be described as a series of facial key points. To extract the facial feature points, we used the real-time face estimation open-source code from the Dlib c++ library [9]. the dlib library contains a pre-trained detector that estimates the coordinate of 68 points that are mapped facial regions of interest.



3.3. OpenCV

"OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-

art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. "[10]

3.4. Raspberry Pi

"The Raspberry Pi is a low-cost, credit-card-sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games." [11]

3.5. EAR - Eye Aspect Ratio

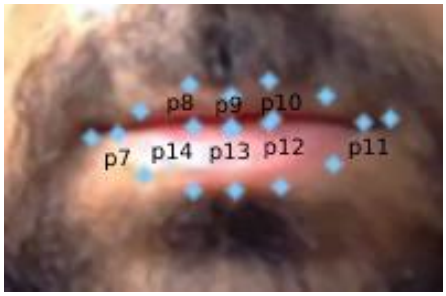
Eye Aspect Ratio It is a scalar quantity obtained by detecting a face from an image, finding the Euclidean distance of the corresponding eye coordinates, and substituting it into the following formula:

$$EAR = \frac{\|P2 - P6\| + \|P3 - P5\|}{2\|P1 - P4\|}$$

3.6. MAR - Mouth Aspect Ratio

MAR is the ratio of vertical distance of mouth to the horizontal distance of the mouth. The three vertical distances and one horizontal distance are considered in MAR estimation. When driver open his mouth, the vertical distances will start increasing and the horizontal distance will decrease slightly.

$$MAR = \frac{\|P14 - P8\| + \|P13 - P9\| + \|P12 - P10\|}{3\|P11 - P7\|}$$



3.7. Head Pose Estimation

In computer vision the pose of an object refers to its relative orientation and position with respect to a camera. You can change the pose by either moving the object with respect to the camera. To detect object motion, we define two matrices:

- **Rotation matrix**

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When $\boldsymbol{\theta}$ is the rotation degree.

- **Translation matrix**
  When $\boldsymbol{v_x}$ is value that represent moving in x axis, $\boldsymbol{v_y}$ is value that represent moving in y axis, and $\boldsymbol{v_z}$ is value that represent moving in z axis

$$\begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. ENGINEERING PROCESS

In the beginning, we started researching the causes of car accidents, we continued into the subtopic of drowsiness while driving accidents. We noticed that this issue is causing a lot of fatal accidents as mentioned in [1], and we desire to solve this problem. Later, our procedure included two steps, firstly we started to study techniques to identify drowsiness. We found that most of the projects in this field put a lot of effort into trying to identify if the driver is sleeping, whereas we wanted to develop a system that is more focused on detecting drowsiness and notifying the driver in advance before an accident could occur.

Secondly, we saw a few options about how to detect drowsiness, and we started to implement and investigate how it works. After a couple of experiments with these technologies, we choose to focus on behavioral detection and consider three major behaviors: blinks per minute, snooze and yawning detection, using the Dlib library [9] that gives us face detection and face point landmarks detection.

After we had some experience using this face post processing eye, mouth, and face detection, we started to design the system architecture. After the architecture design stage, we started to develop the system: Firstly, we created new project on GitHub, and start the implementation in python programming language. The implementation includes StayAwake logics component, and StayAwake GUI component. We imported all dependencies packages that our project required (can be found on requirements.txt file). In the logic side we started to implement all the features mentioned above MAR, EAR, and head pose estimation. After we had all mathematical calculation functions, we continue to implement the two sub systems: "Fatigue Detection", "Fall A Sleep Detection".

After we finished all the logic side of the system we started to work on the GUI, using Tkinter library.

During the implementation process, we faced some difficulties:

When we started the blink detection, we faced issue that happened because the system run in a high frame rate. Each frame recognized as a blink because ear was below the threshold. To solve this issue, we started to use computer time between closing/opening the eye (ear > threshold or ear < threshold) and decided which blink duration is counted as a real blink.

Moreover, in snooze detection, we faced a problem that the blink and the snooze overlap. To face that issue, we used another blink duration for snoozing, choosing different duration for different purposes of blink/snooze.

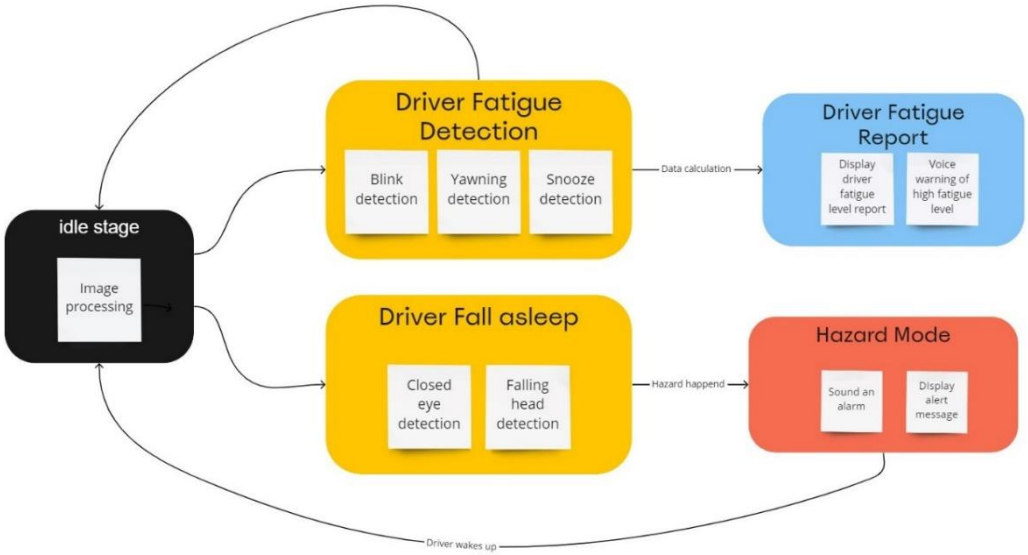
In addition, when we implemented our GUI side of the system, we had an issue when we play a sound (alarming and notifying the driver) it causes the system to freeze and stop the frame recording. To solve this issue, we used multi-thread implementation.

In the development process, we worked on different working environments, and we found that we need to fine-tune the parameters like MAR, and EAR threshold.
In the final project process, the system will run on a specific computer (Raspberry pi) and the MAR and EAR will be set to a fixed size. We got the equipment for our project in the last stages of the project implementation, casus us to implement it on our private computers, and unable to test our system in our chosen hardware components. After we got the equipment, we started to deploy our debug version on the mini pc, we understood that the picked computer, Raspberry Pi 4 (4GB) resulted in poor performance in the image processing stages, in contrast to our laptop computers. In addition, we got a camera that does not have night vision technology, because of a lack of equipment.

5. PROJECT DESIGN

5.1. Project Main flow

This section describes how our system should work, during all driving sessions. There is a camera that captures frames in real-time and makes calculations on the captured frames in the backgrounds using libraries and our logic. The process contains the following steps:
Firstly, the system will capture one frame and extract 68 facial landmark points using Dlib library [9] with "shpae_predictor_68_face_landmark".
Secondly, there are two processes that are running in parallel. The first one is "Driver Fatigue Detection", this process is responsible for rating the driver's fatigue level. during this step, the system logic is to count the number of every single fatigue

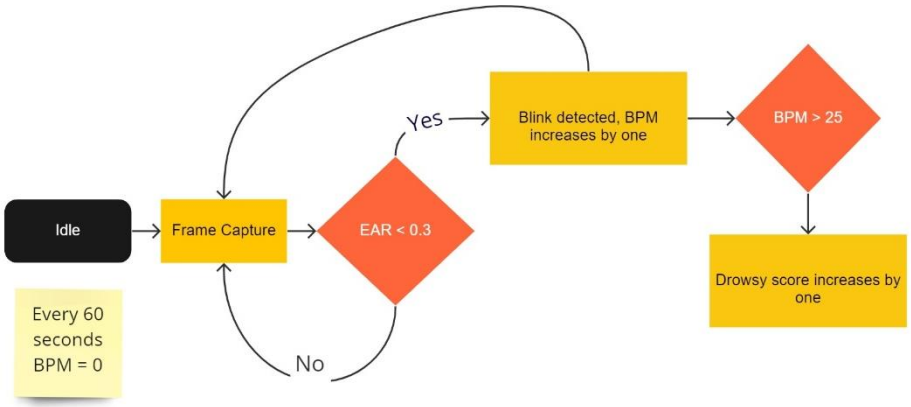


behavior (blink, yawning, snooze). If it exceeds the maximum threshold the system will notify the driver in real-time displaying a fatigue levels graph. The second steps are "Driver Fall asleep", this process is responsible for detecting fatal situations: driver closing eye for more than 2 second or driver's head falling. These two scenarios indicate the driver's attention isn't on driving, and the system will transfer into "Hazard Mode". This mode causes the system to sound a noticeable alarm and display it on the screen monitor.

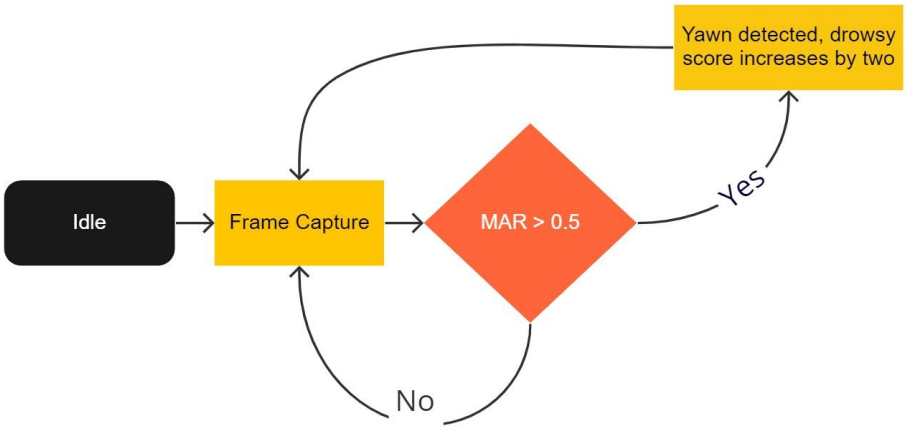
5.1.1. Driver Fatigue Detection

This process is responsible for detecting suspicious drowsy behavior of the driver, it includes the driver's drowsy score, and every drowsy feature will have his score depending on its importance. In addition, this process will display on the monitor every 10 minutes graphs of every drowsy feature score.

- Driver Blink Detection
  Calculate EAR if the value is below 0.2 it indicates that the driver blinked. Count the number of BPM (blink per minute) using that technique, if it exceeds the fixed threshold (25 blinks per minute) drowsy score increase by one.

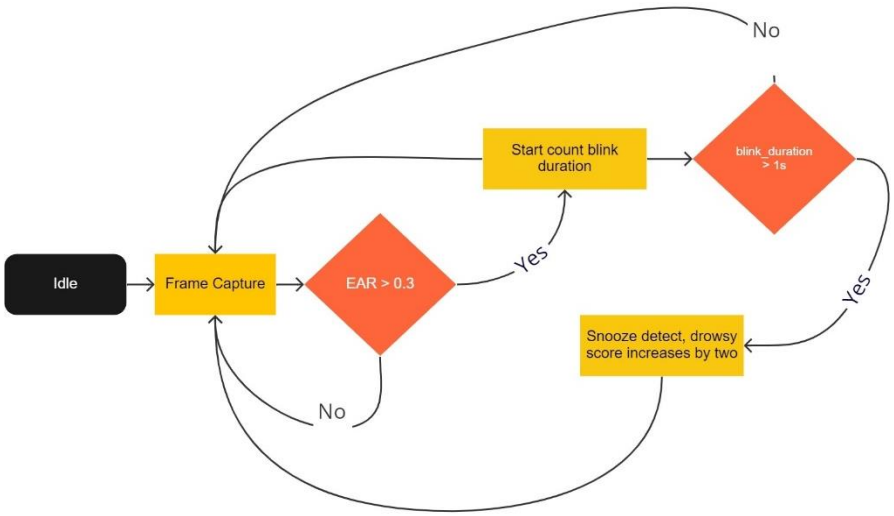


- Driver Yawning Detection
  Calculate MAR if it exceeds 0.5 it indicates that the driver is yawning. the drowsy score increases by two.



- Driver Snooze Detection
  Calculate EAR if the value is below 0.2 It indicates that the driver blinked, if his blink takes longer than 1 second it indicates that the driver is snoozing, and the drowsy score increase by two.

  if the total drowsy score exceeds the limit of 10 points, the system will notify the driver by sounding a voice message that informs the driver that he should take a rest.

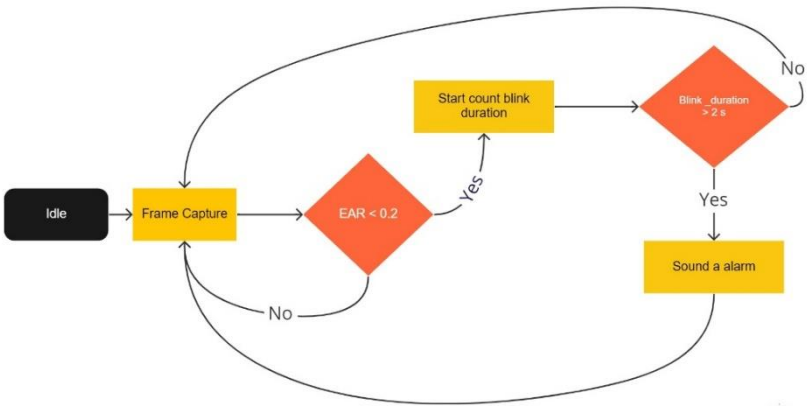


5.1.2. Driver Fall Asleep Detection

This process is responsible for detecting dangerous scenarios and alerting the driver in real-time, there is two state that this sub system can be activated, driver closed eye or driver head fall.

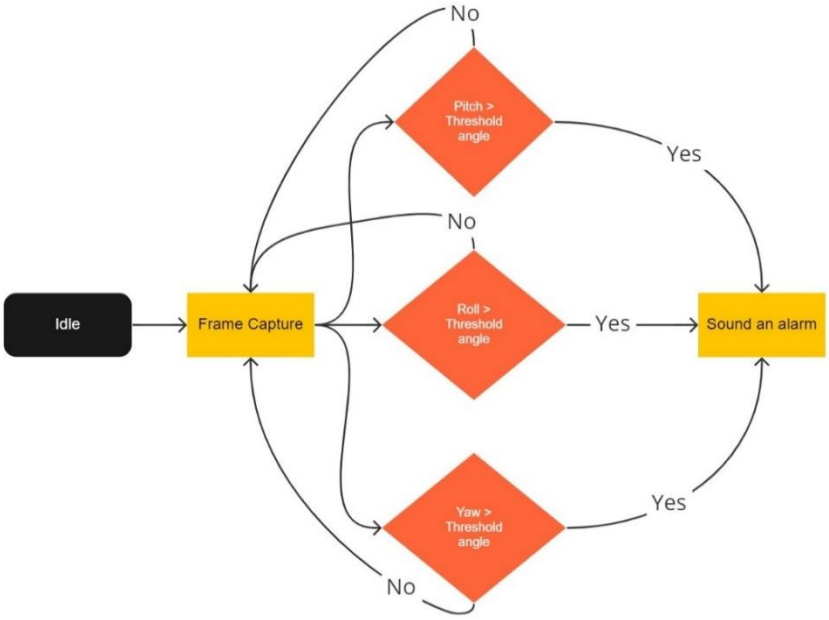
- **Driver Closed Eye**

  Calculate the rate of the blink, when EAR parameter is below 0.2, start count the blink duration. If the exceeds one seconds the system recognize that fall asleep situation occurs, alert the driver by sound an alarm and display it on monitor.



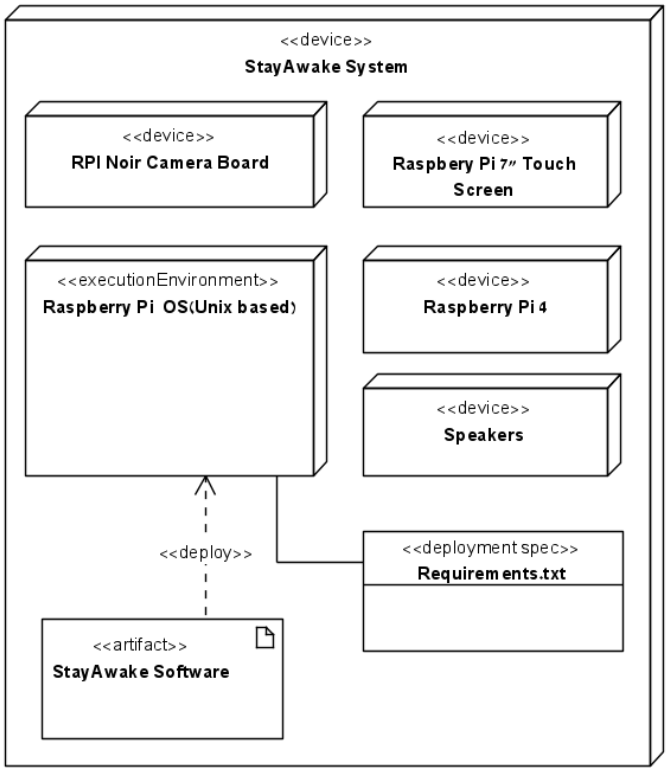
- **Driver Head Fall**

  Calculate the head pitch, roll, and yaw degrees, if some of the degree is below the define threshold it indicates that the driver head fallen to x, y or z direction, and the system recognize that fall asleep situation occur, alert the driver by sound an alarm and display it on monitor.

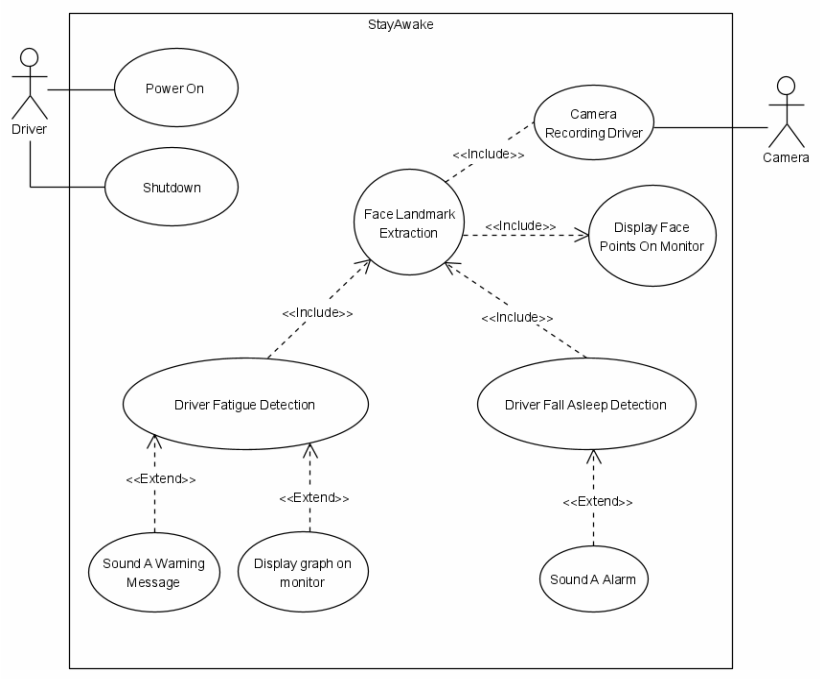


5.2. Project Diagrams

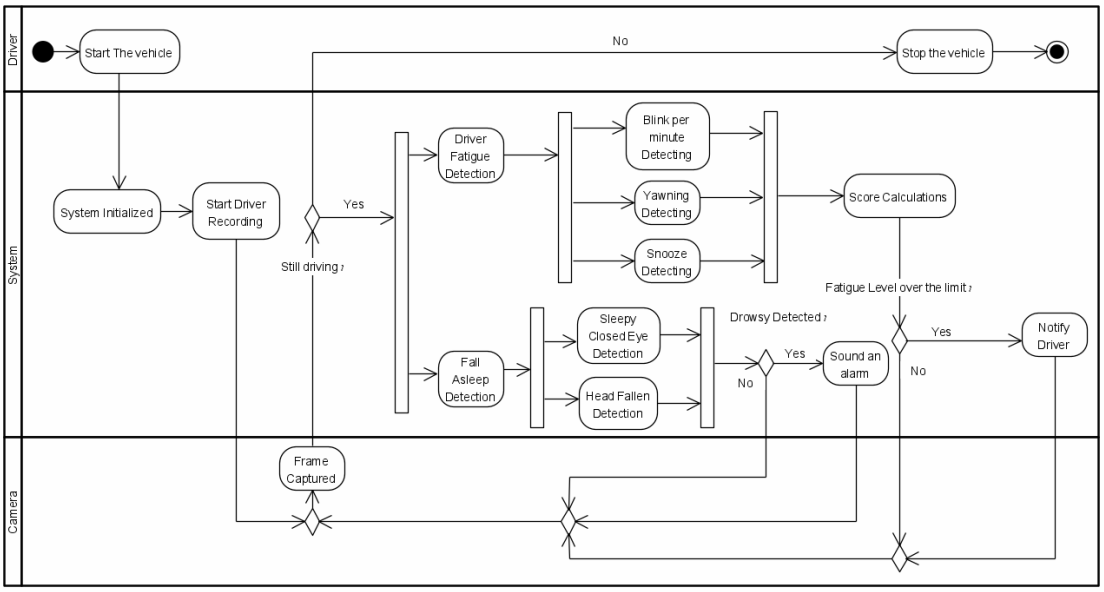
Deployment Diagram



Use Case Diagram



Activity Diagram



6. VERIFICATION PLAN

- **System Tests**

Test Name	Description	Expected Result
CameraTest	Start the camera, verify if it works properly	System display video frames on the monitor as expected
FaceRecognitionTest	After the system is loaded, show a human face	System displays square around the head from the captured frame
FaceLandmarkPointsTest	After the system is loaded, show a human face	System displays all 68 landmark points on the captured frame

- **Functional Tests**

Test Name	Description	Expected Result
EARTest	After the system is loaded, show a human face, close the eyes	EAR < 0.2 when the eye is closed
MARTest	After the system is loaded, show a human face, open the mouth	MAR > 0.5 when the mouth is open
BlinkDetectionTest	After the system is loaded, show a human face, and make a single blink	EAR < 0.2 0.02 < Blink Duration < 0.3 Blink occur, increase blink counter by one
YawningDetectionTest	After the system is loaded, show a human face, and illustrate yawning	MAR > 0.5 0.3 < Yaw Duration Increases yaw counter by one
SnoozeDetectionTest	After the system is loaded, show a human face, and illustrate snoozing	EAR < 0.2 0.3 < Blink Duration < 0.9 Snoozes occur, increase snooze counter by one
HeadFallDetectionTest	After the system is loaded, show a human face, and tilt the head to one side.	Sound an alarm
FallAsleepDetectionTest	After the system is loaded, show a human face, and a closed eye for more than 1 seconds	Sound an alarm
BlinkPerMinuteTest	After the system is loaded, show a human face, and a blink 26 times in one minute	Increases hazard blinking indicator by one

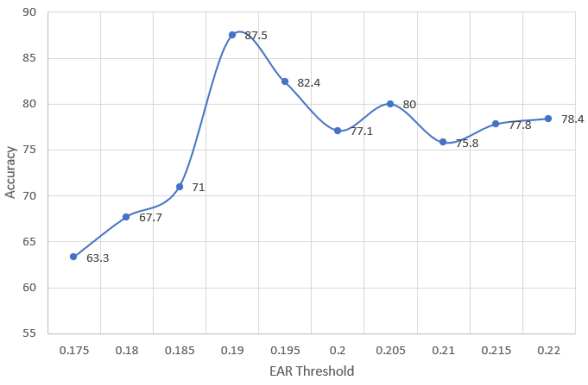
- **Accuracies Tests**

Test Name	Description	Expected Result
BlinkAccurecyTest	After the system is loaded, show a human face, and a blink 100 times	90% of detection succeed
YawningAccurecyTest	After the system is loaded, show a human face, and a yaw 100 times	90% of detection succeed
SleepAccurecyTest	After the system is loaded, show a human face, and illustrate sleeping(closed eye for more than one second) situation for 100 times	95% of detection succeed
HeadFallAccurecyTest	After the system is loaded, show a human face, and illustrate head falling situation for 100 times	95% of detection succeed

Experiment 1: EAR Thresholding

In this experiment we test which EAR threshold is giving the best accuracy result

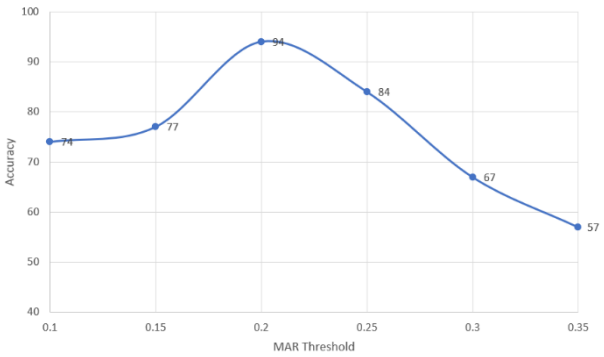
EAR Threshold	Accuracy	True Positive	False Negative	False Positive	True Negative
0.175	63.3	19	11	0	0
0.18	67.7	21	9	1	0
0.185	71	22	8	1	0
0.19	87.5	28	2	2	0
0.195	82.4	28	2	4	0
0.2	77.1	27	3	5	0
0.205	80	28	2	5	0
0.21	75.8	25	5	3	0
0.215	77.8	28	2	6	0
0.22	78.4	29	1	7	0



Experiment 2: MAR Thresholding

In this experiment we test which MAR threshold is giving the best accuracy result

MAR Threshold	Accuracy	True Positive	False Negative	False Positive	True Negative
0.1	74	28	2	8	0
0.15	77	27	3	5	0
0.2	94	30	0	2	0
0.25	84	26	4	1	0
0.3	67	20	10	0	0
0.35	57	17	13	0	0



7. VERIFICATION PROCESS

In verification process we execute all our test that we are mentioned above (Verification Plan), we divide it to three sections: System Test, Functional Test and Accuracy Test. And we will describe every section below:

- **System Test**
  We assemble all the components together: camera, display, computer, and speakers. start the application and we saw that the camera works properly, we are succeeded to recognize faces as expected, and show 68 landmark points on the face. In addition, we play alarm sound to check that the system sound is working properly. All our expected system tests are passed, which indicates that the system is working fine.

- **Functional Test**
  We run our software, After the system loaded and started to recognize driver face, we start to test all the feature that include in the system manually by illustrating blink, yaw, and snooze.
  When we try to detect blink, we check if the EAR getting below 0.2 threshold and the amount of duration time is less than 0.3, if all these conditions are passed, we check that the blink counter is increased by one.

For the snooze detection we check for the same EAR threshold, but instead we are changing the time duration into more than 0.3 seconds and less than one second, we check that the snooze counter is increased by one.

For yaw detection we check if the MAR is above 0.5 and if the yaw duration time is above 0.3, after that we check if the yaw counter is increased by one. To check that hazardous situation like a closed eye for more than one second or a head fall, indicating that the driver falls asleep. We demonstrate those acts and wait for the system to notify us by sounding an alarm.

- **Accuracy Test**

  In the accuracy test plan, we repeatedly use our functional tests, to achieve 90% in all tests. For the blink accuracy test, we used BlinkDetectionTest functional test, repeating it 100 times and checking how many times the system recognized it as a blink.

  For yawing accuracy test we used YawingDetectionTest functional test, repeating it 100 times and checking how many times the system recognized it as a yaw.

  For the head fall accuracy test, we used HeadFallDetectionTest functional test, repeating it 100 times and checking how many times the system sounded the alarm.

  For the sleep accuracy test, we used SleepAccurecyTest functional test, repeating it 100 times and checking how many times the system sounded the alarm.

$$Accuracy(blink) = \frac{\#Blink\ Detection}{\#Blink\ occur}\ ,\ Accuracy(sleep) = \frac{\#Sleep\ Detection}{\#Sleep\ occur}$$

$$Accuracy(yaw) = \frac{\#Yaw\ Detection}{\#Yaw\ occur}\ , Accuracy(head\ fall) = \frac{\#Head\ Fall\ Detection}{\#Head\ Fall\ occur}$$

$$Accuracy(snooze) = \frac{\#Snooze\ Detection}{\#Snooze\ occur}$$

8. RESULT AND CONCLUSIONS

A car accident can occur because of drowsiness driving. Our system tries to solve this issue by using image processing technologies. Detect human face symptoms that indicate higher fatigue levels and notify the driver or surrounding passages. We faced some difficulties with the design and implementation of the project. One of

our system's goals is to fit into the vehicle. Consequently, we choose the hardware component to be slim as possible. In the debugging of the project, we saw that our computational power is not strong enough, and the frame rate is getting slower. As a result, we continue the implementation in our private laptops, where it works properly. With good hardware components, the system shows significant results and a nice driver fatigue representation.

In conclusion, our system provides good results in detecting driver fatigue symptoms, like frequent blinking, snoozing, and yawing. The driver gets an informative message that indicates his drowsy levels during all driving sessions. In addition, the system recognizes hazardous situations when the driver falls asleep, alerting him in real time. The system provides poor results when the face isn't recognized well. For future work, need to put more effort into the face recognition field, to increase the system's performance and accuracy. The provided system is in the early stages, a prototype. It is separated from vehicle components and required for deployment and installation as a part of the car's components. In future work, the system can be pre-built in the vehicle display.

9. USER DOCUMENTATION

9.1. User Guide Operation Instruction

General Description

StayAwake is a system installed inside the vehicle, the primary purpose of the system is to detect drivers' fatigue symptoms, analyze them and notify the driver about his drowsy level during all driving sessions. In addition, the system detects critical sleep signs like head falls or closed eyes. and sounds an alarm. The system records the driver while he's driving using the camera, each frame proceeds, capturing the driver's face, and is being analyzed. it catches blinks, yaws, and snoozing, and calculates it to a drowsy score every minute, if it exceeds the limit it's adding to the driver's drowsy level which includes five different levels. and notify the driver with a suitable message. Moreover, if the driver's head falls or he has closed his eyes for more than one second the system turns into hazard mode and sounds an alarm in order to wake the driver up. The system includes the main process unit (Raspberry pi computer) running the python StayAwake application, monitor, camera, and speakers. Target users are every driver.

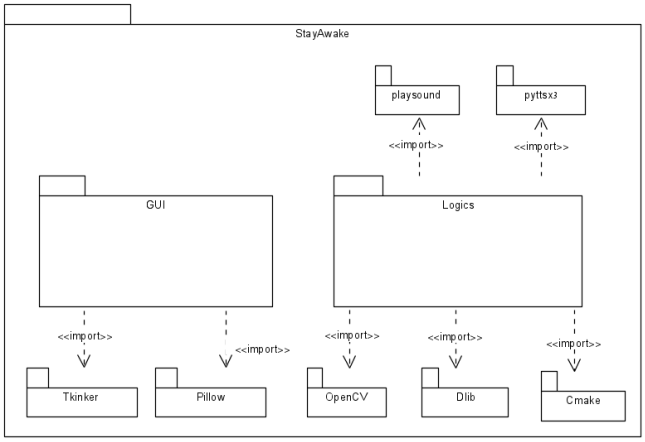
Operational Process

The system includes only one scenario:

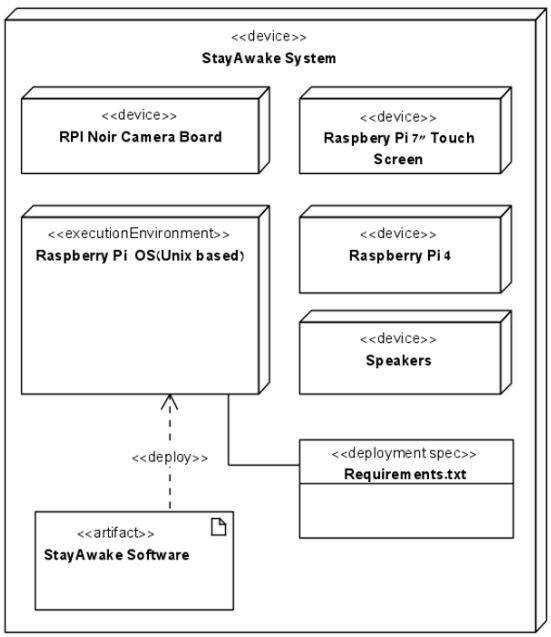
1. Place the system in your vehicle and connect it to the vehicle power supply.
2. Start the vehicle.
3. The software system will boot automatically when the computer is turned on.
4. Start the driving session, pay attention to system instructions, and stay awake.
5. All fatigue and drowsy information will be shown on the monitor and notifications will be sound when needed.

9.2. User Maintenance guide

Package diagram:



Deployment diagram:



9.3. Project Requirements And Installation Guide

Software requirements:

Operation system: Windows 10 or above / Raspberry Pi OS(Unix based)

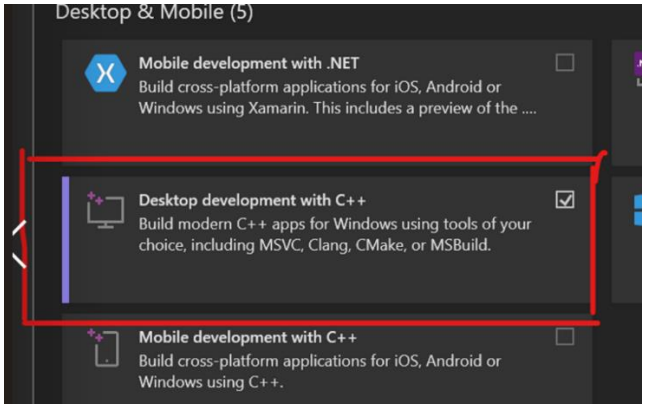
Python 3.6 or above: download here

Python IDE: install pycharm, download here

StayAwake files: download the files from put Github here

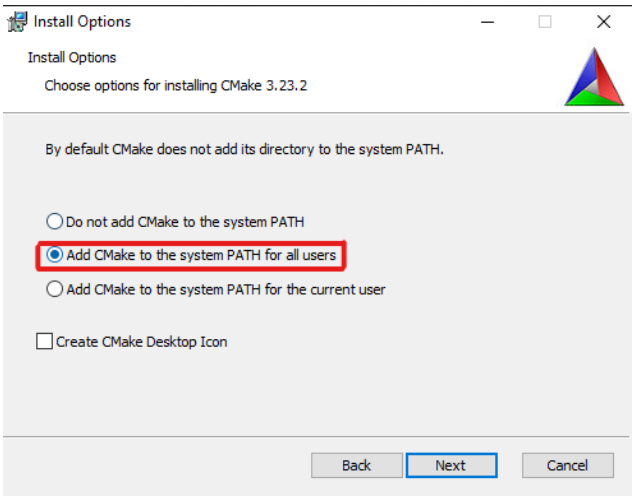
C++ Compiler(visual studio): download here

- download the visual studio code community version after that as you can see in the below image, select Desktop development with c++ while installing VS code.



CMake: download here

- make sure to choose the right version according to your system configuration
- While installing CMake select Add CMake to the system PATH to avoid any error in the following steps.
- Install CMake module: in cmd write the command “pip install cmake”



Dlib installation: Write in cmd “pip install dlib”
Requirements.txt installation: to install all other packages required to run the StayAwake system, you need to install the file “requirements.txt” in the project repository to install it using the command “pip install -r requirements.txt”.

Hardware requirements:

- Camera: RPI Noir Camera board
- Monitor: Raspberry pi 7” Touchscreen
- Speakers: Bluetooth / USB Speakers
- Computer: Raspberry pi 4GB

After assembling all the hardware components and software requirements installations, run the “StayAwake.exe” file that is included in the project repository.

10. REFERENCES

[1] L. Borrelli, "Drowsy driving statistics and facts 2022," 2 2022. [Online]. Available: https://www.bankrate.com/insurance/car/drowsy-driving-statistics/.

[2] R. Schleicher, N. Galley, S. Briest and L. Galley, "Blinks and saccades as indicators of fatigue in sleepiness warners: looking tired?," 2008.

[3] S. H.R, "Average duration of a single eye blink," bionumbers, 2001. [Online]. Available: https://bionumbers.hms.harvard.edu/bionumber.aspx?&id=100706&ver=4.

[4] U. Budak, V. Bajaj, Y. Akbulut, O. Atila and A. Sengur, "An Effective Hybrid Model for EEG-Based Drowsiness Detection," 2019.

[5] K. Fujiwara, E. Abe, K. Kamata, C. Nakayama, Y. Suzuki, T. Yamakawa, T. Hiraoka, M. Kano, Y. Sumi, F. Masuda, M. Matsuo and H. Kadotani, "Heart Rate Variability-Based Driver Drowsiness Detection and Its Validation With EEG," 2019.

[6] S. Arefnezhad, S. Samiee, A. Eichberger and A. Nahvi, "Driver Drowsiness Detection Based on Steering Wheel Data Applying Adaptive Neuro-Fuzzy Feature Selection," 2019.

[7] M. Suleman, S. Bin Zulifqar and N. Memon, "Real time driver drowsiness detection using computer vision techniques," 2021.

[8] B. KIR SAVAŞ and Y. BECERİKLİ, "Real Time Driver Fatigue Detection System," 2019.

[9] Office of the Director of National Intelligence, Intelligence Advanced Research Projects Activity and U.S. Government, "DLib C++ library," Dlib, [Online]. Available: http://dlib.net/.

[10] O. team, "OpenCV Documentation," OpenCV, [Online]. Available: https://opencv.org/.

[11] T. R. P. F. Group, "Raspberry Pi," Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/.

[12] J. Feng, Z. Guo, J. Wang and G. Dan, "Using Eye Aspect Ratio to Enhance Fast and Objective," 2019.